Università di Trento
ICT International Doctorate School

Feasibility Study and Requirements Analysis

FM Hardware Wholesaler Company

Business process and software system improvement

Requirements Engineering Course Assignment
Prof. John Mylopoulos

Mirko Morandini - Fabiano Dalpiaz

May 2007

0 Table Of Contents

1 The Problem

This document analyzes the feasibility and the requirements for the improvement of the business process and the software system in the wholesaler company *FM.* The business of FM consists of buying computer hardware from producers and selling them to retail stores. In this context the main issue is *Supply Chain Management* (*SCM*), the flow and storage of materials from point of origin to point of consumption. SCM is fundamental to face the problems of business logistics, and covers inventory management, purchasing, transport, warehousing, and their organization and planning.

Currently, the company adopts a traditional logistics business process, established step by step after its foundation, during the initial growth of its business. Software systems for the various stages in the business process were bought by the various departments as they were needed, at different times from different vendors. Therefore, these systems are poorly integrated with one another. This leads to many gaps in the software-assisted business process and hence many key steps rely on work and experience of the employees.
In order to enlarge the customer base and to increase turnover, the company has to increase customer satisfaction, mainly by reducing delivery times and costs. However, following the current business processes, growing customer demands cannot be satisfied in acceptable time. Moreover, the process is not adequate for the current size of the company, resulting in higher costs caused by inefficient warehousing and office work and the need for well-trained employees.

The management decided to improve the company's business processes. To achieve increased efficiency throughout all business activities, appropriate business logistics software has to be installed, in order to integrate the whole *Supply Chain Management* process (purchasing, warehouse management, sales, and transport). To use the strengths of the software best, the business process has to be properly reorganized.
Reorganizing the business processes, the company can achieve minimized delays for placing orders and for packaging, and a reduction of both delivery and stocking times. This should lead to shorter delivery times and lower costs. Nevertheless, prior to the changes, an accurate feasibility study has to be conducted, considering the costs of the software and the effort of employee training.

2 Context

The organizational context is described in detail in Appendix A. The company is organized in three departments: sales, warehousing and purchasing. The identified external stakeholders are suppliers, system designers and the web-mastering company. Internal stakeholders include the all the company management, namely the president and the department managers. The users of the new system would be all members of the organizational structure and the retailers.
The current business processes, describing the usual workflow performed in the company, are analyzed in Appendix B. The problems in the current scenario are analyzed using the PIECES framework, which is proposed to examine the major problems in an environment. In particular, the examined areas are performance, information, economic, control and security, efficiency, and service.

3 Proposed Alternatives:

In order to propose and select alternatives for the system to-be, we split the logistics business process into three parts: purchasing, warehousing, and selling.
We identify alternative solutions for each of these three processes, and evaluate their combination.

3.1 Proposed Alternatives

In order to propose and select alternatives for the system to-be, we split the logistics business process into three parts: purchasing, warehousing, and selling.
We identify alternative solutions for each of these three processes, and evaluate their combination.

3.1.1 Selling:

1. *As-is*: orders are given directly to the salesmen or (via e-mail) to the company's secretary; they are then stored in the OMS system by the same person which received the order. Invoices and payments are managed using OMS.
2. *New web-based ordering system with lightweight clients for salesmen*: the entire selling process is handled by this new system. Orders are always managed through the website, which can be accessed by customers, secretaries, and salesmen, through a standard browser or light PDA-based interface client. The latter solution is specific for salesmen, and gives them all up-to-date information on products, prices, available quantities, and waiting times. Invoices and payments are handled by this system through the same interface, with different views depending on a role-based access.
3. *Pure web-based ordering based on the legacy system*: the way customers can give their orders to the company is via a web-based interface, namely an e-commerce web site. This site obtains products, prices and available quantities from the OMS, and directly forwards the received orders to the OMS, which manages invoices and payments as usual.
4. *Hybrid web-based ordering based on the legacy system*: customers and salesman can use a system like in option 2, but OMS is still used as the base system that holds the databases and provides the interface for the secretaries.

3.1.2 Warehousing:

1. *As-is*: OMS is used by the employees in the warehouse department to manage the whole warehouse activities. Arriving goods are stored in the system by the warehouse staff, whereas customer orders arrive directly from the selling department through OMS, are printed to prepare deliveries, and the quantities are automatically decreased.
2. *New browser-based warehousing system*: provides the same functions as OMS, and is integrated with the system which manages supply orders. Moreover it provides support to keep an optimal warehouse filling level, giving statistics and hints directly to the purchase department. Goods inbound is managed directly by the purchase system, and warehouse workers only check the correct quantities.
3. *Improved OMS version*: OMS is added with functionalities to support the optimal warehouse filling maintenance. Moreover, it is completely integrated with the purchase system (like in option 2).

3.1.3 Purchasing:

1. *As-is*: SMS is used by the employees of the purchase department to manage the orders given to the suppliers, and to manage invoices and payments. Inbound goods are not managed by the purchase department, but directly by the warehouse.
2. *New browser-based purchase system*: the system is the same as option 2 in warehousing. The functions for purchasing (invoices and payments management) are available in this system, with the same functionalities as in SMS.

3. *Improved SMS version*: SMS is directly integrated with an improved version of OMS. The goods inbound is connected to invoices processing.

3.1.4 Analyzed alternatives combinations:

- Alternative 1: The whole *system as it is* (Selling: as-is, Warehousing: as-is, Purchasing: as-is).
- Alternative 2: *New browser-based system* for ordering, warehousing and purchasing with lightweight clients (Selling: New web-based ordering system with lightweight clients, Warehousing: New browser-based warehousing system, Purchasing: New browser-based purchase system).
- Alternative 3: *Pure web-based ordering based on improved legacy systems*: (Selling: Pure web-based ordering based on the legacy system, Warehousing: Improved OMS version, Purchasing: Improved SMS version).
- Alternative 4: *Hybrid web-based ordering based on the old system:* (Selling: *Hybrid web-based ordering based on the legacy system*, Warehousing: as-is, Purchasing: as-is).
- Alternative 5: *Hybrid web-based ordering based on improved legacy systems* (Selling: *Hybrid web-based ordering based on the legacy system*, Warehousing: Improved OMS version, Purchasing: Improved SMS version).

3.2 Alternatives evaluation

The feasibility study **(Appendix C)** leads to the following evaluation of the alternatives, represented in a table with the weighted sum of scores obtained by the different alternatives:

Feasibility type	**Weight**	*Alternative 1*	*Alternative 2*	*Alternative 3*	*Alternative 4*	*Alternative 5*
Technical	20%	50%	30%	75%	60%	55%
Operational	30%	40%	55%	65%	60%	85%
Schedule	15%	100%	30%	70%	85%	65%
Economic	35%	40%	70%	20%	65%	80%
Total Score	100%	**51%**	**52%**	**52%**	**66%**	**74%**

Table 1: Weighted sum of scores obtained by the evaluation of the alternatives.

The evaluation of the different alternatives provides interesting results. The first three alternatives get nearly the same (low) score, even if the components producing the overall result are balanced in very different ways: Alternative 1 has a very good ranking for the schedule (no changes are required) and average low percentages for the other components; Alternative 2 is economically good, but has technical and schedule problems (it is the development of a new solution); Alternative 3 has very bad economic perspectives, due to the firing of salesmen: this choice reduces personnel costs, but decreases also the incomes. Alternatives 4 and 5, both basing on the same system for the selling department, get higher scores. The alternative we suggest is alternative 5, which combines a modern web-based e-commerce interface and lightweight clients for the selling department with an improved version of the old software system. This system gained the best score for economical and operational feasibility, technically it is possible and scheduling is not a major issue for the management (with a weight of only 15%). Alternative 4 proposes the use of the old systems for the warehousing and purchasing departments. Although this choice lowers development costs and learning effort, the business processes of these two departments are not restructured and therefore it does not satisfy the stakeholders' main goal of having better cost- and time-efficiency: this is another reason to propose Alternative 5 to the customer.

4 The Solution

4.1 Chosen alternative

The alternative recommended for implementation is alternative 5: *Hybrid web-based ordering based on improved legacy systems* (Selling: *Hybrid web-based ordering based on the legacy system*.

From here on we will examine only the part of the project concerning the process where the selling department is involved, to reduce overall complexity. We will provide an analysis for the entire business process, following all the necessary steps.

Chosen solution for the selling activity:

Hybrid web-based ordering based on the legacy system
Orders can be managed through a website, which can be accessed by customers, secretaries, and salesmen, through a standard browser or light PDA-based interface client. The latter solution is specific for salesmen, and gives them all up-to-date information on products, prices, available quantities, and waiting times. Invoices and payments are handled by this system through the same interface, with different views depending on a role-based access. The old OMS software is kept as base system, and secretaries can still use it to perform management activities.

Appendix D contains the detailed requirements elicitation for the selling system, to provide further information on the chosen solution. Goal models show the goals of each of the stakeholders with respect to the software system. The leaf-goals are the starting point to define the use cases of the system-to-be. The modelled users are the sales agent, the secretary and the retailer customers. The class diagram with OCL constraints depicts the general architecture of the system, the components of the system and the relations between them. Sequence diagrams model the normal and exceptional temporal execution of the principal use cases. Finally, an activity diagram represents the typical business process with the interactions of the users, after the introduction of the new system.

5 Composite System Requirements

1. **Personal Digital Assistant:** each sales agent should be provided with a PDA device, which should support the selling process. All important data should be available in the PDA and it has to be at most 1 day old, because the sales agent will rely on the information present in the PDA. To reach this objective, each PDA should be able to synchronize its data using a wireless IEEE 802.11b connection, which enables the contact with the central server.
2. **Daily Synchronization of the PDA database:** the sales agents have to synchronize their PDA devices with the system every morning before any business activity, to get the updated list of customers, the updated prices and quantities of the products, and the special offers of the company.
3. **Daily Synchronization of the central orders database:** at the end of the working day, each sales agent should synchronize the central database with the orders he placed during the day. This activity is suggested to be performed more than once a day, as soon as a wireless connection is available.
4. **Sales agent working-day planning procedure:** the schedule of the working day has to be made basing on the list of suggested customers retrieved in the PDA, consistently with the meetings already planned directly after customer requests.

5. **Propose special offers:** in order to reach one of the major objectives of the new system, namely increased business process efficiency, the sales agent has to actively propose the special offers of the company, to introduce new products to the customers and possibly increase the sales volume.
6. **Web Server:** the web server should interface directly the database, and needs to be available 24/7. Reliability and security are fundamental properties to be provided. The performance of the server should allow at least 50 concurrent customer accesses.
7. **Send invoice via ordinary mail:** the secretary should print the invoices and send them to the customers via ordinary mail for every correctly delivered order.
8. **Check payment status:** the secretary is in charge of verifying the status of the invoices payment. She should also insert payment receipts into the system, changing the order status. If payments are not executed within the due date, she should send the customer a reminder for the payment. If the invoice is not paid within two reminders, she should inform the sales department manager.

6 Software Requirements Specification

6.1 External Interface Requirements

6.1.1 User Interfaces

The user interfaces required for the system to-be consist of the web interface allowing customers (and also sales agents) to place online orders, and of the PDA interface for sales agents.
The web interface should allow the users to view the available products with corresponding updated quantity and expected delivery time, and it has to explicitly highlight the special offers, in order to suggest them to the customers. The web interface should be easy to use and allow customers an easy order process, giving visibility to the shopping cart.
The interface of the PDA should allow the sales agent to see the customer base, highlighting the suggested customers to contact and the reasons why he should contact them. The list of products should also be available in another screen, with a tree-like output organizing products by category, and it should show the special offers.

6.1.2 Hardware Interfaces

The new system should be in contact with the machine where the database is stored. The connection will be provided via Ethernet cables, using a TCP/IP connection.
The PDA should contact the central database, and the connection will be performed using Wireless IEEE 802.11b standard.

6.1.3 Software Interfaces

The PDA software should be compatible with the Windows Mobile 5 operating system, the environment where it will be designed to work. The web interface has to work on Apache Web Server and uses PHP 5.

6.1.4 Communication Interfaces

The database should be accessible to both the web interface and the PDA (in remote). All communications will use TCP/IP protocol stack. The PDA will contact the database using port 6895, and hence the company firewall will be set up in order to allow that kind of connection. The web interface is inside the company and will use port 6896 to connect to the database.

6.2 *Functional requirements* *(grouped by sub-system)*

6.2.1 WEB STORE

6.2.1.1 Login mechanism

To enter the business-to-business web store from the company web site, the user has to log in with password and username, which can be requested only directly to the sales agent. The username should be saved in a cookie to allow a faster login procedure for repeated logins. The password has to be requested at every login for security reasons.

6.2.1.2 Role-based access

The web-store interface functionalities depend on the role associated to each user name. Available roles are Customer, Sales Agent and Administrator. The customer interface allows consulting and searching the catalogue and placing orders for himself. The sales agent interface allows to consult the whole catalogue, the special offers and to place orders for a customer in the customer base, providing a customer search interface.

6.2.1.3 Consult catalogue and place web-based orders

The web catalogue should have two views, one customized view containing the products usually bought by the respective customer, some additional products of the same genre, new products and possibly other products that could be suggested to the customer, basing on a global statistical analysis of the orders. The second view should contain all orderable products. In both views, products should be displayed divided into categories, and internal to the categories divided by manufacturer. A tree view should help for navigation and overview. Products are displayed with their main characteristics and their price; detailed information should be obtained by clicking on the item. On each the row displaying a product, there should be an editable field where to put the quantity of items to order (different to B2C solutions!). On top and bottom of the catalogue view there has to be a button to insert all products with quantity >0 into the orders list (shopping cart). Next to this button there is a button to view the shopping cart. The shopping cart page has to display all ordered items, the total amount spent with and without taxes, possible discount and further expenses (packaging, shipping). Prior to placing the order there must be a confirmation screen which displays also the number of different items and the total amount to pay. In the moment the order is confirmed, it has to be directly inserted to the system's order database. If the insertion succeeds, a printable page with a reminder of the ordered products (similar to the final invoice) has to be shown. Otherwise, a warning has to be shown. The shopping car content has not to be lost at a database entry error, but has to be available for changes and for retrying an insertion. If the insertion is not possible, there has to be shown an option for printing the actual shopping cart, to give it to a sales agent.

6.2.2 PDA INTERFACE

6.2.2.1 Authentication

Authentication is based on a traditional login procedure with password and user name (precompiled after the first login) and required when starting the client program. This login should be automatically aligned to the company's system login. Moreover, to confirm an order or a change a user profile, a 3-digit key code is required to prevent misuse.

6.2.2.2 Manage orders

The catalogue has to be shown in a structured way basing on folders which group the products. The interface should be clear and allow fast ordering. Furthermore, selection of the customer and a list of all actual orders of that customer should be provided.
The orders managing component should also provide daily statistics on the number of orders and the total ordered amount.

6.2.2.3 Manage customer base

The PDA customer base is daily synchronized with the system customer base. It should be possible to view all customer data, to modify it and to add new customers with all their data.
Moreover, the PDA should hold the agenda of the customers to visit, suggested by the system. This list should be modifiable, deleting customers, moving them to another time slot, and adding customers from the customer base or new ones.

6.2.2.4 Synchronization

The synchronisation operation should include the following functionalities:

- update the PDA product list (with actual prices) downloading all changes from the system
- copy the committed orders from the PDA to the system
- copy the new orders of the customers assigned to the authenticated sales agent, currently in process, to the PDA and update all the order status information.
- synchronize the customer base with the system customer base, adding the new customers to both bases and synchronize changes between the systems. If an entry was changed in both databases, the information in the system db should be preferred. A warning should be shown on the PDA, showing the fields that are in conflict.

6.2.2.5 Offers management

Offers have to be displayed in a special folder of the product catalogue, accessible with one click from every view in the catalogue. They are synchronised together with the product list and get the same treatment as standard products in the ordering process. The products in special offer should also be visible in the view of its product category folder.

6.2.3 OLD SYSTEM CLIENT APPLICATION

6.2.3.1 Authentication

Authentication is made once at the start of the client application, with system login name and password.

6.2.3.2 Orders management

Orders should be viewable per customer, per status and per order day. They should be modifiable prior to confirmation.

6.2.3.3 Invoice printing

All orders successfully assembled and ready to be delivered (information comes from the warehouse management system, together with some possible minor changes in the order, e.g. for unavailability of a product) have to be shown in a special Invoices-printing list. It is possible for the secretary to print these invoices with one click. The order status is automatically changed with this operation.

6.2.3.4 Payment management

Open (i.e. not paid) invoices should be viewed in a special screen, ordered by customer. For each order, payment can be confirmed or there can be printed an official payment request, formatted to be sent by ordinary mail.

6.2.3.5 Products insertion

A screen shows the product database, with its content, organized in folders. It is possible to add a new product with all its information in any folder, to add product group folders, and to modify product information. Products should be searchable by product id, by name and by group type.

6.2.3.6 Customer insertion

Customer information can be viewed and modified. New customers can be inserted. Customers should be searchable by their id, their name and by city.

6.2.3.7 Price management

For price management there should be a screen independent from the product database screen. It contains all the necessary fields for price management and change, such as basis price, taxes, discount with time-constraints, profit in %, end-user price. Changes to one of these fields should automatically reflect to the other fields. If the end-user price is modified more than +/-10%, confirmation should be requested, showing the old and the new price.

6.3 Non-functional Requirements

6.3.1 Availability

The software system should be 24/7 available. In particular, this requirement holds for the database system and the web interface. With the possibility of placing orders via web, the customers do not want restrictions on the times where they can order. Moreover, the sales agents should be able to synchronize their PDAs with the system any moment, especially if they don't have continuous internet access.

6.3.2 Security

An adequate level of security has to be guaranteed in every component of the system. In particular, the passwords for the users accessing the web interface should be encrypted via hashing, in order to avoid the determination of the original password. Moreover, SSL should be used to protect the flow of data necessary for authenticate the users. All the systems internal to the company is protected by a robust firewall, which stands between the internal systems and the web. The salesmen PDAs have to be protected by requesting a proof of the authenticity of the user: password protected access can be sufficient, but a higher level of security can be achieved using fingerprinting recognition. The old systems have to be enhanced requiring an username/password based access to employees.

6.3.3 Maintainability

One of the worse issues in the old information system consisted in a low degree of maintainability, which required an accurate feasibility study and the proposal of solutions to face this problem. The new systems should assure maintainability, in order to avoid similar issues to arise in the future. The system will be developed using object oriented techniques for the PDA part, and with well commented and structured PHP code for the web interface.

6.3.4 Scalability

Since the company is planning to increase its customer base and the number of orders, scalability should be assured by the system. In the chosen solution, the best efforts that can be made consist of enabling multiple customers to access the web site at the same time. The database will not be modified, but at least the introduction of a web server can manage the html/php overload.

6.3.5 Reliability

Reliability of the system is guaranteed by the DBMS. The existing DBMS is transaction based, and implements backup and distribution mechanisms to prevent off-line situations. The new web server should avoid stall situations for the online customers.

6.3.6 Compatibility

The system is guaranteed to be compatible with the old production systems, but it should also avoid compatibility issues with other systems. In order to satisfy this property, some additional supported file types have to be added to the current ones. For instance, xml files are not yet supported but they are needed to be.

6.3.7 Usability

The users of the system are not computer experts, and hence particular attention should be paid to usability. In particular, the web interface should be clear, concise, and easy to use The same properties should be guaranteed by the PDA interface.

7 Appendix A: Organizational Structure

7.1.1 Organisational structure

The identification of stakeholders, namely of the persons which are interested (directly or indirectly) in the logistics business process, is the first step to be performed in the feasibility study. All the following activities should consider the people involved in the context, which can have benefits or drawbacks from the introduction of new software systems.
The stakeholders identification process is subdivided into the representation of the internal organizational structure of the company and the identification of external stakeholders, internal stakeholders, and users.

7.1.2 Internal organizational structure:

Figure 1 shows the organizational structure of the examined company. We assume that each operational entity (secretary, worker, agent) can represent more than one physical entity. The structure is quite typical, with a president leading the company business and with representation functions, a personal secretary which organizes the schedule of the president, and a subdivision in three departments.

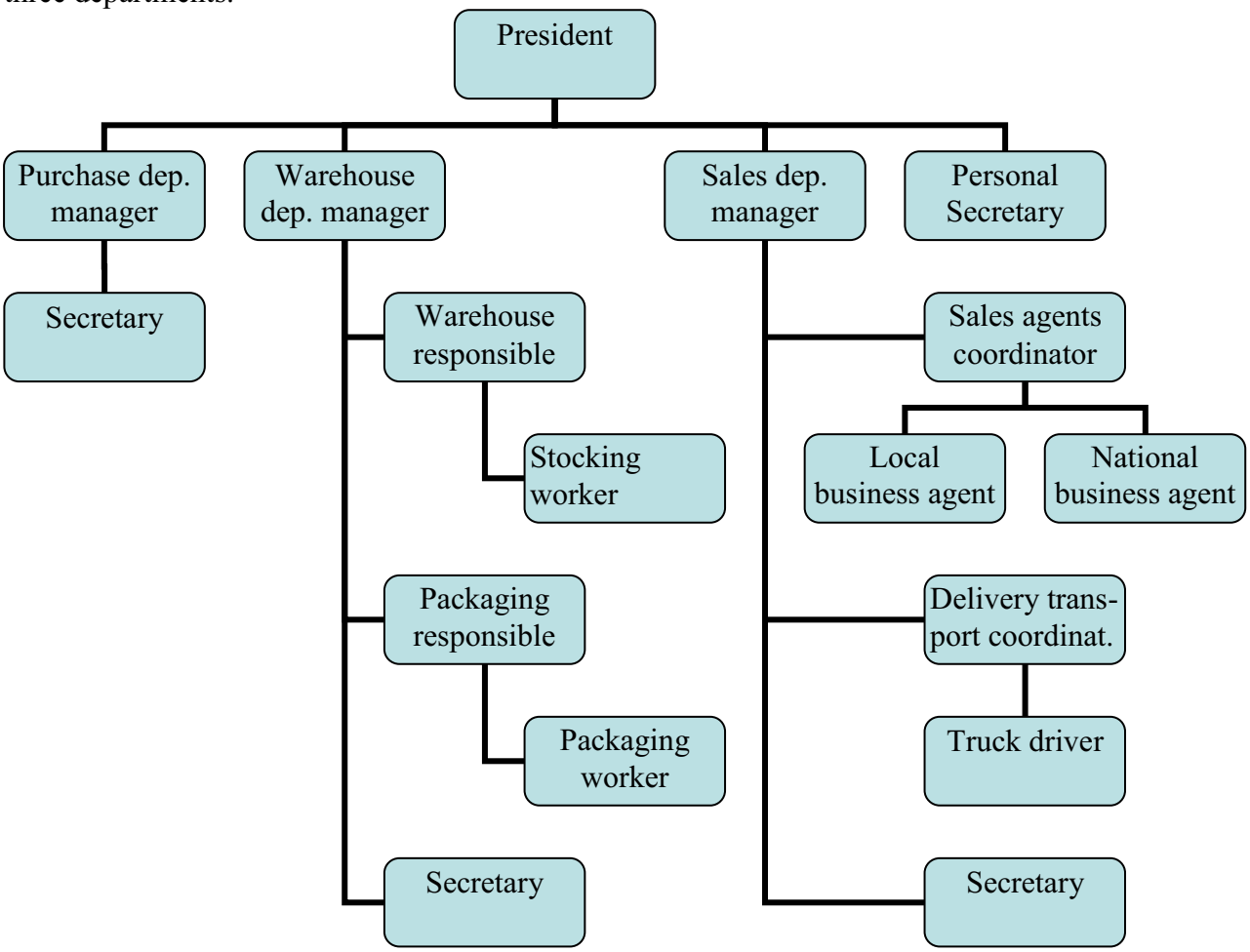


Figure 1: Internal Organizational Structure

The purchase department is headed by the purchase department manager, which delegates to his secretary the operational duties. The warehouse department has also a manager, which coordinates the warehouse responsible, the packaging responsible, and his secretary. The warehouse responsible in turn coordinates the stocking workers, and the packaging responsible manages the packaging

workers. The sales department is headed by a manager, whose direct subordinates are the sales agents coordinator (which manages both local business agents and national business agents, working in different geographical contexts), the delivery transport coordinator (which organizes the work of the truck drivers), and the secretary.

7.2 External stakeholders:

Suppliers (Hard- and Software producers): they should be able to satisfy requests coming from the purchase department as soon as possible, and respecting the contracted delivery times.

System designers (Fabiano & Mirko): they have to provide a solution which can provide an efficient logistics business process.

Web-mastering company: the current web-site of the company might be improved and/or updated, in order to shorten the ordering time.

7.3 Internal stakeholders:

President: he is the head of the company, and responds to the customers about the efficiency and quality of the company.

Purchase dep. Manager: he is in charge of providing fast and convenient provision of hardware from the providers.

Warehouse dep. Manager: he should ensure a proper filling of the warehouse, and at the same time reduce stocking times. Furthermore, he has to organize an efficient repackaging of the stocks.

Sales dep. Manager: he is in charge of providing goods to the customers in short time.

7.4 Users

All organizational structure members: they use the software systems, which should support them in their operational and decisional activities.

Customers (Retailers): in the case we allow the customer to access in some terms our systems, they become users.

7.5 Other information sources

Legacy System providers: they could be asked to change the systems they produced for the company, and are responsible of maintenance; the legacy systems influence the performance of the supply chain in the company.

Suppliers: they could be asked to provide interfaces to their systems, but they are not users of the system, because their typical size is much bigger than the company's one, and hence they would not accept to use it.

8 Appendix B: Current Situation and Problems

8.1 Current situation, typical work scenario:

- Customers enter into contact with the company through the sales agents (local or national, depending on where the customer is located), and rely on the agents to make orders. Alternatively they can send their orders by fax or by mail to the sales secretary, without adopting a predefined layout to fill the order form.

- The orders are stored in the orders management system (OMS) directly by the business agent or by the sales department secretary (when orders are received via mail or fax).

- OMS automatically checks the availability of the products in the warehouse. If all ordered products are available, the order is automatically forwarded to the stocking workers, which collect the requested products in the warehouse and prepare the order for delivery. If some of the products are not available, the warehouse responsible gets a warning message and has several options: she can directly place an order to the suppliers or ask the purchase department to supply a new stock of the requested product.

- Often stock packages are larger than the quantities requested by small customers, and in the case the warehouse responsible should request the packaging responsible to split the stocks into smaller units.

- The warehouse responsible should take care of a proper filling of the warehouse. She relies mainly on her experience to define minimum product stocks and the amount of products to purchase in advance. Selection criteria involve the satisfaction of customer needs in minimal time, and the reduction of risks given by old unsalable products in the warehouse.

- The purchase department is responsible to manage all incoming products. They observe the market, select suppliers, bargain prices, and pay invoices. The company has a simple Suppliers Management System (SMS), which keeps track of data about suppliers and incoming goods invoices.

- The received stocks are directly delivered to warehouse workers, which are responsible for recording the quantities using the OMS and checking errors controlling invoices and supplied goods.

- A further service of the company is to offer preconfigured hardware packages to retailers at special offer prices. This is possible only with a good collaboration among purchase department (that has to identify supplier offers), the warehouse (that has to be optimally used) and the sales department (that has to guarantee that they are able to sell a high number of packages and actually needs too high coordination costs).

- As the warehouse is ready to deliver a complete order, the sales department organizes delivery to the customer, shipping notes and invoices. It also has to ensure correct payments and to block new orders from customers that do not pay in time.

- OMS is a basic warehouse management software: its functionalities include accounting of customers orders, and warehousing with goods inbound and outbound. It does not provide

hints for the time and quantity of supply orders, statistics, and assisted order placement which could improve logistics efficiency. SMS manages the purchasing activities.

8.1.1 What the management wants:

The management asked to perform a feasibility study to verify the possibility of increasing efficiency throughout all business activities, including the reduction of both delivery and stocking times. There is hence a very high level objective, which is "improved efficiency".

8.2 Identification of problems (using PIECES):

Performance: in our context it concerns the effectiveness of the company's business processes. We need to reduce both delivery and stocking times, because in the actual situation they are too high.

Information: customer orders are not accurately captured, because they rely on manual procedures handled by secretaries and salesmen. Inbound invoices and product quantities are manually managed by the purchase department and the warehouse using two different software systems (SMS and OMS, respectively). Outputs lack of relevant and well organized information for the purchase department, which places orders on the basis of the personnel experience. In the current system there is no support to the creation and management of special offer packages. Stored data is not well organized, because the current software holds only data about the actual warehouse filling and does not track warehouse history and does not compute helpful statistics.

Economic: the current costs are too high, because the stocking and delivery business processes are not efficient. Moreover, warehousing costs could be unknown due to the lack of predictions about stocking dynamics. By lowering delivery times and costs new customers will be attracted and sales could be increased.

Control and Security: product quantities stored in SMS can be inconsistent with quantities recorded in OMS, because they are two independent systems and delivered quantities are compared to invoices only in the warehouse department. The decision-making workflow is not regulated, and the way of working relies on people experience.

Efficiency: time is wasted in interaction between customers and sales department, because of the lack of an automated order system. The same issue holds between the purchase department and suppliers. If warehouse stocks level is too low, the warehouse responsible gets only an alert, but the interaction with the purchase department is not computer supported, leading to a communication effort which increases the waiting time.

Service: OMS and SMS do not provide intercommunication functions, and they lack of compatibility and coordination with customers and suppliers systems.

9 Appendix C: Feasibility study

We examine each of the 5 identified alternatives (which comprise selling, warehousing, and purchasing) performing technical, economic, schedule, and operational feasibility analysis. To each type of feasibility study a score between 0 and 100 is given.

At the end of the feasibility study analysis we will propose a recap table, where all the types of feasibility are weighted and allow us to give an overall opinion on which are the alternatives we suggest and which ones we discourage.

Feasibility Study	Wt	Alternative 1	Alternative 2	Alternative 3	Alternative 4	Alternative 5
Technical Feasibility **Technology:** is the technology needed in the proposed solution mature? Is it a state-of-the-art technology? **Expertise:** do we possess the necessary expertise to develop such a system? **Compatibility:** is the proposed solution compatible with other systems in the company?	20%	The technology is old but mature, and could have problems in scalability. Expertise regarding system use is high, while changes to the system require legacy-systems experts. There are known compatibility issues with new technologies, because the system works on a old architecture. OMS and SMS are not compatible; all information has to be transferred without computer support.	Brand new technology allows building such a system, even if many interfaces are not yet standardized. The development company is expert in proposing web-based systems, but needs to hire an expert in lightweight clients. We also need expert requirement engineers that analyzes the functionalities of OMS and provides the requirements for the new system. Furthermore, an expert in business logistics is required to analyze the process and propose indexes and techniques to improve it. The system has to integrate all the functionalities of OMS and SMS, and to interface with all devices used in the company (label printers, laser scanners).	The required technologies for the selling system (web-based systems, AJAX) are well established, widely used, and state-of-the-art. The remaining systems rely on old technologies, which could have problems with scalability. We possess the necessary expertise to develop the web-based system, but require a experts to integrate it with OMS. An expert for legacy systems is needed. Small compatibility problems are given by the integration of the web interface with OMS. Other problems could arise interfacing SMS and OMS.	The technology for the selling system is state-of-the-art, but relies on the existing system (OMS). Warehousing and purchasing continue using their old systems. The development company is expert in proposing web-based systems, but needs to hire an expert in lightweight clients. We also need an expert to integrate the system with OMS. Support and maintenance to the old systems require legacy-systems experts. Small compatibility problems are given by the integration with OMS. Moreover, the old systems should be able to manage the lightweight clients. The incompatibility between OMS e SMS still persists.	The required technologies for the selling system (web-based systems, AJAX) are well established. Lightweight clients use non-standardized interfaces and brand-new technology. The remaining systems rely on old technologies, which could have problems with scalability. We possess the necessary expertise to develop the web-based system, but require a experts to integrate it with OMS. An expert for legacy systems and one for lightweight clients are needed. Small compatibility problems are given by the integration of the web interface with OMS. Other problems could arise interfacing SMS and OMS.
Score:		**50%**	**30%**	**75%**	**60%**	**55%**

Feasibility Study	Wt	Alternative 1	Alternative 2	Alternative 3	Alternative 4	Alternative 5
Operational Feasibility **Acceptance:** will managers accept (or reject) the proposed solution? **Functionality:** to which degree will the system satisfy the needs of the company? Will the system work properly? **Change:** how will the working environment change? Will end users use the system?	30%	The management of the company is sure a new solution is needed, and would probably reject to keep the system as-is if costs of change are not too high. Many processes are not supported by the system, and need to be manually handled, for example the decision of when and how much to order. There is no change in the working environment, but as well no improvements will be made.	The managers of this company like new technologies, but are afraid that something goes wrong if big changes in the software system and in the working process are made. A brand new system should satisfy all the needs of the company, and improve the efficiency of business processes. The success of the system depends on a complete and precise requirements analysis which leads to a correct substitution of OMS and SMS. Wide changes are made in the way of working: salesmen should adopt lightweight clients, customers are encouraged to place orders in Internet, and secretaries have to use the new system. Warehouse workers and the purchase system are assisted in their activities, but have to accept the new system. There may be people rejecting the changes, and learning effort is required for all employees.	The managers think the system can decrease personnel costs, but can also reduce the market, because selling experts are no more required. They would be satisfied with the old system if it were well integrated and performance remains acceptable. The system will work properly if all customers use it. In the case salesmen and email orders are still necessary there are no improvements in the selling process. The integration of OMS and SMS aids warehousing and purchasing activities. Changes on customer side have to be accepted: they have to be instructed to use the only way of ordering (web-based). Moreover, the purchasing department should accept the new way of working.	The managers of this company like new technologies, but they can have doubts on the efficiency of the old system. Moreover, they will probably not accept the lack of integration between OMS and SMS. The new ways of ordering should better satisfy the needs of customers and the efficiency of the sales department. The selling system will work properly, if the new technologies are accepted. Keeping the old systems implies manual handling of many activities. Customers have an additional way of placing orders, salesman should accept the use of lightweight clients, while secretaries, workers, and the purchase employees will still work on the old system.	The managers of this company like new technologies, but they can have doubts on the efficiency of the old system. On the other side, they like the idea of improving and integrating the old systems. This solution implies correct working of the sales activities (like in Alternative 4), and improves the functionalities of the old systems (like in Alternative 3). Customers have an additional way of placing orders, salesman should accept the use of lightweight clients, while the other employees should adapt to the use of the additional functions of SMS and OMS.
Score:		**40%**	**55%**	**65%**	**60%**	**85%**

Feasibility Study	Wt	Alternative 1	Alternative 2	Alternative 3	Alternative 4	Alternative 5
Schedule Feasibility **Time/Schedule:** how long does it take to get the technical expertise? How long will the project last? Are there overrun risks?	15%	The system is ready, there are no changes.	Requirements analysis and programming require experts and long time. There are strong overrun risks, and time-consuming learning effort is required for all users of the system.	Small changes required for the selling system, where low time effort in programming the web interface. The old system needs time to be improved, but this needs less time than developing a new system.	For the selling system like Alternative 3, but more programming effort for creating the lightweight client software and its server system. No effort is required for the systems of the other departments.	Takes the time costs from Alternative 4 for the selling system, and from Alternative 3 for the improvement of OMS and SMS.
Score:		**100%**	**30%**	**70%**	**85%**	**65%**

Feasibility Study	Wt	Alternative 1	Alternative 2	Alternative 3	Alternative 4	Alternative 5
Economical Feasibility **Development costs:** this cost includes the programmers' salary, additional software licenses, hardware. **Operational costs:** this cost is composed by maintenance and personnel costs for operation, support, and training. **Benefits:** they include increased sales, higher efficiency, and savings due to fewer personnel needed. Cost-Benefit analysis is made through a detailed payback analysis.	35%	No development costs, maintenance costs are quite high but constant (15.000€/year). Personnel costs are high due to inefficient working process: experts provided an estimate that operational costs can be lowered by 50.000€/year with a perfectly efficient business process. There are no benefits, because this is the current situation.	Develop a new system from scratch: Requirements engineers 4 x 30 days x 300€, Designers 4 x 30 days x 250€, Programmers 10 x 90 days x 150€. Deployment: 5 x 10 days x 180€. New hardware: 1 Server @ 12.000€, 3 PDA clients @ 500€. Maintenance: 5.000€/year, operational costs estimated 5.000€/year above a perfectly efficient business process. Benefits supposed to increase (see details in the tables).	Develop an e-commerce selling interface: RE & Design: 2 x 14 days x 250€, programming & graphics: 4 x 8 days x 150€. Deployment: 2 x 4 days x 180€. New web server: 4000€. Improve OMS and SMS: Requirements engineers 2 x 30 days x 300€, Designers 4 x 14 days x 250€, Programmers 4 x 60 days x 150€. Deployment: 2 x 10 days x 180€. Maintenance: 10.000 for the legacy system + 1.000€ for the e-commerce interface Operational costs: including firing of the 3 salesman: 50.000€ below the estimated perfectly efficient business process. Benefits are supposed to decrease.	Develop a system for lightweight clients and the e-commerce interface: Requirements engineers 2 x 20 days x 300€, Designers 3 x 15 days x 250€, programming & graphics: 5 x 30 days x 150€. Deployment: 2 x 3 days x 180€. New web and lightweight client server: 5.500€. Maintenance: 15.000 for the legacy system + 3.000€ for the e-commerce interface. Operational costs will be 35.000€/year above a perfectly efficient business process. Benefits supposed to increase.	Develop a system for lightweight clients and the e-commerce interface: Requirements engineers 2 x 20 days x 300€, Designers 3 x 15 days x 250€, programming & graphics: 5 x 30 days x 150€. Deployment: 2 x 3 days x 180€. New web and lightweight client server: 5.500€. Improve OMS and SMS: Requirements engineers 2 x 30 days x 300€, Designers 4 x 15 days x 250€, Programmers 4 x 60 days x 150€. Deployment: 2 x 10 days x 180€. Maintenance: 10.000 for the legacy system + 3.000€ for the e-commerce interface. Operational costs will be 15.000€/year above a perfectly efficient business process. Benefits supposed to increase.
Score:		**40%**	**70%**	**20%**	**65%**	**80%**

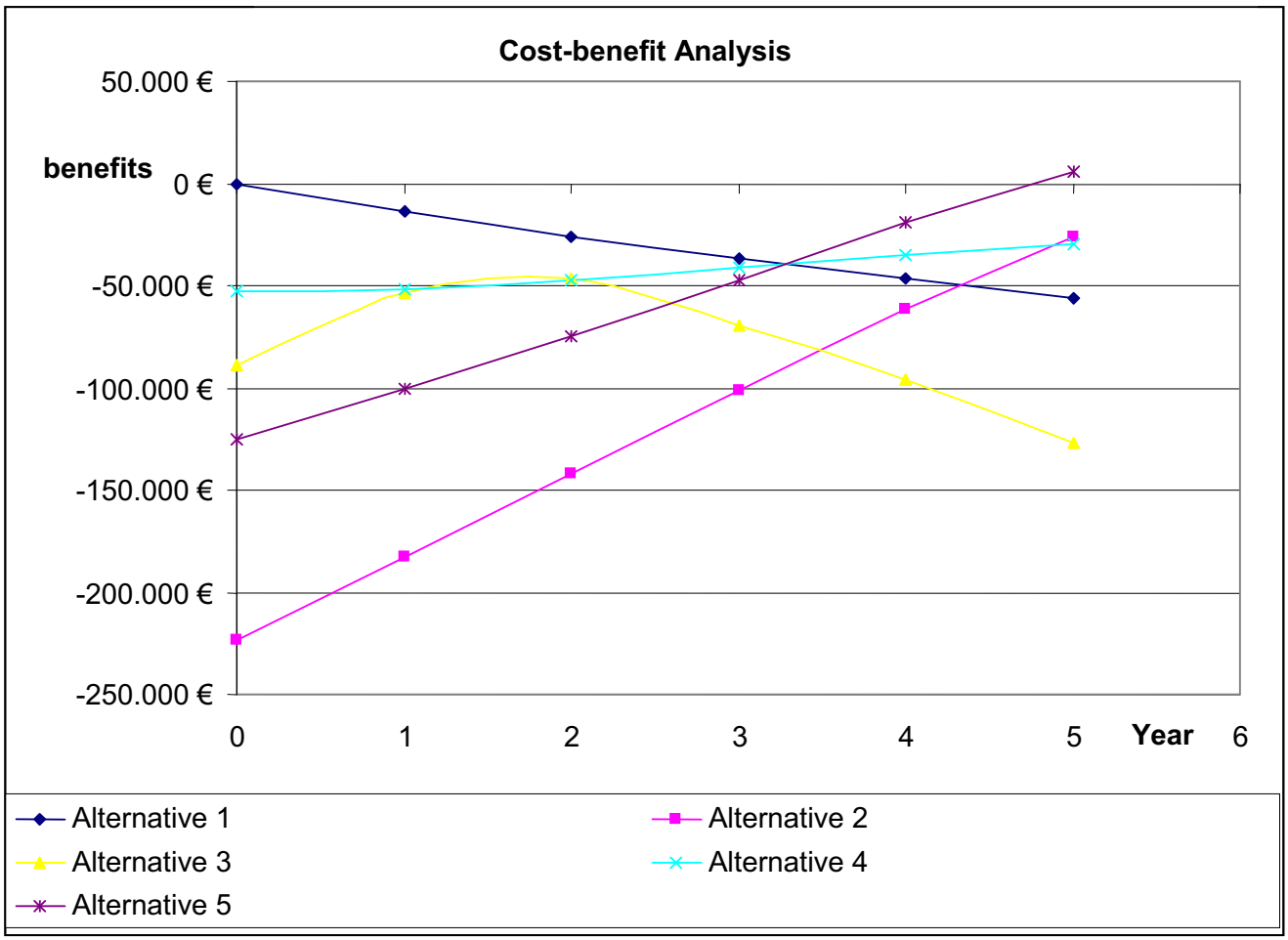


Figure 2: Cost-benefit analysis diagram

The cost-benefit analysis diagram in Figure 2, shows that the most cost-efficient solution after 5 years is Alternative 5, which reaches the break-even point within five years and is more effective than keeping the old system already after slightly more than three years. Another good choice on the long term would be the development of a new system (Alternative 2), although the high initial cost needs around 6 years to be amortized and 9 years to be better than Alternative 5 with the estimated benefits and maintenance costs. We consider this period too long compared to the typical life cycle of this type of software systems.

9.1 Detailed cost-benefit analysis

Detailed cost-benefit analysis for all five alternatives. Costs and benefits are evaluated through a period of 6 years, basing on a reduction of the present value by 9.7%. Development costs (salaries and hardware) are detailed in additional tables.

9.1.1.1 System as-is

Year	0	1	2	3	4	5
Development Cost	0 €					
Maintenance cost		15.000 €	15.000 €	15.000 €	15.000 €	15.000 €
Operational cost	0 €	0 €	0 €	0 €	0 €	0 €
Present Value	1,00 €	0,90 €	0,82 €	0,74 €	0,66 €	0,60 €
time adj cost	0 €	13.545 €	12.231 €	11.045 €	9.973 €	9.006 €
Total Present Value Cost	0 €	13.545 €	25.776 €	36.821 €	46.794 €	55.800 €

Benefits						
T-adj Benefits	0 €	0 €	0 €	0 €	0 €	0 €
Cumulative Benefits	0 €	0 €	0 €	0 €	0 €	0 €
Net Costs+Benefits	0 €	-13.545 €	-25.776 €	-36.821 €	-46.794 €	-55.800 €

New web-based system

Year	0	1	2	3	4	5
Development Cost	223.500 €					
Maintenance cost		5.000 €	5.000 €	5.000 €	5.000 €	5.000 €
Operational cost	0 €	-45.000 €	-45.000 €	-45.000 €	-45.000 €	-45.000 €
Present Value	1,00 €	0,90 €	0,82 €	0,74 €	0,66 €	0,60 €
time adj cost	223.500 €	-36.120 €	-32.616 €	-29.453 €	-26.596 €	-24.016 €
Total Present Value Cost	223.500 €	187.380 €	154.764 €	125.311 €	98.715 €	74.700 €
Benefits		5.000 €	10.000 €	15.000 €	20.000 €	20.000 €
T-adj Benefits	0 €	4.515 €	8.154 €	11.045 €	13.298 €	12.008 €
Cumulative Benefits	0 €	4.515 €	12.669 €	23.714 €	37.012 €	49.020 €
Net Costs+Benefits	-223.500 €	-182.865 €	-142.095 €	-101.597 €	-61.704 €	-25.680 €

Development cost calculation

	persons	**days**	**daily cost**	**sum**
R.E.	4	30	300	36000
Design	4	30	250	30000
Programming	10	90	150	135000
Deployment	5	10	180	9000
Hardware server				12000
Hardware clients				1500
				223500

9.1.1.2 Pure web-based + improved OMS+SMS

Year	0	1	2	3	4	5
Development Cost	88.840 €					
Maintenance cost		11.000 €	11.000 €	11.000 €	11.000 €	11.000 €
Operational cost	0 €	-100.000 €	-100.000 €	-100.000 €	-100.000 €	-100.000 €
Present Value	1,00 €	0,90 €	0,82 €	0,74 €	0,66 €	0,60 €
time adj cost	88.840 €	-80.367 €	-72.571 €	-65.532 €	-59.175 €	-53.435 €
Total Present Value Cost	88.840 €	8.473 €	-64.098 €	-129.630 €	-188.806 €	-242.241 €
Benefits		-50.000 €	-80.000 €	-120.000 €	-130.000 €	-140.000 €
T-adj Benefits	0 €	-45.150 €	-65.233 €	-88.358 €	-86.436 €	-84.056 €
Cumulative Benefits	0 €	-45.150 €	-110.383 €	-198.740 €	-285.176 €	-369.232 €
Net Costs+Benefits	-88.840 €	-53.623 €	-46.284 €	-69.110 €	-96.371 €	-126.991 €

Development cost calculation

	persons	days	daily cost	sum
R.E.	2	30	300	18000
Design	6	14	250	21000
Programming	4	68	150	40800
Deployment	2	14	180	5040
Hardware server				4000
Hardware clients				0
				88840

9.1.1.3 Hybrid selling system + system as it is

Year	0	1	2	3	4	5
Development Cost	52.330 €					
Maintenance cost		18.000 €	18.000 €	18.000 €	18.000 €	18.000 €
Operational cost	0 €	-15.000 €	-15.000 €	-15.000 €	-15.000 €	-15.000 €
Present Value	1,00 €	0,90 €	0,82 €	0,74 €	0,66 €	0,60 €
time adj cost	52.330 €	2.709 €	2.446 €	2.209 €	1.995 €	1.801 €
Total Present Value Cost	52.330 €	55.039 €	57.485 €	59.694 €	61.689 €	63.490 €
Benefits		4.000 €	8.000 €	12.000 €	12.000 €	12.000 €
T-adj Benefits	0 €	3.612 €	6.523 €	8.836 €	7.979 €	7.205 €
Cumulative Benefits	0 €	3.612 €	10.135 €	18.971 €	26.950 €	34.155 €
Net Costs+Benefits	-52.330 €	-51.427 €	-47.350 €	-40.723 €	-34.739 €	-29.336 €

Development cost calculation	persons	days	daily cost	sum
R.E.	2	20	300	12000
Design	3	15	250	11250
Programming	5	30	150	22500
Deployment	2	3	180	1080

Hardware server	4000
Hardware clients	1500
	52330

9.1.1.4 Hybrid selling system + improved OMS+SMS

9.1.1.5 9.1.1.6 9.1.1.7 9.1.1.8

Year	0	1	2	3	4	5
Development Cost	124.930 €					
Maintenance cost		13.000 €	13.000 €	13.000 €	13.000 €	13.000 €
Operational cost	0 €	-35.000 €	-35.000 €	-35.000 €	-35.000 €	-35.000 €
Present Value	1,00 €	0,90 €	0,82 €	0,74 €	0,66 €	0,60 €
time adj cost	124.930 €	-19.866 €	-17.939 €	-16.199 €	-14.628 €	-13.209 €
Total Present Value Cost	124.930 €	105.064 €	87.125 €	70.926 €	56.298 €	43.090 €
Benefits		5.000 €	10.000 €	15.000 €	20.000 €	20.000 €
T-adj Benefits	0 €	4.515 €	8.154 €	11.045 €	13.298 €	12.008 €
Cumulative Benefits	0 €	4.515 €	12.669 €	23.714 €	37.012 €	49.020 €
Net Costs+Benefits	-124.930 €	-100.549 €	-74.456 €	-47.212 €	-19.287 €	5.930 €

Development cost calculation

	persons	days	daily cost	sum
R.E.	4	25	300	30000
Design	7	15	250	26250
Programming	6,5	60	150	58500
Deployment	2	13	180	4680
Hardware server				4000
Hardware clients				1500
				124930

9.1.1.9 Summary of net costs for the five alternatives:

Year	0	1	2	3	4	5
System as-is	0 €	-13.545 €	-25.776 €	-36.821 €	-46.794 €	-55.800 €
New web-based system	-223.500 €	-182.865 €	-142.095 €	-101.597 €	-61.704 €	-25.680 €
Pure web-based + improved OMS+SMS	-88.840 €	-53.623 €	-46.284 €	-69.110 €	-96.371 €	-126.991 €
Hybrid selling system + system as it is	-52.330 €	-51.427 €	-47.350 €	-40.723 €	-34.739 €	-29.336 €
Hybrid selling system + improved OMS+SMS	-124.930 €	-100.549 €	-74.456 €	-47.212 €	-19.287 €	5.930 €

10 Appendix D: Requirements Elicitation

10.1 Goal modelling

Goal modelling will be done with the purpose of analyzing the objectives of the stakeholders, and identify how various alternatives in the chosen solution can contribute to the non-functional requirements (the soft goals), and decide which alternative is the most appropriate. From the leaf-level goals we can identify the use cases of the stakeholders, and propose a whole picture which relates use cases and actors. The class diagram will identify the entities involved in the to-be system, depict with more detail the relationships among them, and add constraints using OCL. The activity diagram shows the dynamics of the whole business process (namely, the selling activity), while sequence diagrams are useful to understand the working sequence for the various use cases. The goal modelling activity starts by identifying the actors whose goal rationale will be explained using goal models. In our case, the most involved actors are the salesmen, the customers, and the secretaries. For each of them we provide a goal model together with a description / comment.

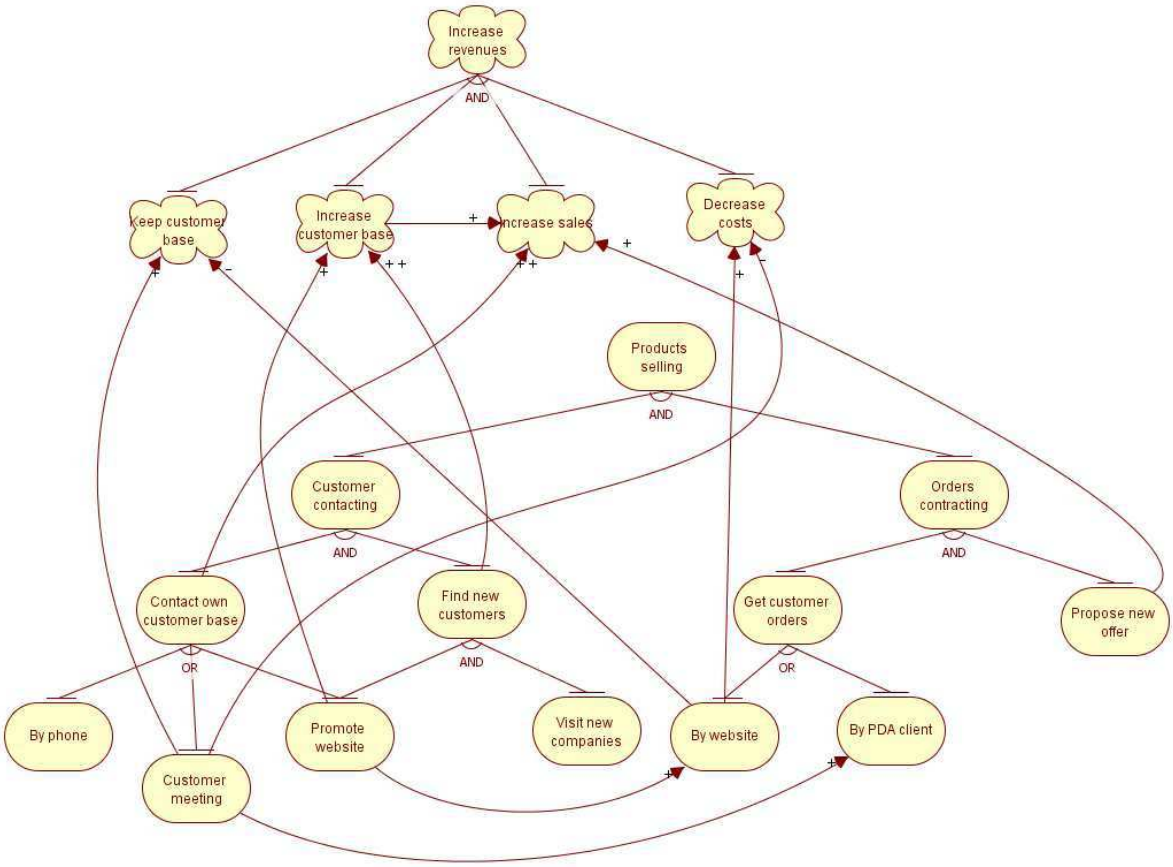


Figure 3: Goal model for the SalesAgent

Figure 3 depicts the goal model for the SalesAgent. The top-level goal is *Products selling*, since this is what is expected from this kind of workers from the management. This goal has two subgoals, namely contacting the customers and the contracting of orders, because they should both get into contact with possible customers and convince them to order with a contracting activity. The customers should be contacted both by using the agent's customer base and by finding new customers. In the case of *contact own customer base*, the customer can be contacted by phone, setting up a meeting, or promoting the web site. In order to find new customers, the sales agent should both promote the website and visit new companies, proposing the products sold by the represented organization. With respect to *orders contracting*, the agent should both get customer orders and propose new offers; when getting orders, they either can come by the website or be inserted into the agent's PDA client.
The promotion of the web site contributes positively to getting orders by website, because customers are encouraged to use the web without involving the agent, and when doing this way we get a positive contribution to the soft goal *Decrease costs*, which is part of the *Increase revenues* soft goal. Setting up a meeting with the customer contributes positively to getting orders by PDA client and helps to keep the customer base (soft goal part of *Increase revenues*), but contributes negatively to the soft goal *Decrease costs*, because of the higher cost in meeting the customer. Contacting the customer through own customer base makes the soft goal *Increase Sales*, whereas *Propose New Offer* contributes positively to it; that soft goal is also part of the top level soft goal *Increase Revenues*. The goal *Find new customers* makes the soft goal *Increase customer base*, which in turn contributes positively to *Increase sales*, and is also part of the top level soft goal. *Promote website* helps to increase the customer base, because new customers can be attracted by the short time needed to consult the catalogue and place orders via web, but contributes negatively to *Keep customer base*, because some customers would like to maintain the traditional sales agent-based ordering system.
Regarding the identified soft goals of the company, the best solution would be using the web-based selling system as much as possible, but some customers will not use it, and hence the role of sales agents is still important.
The goals that will be realized into the use case diagrams are *customer orders by PDA* and *by website* for the management of customers' orders, and *Propose new offer* to increase the sales. The leaf goals that are not realized are the ones which do not concern the software system.

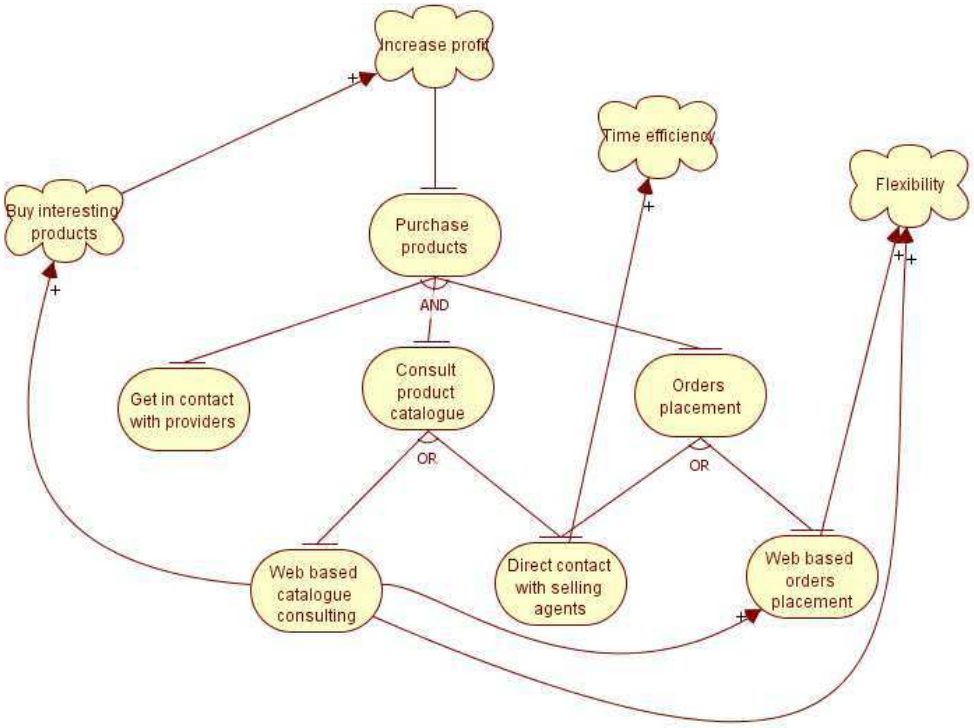


Figure 4: Goal model for actor Customer

The top-level goal of the Customer in the examined process, as can be seen in Figure 4, is *Purchase products*. This involves *Get in contact with providers*, *Consult product catalogue* and the *Orders placement* activity. The catalogue can be consulted either by using the web-based interface or through a direct contact with selling agents. The placement of orders can be done either with direct contact with selling agents, or placing the orders by web.
Consulting the catalogue via web helps the soft goal *Buy interesting products*, which helps to *Increase profit*: this one is a soft goal which is achieved by purchasing products and other goals (such as "selling products"), not depicted in the diagram because not regarding the purchasing activity. Consulting the web-based catalogue also helps the placement of web-based orders and the soft goal *Flexibility*. The same positive contribution to flexibility is given by web based orders placement, while the direct contact with selling agents helps *time efficiency*, because the selling process is usually shorter (the agent can give details about products, can suggest offers, and so on).
The goal model shows how customers need both the possibility of standard purchasing using agents and a web based system, since they both give positive contributions to different soft goals.
The realized leaf goals are web based catalogue consulting and web based orders placement, since the other ones do not involve the necessity of the software system.

Figure 5 depicts the goal model for the secretary. In this case, there is not only one top-level goal, because the secretary should perform two distinct activities, namely *manage customer orders* and *manage web site*.
In order to manage customer orders, she should *verify orders status*, *insert orders*, *print invoices*, and *check invoices payment*. The activity of checking payments contributes negatively to her efficiency, because it can require several phone calls to the customer and also contacts with lawyers if needed. The insertion or orders can be done either by using the web-based system, which positively contributes the soft goal *Easy procedures* and the goal *Verify orders status*, or by using OMS. The latter solution helps efficiency, because secretaries are expert in using it, and can insert orders very quickly.

The management of the web site also contributes negatively to *Efficiency*, because the operation is slow. However, this is a limited drawback of the solution and it was present also in the old system. That goal consists of inserting new products, updating the prices, and inserting customer contacts.
The realized goals are all the ones concerning the web site management, and the management of customer orders via OMS. The only choice which is not realized is web-based orders insertion, to avoid the secretary to learn the new interface.

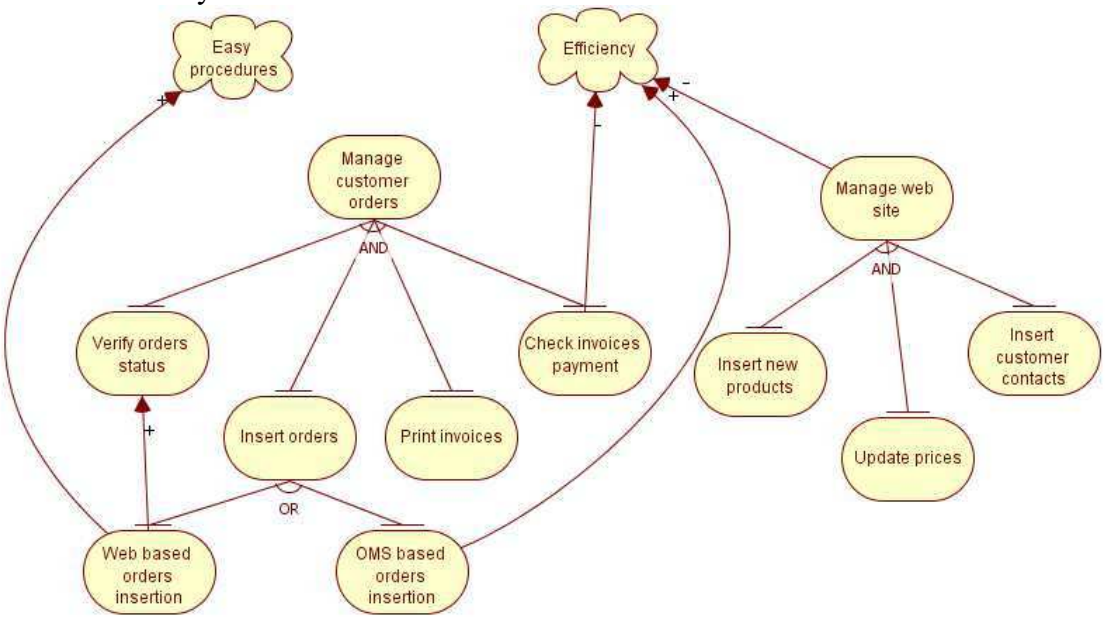


Figure 5: Goal model for actor Secretary

10.1.1 Use case diagram

Figure 6 shows the use case diagram for the system, where the leaf-level goals of the actors identified in the goal modelling are used to generate use cases, and the actors still remain the same ones.

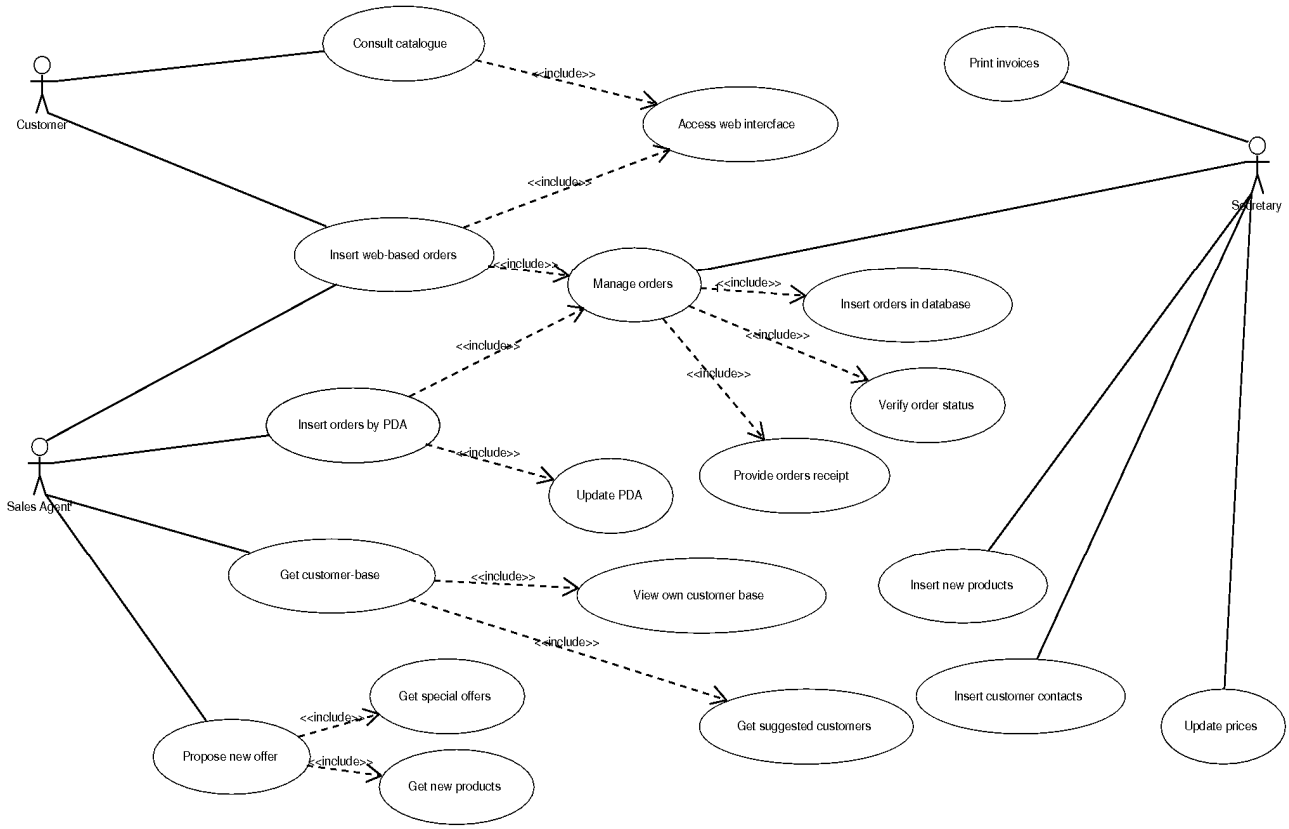


Figure 6: Use case diagram for the system

The *Customer* can use the system in two ways, namely consulting the catalogue and inserting web-based orders. Both use cases include *Access web interface*, and the insertion of orders includes the *Manage orders* use case. In turn, this includes *Insert orders in database*, *Verify order status*, and *Provide orders receipt*. In the case of the customer, these operations can be done only with respect to the orders placed by him (which do not need to be inserted by him, however).

The *Sales Agent* uses the system to insert orders either by *Insert web-based orders* or by *Insert orders by PDA*; the latter use case includes the management of orders and the update of the PDA, in order to have products and prices updated when the customer has to place an order. Another function required by the Sales Agent *Get customer base*, which includes viewing its own customer base, and get suggestions about possible suggested customers. The last use case of the Sales Agent concerns the proposal of new offers, which includes the possibility *Get special offers* from the company database and to *Get new products* which could be of interest for the customers.

The *Secretary* uses the system to print invoices for the customers, after the order has been inserted into the invoice and need to be formalised. Other use cases, coming from the bottom-level goals, are to *Insert new products* in the system, to *Insert customer contacts*, and to *Update prices*. These use cases should be however performed using the old OMS interface.

The Use cases scenario does not reach a very deep detail, because the next section proposes a more refined structure based on a class diagram enriched with OCL constraints, and then dynamic models are proposed to give a non-static view of the system.

10.2 Class diagrams with OCL constraints

From the goal analysis and the use cases diagrams we derived some basic entities, which have been modeled in Figure 7 in terms of classes, and the relationships among them are represented through typical relations among classes (association, generalization, aggregation). A superclass named *WebInterface* is the parent of *OnlineStoreInterface* and *ManagementInterface*, which represent, respectively, the web interface for customers to perform the orders placement activity and the interface used by the employees of the company to manage sales-related activities through the web. The class *OldInterface* is the old system, namely OMS, which still works in the solution chosen during feasibility study. Each interface has an unidirectional association with the class *SystemDatabase*, where data about customers, products, and orders is stored: the interfaces should query the database to get / update / delete information. There are several entities aggregated into the SystemDatabase: *Customer* identifies the companies which play the role of customer with respect to the analyzed organization, *Invoice* is the legal document representing the bill for a bunch of orders (class *Order*), *Product* represents the product stocks which are sold to the customers, which can be seen as a single entity from the point of view of the analysis. The class *SpecialOffer* specializes the class *Product*, identifying a special price during some periods for particolar stocks.

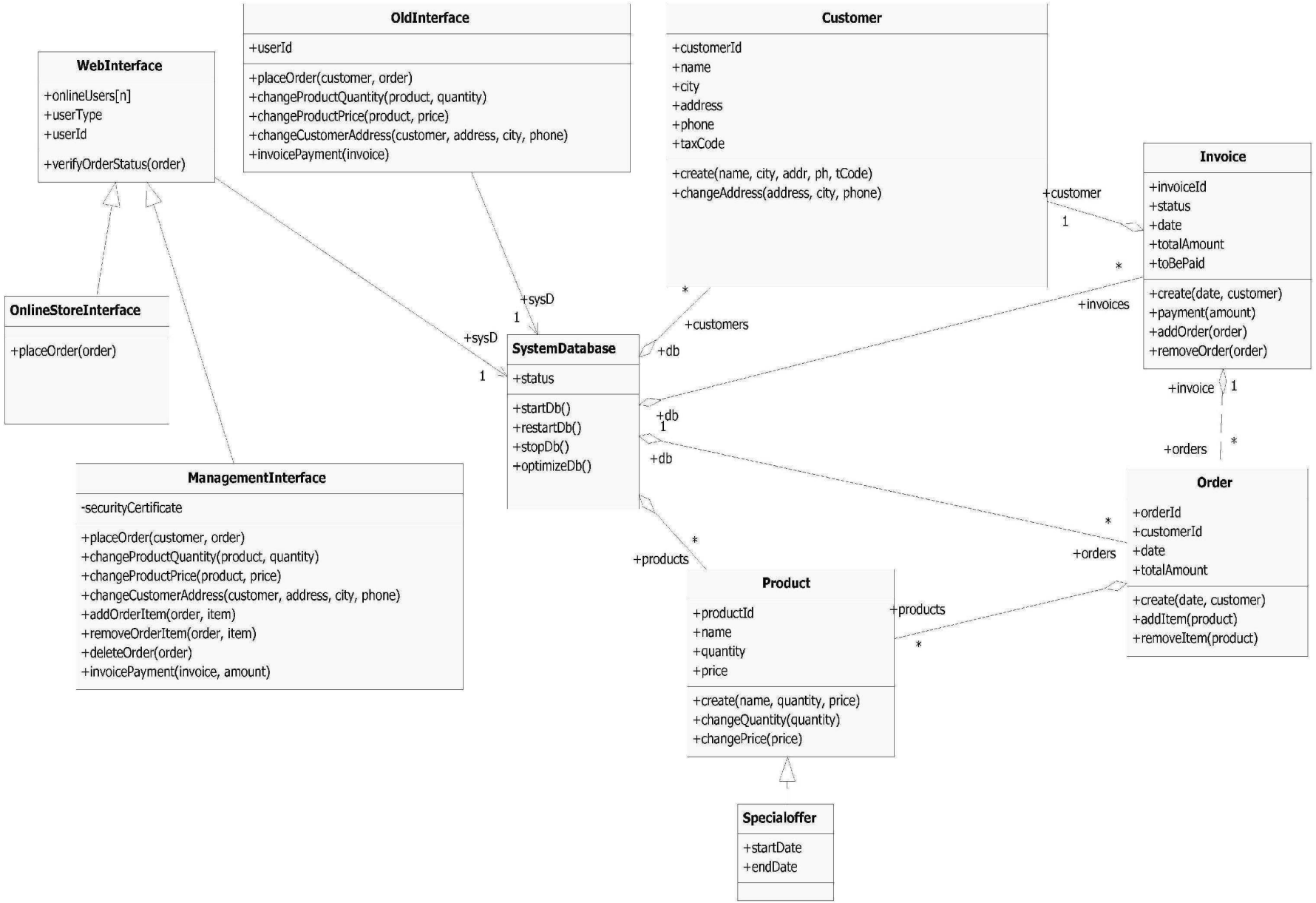


Figure 7: Class diagram of the to-be selling system

After this brief general description of the entities inside the class diagram, we can now go into deeper details for the single classes, and enrich the notation with the aid of the Object Constraint Language (OCL), which can be used to define invariants, preconditions, and postconditions.

WebInterface

```
self.onlineUsers -> exists(self.userId)

WebInterface::verifyOrderStatus(order)
 pre:  self.sysD.orders -> includes(order)
 post: true
```

The web interface is composed by an attribute named *onlineUsers*, which is an array containing the identifiers of the users currently logged in the system through the web interface. Moreover, there is an attribute *userType* representing the type of the user logged in (the allowed values will be shown in the description of the subclasses), and the identifier for the user logged in the instance of the class (*userId*). A method *verifyOrderStatus* allows the verification of the status for a particular order (the parameter *order* is an object of class *Order*). The OCL constraints require the current user id to be in the list of the online users, and the existence of the order for which the status should be shown for the method *verifyOrderStatus*.

OnlineStoreInterface

```
self.userType= 'Customer'

OnlineStoreInterface::placeOrder(order)
 pre:  self.sysD.orders -> not includes(order)
 post: self.sysD.orders -> includes(order) and order.customerId=self.userId
```

The first kind of class specializing from *WebInterface* is the interface to allow customers to access the online store. There is a method *placeOrder* with only the parameter *order*, because each customer can place orders only for himself. The first invariant requires the derived *userType* attribute to get the value 'Customer', while the *placeOrder* method is detailed through a constraint. The precondition is that the *order* object passed as parameter is not present in the database used by the interface, and the postcondition is that after the method execution, the *order* exists and the *customerId* attribute for the order is the *userId* of the current class, namely the logged user.

ManagementInterface

```
self.userType = {SalesAgent, Secretary}

ManagementInterface::placeOrder(customer,order)
 pre:  self.sysD.orders -> not includes(order) and self.sysD.customers -> includes(customer)
 post: self.sysD.orders -> includes(order) and order.customer=customer

ManagementInterface::changeProductQuantity(product, quantity)
 pre:  self.sysD.products -> includes(product) and if quantity < 0 then product.quantity + quantity
  > 0
 post: product.quantity = product.quantity + quantity

ManagementInterface::changeProductPrice(product, price)
 pre:  self.sysD.products -> includes(product) and price > 0
 post: product.price = price

ManagementInterface::changeCustomerAddress(customer, address, city, phone)
```

```
 pre:  self.sysD.customers -> includes (customer) and address <> empty and city <> empty and phone
  <> empty
 post: customer.address = address and customer.city = city and customer.phone = phone

ManagementInterface::deleteOrder(order)
 pre:  self.sysD.orders -> includes(order) and order.invoice = empty
 post: self.sysD.orders -> not includes(order)

ManagementInterface::addOrderItem(order, product)
 pre:  self.sysD.orders -> includes(order) and self.sysD.products -> includes (product) and
  self.sysD.orders.products -> not includes (product) and order.invoice = empty
 post: self.sysD.orders.products -> includes (product) and self.sysD.orders.totalAmount =
  self.sysD.orders.totalAmount + product.price

ManagementInterface::removeOrderItem(order, product)
 pre:  self.sysD.orders -> includes(order) and self.sysD.products -> includes (product) and
  self.sysD.orders.products -> includes (product) and order.invoice = empty
 post: self.sysD.orders.products -> not includes (product) and self.sysD.orders.totalAmount =
  self.sysD.orders.totalAmount - product.price

ManagementInterface::invoicePayment(invoice, amount)
 pre:  self.sysD.invoices -> includes (invoice) and invoice.status <> 'Paid' and  invoice.toBePaid
  >= amount > 0
 post: invoice.toBePaid = invoice.toBePaid - amount and if invoice.toBePaid = 0 then invoice.status
  = 'Paid' else invoice.status = 'Partial'
```

The management interface has a private attribute representing a security certificate, which has to be downloaded from the server to let the users have administrative privileges, after inserting a username and a password. The type of user can be both ‘SalesAgent’ and ‘Secretary’, but it cannot be a customer. The *placeOrder* method places an order for a particular customer, pre-verifying the existence of the customer and the not existence of the order, and requiring the presence of the order associated to the customer after the execution. *changeProductQuantity* checks the existence of the product stock for which the quantity of components needs to be changed, and allows both adding and removing items specifying positive or negative values for the parameter *quantity* (in the precondition particular attention is paid to avoid negative quantity values in the product class). The management interface allows changing the price for a product stock, with the method *changeProductPrice*, which verifies the existence of the product and checks the price to be bigger than zero, and after execution, the price for the product stock should be updated as requested. With respect to the customers, *changeCustomerAddress* allows to change the address of a specific customer, after verifying that the customer exists and the parameters are not empty, and after the invocation the attributes *address*, *city*, and *phone* of the object *customer* should be correctly updated. The method *deleteOrder* removes an order from the database if not yet associated to an invoice, while *addOrderItem* and *removeOrderItem* allow the adding and removal of product stocks from an order, verifying the existence of the order, of the product stock, and the correct execution of the required operation, updating also the total amount of the order. *invoicePayment* is used to account a partial or total payment of an invoice, verifying the existence of an invoice which should be (partially) paid, the amount to be paid being bigger than zero but less than or equal to the amount to be paid, and updating the invoice after the execution. In particular, the amount to be paid is updated, and then the status of the invoice can be changed: if the invoice has been completely paid the status is set to ‘Paid’, otherwise is set to ‘Partial’.

OldInterface

```
/** the same pre- and post-conditions of ManagementInterface, for the available methods**/
```

The old OMS interface is only used inside the company, and allows to perform a subset of the operations of the *ManagementInterface* class. This is the reason why the constraints are not shown.

In the class definition, there is the attribute *userId* to identify the user, and the methods to place an order, change the quantity and price of a product stock, change a customer address, and pay an invoice.

SystemDatabase

```
self.status = {On, Error, Off}

SystemDatabase::startDb()
 pre:  self.status = 'Off'
 post: self.status = 'On'

SystemDatabase::restartDb()
 pre:  self.status = {'Error', 'On'}
 post: self.status = 'On'

SystemDatabase::stopDb()
 pre:  self.status = {'Error', 'On'}
 post: self.status = 'Off'

SystemDatabase::optimizeDb()
 pre:  self.status = 'Off'
 post: true
```

The system database is represented by a class, which is quite simple and has a status (on to represent the working condition, off to represent when it is stopped, error to signal a malfunctioning behavior). The method *startDb* can be executed when the status is 'Off', and sets the status to 'On'. *restartDb* needs the status to be different from Off, and restarts the database setting the status to On after being executed. The same precondition holds for *stopDb*, but the database is then turned off. The optimization of the database (*optimizeDb*) has to be performed when the database is off.

Order

```
self.orderId > 0
self.totalAmount >= 0.00
if self.invoice <> empty then self.invoice.customer.customerId = self.customerId

Order::create(date, customer)
 pre:  self.db.customers -> includes(customer)
 post: self.date = date and self.customerId = customer.customerId

Order::addItem(product)
 pre:  self.products -> not includes(product) and self.invoice = empty
 post: self.products -> includes(product) and self.totalAmount = self.totalAmont + product.price

Order::removeItem(product)
 pre:  self.products -> includes(product) and self.invoice = empty
 post: self.products -> not includes(product) and self.totalAmount = self.totalAmount -
  product.price
```

An order is characterized by an unique identifier (*orderId*) that should be a positive number, a *customerId* related to the customer that placed the order, a *date*, and a *totalAmount* bigger or equal to zero. If the order is associated to an invoice, then the *customerId* should be the same linked to the invoice. The method *create* generates a new instance (is the constructor of the class) with a particular date and with a customer, which should exist in the database, and requires the date to be set, but does not associate any product stock. The adding and removal of products is done through the invocation of *addItem* and *removeItem*, which respectively add the product specified in the

parameter to the order (and requires it not to be present before and is not yet associated to any invoice), and updates the order total amount by adding or subtracting the price of the product stock.

Product

```
self.quantity > 0

Product::create(name, quantity, price)
 pre:  quantity > 0 and price > 0
 post: self.name = name and self.quantity = quantity and self.price = price

Product::changeQuantity(quantity)
 pre:  if quantity < 0 then quantity + self.quantity > 0
 post: self.quantity = self.quantity + quantity

Product::changePrice(price)
 pre:  true
 post: self.price = price
```

The *Product* class represents a product stock, identified by an unique *productId* and textually associated to a *name*, with a number of components specified by the *quantity* (bigger than zero), and with the total *price* for the product stock. The constructor *create*, if the quantity and price are both bigger than zero, generates a product stock entity; *changeQuantity* allows both adding and removing components to the product stock; with the only requirement that we cannot remove more than what is available.

Invoice

```
self.invoiceId > 0
self.totalAmount >= 0.00
self.orders -> includeAll(self.customerId)
self.status = {Paid, Partial, Not Paid}
self.totalAmount = sum(self.orders.price)
self.toBePaid <= self.totalAmount

Invoice::create(date, customer)
 pre:  self.db.customers -> includes(customer)
 post: self.status = 'Not Paid' and self.totalAmount = 0 and self.customer = customer and self.date
  = date and self.toBePaid = 0

Invoice::payment(amount)
 pre:  self.toBePaid >= amount > 0 and status <> 'Paid'
 post: toBePaid = toBePaid - amount and if toBePaid = 0 then status = 'Paid' else status = 'Partial'

Invoice::addOrder(order)
 pre:  self.orders -> not includes(order) and order.customerId = self.customer.customerId
 post: self.orders -> includes(order) and self.totalAmount = self.totalAmount + order.totalAmount
  and self.toBePaid = self.toBePaid + order.totalAmount

Invoice::removeOrder(order)
 pre:  self.orders -> includes(order) and self.status <> 'Paid'
 post: self.totalAmount = self.totalAmount - order.totalAmount and self.toBePaid = self.toBePaid -
  order.totalAmount and self.orders -> not includes(order)
```

The *Invoice* class is characterized by the positive unique identifier *invoiceId*, the status of the invoice (Paid, Partial – only a part has been paid, Non paid), the date when it was prepared, the totalAmount being the sum of the associated orders, and the part of the invoice to be paid (*toBePaid*). The constructor method *create* takes a date and a customer as inputs: it can be executed when the customer exists in the database, and sets the status to 'Not Paid', the *totalAmount* to zero

(it will be increase by adding the orders), the customer object, the date, and the *toBePaid* attribute to zero. The *payment* method allows the partial payment of (part) of the invoice amount, updating the status of the invoice. The *addOrder* verifies that the order has not been added before, and that there is a match between the order and the invoice customer, then adds the order and increases both the total amount and the amount to be paid. *removeOrder* works very similarly, but removes the order from the invoice.

SpecialOffer

```
self.startDate < self.endDate
```

The special offer is a subclass of a product, which is created to sell more easily the stock. It has a *startDate* and an *endDate*, after which the offer is no more valid. The *startDate* should be before the *endDate*.

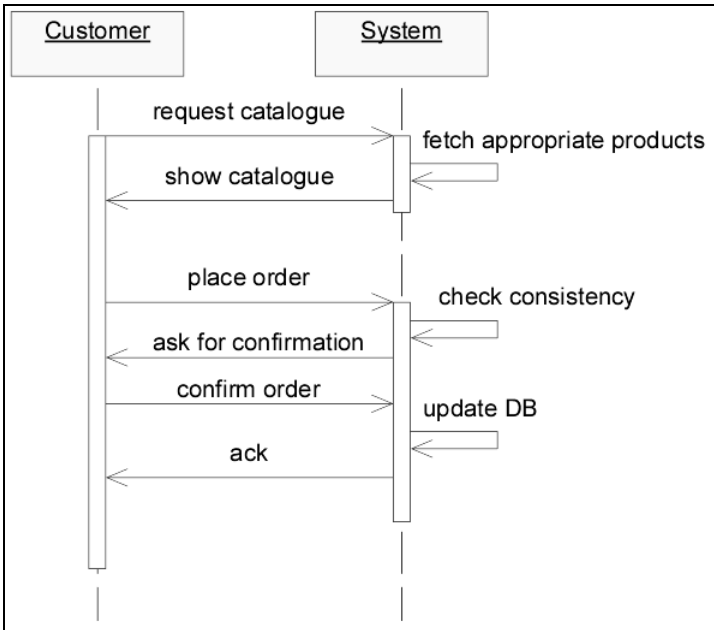
Customer

```
self.customerId > 0

Customer::create(name, city, addr, ph, tCode)
 pre:  name <> empty and city <> empty and addr <> empty and ph <> empty and tCode <> empty and
  self.db.customers -> forAll(taxCode <> tCode)
 post: self.name = name and self.address = addr and self.phone = ph and self.taxCode = tCode

Customer::changeAddress(address, city, phone)
 pre:  address <> empty and city <> empty and phone <> empty
 post: self.address = address and self.city = city and self.phone = phone
```

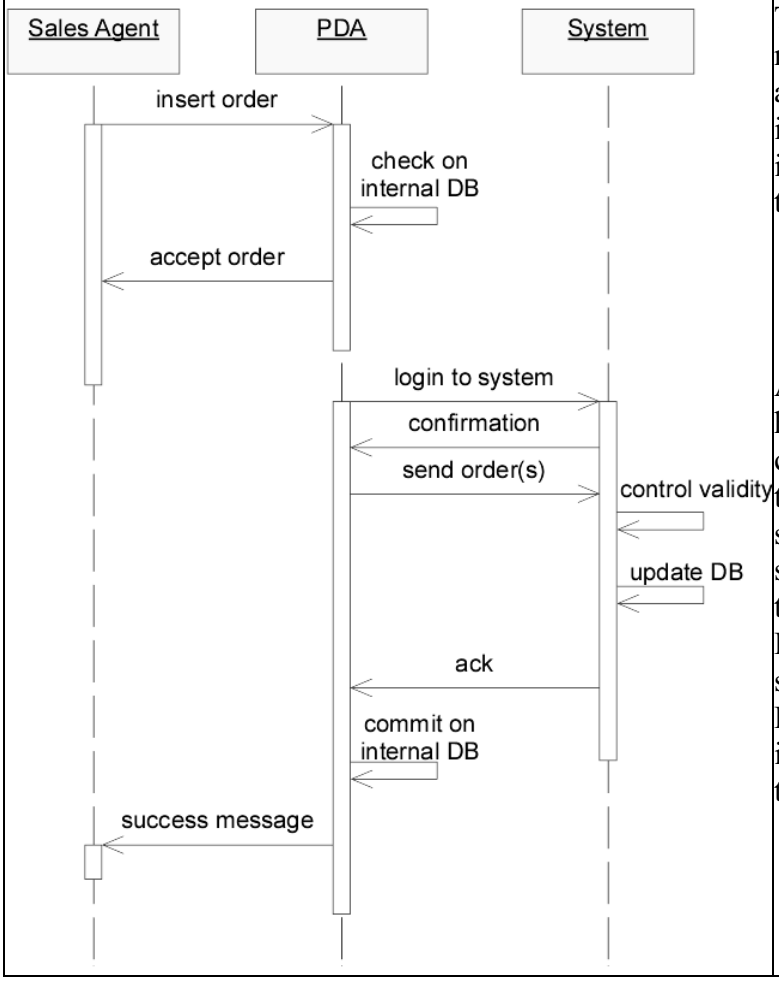
The class *Customer* contains data about the companies which buy supplies from our company. The *customerId* is the unique positive identifier, and there is a bunch of attributes to characterize the company. The *create* constructor verifies that the parameters (name, city, address, phone, tax code) are not empty and that the tax code is not already in some other recorded companies; after the execution the data is updated. The *changeAddress* method works by changing the attributes of the class, after verifying that the parameters are not empty.

10.3 Sequence diagrams

In the next step, we define sequence diagrams for each of the most important use cases previously modeled. The sequence diagrams are useful to detail the execution of a use case. They are simple models of the sequences of interactions between different users and the system. Exceptional execution of the use cases can be modeled in additional sequence diagrams to better explain the sequence of steps needed.

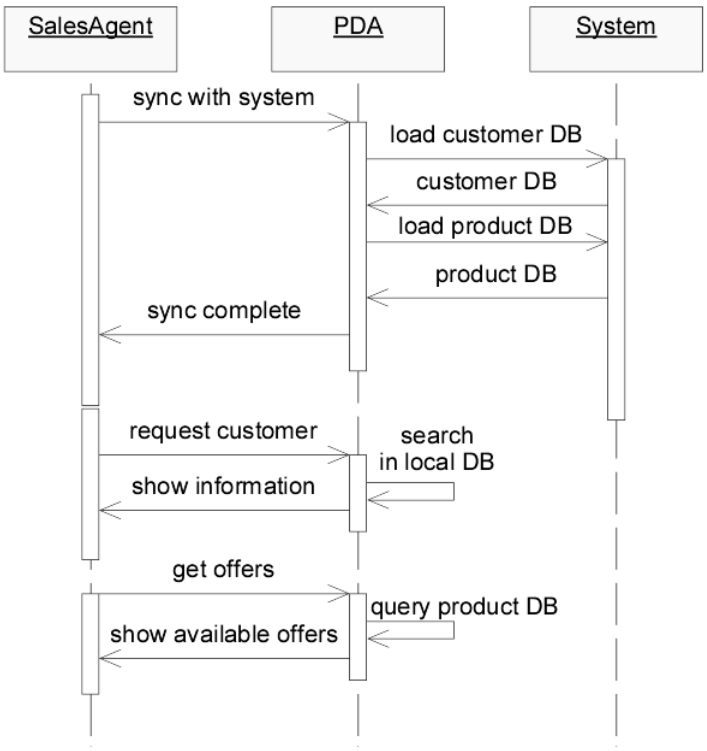


This sequence diagram details the use cases “consult catalogue” and “insert web-based orders”, both with successful execution. If a customer requests to show the catalogue via the web-interface, the system selects the products appropriate for the customer (emphasizing products in relation to former orders and new offers), and shows the web catalogue. After placing the order, the system checks some constraints within the database (quantity,…) and asks for confirmation. If the order is confirmed, it is written to the DB and an acknowledgement is returned to the customer.



This and the following diagrams detail the most important interactions of the sales agent with the system. He uses a PDA to insert orders. The PDA controls the order, inserts it into its internal DB and confirms the successful insertion to the agent.

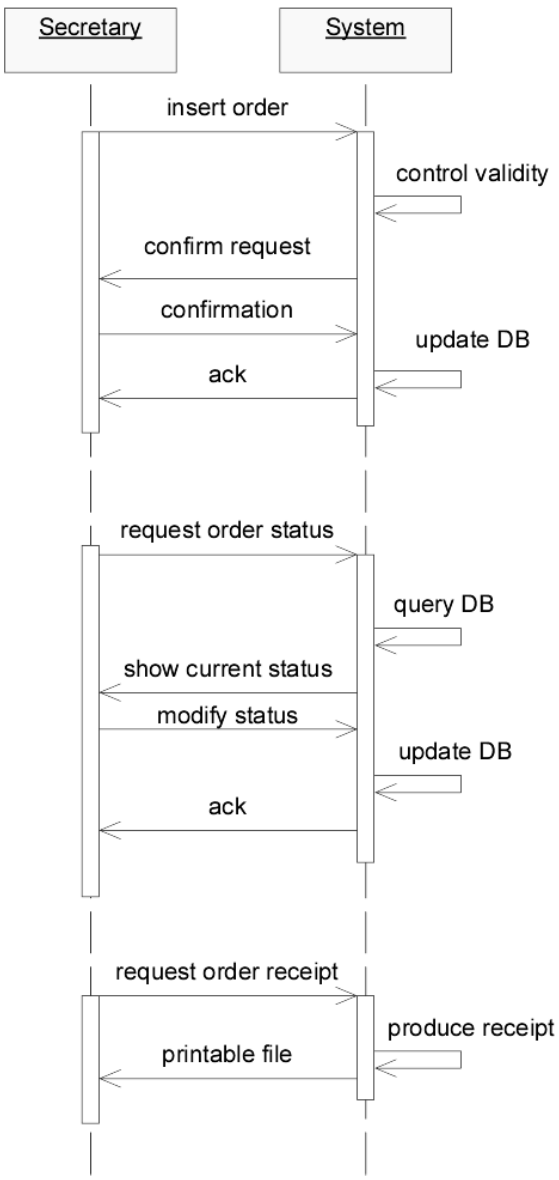
At the end of the working day, the PDA has to load the new orders into the company database. When connected to the company network, it logs in to the system and gets confirmation. Then it sends the orders. In the system, validity of these orders is further checked and the DB is updated. Acknowledgement of the successful transaction is given to the PDA, which commits the order also to its internal DB and gives a success message to the agent.



In this diagram, three independent sequences of interactions are described. To synchronize the PDA product database to the system's one, the user executes a synchronization facility. Synchronization involves loading customer and product DB from the system. At completion, the agent is notified.

When on the road, the agent can now request to the PDA request information on the customers to visit.

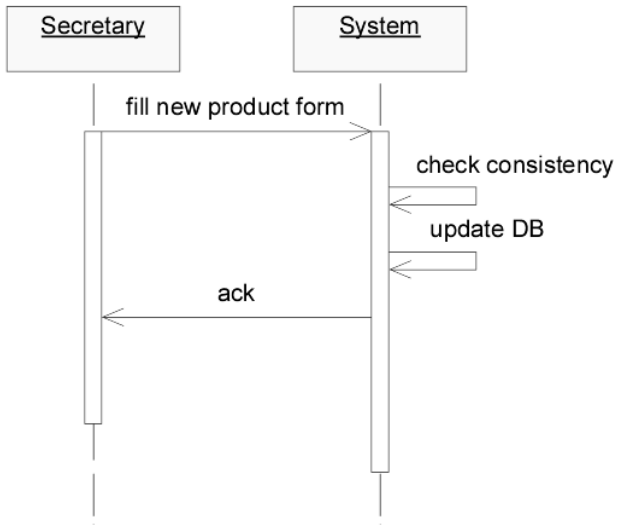
Another simple interaction of the sales agent with the PDA is needed to get the company's actual product offers.



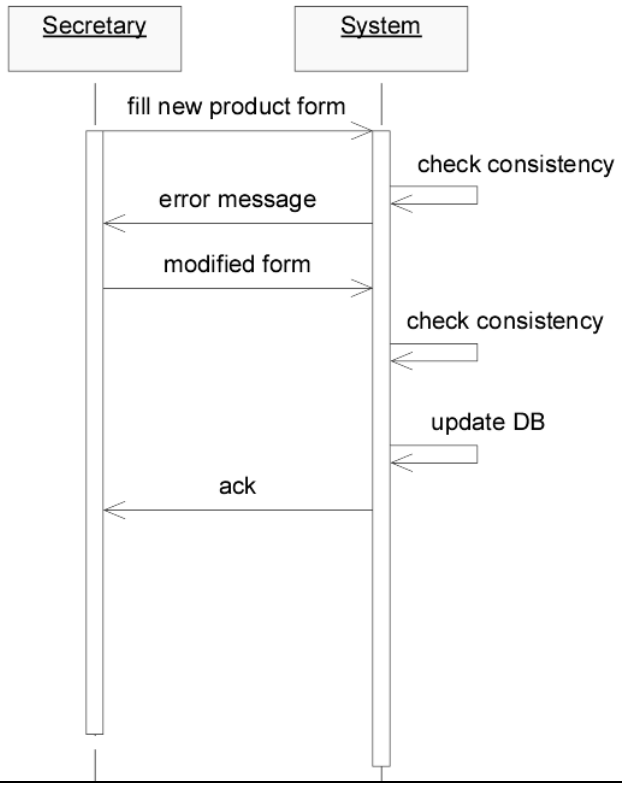
The following three diagrams show the main interactions of the secretary with the system. First, the secretary can insert orders directly into the system. This can be needed if customer orders are made by telephone or by mail, mainly in the transition phase from the traditional to the new ordering process (by web and PDA). The system requests confirmation before committing on his DB, like in the interactions previously described for the customers with the web service.

Moreover, the secretary can view the status of the a particular order and modify it.

Finally, an order receipt can be requested. It is created by the system and returned as a printable file, ready to be sent by e-mail, faxed or directly printed to be mailed.



The secretary can store new products in the database, filling a new-product form. Internally, the system checks if the product already exists, if all forms are filled and if all information is consistent. If all is okay, the database is updated and an acknowledgement is given to the secretary.



This diagram models a simple exceptional interaction occurring if the secretary commits an error at product insertion or inserts a product already in the database, with different properties.
Here, after the consistency check, the system gives an error message where it highlights the possible error. After modification by the secretary, the form is re-checked and then, if now it is correct, inserted into the database and an acknowledgement is given like in the normal interaction, where no errors occurred.

10.4 Activity Diagram

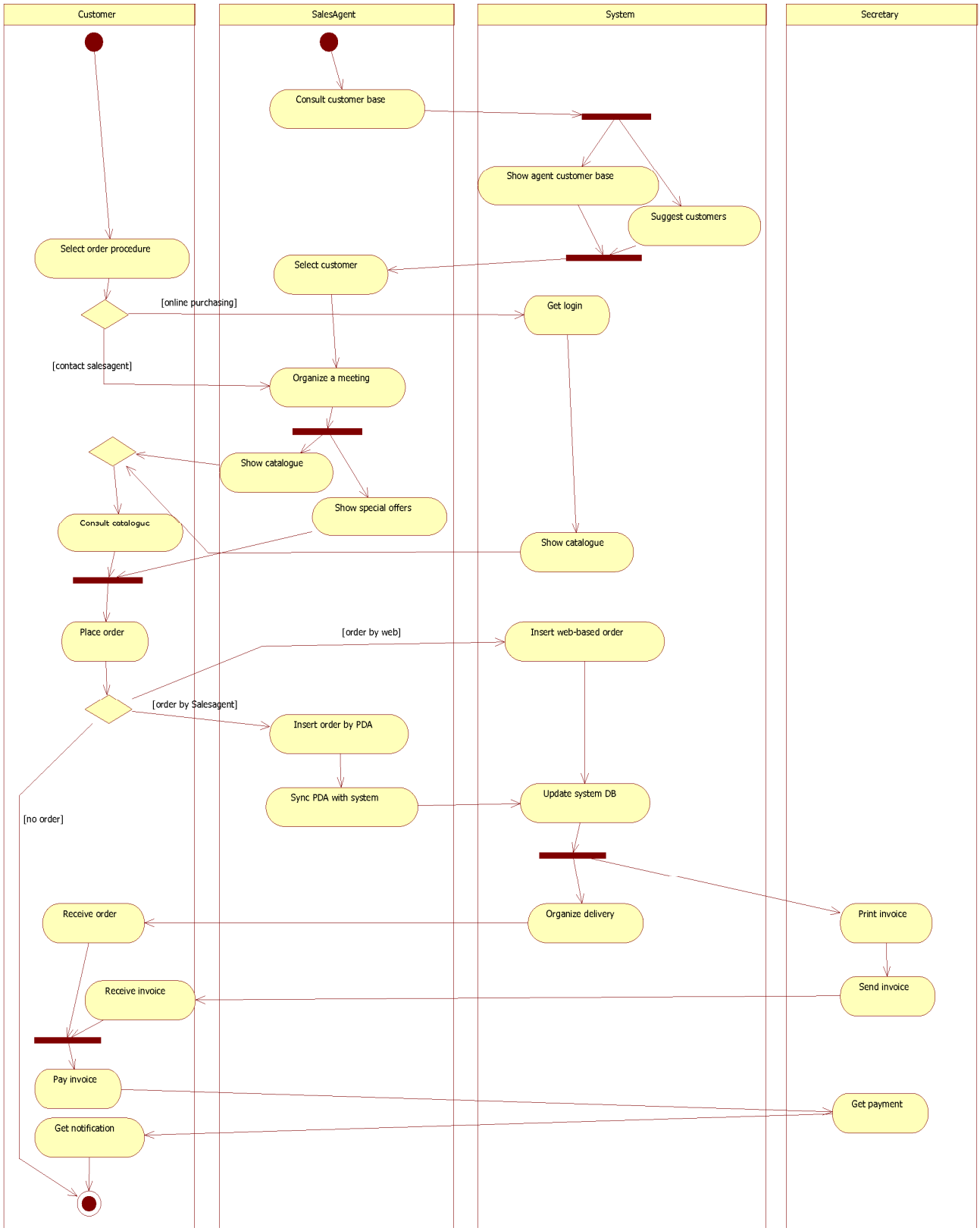


Figure 8: Activity diagram, focusing the product ordering process

The activity diagram in Figure 8 shows the sequence of activities along the product ordering business process. It was modeled starting from the use cases previously defined.
Product ordering starts with establishing a communication (usually basing on an existing relationship) between the customer and the company. Customers can get in contact with the company by setting a meeting with the sales agent or by using the web-shop interface, where they have to log in and can visualize the actual catalogue.
Meetings between customers and agents can be also set up in a different way, starting from the sales agent, who consults the customer base of the system to get customers opportune to contact. Depending on the suggestions made by the system, on the road to take (the old traveler-salesman problem) and his personal impression, the agent selects some customers and tries to set up a meeting with them.
Independent of the initiator, at the meeting the catalogue and the special offers are shown. The catalogue is consulted by the customers and an order is (hopefully) placed, taking into consideration also the available special offers. If the customer does not want to place orders, the scenario ends. The order can be again placed by web or directly to the sales agent. Orders given via web are directly stored into the system database, whereas the sales agent inserts the orders in his personal PDA. At the end of the working day the PDA has to be connected to the company system to download the orders stored and update the system database. At the moment the orders are in the system DB, two activities start in parallel: the delivery is organized (and this process is not further detailed here, because it concerns the companies warehousing and purchasing department) and the invoices are printed. These invoices are sent to the customer after the ordered products were correctly received. As next activity of the process, the customer has to pay the invoice. The secretary finally gets the payment and notifies the customer for the successful transaction.