

Internet Programming

---

**Duelco**  
**Company Web Shop Integration**

---

Fall 2005  
SW7, project group sw701a  
Department of Computer Science  
University of Aalborg





**SEMESTER TOPIC:**

**EN:** Internet Programming

**DK:** Internet Programming

**TITLE:**

Duelco - Company Web Shop Integration

**PROJECT PERIOD:**

SW7,  
September 2nd - December 20th 2005

**PROJECT GROUP:**

sw701a

**GROUP MEMBERS:**

Mads Schaarup Andersen,  
[masa@cs.aau.dk](mailto:masa@cs.aau.dk)

Kristian Bødker,  
[boedker@cs.aau.dk](mailto:boedker@cs.aau.dk)

Martin Madsen,  
[epsen@cs.aau.dk](mailto:epsen@cs.aau.dk)

Daniel Rødtne Poulsen,  
[drpo02@cs.aau.dk](mailto:drpo02@cs.aau.dk)

Mads Vøge  
[voege@cs.aau.dk](mailto:voege@cs.aau.dk)

**SUPERVISOR:**

Lone Leth Thomsen  
[lone@cs.aau.dk](mailto:lone@cs.aau.dk)

**ABSTRACT:**

This report documents the development of the new web shop for the company Duelco. The system is based on a 5-tier architecture, and is designed with emphasis on flexibility and usability. The system is based on an analysis of Duelco's current system and their requirements.

Regarding flexibility, the tier interfaces are designed using XML Schema, and all communication is done using XML. Furthermore the project focuses on the use of XML and related technologies. Although the use of XML caused some problems, overall it helped the development.

Despite software license limitations, the final system ended up as a working prototype thanks to the loosely coupled 5-tier architecture, which meant that only two tiers had to be changed to get a working system.

**NUMBER OF COPIES:** 8

**NUMBER OF PAGES:** 70



# PREFACE

---

The report is written by group sw701a as a part of the SW7 semester, fall 2005 at the Department of Computer Science, Aalborg University. The topic of the SW7 semester is Internet programming, and the semester project is about developing a web shop.

This report documents the development of a web shop based on an 5-tier architecture that is scalable and easy to interface with. The web shop developed is a prototype prepared for integrating with an ERP system. The solution is targeted and inspired by the danish company *Duelco*.

Furthermore, the report contains a reflection over the process of making the web shop, and over the SW7 semester as a whole.

## Duelco

The project was carried out in cooperation with the company Duelco, who supplied the group with input to the web shop functionality and user interface. They provided a copy of their setup in the ERP system Navision Attain, along with some test data. We would like to thank Duelco for the cooperation.

## Reading Instructions

Because of the technical level of the contents, it is necessary for the reader to have background knowledge in the field of computer science. In particular, knowledge of Internet technologies is required.

The report should be seen as a documentation of the whole process of making the new Duelco web shop, and should enable the reader to make a similar system. Furthermore, the report should enable the reader to make changes to the system e.g. exchanging a tier in the architecture.

The report is divided into 10 chapters:

**Introduction** Introduces the project, and the problem in hand. Explains the focus of the project.

**Duelco's Current System** Description and analysis of the Duelco's running system.

**Analysis of the New System** Analysis of the new system according to the analysis of the current.

**Architectural Design** Description of software quality attributes, architecture, and interfaces.

**Tier Design** Design of the individual tiers.

**User Interface Design** Design of the web shop user interface.

**Implementation** Description of the implementation.

**Test** Description of how the tests were carried out.

**Reflection** Reflection of the project process.

**Conclusion** Conclusion.

## Type Settings

References in the report are marked as [X], where X is the number of the reference, which can be found in the bibliography in the back of the report. Furthermore the following type settings were used:

- Keyword are written using the *italic* font.

## CD Content

A CD is supplied along with the report. This contains:

- report/report.pdf – This report in portable document format.
- report/report.ps] – This report in post script format.
- source/ – The source code developed during the project.

## Signatures

---

Mads S. Andersen

---

Kristian Bødker

---

Martin Madsen

---

Daniel R. Poulsen

---

Mads Vøge



# CONTENTS

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Duelco's Current System</b>	<b>5</b>
2.1	Business Process Management . . . . .	5
2.2	The Website . . . . .	6
2.2.1	The Entry Page . . . . .	6
2.2.2	The Main Site . . . . .	7
2.2.3	Issues with the Website . . . . .	7
2.3	The Shopping Module . . . . .	8
2.3.1	Product Pages . . . . .	8
2.3.2	For Registered Users Only . . . . .	9
2.3.3	Issues with the Shopping Module . . . . .	9
2.4	The ERP System . . . . .	10
2.5	Requirements for the New System . . . . .	10
<b>3</b>	<b>Analysis of the New System</b>	<b>13</b>
3.1	System Definition . . . . .	13
3.2	Problem Domain . . . . .	13
3.2.1	Environment . . . . .	14
3.2.2	Class . . . . .	14
3.2.3	Structure . . . . .	15
3.3	Application Domain . . . . .	16

---

3.3.1	Usage . . . . .	16
3.3.2	Functions . . . . .	17
3.3.3	User Interface . . . . .	17
<b>4</b>	<b>Architectural Design</b>	<b>21</b>
4.1	Software Quality Attributes . . . . .	21
4.2	Technical Platform . . . . .	24
4.2.1	Equipment . . . . .	25
4.2.2	Basic Software . . . . .	25
4.2.3	Design Language . . . . .	25
4.3	Component Architecture . . . . .	25
4.3.1	N-tier Architecture . . . . .	25
4.3.2	Split-up in Tiers . . . . .	25
4.3.3	Interface Design . . . . .	26
4.3.4	Delegation of Responsibility . . . . .	27
4.4	Summary . . . . .	27
<b>5</b>	<b>Tier Design</b>	<b>29</b>
5.1	Server Presentation and Client Presentation Tier . . . . .	29
5.1.1	Interface with Business Logic Tier . . . . .	30
5.1.2	XSLT . . . . .	30
5.1.3	Asynchronous Update of User Interface . . . . .	31
5.2	Business Logic Tier . . . . .	31
5.2.1	Class Design . . . . .	31
5.2.2	Interface . . . . .	32
5.2.3	URL Interpretation . . . . .	32
5.3	Data and Data Access Tiers . . . . .	33
5.3.1	Class Design . . . . .	33
5.3.2	Interface . . . . .	34
5.3.3	Data Design . . . . .	34
<b>6</b>	<b>User Interface Design</b>	<b>37</b>
6.1	Interface Design Process . . . . .	37
6.1.1	Paper Prototype . . . . .	37

---

6.1.2	Non-Interactive Prototype . . . . .	37
6.1.3	Interactive Interface . . . . .	38
6.2	Usability . . . . .	39
6.3	Web Specific Considerations . . . . .	41
6.3.1	Cross Browser Support . . . . .	42
6.3.2	Graceful Degradation . . . . .	42
6.3.3	Standard Compliance . . . . .	42
<b>7</b>	<b>Implementation</b>	<b>43</b>
7.1	Overall Implementation Decisions . . . . .	43
7.1.1	Tier Installation . . . . .	43
7.1.2	Error Handling . . . . .	44
7.1.3	Programming Language . . . . .	45
7.2	Server Presentation Tier . . . . .	45
7.2.1	URL Rewriting . . . . .	45
7.2.2	Sessions . . . . .	46
7.3	Business Logic Tier . . . . .	47
7.3.1	Testability and Maintainability . . . . .	47
7.3.2	Configurability . . . . .	48
7.4	Data and Data Access Tiers . . . . .	48
7.4.1	Problems . . . . .	49
7.4.2	Alternative Solution . . . . .	49
7.4.3	Database . . . . .	50
7.4.4	Data Flow . . . . .	50
<b>8</b>	<b>Test</b>	<b>53</b>
8.1	Testing Strategy . . . . .	53
8.1.1	Unit Testing . . . . .	53
8.1.2	Integration Test . . . . .	54
8.1.3	NUnit . . . . .	54
8.2	Testing Tactics . . . . .	54
8.3	Usability Test . . . . .	54
<b>9</b>	<b>Reflections</b>	<b>59</b>

---

9.1	Project Process . . . . .	59
9.2	Project Goals . . . . .	59
9.2.1	To Develop a Prototype Web Shop with Focus on Usability . .	59
9.2.2	To Base the Web Shop on a Scalable N-tier Architecture Easy to Interface With . . . . .	60
9.2.3	To Interface with the ERP System Microsoft Navision . . . .	60
9.2.4	To Maintain Standards and Use Best-Practice Techniques Re- garding Web Presentation . . . . .	60
9.2.5	To Explore and Evaluate XML and Related Technologies . . .	61
9.3	Software Engineering Methods . . . . .	62
9.3.1	OOAD Method . . . . .	62
9.3.2	Spiral Model . . . . .	62
9.3.3	Pair Programming . . . . .	63
9.4	Tools . . . . .	63
9.4.1	Software Tools . . . . .	63
9.4.2	Server Software . . . . .	65
9.4.3	Server Hardware . . . . .	65
9.5	Future Development . . . . .	66
9.5.1	Known Issues . . . . .	66
9.5.2	Future Improvements . . . . .	66
9.5.3	Improving User Interface Responsiveness . . . . .	67
<b>10</b>	<b>Conclusion</b>	<b>69</b>
<b>A</b>	<b>Flow Diagrams</b>	<b>71</b>
<b>B</b>	<b>XML Schemas</b>	<b>79</b>

# INTRODUCTION

---

During the last 10 years the Internet and in particular the World Wide Web has gone from being used by researchers to being accessible to ordinary people all over the world. The development has entailed a lot of new uses, and one of these is e-commerce which is usually realized through a web shop. Developing such a web shop is the goal of this project. The web shop developed is for the danish company Duelco.

## Duelco

Duelco is an engineering and trade company which run a web shop based on an ERP System. This web shop is however outdated in a number of ways. The user interface has several usability issues, and the ERP system primarily used in the company is not the ERP system that the web shop interfaces with. I.e. two different ERP systems are used in parallel.

The purpose in relation to the company is therefore to make a web shop prototype which presents a new user interface, and which interfaces with the company's primary ERP system.

## Web Shops

Web shops can be everything from very simple applications interfacing directly with a database to having more complex and flexible architectures, but at a minimum web shops usually consist of at least a user interface and data handling functionality. In relation to this, this project will focus on building an application with an n-tier architecture to enforce certain software quality attributes.

The interfaces used between tiers should be based on widely accepted standards in order to support current and future development. In the context of web applications, web services are a natural choice for application to application communication.

## XML

XML is a standardized way of creating structured data formats. XML has a wide range of tools supporting the development, processing, interpretation and validation of XML data formats. XML has also gained wide acceptance in the industry due to its flexibility and vendor neutral nature.

In this project the focus lies on the XML technologies: XSLT, XPath, and XML Schema. The reason behind this is that these technologies are somewhat stable and that the experiences gained by the use of these can be used in later projects.

Traditionally, XML can be thought of in two ways: The concrete textual representation, and the conceptual tree model. The aim of this project will be to work with XML only as the conceptual tree model (DOM), and thereby not allow changes directly on the string representation of the data. This will make abstraction of the system higher, and allow a better use of XML tools and standards.

## Analysis and Design Method

The method used in the analysis and design phases of the of the project is based on the Object Oriented Analysis and Design method[1].

Alternative methods were also considered. One of these was Extreme Programming (XP). This was however dismissed based on three considerations.

The group consisted of an uneven number of members. This would somewhat hamper pair programming or at least result in a maximum of two programming teams.

The XP method (and other agile methods) requires a very tight feedback cycle with the customer. And such frequent contact with the customer, was not possible with the customer in hand.

Further more none of the group members had the sufficient experiences with XP to allow this method to be used without tying up many resources to using the method correctly and divert focus from the actual development. The group members were familiar with the OOAD method, and as learning a new method was not a goal of this project, the XP method was not used.

## Project Goals

This section summarizes the focus of the project by stating the goals and the motivation behind these.

The goals of this project are:

**To develop a prototype web shop with focus on usability** One of the goals in this software engineering project is to develop a web shop. By development is meant to ac-

tually analyze, design and implement a prototype that ends up working. An important part of this is to implement the designed solution, and show that it works in practice.

Developing a web shop is a common task under the theme of Internet programming, which makes it easy to approach, and therefore allows for putting more effort into other things, such as usability. Usability is a major issue as more and more people without any computer experience or training are now using computers and their associated applications – there amongst web applications and web shops.

**To base the web shop on a scalable n-tier architecture easy to interface with** Basing the web shop on a complex architecture takes it from being a common and trivial task to being an interesting and educating challenge. A scalable architecture is of high relevance if the performance of the web shop should be able to be increased as more users begin to use it. Dividing the architecture into separate tiers of functionality in a so called n-tier architecture allows for the most loaded parts of the system to be multiplied and load-balanced easily. At the same time, the split-up into tiers makes it easy to interface the system with other systems, because only one tier has to be modified and adapted to interface with another system.

**To interface with the ERP system Microsoft Navision** Many companies use an ERP system to manage their business processes, and in the case of the company Du-elco, Microsoft Navision is used. As the ERP system is supposed to tie all the business processes together in one place, it is important that also the web shop is designed to inter-operate with it. By interfacing with Microsoft Navision, the web shop should be able to utilize the business logic that is already implemented in Microsoft Navision. In that way, the web shop can be seamlessly integrated in the business processes of an already established company.

**To maintain standards and use best-practice techniques regarding web presentation** Web browsers are known to be very tolerant of incorrect HTML code and try to display the HTML at best effort anyway. The problem is that different browsers do this differently, and the web pages might not look as intended. This can be avoided by adhering to the standards for HTML and other web presentation technologies.

Best-practice techniques will be used to maximize the separation of presentation and content. By doing this, presentation logic that is used on several pages on the website can be cached in the browser, and will thus optimize the load time.

**To explore and evaluate XML and related technologies** As XML has grown to become the de facto standard for data representation, and especially for e-business integration, it demands attention in a project like this. Many XML related technologies have emerged, and many tools and programming languages have adapted accordingly to support these. To explore and evaluate XML and related technologies, the system

developed in this project should base its communication and data manipulation hereon. More specifically, this amounts to:

- Use XML as data format in general
- Work with XML as a tree data structure, and not as plain text
- Use XML web services
- Use XSLT for transforming XML data into XHTML presentation
- Design interfaces using XML Schema

# DUELCO'S CURRENT SYSTEM

---

As this project takes starting point in the sales order processes of Duelco, it is appropriate to start out reviewing how these are managed. This is done based on a meeting with Duelco, and by exploring their current website, shopping module, and their ERP system.

The meeting with Duelco took place in the initial phase of the project. Here Duelco gave a presentation of the company with some insight into their business processes and a brief explanation of their current solution for business and sales management. Additionally, Duelco listed some requirements they had for a new and improved solution.

## 2.1 Business Process Management

Duelco's business processes are managed through an ERP system called Microsoft Navision Attain version 3.6 with a version 3.7 kernel. Navision Attain is used for warehouse management, sales order management, customer relations management and most other business processes in the company. One particular thing, that is *not* running on Navision Attain, is the shopping module on the company website.

Their current shopping solution is based on Lotus Notes, which manages its own database and is completely separated from the Navision Attain ERP system. All orders made through the shopping module are manually transferred to the ERP system. Actually, this is part of the sales order processing; when a customer places an order in the web shop, the order is considered a quotation. This quotation is then reviewed, confirmed as an actual order, and entered into Navision Attain as an order by an employee.

## 2.2 The Website

This section briefly describes the content and structure of Duelco's current website[2]. The language of the website is Danish, but in this description of the site all referenced labels and texts are translated into English.

Antal	Varenr	Beskrivelse	Info	Enhed	Bestilling	Pris excl. moms
<b>Maskinsikkerhed, Datasensor</b>						
1	95ASE1000		?			DKK
1	95B030000		?			DKK
1	S937530090		?			DKK
1	95A251240		?			DKK
1	G5220262		?			DKK
1	J950535393		?			DKK
1	S940700023	Datasen.R1-PLASTIC REFLECTOR 23 mm	?	STK		DKK 41,95
1	S940700075	Datasen.R5-PLASTIC REFLECTOR 75 mm	?	STK		DKK 64,50
1	970500020	Datasensor BWS-T4-MT	?	STK		DKK 6285,20
1	970500030	Datasensor BWS-T4-N	?	STK		DKK 5355,55
1	970500040	Datasensor BWS-T4-N-MT	?	STK		DKK 6053,05
1	95A251380	Datasensor CS-A1-02-G-03 axial M12 4 poles unshield 3m	?	STK		DKK 74,20
1	95A251270	Datasensor CS-A1-02-G-05 axial M12 4 poles unshield 5m	?	STK		DKK 87,90

Figure 2.1: A screenshot of the product listing in Duelco's current solution.

### 2.2.1 The Entry Page

The website has an entry page, that welcomes the user to Duelco's website. It has a login form, on which registered Duelco customers can authenticate themselves with a user name and a password. After authenticating, the user is directed to the main part of the website, which is described later in this section.

Instead of logging in, the user has a number of other options available as hyperlinks. The first is *Guest Entrance*. This directs the user to the main part of the website logged in as a guest user. In contrast to registered users, a guest user is not able to see prices on products, make orders, and access customer related data. In all other areas, the site is identical for guests and registered users.

The second hyperlink is *Subscription*. This brings up a form, the user can fill out to apply for becoming a registered user. Duelco then checks the application, and normally only existing customers are allowed to become registered users. This is to prevent competitors from spying on the prices.

The third hyperlink *Contact*, presents the user with Duelco's contact information. The last hyperlink, *Duelco Safety Techniques*, leads the user to an English version of the

website, which is disregarded in this project.

### 2.2.2 The Main Site

The main part of the website is built using frames, that split the site up in three parts: *top*, *left* and *right*. The top frame contains the company logo and a navigation menu. It also displays the user name of the user logged in. The left frame is dedicated the shopping module. It functions as a sidebar containing a product search menu and some shopping cart information with related options. The right frame is the main frame displaying the content of the current page. The content of this frame changes when selecting something from the navigation menu in the top frame and when accessing the shopping options in the left frame. The content of the main frame is also changed by activating any hyperlinks in the content of the currently displayed page.

The navigation menu gives access to all the content of the website, which is not part of the shopping module. It has nine different menu items leading to pages with *company profile*, *news*, *product categories*, *contact information*, *product brochures*, *suppliers*, *terms of sale and delivery*, and *help on using the website*. The content of these pages will be categorized as *general content* and will not be described further, as focus is on the shopping module on the website.

### 2.2.3 Issues with the Website

The first thing to notice on the website is the entry page. The user cannot access the main part of the website before choosing either guest or registered user. This requires an extra and unnecessary effort to access the actual page. Furthermore, selecting *Guest Entrance* could give the user the impression, that he is not really welcome and only allowed to see a limited version of the website, which is not true. Only a few things in the shopping module are hidden. A solution to this problem is to assume that the user is a guest and skip the entry page. If the user then wants to log in, this can be done at any time while browsing the website.

The use of frames on split up the website is a highly discussed topic[3]. In the case of this website, it ensures that the navigation menu and the sidebar is always visible even when scrolling down the text in the main frame. However, the use of frame leaves the website difficult to bookmark, difficult for a search engine to index, and difficult for simple web browsers to display. This should be reason enough to eliminate the use of frames.

Examining the HTML source code of the website reveals two additional issues. Firstly, HTML tables are used to do layout. According to W3C this should be avoided[4]. The semantics of tables is to present tabular data. Layout should instead be done using CSS.

Secondly, the code has a high code-to-content ratio, which means that a lot of HTML code is used to present a relatively small amount of content. This is caused by extensive

use of tables and by use of inline styling information. The bulky code increases the load time of the HTML document, and furthermore, it results in a lower rating on many search engines because the density of keywords is lower when the code-to-content is high[5]. This is another reason to reduce the use of tables. Also, extracting the inline styling information and putting it in a separate CSS file will reduce the size of the HTML document. The CSS file can be cached by the browser and only has to be loaded once.

A last thing to notice is the nine menu items. Some of the titles are very general, and does not tell the user what content to expect from the corresponding pages. Also, nine pages is quite many to look through to find, what one is looking for. When examining the content of the pages, it seems that the content could be grouped differently in maybe half as many pages.

## 2.3 The Shopping Module

The shopping module is accessed through the sidebar in the left side. The sidebar contains a product search menu consisting of three drop-down boxes, a text field, and two buttons: reset and search. The three drop-down boxes lets the user select *main category*, *sub category* and *manufacturer*. Initially, all of the main categories, sub categories and manufacturers are available for selection. When a main category is selected, only the sub categories under that category and the manufacturers that have products in that category are available. In the text field a free text string can be entered. Clicking the search button displays a listing of the products that satisfy the search criteria in the main frame. The free text search is matched against the description text of a product.

### 2.3.1 Product Pages

The product listing consists of a table with seven columns with the following labels: *Quantity*, *Product Number*, *Description*, *Info*, *Unit*, *Order*, and *Price ex. VAT*. The *Quantity*, *Order*, and *Price ex. VAT* columns are empty if the user is a guest. If the user is a registered user, the *Quantity* column contains a text field for each row with default value 1. This number can be changed and it specifies the number of products ordered by clicking the shopping cart icon now displayed in the *Order* column. The *Info* column contains an info icon with a hyperlink to a product details page with additional information about the corresponding product.

The product details page has ten fields, that are labeled: *Product Number*, *Designation*, *EAN number*, *Unit*, *Price per Unit*, *Weight*, *Volume*, *Catalogue*, *Description*, and *Documentation*. The *Product Number*, *Unit*, and price field contain the same data as the corresponding columns in the product listing. The *Description* field now contains the name of the manufacturer, and the *Designation* field contains the text of the *Description* column. The *Catalogue* field is rarely used but contains a reference code,

presumably to some Duelco or supplier catalogue. The *Documentation* field contains a hyperlink to a product brochure in PDF format. The fields not described are self-explanatory. The product details page also contains the *Quantity* column field and the shopping cart icon from the *Order* column. Lastly, a picture of the product or product group is displayed.

### 2.3.2 For Registered Users Only

If the user is a registered user, the sidebar also contains some shopping cart summary information, a hyperlink to the shopping cart, a hyperlink to previous orders, and a hyperlink to a page called *Terms of Business*. The shopping cart summary information consists of a field showing the last product added to the shopping cart and a field showing the total price of products currently in the shopping cart. The summary information is automatically updated when a product is added to the shopping cart without reloading the main frame content. Thus, the user can continue shopping undisturbed.

The shopping cart hyperlink displays a page with the content of the shopping cart in the main frame. This page also calculates the total price inc. VAT. From this page the content of the shopping cart can be changed or deleted, and the user can proceed to check-out, which loads a new page.

The check-out page has a form with billing and delivery information and some additional information for the user to fill out. The page also shows an outline of the order corresponding to the content of the shopping cart. From this page the user can submit the order or cancel it. Clicking *Cancel* does not cancel the shopping session, but directs the user back to the shopping cart page. Submitting the order presents the user with a page saying, *Thank you for your order*.

The hyperlink in the sidebar for previous orders changes the main frame to a page with a table listing any previous orders the user has made. The columns are: *Web Order Number*, *Requisition Number*, *Date*, *Total*, and *Show Order*. The *Show Order* column contains an icon that links to an order details page. This page shows the same information as the check-out page did, when the order was submitted, in static form.

### 2.3.3 Issues with the Shopping Module

Regarding the shopping module, a few points of improvement were identified.

The product search menu does not give any overview of the different categories and sub categories available. This could be a good thing to have for guests to browse through.

Some of the columns in the product listing page should be changed. Instead of supplying an info icon as hyperlink to product details, the description text should be the hyperlink, and the *Info* column could be spared. The *Quantity* column should be placed next to the *Order* column to indicate the number of products to order.

On the product details page, the inconsistency in use of labels should be fixed. Fur-

thermore, Duelco desired a field displaying if the product is in stock. This corresponds to their desire to integrate the shopping module with their ERP system, which contains storage information.

Further requirements from Duelco was for registered users to see both the net price of the products and the price counting in the discount the user has from the customer contract with Duelco.

## 2.4 The ERP System

The review of the ERP System is based on a copy of Duelco's configuration including some test data, which has been imported into a local installation of the ERP system. It was only possible to explore some of it because of a limited license for the ERP system (see 9.2.3 on page 60).

The current website and shopping module is, as mentioned, based on Lotus Notes. It should instead be based on Duelco's ERP system, and get its data and functionality from that. The ERP system manages a database with all data of the company. Regarding data relevant for the shopping module is holds data on *products*, *prices*, *orders*, *storage*, and *customers*. The current functionality could not be investigated because of the limited license.

For the ERP system to replace Lotus Notes it must additionally hold data on product categories, registered users of the shopping module, and general content for the website. The product categories should reflect the main and sub categories present in the current shopping module. The data required for registered users is the same as for customers with the addition of authentication information. The general content is more diverse and probably requires some new tables in the database of the ERP system. Furthermore, the ERP system must be able to differentiate between orders and quotations because of the ordering procedure mentioned in Section 2.1 on page 5.

The functionality needed for the shopping module concerns user authentication, product searching, price calculation and order creation. How much of this functionality is already present is unknown.

## 2.5 Requirements for the New System

To sum up the points of improvement found in the review of Duelco's current system above, here is a list of requirements for the new system.

Regarding the website in general, the new system should:

- Eliminate entry page and assume the user is a guest.
- Avoid using HTML frames.

- Use CSS instead of HTML tables and in-line styling to do layout.
- Regroup the general content to fewer pages.

Specifically for the shopping module, the new system should:

- Have a page with overview of the different product categories.
- Rearrange the columns in the product listing.
- Use consistent naming across the different pages.
- Include storage information on products.
- Display both net price and customer discount price on products.

For the ERP system to accommodate this, it must have the following data and functionality:

- Authentication information for registered users.
- General content data for the website.
- Product categories information.
- Functionality for searching products.
- Prices and discounts calculation.
- Creation of quotations instead of orders.

Besides these points of improvement, the existing shopping module is well structured and presents the content in a reasonable way. As there is no reason to change what works, the description given in this chapter should be used as a guideline when designing the new system. Preserving the general structure and layout when designing the new system will moreover make it easier for existing to users to use it.



# ANALYSIS OF THE NEW SYSTEM

---

In this chapter the new system is analyzed. The analysis is based on the OOAD method. This is done by making a system definition and analyzing the problem and application domains. The analysis is based on the analysis of the current system in Chapter 2 on page 5.

The following is used from the method: First a system definition is made, and then the problem domain and application domain are analyzed.

## 3.1 System Definition

A web shop selling electrical installation equipment, based on the ERP system Microsoft Navision, which holds information about customers and products. The web shop user interface should allow easy navigation, product browsing and order creation.

## 3.2 Problem Domain

The analysis of the problem domain was done of a system which does not interface with an existing Navision Attain database. The idea behind this is to identify the data which is needed in the model, and to later use this knowledge to make the necessary adjustments to the Navision Attain database, described in Section 5.3 on page 33.

In the following section an analysis of the problem domain will be given. Classes, class structure and events are then extracted from this overview. Behavioral analysis is not done since the classes appeared to have trivial behavior.

### 3.2.1 Environment

The web shop must contain information of users, orders, and products. Users can be of two separate types; registered costumers and guests to the web shop.

Guests in the web shop can browse products, and view the static data such as company profile and news. Registered users have the same functionality available to them as guests, additionally they have a shopping cart available in which they can add and remove products.

Based on the content of their shopping cart, a registered user can get a price for the contents of his shopping cart. They can place an order for the contents of their shopping cart, and review old orders.

Products must be categorized according to a main and a sub category.

### 3.2.2 Class

This overview shows a number of different classes in the system.

**Guest** Guests have the basic functionality available to them such as the static content of the website and product browsing.

**Registered User** Registered users must be uniquely identified by a user name. They must also have a password that confirms their identity. Besides this, contact information such as address and phone number must be supported. They also have old orders associated with them, and a shopping cart when using the web shop.

**Product** A product must be uniquely identified by a product id number. Information describing the product must be supported, such as price and general description. Products have a main and a sub category associated.

**Shopping Cart** A shopping cart can hold pairs of products and quantities. It is also able to calculate a price based on the users identification, and these contents.

**Order** An order in the system is one that has been places by a registered user through the system. An order contains pairs of products and quantities, just like a shopping cart.

In addition to these, classes such as news and page could be considered. As they do not have any apparent complexity or association with the mentioned classes these will be ignored.

	Reg. User	Guest	Order	Cart	Product
product added	X			X	X
product removed	X			X	X
order placed	X		X	X	
logged in	X	X			
logged out	X	X			
old order requested	X		X		
products searched					X

Table 3.1: Table of events.

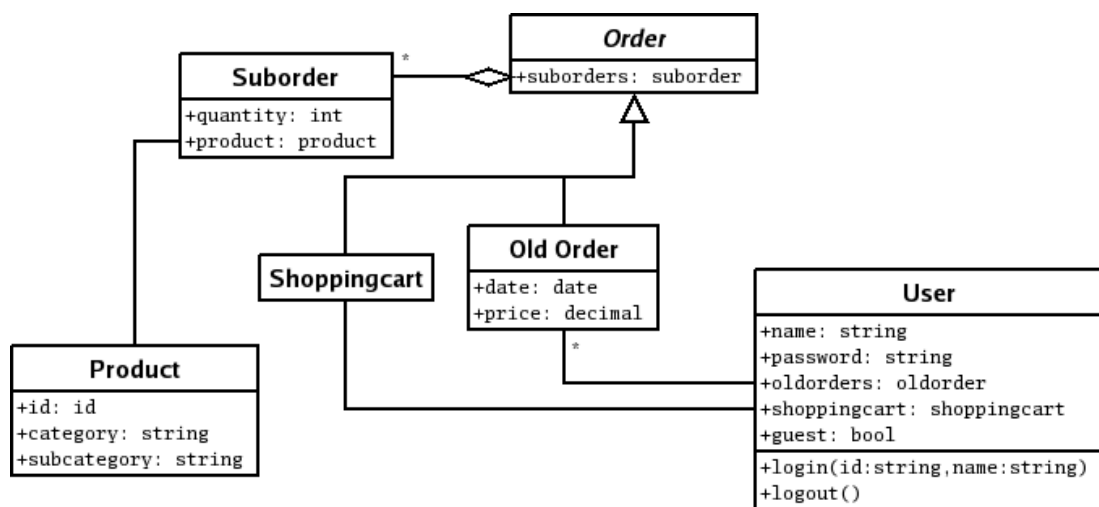


Figure 3.1: Structure of the classes of the problem domain

### 3.2.3 Structure

In Table 3.1 events and their associated classes can be seen.

Based on the preceding analysis of classes and the event table, registered users seems to be guests with extended functionality available. It is decided to model this, by making the two classes into one class, with a field depicting whether it is a registered user or a guest. This class will then have to states: logged in and guest. The login and logout methods change between these states.

Also orders and shopping carts seems to be partly similar, and can be modeled as having a common super class. Finally, a class called Suborder is added to model pairs of products and quantities.

The structure of the system can be seen in Figure 3.1.

	Guest	Registered User	Administrator
view old orders		X	
assemble order		X	
place order		X	
view cart		X	
register user	X		X
browse products	X	X	
add/remove news			X
browse site	X	X	
log in	X		
log out		X	

Table 3.2: The use patterns available to each actor in the system.

### 3.3 Application Domain

The following is an analysis of the application domain, and will examine the actors, use patterns and functionality.

#### 3.3.1 Usage

The following actors are identified:

**Registered User** A registered user on the website.

**Guest** An unregistered user on the website.

**Administrator** An administrator for the website.

The use patterns available to these actors can be seen in Table 3.2.

The more complex patterns will be elaborated in the following.

**Browse Products** Products can be browsed either by main and sub category, or by free text search.

**Assemble Order** Products can be added to or removed from the shopping cart. This use pattern is only available to registered users, as the prices for items may depend on the users deal with Duelco. For the same reason, the *place order* and *view cart* usage patterns are also exclusive for registered users.

**Register User** New users created by a guest through the website must be activated by an administrator before it can be used to place orders. This is due to the fact that

Function	Complexity	Type
create/update user	Simple	Update
calculate prices	Unknown	Calculate
place order	Simple	Update
retrieve user	Simple	Read
retrieve history	Medium	Read/calculate
retrieve order	Simple	Read
search products	Medium	Read/calculate
retrieve product	Simple	Read

Table 3.3: Functions in the Application Domain

price systems must be negotiated with Duelco. This is however not a concern for the analyzed system.

**View Old Orders** Registered users must be able to browse their old orders. They must be able to see an overview of all of their orders, or to see details for a single order.

### 3.3.2 Functions

Table 3.3 shows a list of the functions in the application domain.

**Calculate Prices** The calculate prices function is marked as unknown complexity, as the exact system of pricing at Duelco is currently unknown. It is at least of medium complexity, as it will depend on both the user, and the number of items ordered.

**Search Products** Products can be searched by a combination of category, subcategory and free text.

**Retrieve History** This function will involve creating an overview of all the old orders of a single user.

### 3.3.3 User Interface

This section uses a slightly different approach than the OOAD method since many aspects of the user interface is set by the fact that the interface should be viewed through a web browser with the limitations in technology this enforces.

The user interface goals of the system regarding human factors and usability consists the following:

- Existing costumers, familiar with the existing system, should be able to use the developed system with little extra effort.
- The affordance of the navigational elements should be improved compared to the existing system.
- Improve cross browser performance by identifying technology-related browser-issues and handle these, or provide alternative content.

### **User Identification**

Focusing on two types of users, guests and registered users.

### **Interaction Patterns**

The system consists of two main parts, the web page as a whole and the web shop.

**Web Page** The main interaction pattern of the website as a whole is browsing. The main goal of this interaction pattern is to allow the users to explore and select information relevant to the individual users information needs.

In order to support the user's ability to browse the information on the website a logical grouping of related information needs to be made. This grouping of information into different groups and sub groups was done by identifying the different types of information present on the existing web page and rearranging these in an information hierarchy for the new website.

**Information Hierarchy** One extreme way of presenting the information is to provide all the information at once. This would however result in an overwhelming amount of information and not allow the user to accomplish the task of finding any particular information and hence decrease the utility of the system.

The other extreme would be to separate the information into tiny fragments and then arrange these into myriads of groups and subgroups across a wast tree-structure.

The resulting information hierarchy consists of a tree structure with a relative low branching factor (around 1:5) with a clear grouping into 4 categories with sub-categories:

- Forside (front page)
- Produkter (products)
- Web shop
- Firmaprofil (company profile)

One of these categories is the actual shop which will be analyzed in detail in the following section.

**Shop** The web shop differs from the rest of the website since the goal shifts towards selling products. Therefore the customer must be able to easily find products and add these to an order.

The main use patterns in the web shop part of the system is identified as:

- Interaction elements
- Login
- Selecting products
- Checkout
- Viewing shop history

The user interface should facilitate two different user groups: *Guests* and *Registered Users*, since not all information is relevant to both of the groups. The user interface should also be able to show information about *products*, *shopping cart* content and *orders*.



# ARCHITECTURAL DESIGN

---

In this chapter the design of the overall architecture of the system will be discussed. The different sub-tasks involved in designing the architecture are: Defining quality attributes, components identification and constructing the actual architecture in relation to the software quality attributes and overall project goals.

## 4.1 Software Quality Attributes

This section will examine the software quality attributes of the system. The Open Process Framework was chosen as a source of software quality attributes [6].

An overview of how the software quality attributes are prioritized can be found in Table 4.1 on the next page. The purpose of this is to set the boundaries of the project.

### **Accessibility**

When designing the system special considerations should not be made in this area.

- Priority: Irrelevant

### **Configurability**

Since Duelco should be able to change the content of the web site it is important that the data can be altered without recompiling the system.

- Priority: Important

	Very Important	Important	Less Important	Irrelevant
Accessibility				X
Configurability		X		
Correctness	X			
Efficiency				X
Extensibility				X
Interoperability	X			
Maintainability		X		
Performance				X
Portability				X
Reliability		X		
Reusability	X			
Robustness			X	
Scalability	X			
Security				X
Testability		X		
Usability	X			

Table 4.1: Software Quality Attributes of the system

### Correctness

Correctness is very important in this project as one of the purposes is to explore the uses of XML and related technologies. This is only possible if the XML being generated and processed conforms standards since this is a precondition for many of the tools and technologies.

- Priority: Very Important

### Efficiency

One of the main purposes of this project is to explore XML and related technologies. This implies some overhead in data processing and as a consequence the efficiency of the system has been down prioritized.

- Priority: Irrelevant

### Extensibility

It is not considered important that the system should be extensible.

- Priority: Irrelevant

### **Interoperability**

This is very important because the system should be able to interoperate with various data storage systems.

- Priority: Very Important

### **Maintainability**

It is important that the different parts of the system can be rewritten, hereby making the system more maintainable.

- Priority: Important

### **Performance**

As with efficiency, performance is not considered important as it has higher priority to examine XML.

- Priority: Irrelevant

### **Portability**

The system is specifically designed for Duelco, and is not a general application to be used by many different companies with different platforms. Therefore portability is irrelevant.

- Priority: Irrelevant

### **Reliability**

Because the web shop is used by existing and potential customers of Duelco, problems with with reliability will give the customers a negative impression of the company. Therefore reliability is important.

- Priority: Important

### **Reusability**

It is considered very important that the system should be easily changed. An example would be to change the user interface without having to change anything else.

- Priority: Very Important

**Robustness**

It is not considered very important that the system should run under abnormal circumstances. This is not in the scope of the project.

- Priority: Less Important

**Scalability**

Since scalability is one of the specific goals (see Chapter 1 on page 2) of the project this is considered very important.

- Priority: Very Important

**Security**

As this application sends sensitive information over a network security would normally be considered important. The scope of this project is however to develop a working prototype which demonstrates the features, and because of that security is not considered important.

- Priority: Irrelevant

**Testability**

The system has to be testable in order to facilitate the testing of the functionality and thereby identify defects early in the development cycle. This aspect is considered important.

- Priority: Important

**Usability**

This is also a specific goal of the project and is therefore considered very important (see Chapter 1 on page 2).

- Priority: Very Important

## 4.2 Technical Platform

This section describes the technical platform of the system.

### 4.2.1 Equipment

The system is designed to run on any platform as it is very flexible how the design of the architecture should be realized. The system however requires a web server running on the computers on which the system has to run. One of the computer in the system does however have to run Microsoft Navision Attain, and because of this that particular machine has to be running a Microsoft Windows operating system.

### 4.2.2 Basic Software

It was planned that the system should be developed using the Microsoft Visual Studio .NET environment. The design was based on using web standards as a way of displaying the website.

### 4.2.3 Design Language

The figures in the design are based on the UML language.

## 4.3 Component Architecture

In this section the system is decomposed into components. The idea here is to distribute the responsibility of the system into components with each their area of responsibility. As it was a part of the project goals to make a scalable architecture (see Chapter 1 on page 2) it was chosen to base the system design on an N-tier architecture.

### 4.3.1 N-tier Architecture

This is also known as a layered architecture. Here each component is a layer with well defined interfaces with the layers immediately above and below. The interface to the layer below describes the operations to which the component has access, and the interface to the layer above describes the operations that the component provides to that layer.

### 4.3.2 Split-up in Tiers

Next the split-up in areas of responsibility is made. This is based on the standard component split-up from the OOAD method [1] where there is a UI, a functional, and a model component. The naming of the tiers is inspired by the article “N-Tier Architecture - How to get there” [7]. Based on the analysis the different components have the following responsibility:

**UI - Presentation Tier** Contains the functionality needed for the UI. In this case this component has the responsibility of showing web pages in a web browser.

**Functional - Business Logic Tier** Has responsibility of the shopping cart, session handling, web page creation. The component is also responsible of retrieving data from the model component.

**Model - Data Tier** Contains the data of the system.

However, this split-up in tiers would not enforce the software quality attributes of the system. Because of that the next step is to identify which quality attributes are not satisfied, and how the architecture can be changed to enforce these.

**Scalability** Should the amount of traffic load on the system rise to an extend where load balancing on the middle tier is needed this three component architecture is not flexible.

With the three tiers presented above, it is not possible to introduce load balancing in the middle tier without rewriting the entire functional component. To enforce this quality attribute of scalability a new tier is inserted between the Presentation Tier and the Business Logic Tier. This tier will then have the responsibility of creating the web pages, and is named: *Server Presentation Tier*. Furthermore the Presentation Tier is renamed to *Client Presentation Tier*. With the new Server Presentation Tier it is now possible to rewrite this tier with a load balancer without touching the implementation of the Business Logic Tier, and Scalability is hereby enforced.

**Maintainability** How the Data Tier is accessed highly depends on what constitutes this tier. If the system goes from having Navision Attain as the Data Tier to having a SQL Server, the interfacing method is changed. This would require rewriting the entire Business Logic Tier, as this holds this functionality. The functionality used to connect to the Data Tier is therefore moved to a new Tier called *Data Access Tier*.

How the tier design ended up in relation to the reference model can be seen in Figure 4.1 on the next page.

### 4.3.3 Interface Design

The interfaces between the tiers are designed to be as flexible as possible. Because of this the design was based on the Simple Object Access Protocol (SOAP)[8] for a programming language independent communication.

As accessing a web service is a rather expensive operation (among other things due to TCP and HTTP protocol overhead) the interfaces should be designed so that one call from the Client Presentation Tier does not result in multiple calls from the Data Access Tier to the Data Tier. This is called the branching factor as should be kept as

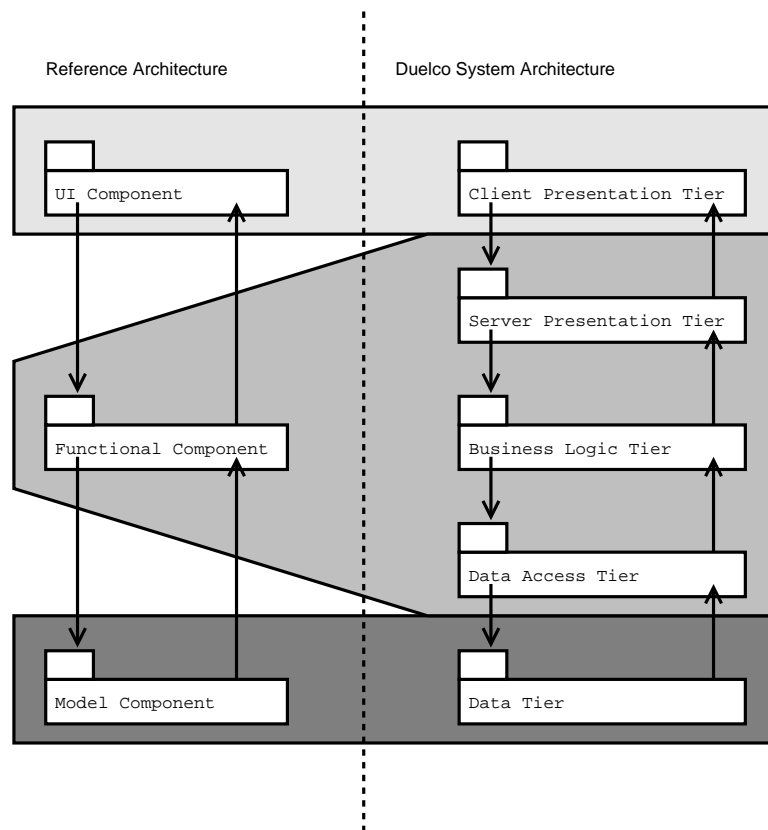


Figure 4.1: Duelco tier design in respect to the reference model.

low as possible ideally one to one. The principle is illustrated in Figure 4.2 on the following page.

#### 4.3.4 Delegation of Responsibility

An important aspect of the proposed architecture is to figure out where different calculations are done. Here it was chosen to leave functionality such as searching and price calculation to the Data Tier. One might argue that this requires more calls down the architecture, but as the information needed for the calculation is stored in this tier, it makes good sense not to store unnecessary information in the above tiers. Furthermore it makes good sense to do searching in the lowest tier instead of sending rather large documents further up in the architecture.

## 4.4 Summary

To sum up it was decided to base the design on an n-tier architecture for increased scalability and maintainability. Furthermore it was decided to make five tiers as the three of the reference model would gather too much functionality in a single tier, hereby

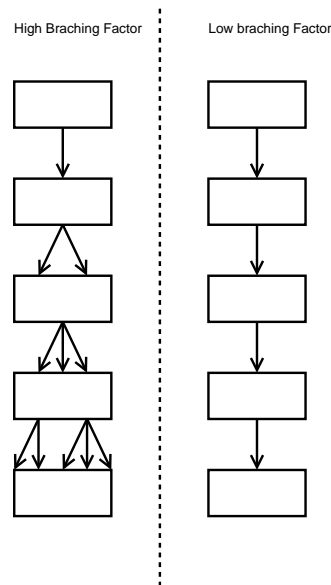


Figure 4.2: Difference between a high and a low branching factor. The figure illustrates how many calls one call from the top tier requires.

contradicting scalability and maintainability. The tier design ended up being the following:

**Client Presentation** Visualises the website on the client machine.

**Server Presentation** Creates the XHTML documents.

**Business Logic** Holds the shopping cart and controls sessions.

**Data Access** Accesses the underlying Data Tier.

**Data** Holds the data. Performs calculations.

# TIER DESIGN

---

In this chapter the tiers identified in the previous chapter will be designed. It was chosen to design the Client Presentation Tier and the Server Presentation Tier together as they are closely related. This was also the case with the Data Access Tier and the Data Tier. The design was done top-down. This was done since it was natural to look on a web page and figure out what data was needed to create it.

The individual tier designs contain a class and an interface description along with aspects which are interesting in the particular tier.

The XML Schemas for interfacing between the different tiers are described in Appendix B on page 79. The flow diagrams explaining calls down the architecture are described in Appendix A on page 71. In the latter explanations of how the schemas correspond to the method call can also be found.

## 5.1 Server Presentation and Client Presentation Tier

The Client Presentation Tier is responsible for presenting the user interface to the user of the system, and the Server Presentation Tier is responsible for telling the Client Presentation Tier what to display. The Server Presentation Tier is also responsible for receiving any requests from the Client Presentation Tier and forwarding these to the Business Logic Tier. The Business Logic Tier then sends a response to the Server Presentation Tier, which presents the response to the Client Presentation Tier.

The Client Presentation Tier consists of third party web browsers, and is responsible for running any client side scripting such as JavaScript, and rendering the user interface, which is web pages. The Server Presentation Tier must therefore provide the display logic code for the web browser, which is HTML.

To sum up, the Server Presentation Tier has two functions:

- To *transform* XML data received from the Business Logic Tier into HTML for

the Client Presentation Tier.

- To *wrap* HTTP requests from the Client Presentation Tier into XML documents and forward these to the Business Logic Tier.

### 5.1.1 Interface with Business Logic Tier

The input received from the Business Logic Tier should consist of a single XML document which adheres to an XML Schema. This XML Schema specifies the message format for communication from the Business Logic Tier to the Server Presentation Tier. It should be able to handle both error messages and normal content. The data needed for the content can be derived from Duelco's current website described in Chapter 2, and from the analysis in Chapter 3. The normal content can be split up into several different kinds of data. This is reflected in the schema as follows:

**Meta Data** Data that is common for the entire website. This part of the schemas can be seen in Figure B.14.

**Shopping Cart** Data about the content of the shopping cart is special, because it is presented at most of the web pages. See Figure B.21 for schema.

**User Info** Data on the user is relevant whenever a user is logged in.

**Page Data** Data that is specific for each different page of the website. Schema is displayed in Figure B.17.

**Simple Content** This consists of general building blocks for simple content such as head lines, paragraphs and listings. See Figure B.22.

**Complex Content** This is content data specific for the different kinds of pages of the website. Each of these have their own schema: Figure B.12, Figure B.13, Figure B.15, Figure B.18, Figure B.19, and Figure B.23.

All this data and different schemas are included into a single schema called `response.xsd`. This can be viewed in Figure B.20.

The communication from the Server Presentation Tier to the Business Logic Tier is described in Section 5.2.2.

### 5.1.2 XSLT

As the Business Logic Tier sends it data as XML, a natural approach is to use XSLT to transform the data from the Business Logic Tier into HTML, which in the form of XHTML is also XML.

The communication from the Server Presentation Tier to the Client Presentation Tier should also adhere to an XML Schema. For this schema the specification for XHTML

1.0 Strict[9] should be used, because this is the version of XHTML that encourages the highest separation of data and layout, as expressed in the project goals in Chapter 1.

A more specific XML Schema could also be designed, so that all web pages are required to be on the same form and include some required elements by following this schema. This is a good way of making multiple tiers generate similar web pages. But since this system only have one tier generating web pages, similarity constraints between various web pages can be implemented directly into the tier. The Server Presentation Tier does this by generating all web pages using one XSL Transformation.

The XSL Transformation consist of mapping data form the response.xsd schema into the schema for XHTML 1.0 Strict. The structure and the final layout of the web pages should correspond the user interface designed in Chapter 6.

### 5.1.3 Asynchronous Update of User Interface

A way of improving the response time of the user interface is to utilize asynchronous HTTP requests. This is done by a JavaScript technique called Asynchronous Javascript And XML (AJAX), which is based on a proprietary XMLHttpRequest object, which is implemented in a very similar manner in different web browsers.

AJAX can improve the interface such that when a user adds an item to the shopping cart, the shopping cart status is updated without a page reload.

This functionality can be implemented by adding functionality to the Server Presentation Tier and Client Presentation Tier. In the Client Presentation Tier JavaScript must be used to retrieve the updated shopping cart information, and to replace the existing shopping cart information. This can be achieved by requesting Server Presentation Tier for the updated information using XMLHttpRequest and replacing the information by DOM manipulations.

## 5.2 Business Logic Tier

The Business Logic Tier is the tier that is responsible for maintaining state for connections to the web server. It is also responsible for mapping HTTP requests from the Server Presentation Tier, to an XML data document containing the data needed to create an XHTML document.

### 5.2.1 Class Design

The Business Logic Tier contains a part of the data model of the problem domain. In particular the Business Logic Tier holds the shopping cart.

The state of a connection, called a session in the following, includes the shopping cart class from in the analysis. The session must also contain information of the user having

the connection, i.e. whether he is logged in, and if so userid and name. This is not the user class from the analysis though, as it resides in the data access tier; it is only a copy of some of the information associated to user.

This leads to the class diagram seen in Figure 5.1.

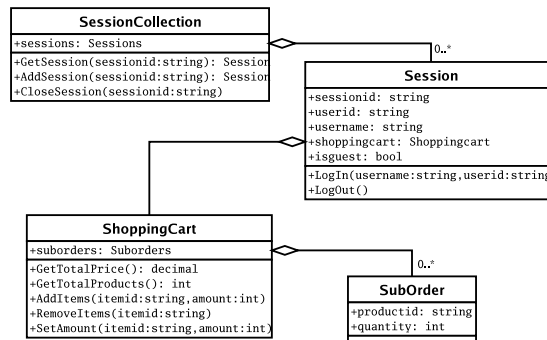


Figure 5.1: Class diagram for sessions in Business Logic Tier

## 5.2.2 Interface

The interface to the Business Logic Tier is very simple, as the Server Presentation Tier simply forwards HTTP requests down after formatting them to XML. The structure of this document can be seen in Figure 5.2. The `formdata` element contains variable provided by the user.

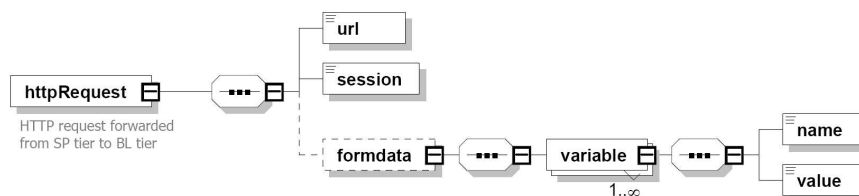


Figure 5.2: HTTPrequest Schema.

## 5.2.3 URL Interpretation

As the Business Logic Tier is responsible for mapping URLs to data documents, a system for doing this must be designed. In Figure 5.3 on the next page, a decision graph for the URL mapping can be seen.

All the decision points branches on the next part of the URL. E.g. the URL `http://www.duelco.dk/produkter/elpanel` will branch down the `produkter` line, then down the `{productid}` line, and a product detail page will be shown.



As the tier is stateless and non of the information in the tier is persistent, it is chosen to use the singleton design pattern. The tier only contains this class, which is used to get *user*, *order*, and *product information* from the Data Tier. The class diagram can be found in Figure 5.4.

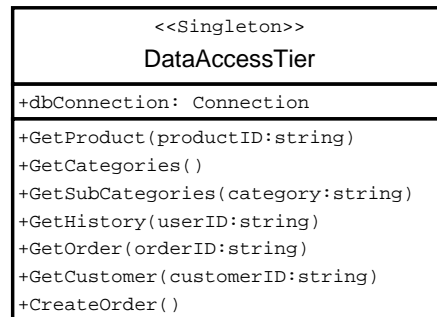


Figure 5.4: Class of the Data Access Tier

The Data Tier is part of the data model, and is represented by a Navision Attain solution, and the class design here is not done by the project group.

### 5.3.2 Interface

Interfacing with Navision Attain can be done in two ways; using data-ports and using XML Ports. As it was decided to do communication using XML, XML Ports is a natural choice given the XML goals of the project. An xml-port is actually just a table with four columns; two columns to name the table and field to map from, and two columns to name the XML node and type to map to.

The XML Ports only define the mapping between an XML document and a relational database – it does not actually call the transformation. This is done by Code Units, which in essence are stored procedures. The communication between the Data Access Tier and the Data Tier is done using the Windows Message Queue and not web services as the other tiers, as this is how communication with Navision Attain is done. To extract information from the XML Ports Code Units have to be written. Code Units is the internal language of Navision Attain, and is fragments of code reading and/or manipulating Navision Attain tables, forms, and content. A Code Unit and an XML Port has to be made for each possible request from the Business Logic Tier to the Data Access Tier.

### 5.3.3 Data Design

As a requirement for the project is to interface with an already existing data model, this section will detail the reverse engineering of relevant parts of this data structure into an ER diagram to get an overview. As this data structure is not designed to interface with a web shop, certain changes might be necessary, and these will be described too.

**Procedure** The procedure involved in reverse engineering the Duelco data structure involved several steps. First the requirements and the existing website was reviewed, to analyze what data was necessary to have in the Navision Attain database. Then the Navision Attain database extract given to us by Duelco was imported in Navision, and the object browser was used to browse for relevant data.

As the tables were not accessible with the Navision license provided at that time, the tables themselves could not be viewed. Forms from the Navision system were however accessible, and these were used to make an ER diagram. This means that the ER diagram might not show the exact structure of the database, but the information in it is in some way available in the database.

It was discussed what changes would have to be made to fulfill the requirements of the system, and these changes were then integrated into a new ER diagram.

**Reverse Engineering** The analysis of the problem domain in shows (see Section 3.2 on page 13), that the central classes of the system are orders, users, and products. Thus this is what was searched for in the Navision database.

The reverse engineered ER diagram can be seen in Figure 5.5. Only entities, keys, and foreign keys are included, i.e. not all attributes are included.

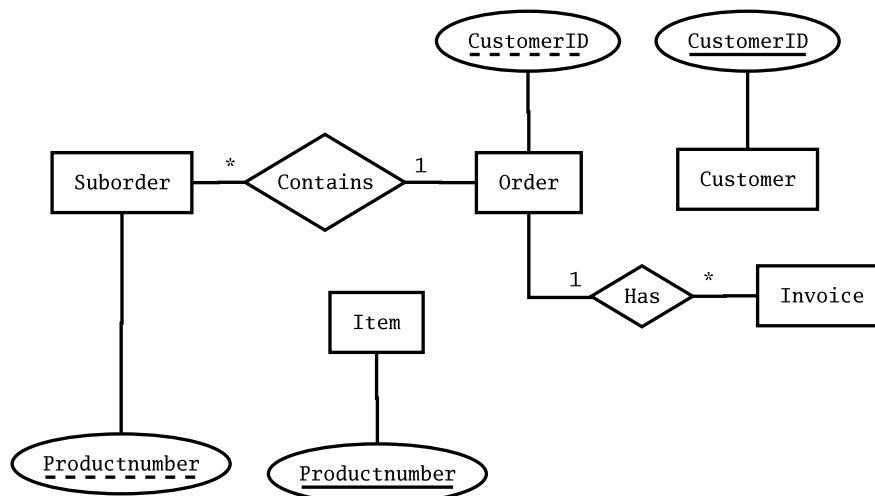


Figure 5.5: ER diagram of the initial Duelco Navision database.

As the figure shows some information seems to be absent in the database. First users does not seem to have a password field, second products does not seem to be categorized. For the web shop to fulfill the requirement of being able to work with Navision as backend, this information has to be added to the database. A proposed new ER diagram for the database can be seen in Figure 5.6 on the next page.

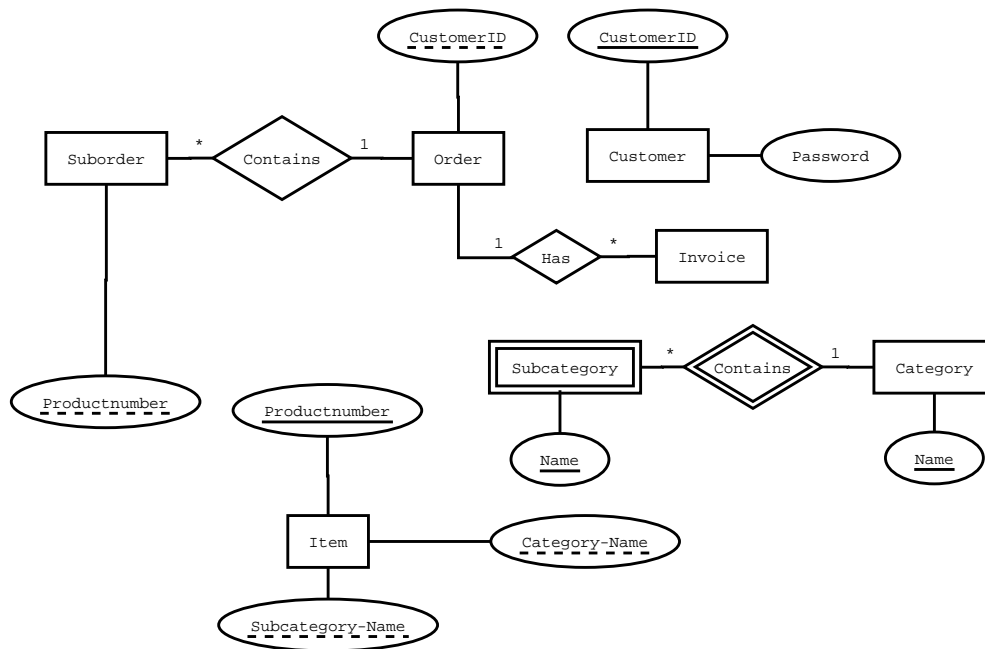


Figure 5.6: ER diagram of the proposed Duelco Navision database.

# USER INTERFACE DESIGN

---

This chapter documents the development of the user interface of the system. Since the system is Internet based the design of the user interface is somewhat different than the design of normal graphical applications. This is mainly due to the fact that the end product (the actual graphical user interface) is somewhat outside the control of the developer due to the different capabilities of web browsers.

## 6.1 Interface Design Process

The interface design process was done as an iterative process using several different techniques in order to test ideas quickly and evaluating these internally within the development group.

### 6.1.1 Paper Prototype

The first iteration of the user interface was made using paper and a pencil in order to allow quick modification.

The main goal of this paper prototype was to get the overall layout of the pages discussed and documented for later use.

An example of one of the paper prototypes developed can be seen on Figure 6.1 on the following page.

### 6.1.2 Non-Interactive Prototype

The second iteration of the user interface consisted of a set of non-interactive graphical representations created using image manipulation software.

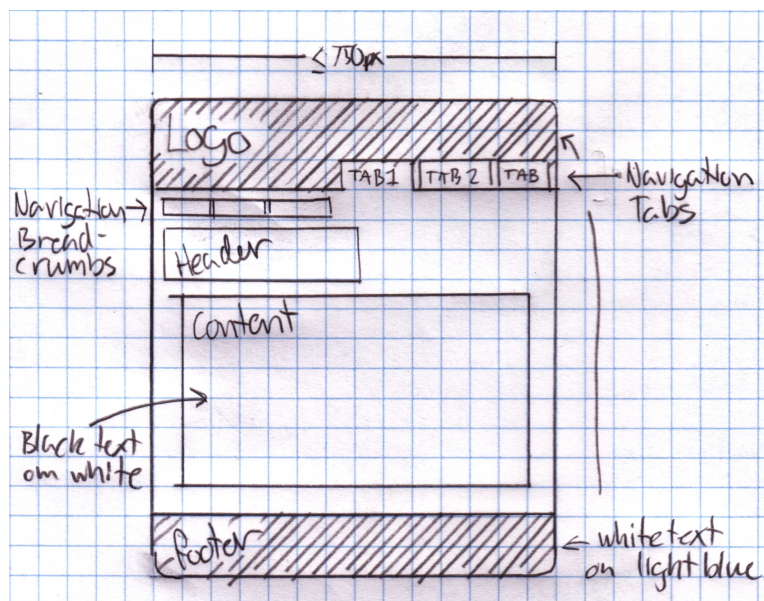


Figure 6.1: One of the early paper versions of the interface.

The main goal of these prototypes was to determine the relative sizes of the different interactive components and to evaluate the aesthetic aspects of the graphical design.

These prototypes were used internally in the development team and this was the first material presented to the user. An example of such a prototype can be seen on Figure 6.2.



Figure 6.2: Example of one of the non-interactive prototypes.

### 6.1.3 Interactive Interface

The final interactive interface was constructed using XHTML, CSS and Graphic elements from the non-interactive prototype. This allowed for a relative easy transition from the non-interactive to the interactive version.

During the development of the actual GUI there were so of issues that had to be handled. In particular since the GUI is actually a rendering performed by browsers and that these does not always render XHTML source as specified by the standards.

In order to identify these browser specific problems early in the development phase the UI were continuously tested as a part of the development in a broad selection of browsers. These were:

- Mozilla (On Windows)
- Firefox (On Unix)
- Opera 8
- Lynx (On Unix)
- Internet Explorer

## 6.2 Usability

The definition of usability used in this report is a so called logical definition (opposed operational) in which usability is defined by a set of attributes.

These attributes are the following:

**Effectiveness** Effective to use

**Efficiency** Efficient to use

**Safety** Safe to use

**Utility** Have good utility

**Larnability** Easy to learn

**Memorability** Easy to remember how to use

Like the software quality attributes from the design, these attributes can be prioritized and presented in a a table, as a Usability Quality Matrix[10].

The Usability Quality Matrix for this project is shown in Table 6.1.

		Very Important	Important	Less Important	Irrelevant	Trivial
Usability	Effectiveness			X		
	Efficiency	X				
	Safety		X			
	Utility	X				
	Lernability	X				
	Memorability			X		

Table 6.1: The Usability part of the *Usability Quality Matrix*.

An in depth description of these attributes can be found in [10].

As shown in the Usability Quality Matrix (Table 6.1) the key attributes for this project is *Efficiency*, *Utility* and *Lernability*. In order to promote these aspects of the system some guidelines was used regarding the design of the user interface.

## Usability Design Principles

During the design of the user interface the following usability design principles were used in order to heighten the above mentioned attributes for the system's user interface.

- Affordance
- Consistency
- Mapping
- Feedback

### Affordance

As determined by the Interface Design Process the four main sections of the web page should be accessible through a tab based interface.

In order to comply with the first of the four chosen design principles the main navigational tabs should be present on all pages sub-pages. In part in order to allow the user to quickly identify which section of the page is active, and in part to comply with the affordance design criteria, allowing easy access to the other sections of the site.

### Consistency

The consistency design principle states that interaction elements with similar effect should have similar appearance. The term consistency can be split up into internal and external consistency describing the consistency within the product and consistency with other similar products.

**Internal Consistency** Navigation elements on all pages should have the same location and structure in order to maintain internal consistency and in order to in coexistence with the logo aid the user in identifying the page as a part of the website.

**External Consistency** External consistency assures that a new user on the website will be familiar with the navigational elements and therefore be capable of using the website without having to acquire knowledge other than previous experience with web surfing.

Examples of external consistency on the web page is the location of the logo (top left), the concept of navigation tabs, and the coloration and underlining of hyperlinks.



Figure 6.3: Example of external use of tabs (From www.Amazon.co.uk)

**Visual Identity** Somewhere in between internal and external consistency lies the visual identity of the website. In this project the visual identity is made to resemble the physical advertising material used by Duelco. The website's color-scheme match these and the use of logo and general visual expression is meant to match company's current material.

### Mapping

The mental model for the navigation elements is tabs or index cards. This is a model used in many operative systems and allows to present different choices and at the same time clearly identify which choice currently active.

Furthermore the tab based interface is an interface-abstraction used by many e-commerce sites (e.g. Amazon.com ) and websites in general and many users will be comfortable with such an interface model.

### Feedback

When a user clicks on a navigation tab the visual representation of the tabs changes so that the current category is shown, as seen in Figure 6.4.



Figure 6.4: The difference between a selected tab *forside* and the other tabs.

Also when a product is put in the shopping cart the interface reflects this by incrementing the product count.

## 6.3 Web Specific Considerations

As mentioned earlier in this chapter there are some considerations which are very specific to web applications. These considerations are described in the following sections.

### 6.3.1 Cross Browser Support

In order to develop a solution which is usable in a wide range of browsers some amount of effort had to be put into this during the design phase.

One thing that was done in this area was to use valid markup language and presentation language and at the same time being aware of the different browser support in this area.

Also no functionality should be exclusive to any browser or periphery web technologies such as PDF or client-side scripting language.

Besides the above mentioned guidelines the two main tactics used to assure interoperability is “*graceful degradation*” and “*standard compliance*”.

### 6.3.2 Graceful Degradation

An approach to creating pages that will work across several browsers is to design for the lowest common denominator. This approach does however not fit our goals about using semantically rich XHTML and CSS and it furthermore excludes several technologies that could be used to heighten the user experience of the pages.

Instead the pages must be designed so that the elements used will degrade gracefully so that a browser without support for any particular technology will fall back to an equivalent representation of the content. An example of this would be adding a textual description on all “img” elements so that a non-graphical browser can use this textual description.

### 6.3.3 Standard Compliance

By complying to the technical specifications for XHTML 1.0 and CSS 2.0 issued by W3C the pages generated will not rely on browser specific error handling when parsing the XHTML source. Then the only variance will be induced by different browsers abilities to present the delivered content.

Furthermore to support more specialized user agents (web crawlers or specialized browsers) semantically rich markup should be used.

An example of semantically rich markup could be marking up a menu, witch in fact is a list of possibilities, as an unordered list using the “ul” tag in XHTML. This unordered list can then be styled using CSS in order to archive the desired visual appearance. Likewise headlines paragraphs and addresses should be presented using the appropriate XHTML tags (“h1”-“h6”, “p” and “address”).

# IMPLEMENTATION

---

This chapter covers the implementation of the system. This will be done through a description of the overall implementation decisions, followed by a description of how each tier is implemented. Changes from the design will be discussed in the individual tiers if changes are made.

## 7.1 Overall Implementation Decisions

It was decided to divide the group in subgroups, that each would develop one of the tiers. As there are five tiers and five group members this means, that each member initially had a tier to work on developing. As some of the tiers are closely tied, and as some of the tiers have less complexity than others, members would often be working together though.

The reason that this parallel implementation was possible was, that the interfaces between the tiers had already been established by creating a number of schemas, that the data sent between tiers should conform to. However, a problem was, that the n-tier architecture is highly hierarchical. This means that the higher tiers are difficult to implement, as they depend on the lower tiers. Because of this, a number of dummy documents and code stubs was created to simulate the lower tiers while these were implemented. This is connected with another decision during the implementation; that testing should be done as soon as possible. This is discussed further in Chapter 8.

### 7.1.1 Tier Installation

The tiers of the system are to be implemented as web services. Combined with the n-tier architecture this makes it possible to have each tier installed on its own server. However only two computers was available during the project. Simulating different servers were still possible though, by setting several websites up on the same server,

but using different ports. Table 7.1 shows how the different tiers are installed. The Server Presentation Tier is installed on port 80 as this is the tier that should respond to user requests.

Tier Name	Location	Port
Server Presentation	Server 1	80
Business Logic	Server 1	81
Data Access	Server 2	80
Data	Server 2	-

Table 7.1: Tier Installation

### 7.1.2 Error Handling

In the developed architecture all the tiers are dependent on each other. Furthermore it is not guaranteed that the tiers are developed in the same framework, and therefore the error handling has to be done in the tiers, and not the web service methods. Due to this the error handling can be done differently in each tier. Because of that a standard way of handling error is used.

The way this is done is by creating an error XML Schema. This should contain the following information:

**Tier** The tier in which the error occurred.

**Description** A description of the error which should be showed to the user of the system.

**Message** An error message used for debugging.

The XML schema can be seen in Figure 7.1.

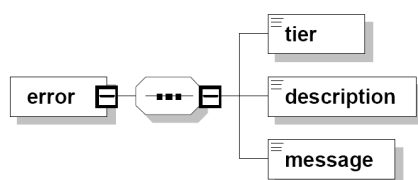


Figure 7.1: XML schema used to report errors.

This error XML document should then propagate up the tiers. E.g. if an error occurred in the Business Logic Tier it should be sent to the Server Presentation Tier and the Client Presentation Tier.

### 7.1.3 Programming Language

When choosing programming language the choice stood between either Java 2 Enterprise Edition or Microsoft .NET. Here it was chosen to base the system on the .NET platform. This was based on the fact that .NET includes the necessary libraries for handling XML, web services, and XSLT. It was also due to prior knowledge of the Visual Studio environment. C# was chosen as the implementation language.

## 7.2 Server Presentation Tier

The following documents the implementation of the Server Presentation Tier. The Client Presentation Tier is not described, because it is already implemented in the form of various web browsers by third party. The only client based logic in this project is JavaScript, which was never implemented.

### 7.2.1 URL Rewriting

Websites in IIS is by default set up to only respond to requests that are directed to an existing file on the server. However, as detailed in the design of the Business Logic Tier in Section 5.2.3, the website must respond to any URL. To allow for this, three things must be done: an HTTP handler must be created, the `web.config` file must be changed, and the ISAPI filter must be adjusted. This is described in the following.

**HttpHandler** First of all a class called `HttpHandler` has to be created, that can redirect requests to the Business Logic Tier. The class doing this can be seen in Listing 7.1. The `GetHandler` method is called whenever the `HttpHandler` receives a request. The request should always be redirected to the url `/Default.aspx`, as seen in line 6. In line 7-8 the path is rewritten, and the old URL is prepended as a variable. The path to the file `Default.aspx` on the hard drive is found in line 9, and finally an actual handler for the new request is returned in line 10-11.

The idea of the `ReleaseHandler` method is that the handlers can be reused. This is supposed to do a good deal for the efficiency of the system, but as this has low priority, no more has been done here.

Listing 7.1: `HttpHandler` code for Business Logic Tier

```
1 public class DuelcoHTTPHandlerFactory : IHttpHandlerFactory
2 {
3     public virtual IHttpHandler GetHandler(HttpContext context,
4         string requestType, string url, string pathTranslated)
5     {
6         string sendToUrl = "/Default.aspx";
7         context.RewritePath("/Default.aspx", String.Empty,
8             context.Request.QueryString + "&oldurl=" + url);
9         string filePath = context.Server.MapPath(sendToUrl);
```

```

10     return PageParser.GetCompiledPageInstance(sendToUrl,
11        filePath, context);
12 }
13
14 public virtual void ReleaseHandler(IHttpHandler handler)
15 {
16 }
17 }

```

**Web.config** The handler created above can handle requests to any URL, but the system still has to be set up to use the handler for requests. This is done by altering the `web.config` file. The lines seen in Listing 7.2 has been added. Lines 4-6 tells IIS, that all request types must be directed to the `HttpHandler` created above. Lines 2-3 make sure, that requests to the static file types used are not redirected, because these should be fetched from the local file system.

Listing 7.2: Addition to `web.config`

```

1 <httpHandlers>
2   <add verb="*" path="*.css,*.gif,*.jpg,*.png,*.js,*.pdf"
3     type="System.Web.StaticFileHandler"/>
4   <add verb="*" path="*"
5     type="Duelco.ServerPresentationTier.DuelcoHTTPHandlerFactory,
6     ServerPresentationTier" />
7 </httpHandlers>

```

**ISAPI Filter** Finally, a problem that occurred was, that requests to URLs such as `http://duelco.dk/forside/` was seen by IIS as a request for browsing a directory, and thus was rejected before reaching the `HttpHandler`. This was fixed by adding an extension mapping to the application. This is done in the IIS configuration. The mapping is set up to use a wildcard as extension, and thus maps any request to to the `aspnet_isapi.dll` without testing whether the file exists. In practice this means, that any requests to the server are forwarded to the `HttpHandler` above.

## 7.2.2 Sessions

Sessions are supported in .NET web applications either by URL rewriting or by cookies. URL rewriting makes the URLs look somewhat strange, while cookies need to be supported by the client to make that work. As the standard is cookies this is used, as no important difference was found between the two models.

The system actually has two sessions per user in the implementation. The .NET standard session between the client and the Server Presentation Tier, and the system specific session in the Business Logic Tier. Mapping between these are easy, by making sure they share the same id. The Server Presentation Tier is responsible for telling the Business Logic Tier when a session is to be closed, because of timeout of shutdown of

browser. When this occurs the web application invokes the method `Session_End`, in the `Global` class of a web application. Sending the session id down to the Business Logic Tier is easily done then.

## 7.3 Business Logic Tier

This section will discuss some issues that arose while implementing the Business Logic Tier. The focus here will be on how some of the quality attributes were promoted using refactoring during the implementation.

### 7.3.1 Testability and Maintainability

The Business Logic Tier functionality consists, as seen in the design, of a single interface method called `httprequest`. However this method branches into several methods each responsible of creating part of the document seen in Section 4.3 on page 25. E.g. a method was made, that created the meta data part of a document, and another, that created the content part of a document, by interpreting the URL from the request.

A problem during implementation was, that it was hard to maintain decoupling. E.g. the meta data part of a document contains a field denoting the type of the content. But the type of the content can only be determined by interpreting the URL. So either the method creating the content had to create part of the meta data too. This would make it very hard to uphold the implementation strategy of doing unit testing while coding, as no method created a complete document fragment, that could be validated.

Another solution would be to do several interpretations of the URL, which would make the maintainability lower, as the same code would be present several places.

Another problem was, that the methods were created to insert their information into hard coded positions in the document. I.e. the meta data method would insert its information into the XPath position `/response/metadata`. Again an issue of maintainability arises, as this makes it harder to change the position of information.

Finally the `XMLDocument` class in the .NET framework has some undesirable properties. E.g. a namespace had to be provided each time a new node was created, instead of just setting a default namespace, to be used until changed.

**The Navigator Class** All these problems led to a refactoring of the Business Logic Tier, where a class called `Navigator` was created. This class is passed down by each method invocation, and includes the following functionality:

**Default Position** A position stack is included. Methods are provided by the navigator for importing notes and for XPath navigation and these will automatically use the position stack as a base point. So by pushing a position setting before calling the method responsible for creating meta data, it can be controlled where the

meta data is inserted. The position can afterwards be popped and the previous position is active again.

**Namespaces** An `XmlNamespaceManager` from the .NET framework is included. The default namespace of this manager is automatically used for creation of nodes. Also short prefixes for all Duelco namespaces are automatically set, and these can be used for easy XPath navigation.

**URL Position** Methods are responsible for interpreting small parts of the URL. Whenever a method has interpreted a part of the URL it can increment the position denoting which part that needs to be interpreted next. E.g. method might be responsible for interpreting the URL fragment `/produkter/...`. Another method might be responsible for interpreting the URL fragment `/webshop/...`. It can then pass on the interpretation of the URL `/webshop/produkter/...` to the first method by incrementing the URL position.

**Settings** The navigator also includes a hash table that can be used to remember information gathered for later use. E.g. the method, that does URL interpretation can set a content type, and the method, that creates meta data can insert this content type into the document. This allows for easier testability.

Besides this the navigator also holds any form data received through the HTTP request, and the session for the user making the request.

### 7.3.2 Configurability

As this is rated important, a number of decisions was made to further this during the implementation of the Business Logic Tier. The idea is to put as much data into external files as possible. Most of the pages of the system will have a corresponding XML file, that defines the static parts of the page. Also a file called `config.xml` provides more general information, that might be used several places, such as company phone number and email.

The format of these files are easily understandable, and thus even less experienced web masters should be able to change settings.

Finally some of the functionality is also placed in external files. Transforming data is done by XSLT, and the XSLT files are external files too. These are not of such an easily understandable format though, and changing them might make the resulting documents unusable by the other tiers.

## 7.4 Data and Data Access Tiers

The following describes the implementation of the Data Access Tier, and problems which interfered with the implementation phase.

### 7.4.1 Problems

A problem occurred with obtaining the Navision license, and as a result XML Ports and Code Units were not available. This made interfacing with Navision through the developed system impossible. This problem is further described in the reflection in Chapter 9 on page 59.

### 7.4.2 Alternative Solution

Different solutions were considered during this phase of the project. Different things had to be taken into consideration. The first thing to consider was the time remaining in the project, but since time had been allocated for implementing the Navision solution, this was not a problem. The following solutions were considered:

#### Alternative ERP System

An alternative solution could be to examine other ERP systems instead of Navision. This would have the feature of being a full working solution with all the aspects intended in the beginning of the project. It did however have some disadvantages:

1. Time was only allocated to interface with Navision, not to set up a system from scratch. This would have been a requirement for this solution to work. This would then have been out of the scope of the project.
2. The reason that Navision was chosen was that the company wanted to maintain the Navision part of their current running system, and getting rid of Lotus Notes. Because of that setting up an alternative ERP system from scratch would not benefit the user in any way.

Based on these those arguments it was chosen that using an alternative ERP system was not a good solution.

#### A Database Solution

Another alternative was to implement a solution which could be used to show the company a working solution of the web shop by storing the data otherwise than in an ERP system. Here a database solution was a possibility. The advantages here were that it would be quite easy to implement. Furthermore it would only be necessary to store the information needed for the web shop. It would be possible to store all the information obtained through the customer, but this would not be used in any way.

The database solution was chosen as it was a rather simple solution, and it was considered a waste doing a complex solution which would have to be rewritten if the system was to be delivered to the company. Furthermore it still contained the information

necessary to have a system which could be demonstrated. To make the solution more realistic the product and customer data used was that which was supplied by Duelco.

It was chosen not to put a lot of effort in implementing the Data Access Tier and the Data Tier as this solution was mainly used to show a working result of the tiers above. Because of this refactoring has not been used in the Data Access Tier, and things such as thread safety is not guaranteed. In the Data Tier there is no guarantee for atomic transactions or referential integrity.

### 7.4.3 Database

The system was implemented using the .NET framework, and as this has build in support for Microsoft SQL Server 2000 through ADO.NET, this was chosen as the database server. Other database servers could also be used, but as it was easiest done with the server mentioned, this was chosen.

The design of the database was based on the ER diagrams from the data design in Section 5.3.3 on page 34. The only thing added is the attributes which hold information necessary for the web shop. The communication between the Data Tier (the database server) and the Data Access Tier is as mentioned done using ADO.NET and SQL Queries. This could also have been done using stored procedures in the database. This would have made a more Navision like solution, but as both the Data Tier and the Data Access Tier would have to be exchanged anyway, the solution requiring the least amount of effort was chosen. Had performance been a design goal, stored procedures would also have been a faster solution.

### 7.4.4 Data Flow

The way the communication between the two tiers was implemented was as mentioned using ADO.NET. When the XML data received is a request for data the information flows as follows (Words written in `typewriter` represents .NET classes):

1. The data received from the Business Logic Tier is transformed into an `XMLDocument`.
2. This is then used to create the appropriate `SQLCommand`.
3. This is send to the database which return with an answer to a query represented as a `DataSet`.
4. Finally the `DataSet` is transformed into an `XMLDocument` which corresponds to the appropriate `XMLSchema`. This is then passed to the Business Logic Tier.

When data is being sent to the database the information flows in a similar way. The only difference is that no information is returned from the database besides a message indicating whether the query was successful. Because of that there is no reason for a `DataSet`, and the `XMLDocument` created only contains either an error or a successful message. An overview of the information flow can be found in Figure 7.2.

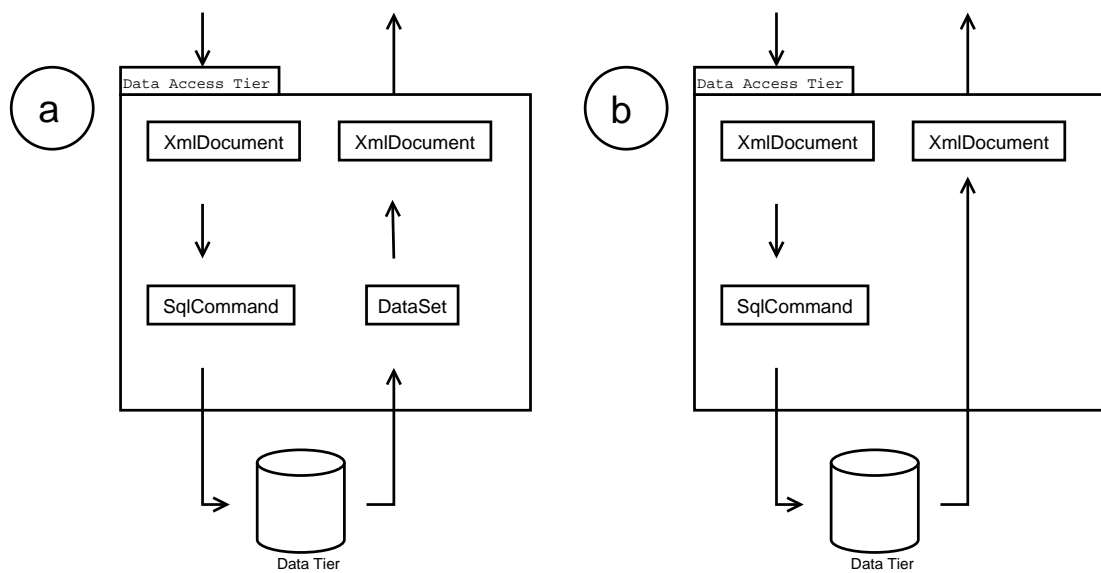


Figure 7.2: Flow of data in the two lower tiers. Figure a shows the flow when extracting information. Figure b shows the flow when adding data.



To ensure the system functions as expected a number of different tests were planned. During the development of the various tiers unit tests were employed to ensure functions were doing as expected. Integration tests were employed when combining two or more tiers to ensure the interfaces were followed. To see if the user interface was easy to use for realistic users of the system usability tests were planned.

## **8.1 Testing Strategy**

This section will discuss the strategies for unit and integration testing. The general strategy was to do testing as soon as possible during the development. However not all tiers were equally testable. The Server Presentation Tier has the responsibility of transforming a response XML document into an XHTML document and thus only consists of a single method. The Client Presentation Tier mainly consists of style sheets. Testing here is done by validating the XHTML documents, and by usability testing. The lower three tiers were however easily testable, though as mentioned in Section 7.3.1 on page 47 a refactoring was done to make the Business Logic Tier more testable.

### **8.1.1 Unit Testing**

Unit testing of the system was done as the system was developed. Whenever a new method was written a test for this method was written as well. Besides making sure the functionality is correct, this also makes refactoring easier.

However the tier architecture made it hard to do unit testing of all methods, as most methods in the higher tiers depends on methods in the lower tiers. This resulted in the creation of stubs, to make testing easier. In particular a stub simulating the Data Access Tier and Data Tier tier was created to return dummy data during the development of

the Business Logic Tier.

### 8.1.2 Integration Test

Integration testing within tiers was done using a bottom-up strategy. The final integration testing between tiers was also done using bottom-up strategy, but as mentioned in the previous section, stubs were used for testing during the development.

### 8.1.3 NUnit

The tool used for unit and integration testing was NUnit [11]. NUnit is ported from JUnit and is able to test code written in .NET. The behavior is controlled by special attributes that can be applied to methods and classes. When a `.dll` file is opened it will automatically find the relevant methods and run them in the right order specified by the attributes.

## 8.2 Testing Tactics

This section will discuss the tactics used for testing. Generally testing tactics fall into two categories: white-box and black-box testing. Both these methods has been used.

White-box testing techniques makes sure that all possible paths through a unit of code is run at least once ensuring code coverage. Technical methods for doing this exists, such as doing graphical analysis of methods. However no methods was found that was complicated enough to warrant an actual formal analysis to find the possible paths through it. In all cases it was rather obvious what inputs to use to make sure all paths was run once.

Black-box testing techniques tests the functional requirements of a unit of code or module. This can e.g. uncover errors in the behavior or the interface of a unit of code. As most methods return an XML document fragment of some sort, most methods can be black-box tested by making a request and validating the returned document against the appropriate XML Schema. Usually some selected fields are also probed.

## 8.3 Usability Test

In this section a proposed usability test will be presented. The actual test was not performed due to time issues and the fact that it was not possible to acquire the necessary test subjects from the user group.

It was however possible to do the initial planning and test design which is documented in the following sections.

## **Test Plan**

The following section contains the plan devised to support the usability testing of the developed system. The test is based on the method described in [12]. The test is not a full usability evaluation as the system is not fully implemented and there is a lot of textual content which is not present in the system.

### **Purpose**

The purpose of the usability evaluation is to identify and classify usability problems in the developed system. Such information can be used in the future development of the system to guide development resources onto the areas of the user interfaces that needs work and as a way of tracking the result of the resources put into the analysis and design of the user interface.

### **Problem Statement/Test Objectives**

The test should determine:

- If the new information hierarchy reflects the end users conceptual model in such a way that the users are able to use the navigation elements in an efficient manner.
- Test if the end users is able to find specific products in the web shop and is able to ordering such products without assistance.
- Test if users are able to use the advanced features in the web shop, in particular retrieving order history.

### **User Selection**

The users used for testing should consist of both existing end users and potential end users not currently familiar with the Duelco web shop. All test subjects should be inside the target audience for the system since the web shop contains a lot of domain specific information and relies on tacit knowledge of the user .

Due to time limitations concerning the usability facilities (one day) and the estimated test-duration is about one hour for each test subject. A set of six users was decided as the number of users that should be used.

### **Test environment/equipment**

The test environment is the AAU-usability lab. This lab is equipped with several video cameras in order to document the actual tests.

The test setup should consist of an office-like environment with a standard PC equipped with Windows XP. The only modification to the test machine is that a modification to the host file should be made in order to redirect all traffic to `duelco.dk` to the server containing the new system.

### Test Monitor Role

The test monitor's role is to brief the test subjects and to guide the user regarding test-specific issues. The monitor should not help the test subject with the actual tasks unless the subject is unable to complete a task with help. In such cases, the test monitor should restrict the help to the absolute minimum.

The test monitor is responsible for making the user *think aloud* in order to identify the user's conceptual understanding of the system and to evaluate user experience.

### Evaluation Measures

All test sessions should be recorded for later analysis, and a brief interview with each test subject should be done as debriefing after each test session.

The interview serves as a way of determining the user's overall impression of the system and should only be a few minutes. The interview should be informal and performed outside the test laboratory.

During this interview, the subject should be allowed to speak freely and to comment on any part of the system or the testing procedure itself.

It is important that the test subject is thanked for his/her participation during this interview.

### Method

The actual test execution consists of four phases for each test subject.

- Test session preparation
- Briefing
- Test execution
- Debriefing

**Briefing** The briefing should be held by the test monitor and must be used to introduce the goal of the test and the testing procedure.

Special care should be put into explaining the *thinking aloud* and that the session is video taped for further analysis.

**Test Execution** The test execution is guided by the test monitor and consists of a the user going through a task list. During the test session the test monitor should restrain himself from helping the user unless the user is stuck. A further description of the test monitor role can be found in the later sections.

**Debriefing** In this phase the test subject should be thanked for his/her participation. Furthermore the subject should be able to present criticism if any.

#### **Task List**

In order to assure that all test subjects are given exactly the same task set a list of the tasks should be given to the test subject in written form.

The test subject should read the task aloud and try to execute the task as he/she finds best.

The task list should consist of 10-15 tasks testing the ability to:

- Find the website for Duelco.
- Find Duelco's telephone number.
- Finding and describing the procedure for obtaining a login to the web shop.
- Log in to the web shop using a provided username and password.
- Finding a specific product.
- Creating an order.
- Creating an order consisting of several products.
- Log out from the web shop.
- Find Duelco's key values.
- Find the details from earlier orders in the system.



# REFLECTIONS

---

In this chapter different aspects of the project are reflected upon. These are: *project goals* and *tools and methods* used during the project.

## 9.1 Project Process

In the initial phase of the project, different tasks and milestones were identified and combined in a time schedule. This helped tracking the project progress, and was useful for planning the different phases of the project.

As part of the time schedule, a limit was set for when the correct Microsoft Navision license should be acquired. In retrospect, this limit should have been much earlier in the project to accommodate an alternative plan. Nevertheless, the group has gained experience in operating under uncertainty, and issues such as resource management will be assigned higher priority in future projects.

## 9.2 Project Goals

In the initial phase of the project a number of Project Goal were stated. Here these are reflected upon.

### 9.2.1 To Develop a Prototype Web Shop with Focus on Usability

In the beginning of the project the goal was to make a fully functional prototype. This was abandoned due to above mentioned factors (Section 9.1). Instead, the functionality was reduced and focus shifted towards utilization of the XML related technologies.

The web shop was from the beginning created with usability in mind, and hereby the usability was improved in relation to the initial website. Further points of improve-

ments could have been identified through a usability test, but this was not done because the relevant lectures in the usability course “Design, Implementation and Evaluation of User Interfaces” were held very late in the project.

In contrast to the current solution, the developed prototype is designed with interfacing with Navision in mind and is based on open well-documented standards. It is therefore easier to adapt to Duelco’s ERP system.

### **9.2.2 To Base the Web Shop on a Scalable N-tier Architecture Easy to Interface With**

Using an 5-tier architecture turned out to be quite useful as the project progressed. As the goals changed from interfacing with Navision to interfacing with a temporary back end, it was only necessary to change the Data Access Tier. This proves the flexibility of the chosen system architecture.

The interfaces between the different tiers were based on XML and web services, rather than on proprietary interfaces specific to an environment or programming language. The use of web services turned out to be easy due to the built-in support in Microsoft Visual Studio .NET 2003. Still it would also be easy to interface with the tiers using another platform as web services is a widely supported technology.

### **9.2.3 To Interface with the ERP System Microsoft Navision**

At the semester start Navision 4.0 was provided by the university. However early in the project it was clear that the license did not permit access to the features needed in order to interface with the Navision system and to configure the business process needed in the Duelco web shop. This had the result that access to the test data from Duelco was inaccessible in some areas, since it contained business process elements not covered by the license.

Despite of getting new licenses twice it was not possible to obtain a license allowing interfacing using XML Ports and custom Code Units. This lack of access to the parts of the ERP system needed to fully implement the suggested solution resulted in a system containing a back-end which is only for demonstration purposes. This is far from the ambition of the group at the semester start but given the circumstances this was the only possible solution.

### **9.2.4 To Maintain Standards and Use Best-Practice Techniques Regarding Web Presentation**

The user interface of the system was from the beginning based on web standards, in particular XHTML 1.0 and CSS 2.0. Furthermore best-practice techniques were also used. This presented an advantage in relation to getting the same visual appearance in

different browser, and it enabled testing the output with XHTML validator.

### **9.2.5 To Explore and Evaluate XML and Related Technologies**

One of the main objectives of this project was to examine XML and related technologies. This was based on the fact that XML has become the Internet defacto standard for data exchange, but also because the project had to incorporate the PE-course, ‘Internet Technologist’<sup>t</sup>. In this course a lot of different XML related technologies were presented. In the following the individual XML goals are reflected upon.

#### **XML as a Tree Data Structure**

Two ways of modeling XML documents in the .NET framework are as DOM’s or as strings. Using DOM’s is of course the best since this allows easy searching using XPath and ensures well-formedness. The use of the .NET DOM was however not as intuitive as could have been hoped. Often it seemed that much code had to be written to do only a little work. This is probably due to the fact, that the .NET DOM is optimized for performance during search more than for actual construction of documents. However it seemed somewhat strange to the group, that namespace managers and documents were not integrated very well. During the implementation this was, to some extent, solved by making a wrapper class for documents and namespace managers.

#### **Using XML Web Services**

The purpose of using XML web services was to make a programming language independent tier interface. In relation to this it was an advantage to use Microsoft Visual Studio .NET 2003 as it has a built-in support for web services.

#### **Using XSLT**

The different tiers in the system communicates by interchanging XML documents. Ultimately the XML document from the lowest tier has to be changed into an XHTML document. XSLT provided a way of changing one XML document to another. An alternative way of doing this would have been to construct the target XML document using the .NET DOM, but this would have been a difficult process. XSLT did this in a nice and relatively easy way. This was made even easier by using tools as described in Section 9.4.1 on page 63.

#### **Interface Design Using XML Schemas**

The intention of using XML schemas to define the interfaces between the tiers, was that it was decided to use XML documents as a way of communicating data between

the tiers. It did however also turn out to be an advantage when testing and debugging. Here it was always possible to figure out in which interface the error occurred.

Many methods of the system was responsible for creating a certain type of XML document, and testing the validity of these is very easy if there is a schema.

### **Working with XML in General**

In general the group felt that using XML have been an advantage, as experience with XML was gained, and as working with XML help make a very flexible architecture.

The main problem with using XML is somewhat cumbersome construction. Also errors in traversals of the tree and transformation are always run-time.

Another way of implementing the internal functionality of e.g. the Business Logic Tier, that wasn't considered during the implementation, would be to use XML serializable classes. This would change many errors from run-time to compile-time. It would also add intellisense support in Visual Studio, and probably also eliminate many of the problems of coupling encountered during implementation, since separating the construction and the insertion of content is trivial using classes instead of documents.

## **9.3 Software Engineering Methods**

The three software engineering methods used in the project where OOAD, spiral model and pair programming. These have some good and some bad qualities which are evaluated here.

### **9.3.1 OOAD Method**

The main development method used during this project is the OOAD method, this method was the basis of the analysis and design phases of the development. The model was used as a guideline, and was especially relevant for identifying the problem area, actors, and the associated use patters.

Using this method provided an organized and systematical approach to the development. The OOAD method was considered to be very useful when only using relevant parts.

### **9.3.2 Spiral Model**

During the development a tight spiral development process was used [13].

This method was used in order to handle the great level of uncertainty regarding the realization of the design.

### 9.3.3 Pair Programming

During some of the more complicated parts of the implementation pair programming was utilized. This helped catch errors much earlier and helped to determine whether a method will work or not. Furthermore it helps the programmers write better code, as two people has to understand the code.

The downside to using pair programming was that it was more time consuming. This might however be compensated for as the code produced has better quality.

## 9.4 Tools

During the development of the proposed system several tools were used.

### 9.4.1 Software Tools

A set of software tools were used during the project. The reflections regarding these can be found in the following section.

#### Microsoft Visual Studio .NET 2003

Microsoft Visual Studio 2003 Professional Edition was an excellent IDE for working with C#, and solutions.

One problem with Visual Studio was deployment of web services that were created on other computers. The problem was that the web service needed to connect to the web server and if Visual Studio could not connect it refused to open the solution as a whole. A simple solution to this would have been to open the .cs files without using the solution, but then Visual Studio's advanced features such as intellisense and syntax highlighting are not in effect.

The problem was however solved by sharing the folders as *Web Folders* and granting the *ASPNET* user privileges to this folder.

In all the group found that working with Visual Studio was a positive experience, though web service handling could have been made easier.

#### XML Specific Tools

Supporting the creation of XML Schemas and XSL-Transformations the tools Altova MapForce and Altova XMLSpy were used [14]. These tools allowed the creation of XML Schemas and transformations more efficient than the built-in XML support in the IDE.

MapForce was used to create XSLT. This was done by mapping one XML schema to another. This demanded that the schemas were created, but as they were defined in the design, this was not a problem. It was the groups opinion that Altova shortened the time used to create these transformations.

XMLSpy was used to create the XML Schemas, and to create figures of these for the report. It differed among the group members whether they liked using XMLSpy for creation of XML Schemas. Some liked it a lot, and some would rather work on the textual representation.

### **Version Control System**

Subversion (SVN) was used as a version control system [15]. This choice was made since Subversion supports functions such as atomic commit operations and file moving and renaming in a much easier way than other version control systems (e.g. CVS). The use of Subversion proved to be quite useful although some problems were encountered initially.

These problems were caused by attempting to move and rename directories without using the built in support in SVN and in some cases this caused the SVN repository to lock up.

**SmartSVN** As a helper application in order to ease the use of the SVN version control system the graphical interface SmartSVN was used [16]. This program allowed the manipulation of the repository, commits and checkouts to be done using a standard directory tree view. This made the use of SVN even easier.

However the ease of use of this application was somewhat counteracted by stability issues and poor quality of error messages displayed when an operation failed.

### **Unit Testing Tool**

As a unit testing tool the N-Unit framework was used [11]. This method of unit testing caused some problems since references to external files would not work properly when the test were run due to the fact that test and the web service is run from different locations.

The N-Unit application itself proved to be somewhat unstable and crashed repeatedly during test executions but this was a minor issue since the errors was not persistent and a restart of the N-unit application typically solved this.

## 9.4.2 Server Software

### Operating System

Two servers were used for the system. One running the top three tiers, and another running the two lower.

The operating system chosen for the top three tiers was Windows 2003 Server. This server was initially intended to run all the web services, but as the project progressed it turned out that the server hardware was quite slow. Because of this it was chosen to run the two lowest tiers on one of the group room computers. As this computer was also being used as a workstation it was chosen to keep this server running Windows XP Professional rather than installing Windows 2003 server. Compared to Windows 2003 server Windows XP Professional it is only possible to run a single web server using Internet Information Services, but as only the Data Access Tier had a web service interface, this was not a problem.

Using Windows 2003 Server turned out to be quite intuitive as the user interface was very similar to that of Windows XP which all group members were familiar with.

The everyday work flow concerning server maintenance and configuration was done via the *Remote Desktop* functionality from Windows XP Professional workstations. This allowed a very flexible and yet intuitive access to the server and this method of administration.

### Internet Information Services

The Web Server used throughout the project was Internet Information Server which is bundled with Windows 2003 Server and Windows XP Professional. None of the members of the group had any advanced previous experience with this server software. But after a brief period of adjusting to the graphical management interface and the access control list of the server operating system the software setup caused only minor problems which were easily handled.

## 9.4.3 Server Hardware

The current hardware platform consists of two 700MHz Athlon based desktop computers with 128Mb RAM. This is very close to the absolute minimum requirements for running Windows 2003 Server. And since one of the machines is running several services (Internet Information Services and Microsoft SQL Server 2000) the response time for the system is very high. The hardware system is however sufficient as a demonstration platform.

## 9.5 Future Development

The system is in its current form not a finished product, as there is a number of known issues that needs to be resolved. This section will detail these along side a number of improvements that would be the logical next step after the issues has been resolved.

### 9.5.1 Known Issues

Known issues denotes actual bugs in the system.

#### Remember Form Data

If the user requests to add items to the shopping cart without being logged in, the user is redirected to the login page. In this process the items she requested added is forgotten. This should instead be remembered so it can be automatically added when a login is entered.

### 9.5.2 Future Improvements

Future improvements are not bugs, but instead areas where the system needs more work.

#### Caching

Caching is a feature that can vastly improve the performance of a system, and that is especially the case in an n-tier architecture. Caching in .Net web services is simple. All that needs to be done is to add a `cacheduration` field to the `webmethod` attribute, that takes the number of seconds to cache each result as input. However, some results cannot logically be cached. Other results, that can be cached, might not have any advantage in being cached, if the change of a cache hit is very low. Thus caching is something that needs some work, to get and advantage from it.

#### Cache XSLT Files

Much of the functionality in the system is handled by xslt transformations. Currently the xslt files used are reloaded from the harddrive every time a transformation is needed. A possible improvement would be to cache the xslt files in the system to make the transformations perform better.

### **Lacking Input Validation**

None of the web shops input forms currently validate their input, this means that a user could order e.g. the string "twelve" of one item. The business logic would in response throw an unhandled exception. Instead a page detailing the error should be shown.

### **9.5.3 Improving User Interface Responsiveness**

A way in which the user interface responsiveness would be implemented by asynchronous update of the user interface. This could be done by utilizing AJAX as described in the design chapter.

#### **Asynchronous Update of User Interface**

One way of improving the response time of the user interface is to utilize asynchronous http requests. This is done by a combination of client-side scripting and server-side functionality. This is typically done using XMLHttpRequest (a javascript or activeX object) and by using XML as the data format. This has caused the technique to be referred to as AJAX - Asynchronous JavaScript And XML.

The AJAX technique can be used to improve the interface such that when a user adds an item to the shopping cart, the shopping cart status is updated without a page reload.

This functionality can be implemented by adding functionality to the Server Presentation Tier and Client Presentation Tier. In the Client Presentation Tier javascript must be used to retrieve the updated shopping cart information. And to replace the existing shopping cart information. This can be achieved by letting the Client Presentation Tier access the Server Presentation Tier for the updated information using XMLHttpRequest functionality and replacing the outdated information by DOM manipulations (typically done via the javascript function "loadFragmentIntoElement()" or similar). This would allow the system to update parts of the user interface without a page reload.

One major drawback of this AJAX technique is that fall-back functionality must be provided to browsers without support for the XMLHttpRequest object, and to browsers without or disabled client-side scripting.

However the added feedback and responsiveness of the system could outweigh the drawbacks and a future version of this system could benefit of this technique when adding items to the shopping cart or when the user changes the amount of a product in the cart.



# CONCLUSION

---

In this report the development of a web shop has been documented. The application developed is a new web shop for the company Duelco with emphasis on flexibility and usability.

The new system is based on a thorough analysis of Duelco's currently running system. This is formulated into a set of requirements, and according to these a new system is analyzed. In the design of the system an n-tier architecture was chosen and the interfaces between the tiers were designed to use SOAP web services using data defined by XML Schemas. These decisions, and a choice of using web standards such as CSS and XHTML enforces the requirement of flexibility.

The user interface of the new web shop was created with special emphasis on usability. This was obtained through a comprehensive analysis of the existing user interface, and by using formal design principles. A usability test was also prepared, but not carried out due to time limitations, and inability to obtain realistic test subjects.

This ongoing focus on usability goals throughout the analysis, design and implementation has heightened the level of insight and practical knowledge in the area of usability and in general human computer interaction.

A goal of the project was to explore and examine XML and related technologies. This was done by treating all data as XML, defining the interfaces using XML schemas, using web services as interfaces, and changing documents using XSLT. Furthermore modifications on the XML documents were always done on a tree representation rather than on the text representation. It was found that though using XML caused some problems during implementation, the use of XSLT for data manipulation and XML Schema for interface specification has been a great advantage during the implementation and in order to support further development.

Much effort was put into the architectural design which lead to a loosely coupled 5-tier architecture with strictly separated areas of responsibilities.

This had the consequence that although the system initially was designed to interface with an ERP system, a prototype demonstrating the new system could be created. This

only required redesigning the Data Access Tier and the Data Tier. In a similar manner it is possible to modify the system to interface with Duelco's ERP system by changing the implementation of the Data Access Tier, showing the flexibility of the developed system.

# FLOW DIAGRAMS

---

This appendix contains a number of flow diagrams that was created during the design phase to get an overview of the information flow in the program on different HTTP requests.

All the diagrams start in the left side with a request consisting of a session id, an URL, and possibly some form data. These data will then flow down through the different tiers according to the arrows. The top tier is the Server Presentation Tier, the middle is the Business Logic Tier, and the lowest is the Data Access Tier. To the right the result is returned upwards. The Business Logic Tier always returns a response XML document that holds the information necessary to create a page, and the Server Presentation Tier always translates this to an XHTML document.

## Order

The order request can be seen in Figure A.1. The customers id is retrieved from the session and used to call downwards. An order document is returned. The schema describing the new order can be seen in Figure B.7. The schema for the order returned can be seen in Figure B.8.

## History

The history request can be seen in Figure A.2. Either the entire history can be retrieved or details for a single order. In both cases the customers id is used for retrieval. In the second case the order id is retrieved from the URL. The history schema can be seen in Figure B.4 and the schema for the order document can be seen in Figure B.8.

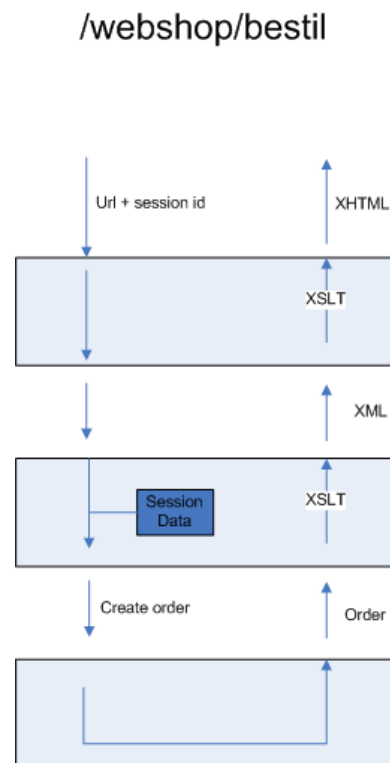


Figure A.1: Flow diagram over the order request.

## Category

The category request can be seen in Figure A.3. This request is rather trivial. The category name is retrieved from the url. The schema for the document returned can be seen in Figure B.2.

## Categories

The categories request can be seen in Figure A.4. The schema for the document returned can be seen in Figure B.3.

## Edit Customer

The edit customer request can be seen in Figure A.5. The customers id is retrieved from the session and used to get the current information. This is then returned upwards. If form data is present the customer has entered new information. If this is valid the Data Access Tier will be told to update the customer entry. The schema for the document returned can be seen in Figure B.6.

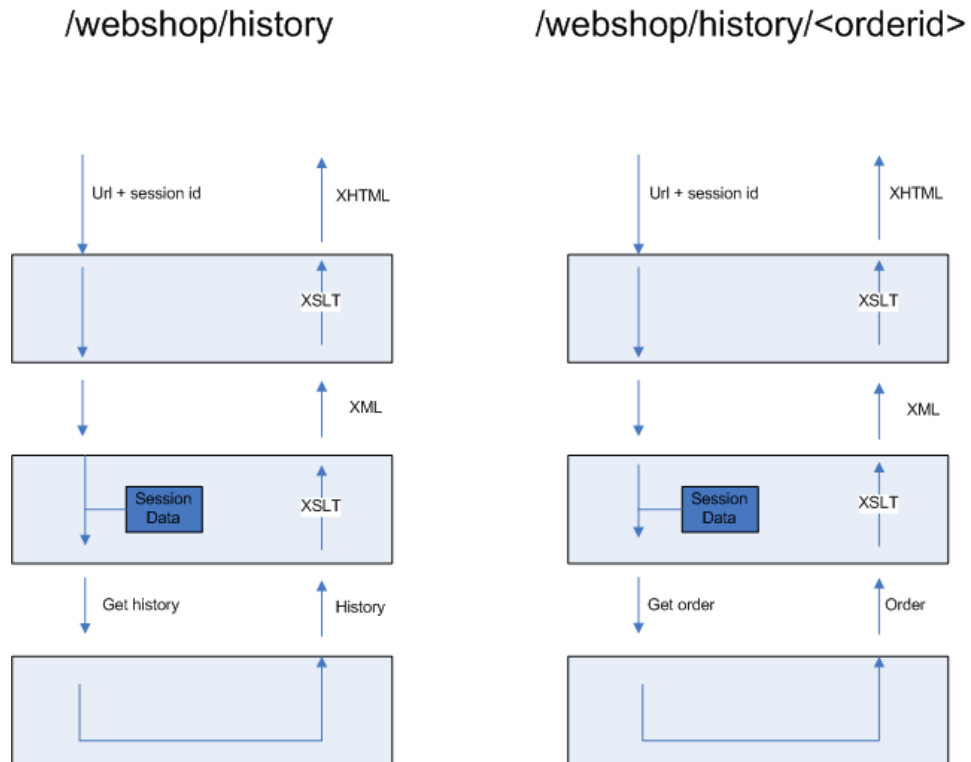


Figure A.2: Flow diagram over the history request.

## Search

The search request can be seen in Figure A.6. If form data is present a query will be sent to the Data Access Tier, and the resulting product list returned upwards. The schema is the same as for sub category request and can be seen in Figure B.11.

## Subcategory

The subcategory request can be seen in Figure A.7. The subcategory and category are retrieved from the URL and the resulting product list from Data Access Tier will be returned upwards. The schema for the document returned is the same as for a search and can be seen in Figure B.11.

## Webshop

The web shop request can be seen in Figure A.8. According to whether the customer is logged in or not either a log in form will be returned or a front page. If form data is present the user has entered log in information. This confirmed by the Data Access Tier and the returned customer document is used to update the session. The schema

/webshop/kategorier/<kategori>

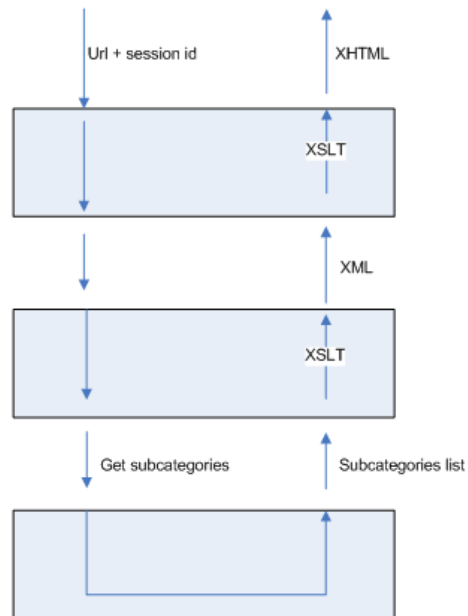


Figure A.3: Flow diagram over the category request.

for the returned customer document can be seen in Figure B.6.

## /webshop/kategorier

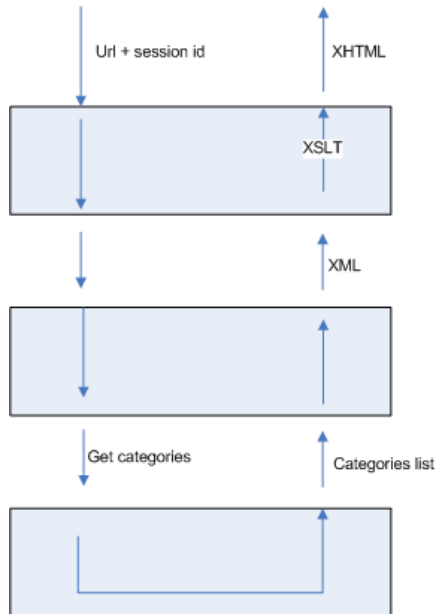


Figure A.4: Flow diagram over the categories request.

## /webshop/bestil

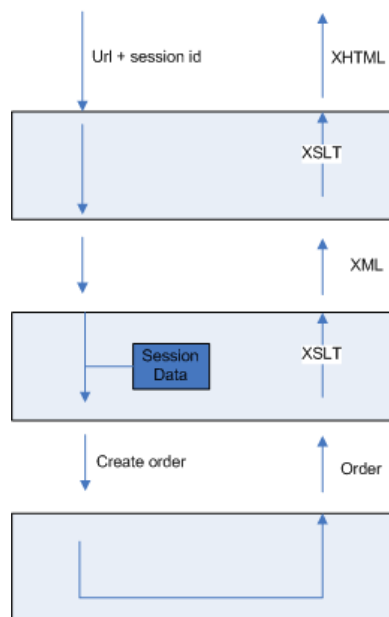


Figure A.5: Flow diagram over the edit customer request.

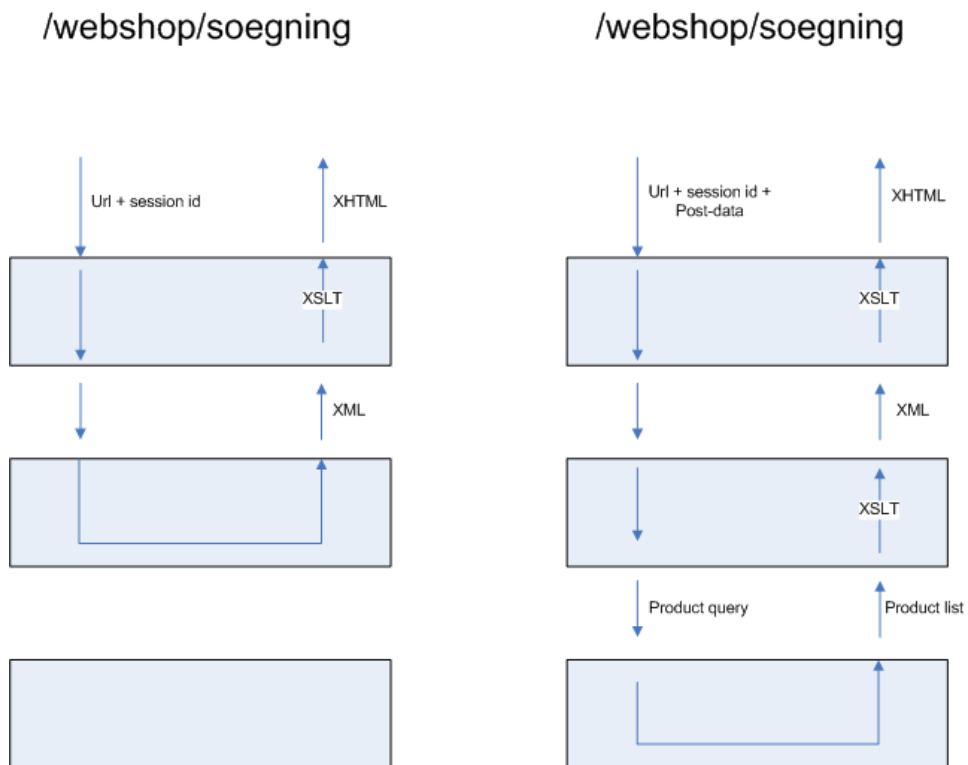


Figure A.6: Flow diagram over the search request.

`/webshop/kategorier/<kategori>/<subkategori>`

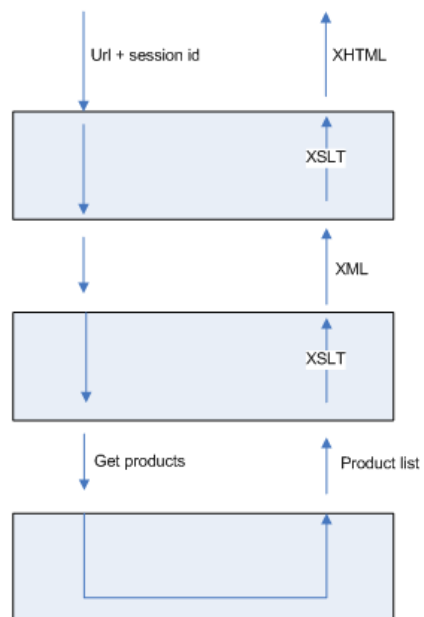


Figure A.7: Flow diagram over the subcategory request.

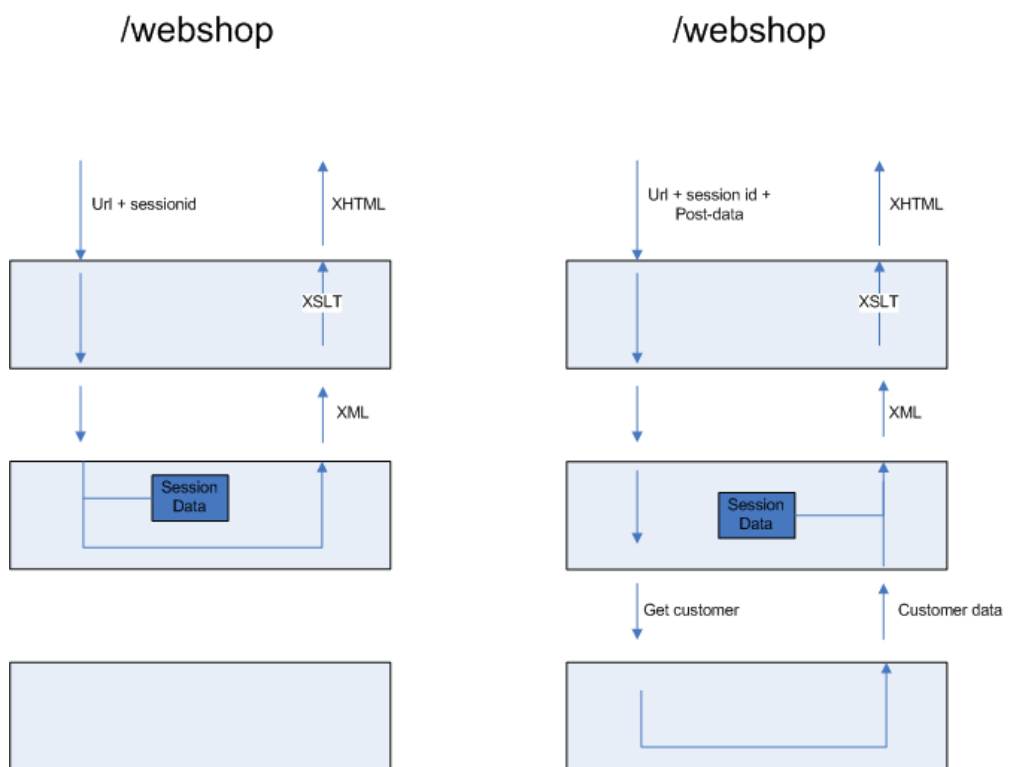


Figure A.8: Flow diagram over the webshop request.



# XML SCHEMAS

---

## Legend

The XML Schemas in the following sections are mostly self explanatory, except for the branching shown on Figure B.1.



Figure B.1: The branching options in XML Schema (left to right) *All*, *Choice* and *Sequence*.

## Data Access to Business Logic



Figure B.2: The Schema for the *category* between the Data Access Tier and Business Logic Tier.

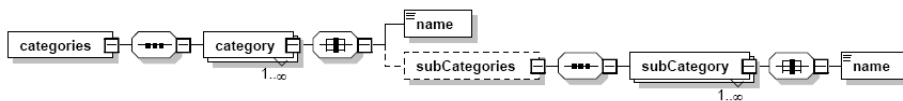


Figure B.3: The Schema for the *categories* between the Data Access Tier and Business Logic Tier.

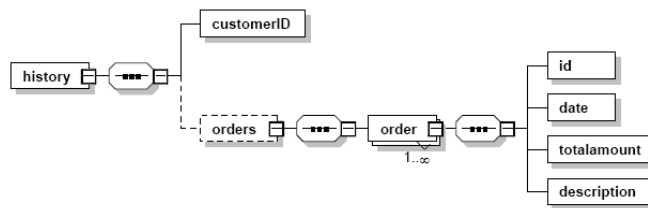


Figure B.4: The Schema for the *history* between the Data Access Tier and Business Logic Tier.

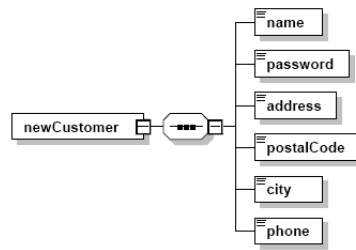


Figure B.5: The Schema for the *newCustomer* between the Data Access Tier and Business Logic Tier.

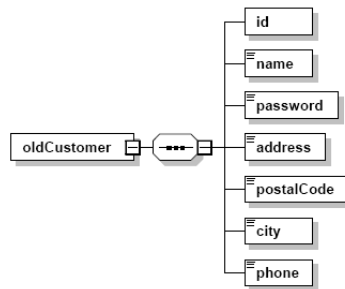


Figure B.6: The Schema for the *oldCustomer* between the Data Access Tier and Business Logic Tier.

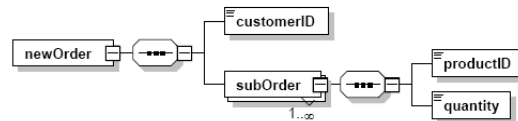


Figure B.7: The Schema for the *newOrder* between the Data Access Tier and Business Logic Tier.

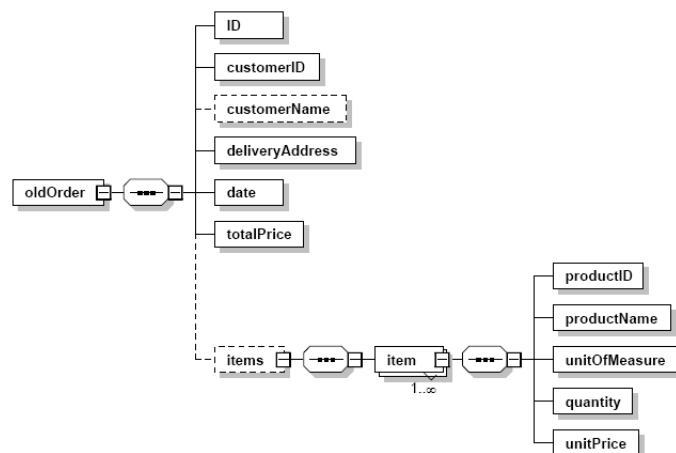


Figure B.8: The Schema for the *oldOrder* between the Data Access Tier and Business Logic Tier.

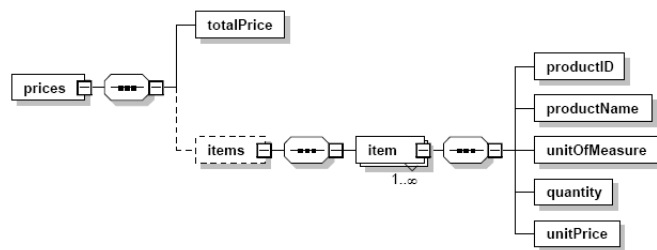


Figure B.9: The Schema for the *prices* between the Data Access Tier and Business Logic Tier.

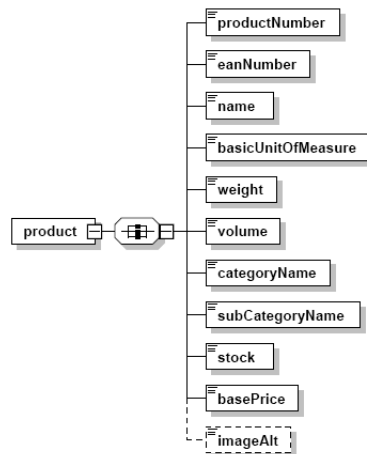


Figure B.10: The Schema for the *newOrder* between the Data Access Tier and Business Logic Tier.

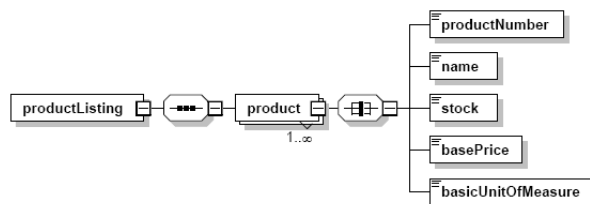


Figure B.11: The Schema for the *productListing* between the Data Access Tier and Business Logic Tier.

## Business Logic to Server Presentation

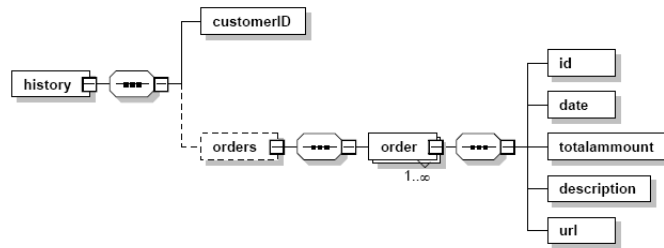


Figure B.12: The Schema for the *history* between the Business Logic Tier and Server Presentation Tier.

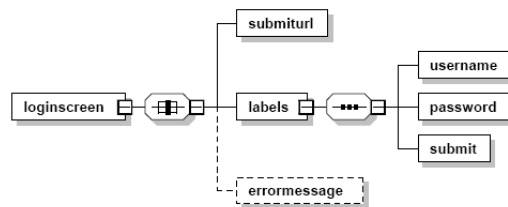


Figure B.13: The Schema for the *loginscreen* between the Business Logic Tier and Server Presentation Tier.

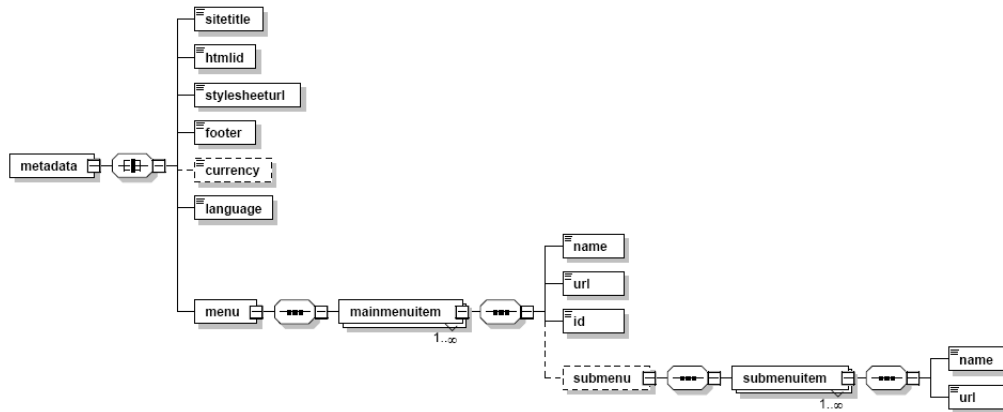


Figure B.14: The Schema for the *metadata* between the Business Logic Tier and Server Presentation Tier.

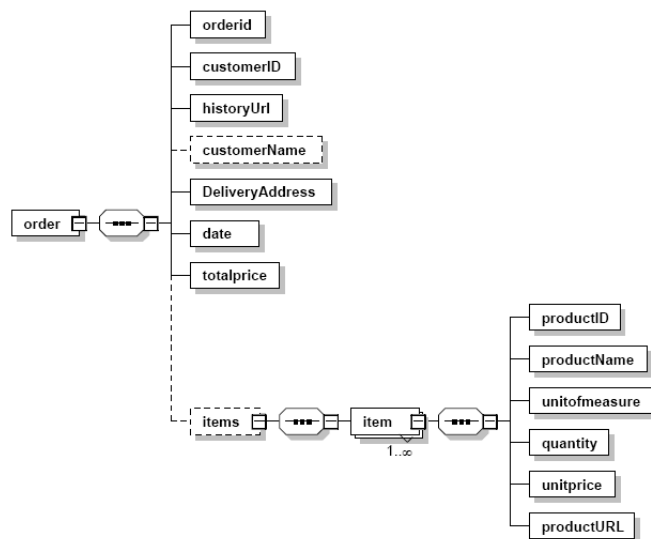


Figure B.15: The Schema for the *order* between the Business Logic Tier and Server Presentation Tier.

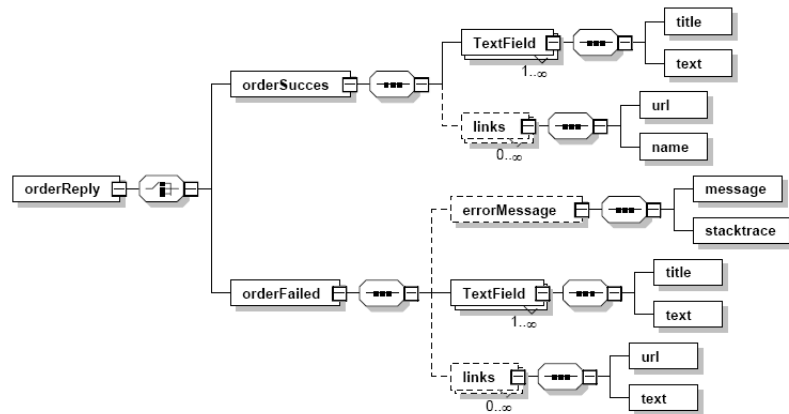


Figure B.16: The Schema for the *orderReply* between the Business Logic Tier and Server Presentation Tier.

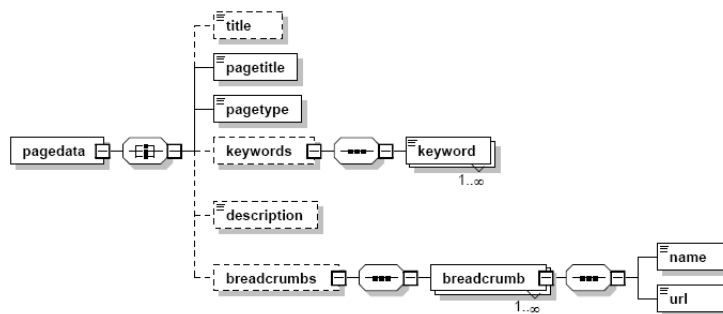


Figure B.17: The Schema for the *pageData* between the Business Logic Tier and Server Presentation Tier.

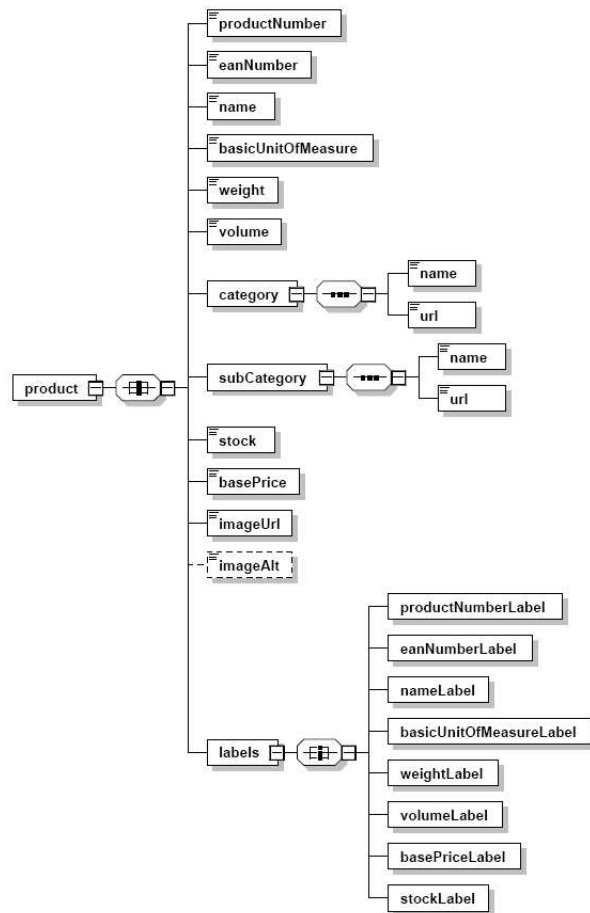


Figure B.18: The Schema for the *product* between the Business Logic Tier and Server Presentation Tier.

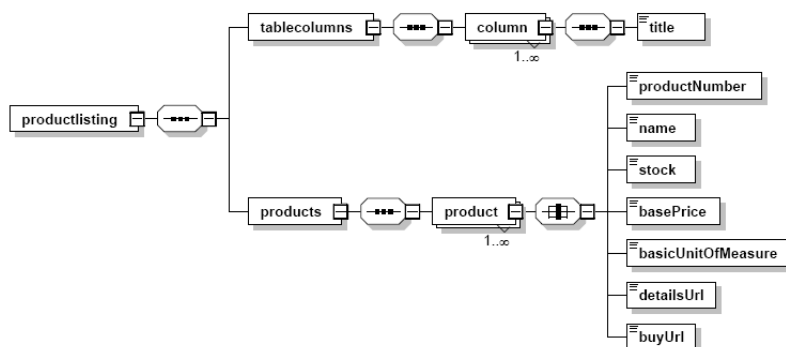


Figure B.19: The Schema for the *productListings* between the Business Logic Tier and Server Presentation Tier.

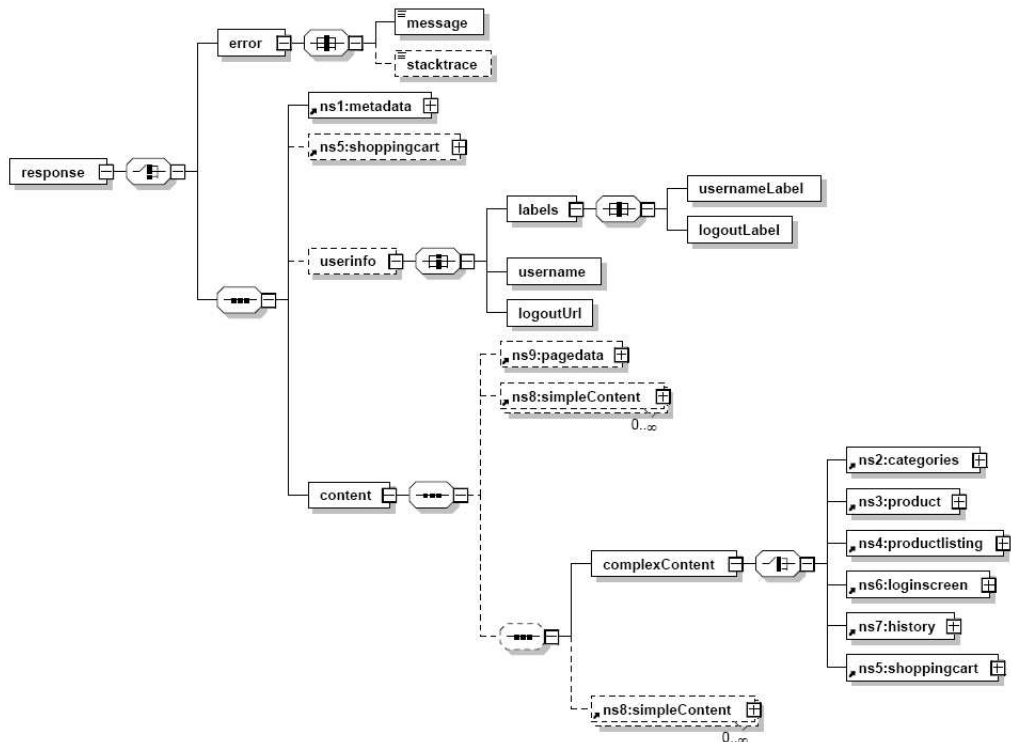


Figure B.20: The Schema for the *response* between the Business Logic Tier and Server Presentation Tier.

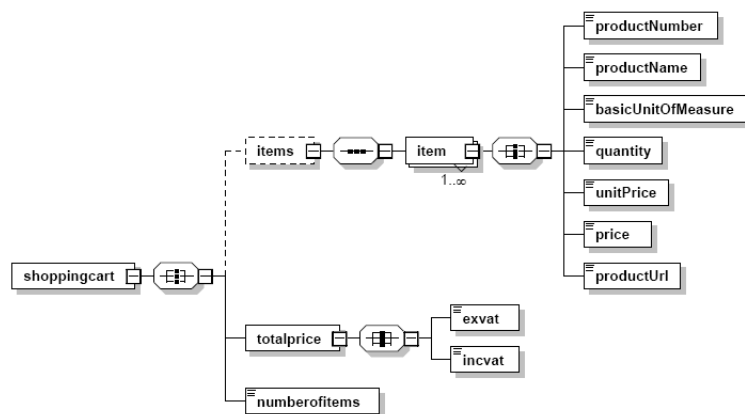


Figure B.21: The Schema for the *shoppingcart* between the Business Logic Tier and Server Presentation Tier.

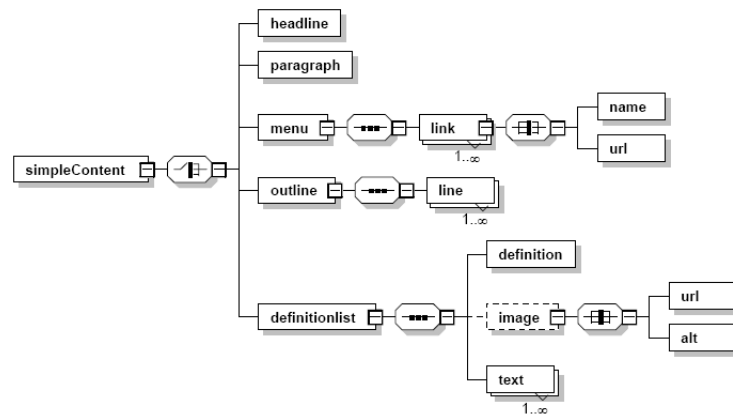


Figure B.22: The Schema for the *simpleContent* between the Business Logic Tier and Server Presentation Tier.

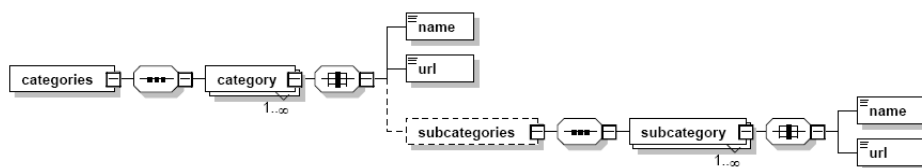


Figure B.23: The Schema for the *categories* between the Business Logic Tier and Server Presentation Tier.

# BIBLIOGRAPHY

---

- [1] Lars Mathiassen, Andreas Munk-Madsen, Peter Axel Nielsen, and Jan Stage. *Object Orienteret Analyse og Design*. Marko ApS, 2001.
- [2] Duelco website, December 2005. <http://www.duelco.dk/>.
- [3] Jakob Nielsen. Why Frames Suck (Most of the Time), December 1996. <http://www.useit.com/alertbox/9612.html>.
- [4] W3C. Tables in HTML documents. <http://www.w3.org/TR/REC-html40/struct/tables.html>.
- [5] Brandon Olejniczak. Using XHTML/CSS for an Effective SEO Campaign. Website, September 2003. <http://www.alistapart.com/articles/seo/>.
- [6] Donald Firesmith. Open Process Framework. Website, December 2005. <http://www.donald-firesmith.com/>.
- [7] [www.highermath.com](http://www.highermath.com). N-Tier Architecture: How to get there. Website, December 2005. <http://www.highermath.com/ps/ntier.htm>.
- [8] W3C. Soap specifications, December 2005. <http://www.w3.org/TR/soap/>.
- [9] W3C. XHTML 1.0. <http://www.w3.org/TR/xhtml1/>.
- [10] Jenny Preece, Yvonne Rogers, and Helen Sharp. *Interaction Design: Beyond Human-Computer Interaction*. John Wiley and Sons, 2002.
- [11] Nunit.org. NUnit. Website, December 2005. <http://www.nuint.org>.
- [12] Jeffrey Rubin. *Handbook of Usability Testing*. John Wiley & Sons, Inc., 1994.
- [13] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw Hill, 2005.
- [14] Altova. Altova.com. Website, December 2005. <http://www.altova.com>.

[15] Subversion. Website, December 2005. <http://subversion.tigris.org/>.

[16] SyntEvo GmbH. Smartsvn. Website, December 2005. <http://www.smartsvn.com/smartsvn/index.html>.