

**AN XML BASED ELECTRONIC CATALOG
MANAGEMENT SYSTEM**

**M.Sc. Thesis by
Nurettin Burak Bari, B.Sc.**

Department : Computer Engineering

Programme: Computer Engineering

MAY 2004

**AN XML BASED ELECTRONIC CATALOG
MANAGEMENT SYSTEM**

**M.Sc. Thesis by
Nurettin Burak Barı, B.Sc.**

(504011406)

Date of submission : 26 April 2004

Date of defence examination: 20 May 2004

Supervisor (Chairman): Prof. Dr. Bülent Örencik

Members of the Examining Committee Assoc. Prof.Dr.Coşkun Sönmez

Prof.Dr. Ali Okatan (B.Ü.)

MAY 2004

**XML TABANLI ELEKTRONİK KATALOG YÖNETİM
SİSTEMİ**

YÜKSEK LİSANS TEZİ
Müh. Nurettin Burak BARI
(504011406)

Tezin Enstitüye Verildiği Tarih : 26 Nisan 2004
Tezin Savunulduğu Tarih : 20 Mayıs 2004

Tez Danışmanı : Prof.Dr. Bülent ÖRENCİK
Diğer Jüri Üyeleri Doç.Dr. Coşkun SÖNMEZ
Prof.Dr. Ali OKATAN (B.Ü.)

MAYIS 2004

ACKNOWLEDGEMENTS

I would like to thank my advisor Professor Bülent Örencik for his important comments and support. I also thank Devrim Ergel, Mutlu Önder and all staff of Parsera Information Technologies for their priceless remarks. Finally my big thanks belong to my family for their support.

April, 2004

Nurettin Burak Bari

TABLE OF CONTENTS

ABBREVIATIONS	vii
LIST OF TABLES	viii
LIST OF FIGURES	x
ÖZET	xii
SUMMARY	xiii
1. INTRODUCTION	1
2. FEATURES OF OUR ECMS	3
3. DATA STRUCTURE	4
3.1 Elements of Data Structure	5
3.1.1 Atomic Elements	5
3.1.1.1 Text	5
3.1.1.2 Number	5
3.1.1.3 Date	5
3.1.2 Non-Atomic Elements	5
3.1.2.1 Language	5
3.1.2.2 Unit	5
3.1.2.3 Member	5
3.1.2.4 Member Group	6
3.1.2.5 Catalog	6
3.1.2.6 Catalog Tree	6
3.1.2.7 Group	6
3.1.2.8 Property Group	6
3.1.2.9 Product Family	6
3.1.2.10 Property	6
3.1.2.11 Value	6
3.1.2.12 Product	6
3.1.2.13 Picture	7
3.1.2.14 File	7
3.1.2.15 XSL View	7
3.2 Hierarchy of the Data Structure Elements	7
3.3 Rule Base	8

3.4	Property Groups	9
3.4.1	Property Group Mode 0	9
3.4.2	Property Group Mode 1	10
3.4.3	Property Group Mode 2	10
3.4.4	Property Group Mode 3	11
3.4.5	Property Group Mode 4	11
3.4.6	Property Group Mode 5	12
3.4.7	Property Group Mode 6	12
3.4.8	Property Group Mode 7	12
3.4.9	Property Group Mode 8	13
3.5	Forkability	13
3.6	Complete Example	14
4.	XML IMPLEMENTATION	16
4.1	Realizing Data Structure Elements in XML Format	16
4.1.1	Unit	16
4.1.2	Property group Mode 1, 2	17
4.1.3	Property group Mode 3, 4	17
4.1.4	Property group mode 6	17
4.1.5	Property group mode 8	18
4.1.6	Property	18
4.2	Advantages of XML	18
5.	GENERAL ARCHITECTURE	22
6.	DEVELOPMENT ENVIRONMENT	24
7.	MANAGER ARCHITECTURE	25
7.1	XML DB	25
7.2	DB-Bridge	25
7.2.1	Locking & Versioning Mechanism	26
7.2.2	Paging Mechanism	29
7.2.3	Connection Pool Mechanism	30
7.2.4	Platform Specific Mechanism	30
7.3	Manager	32
7.3.1	Navigation Related Operations	32
7.3.2	Product Family Related Operations	33
7.3.3	Product Related Operations	33
7.3.4	Supplier Related Operations	33

7.3.5	Data Sharing Operations	33
7.3.6	Administrative Operations	33
7.4	Integration	33
8.	WEB CATALOG	35
9.	CD CATALOG	40
10.	TEST RESULTS	42
10.1	ECMS Performance Test	42
10.2	ECMS Performance Test Variables	43
10.2.1	Test Scenarios	43
10.2.1.1	Scenario 1	43
10.2.1.2	Scenario 2	43
10.2.2	Number of Products	43
10.2.3	PageSize Value	43
10.2.4	DB-Bridge Server	44
10.2.5	Manager Client	44
10.2.6	JVM	44
10.2.7	Network Connection	45
10.3	Test Results	45
10.3.1	Scenario 1-A	45
10.3.2	Scenario 1-B	46
10.3.3	Scenario 1-C	47
10.3.4	Scenario 1-D	49
10.3.5	Scenario 2	50
10.4	ECMS Performance Test Summary	51
10.5	Web Catalog Server Test	52
10.5.1	Test Group 1	52
10.5.2	Test Group 2	53
10.5.3	Test Group 3	53
10.6	Web Catalog Server Test Variables	53
10.6.1	Test Scenario 1	53
10.6.2	Test Scenario 2	54
10.6.3	Test Scenario 3	54
10.6.4	Test Scenario 4	54
10.6.5	Number of Products	55
10.6.6	Catalog Web Server	55
10.6.6.1	Server 1 Configuration	55

10.6.6.2	Server 2 Configuration	55
10.6.7	Memory	55
10.6.8	JVM	56
10.6.9	Network Connection	56
10.7	Web Catalog Server Test Results	56
10.7.1	Scenario 1 Results	56
10.7.2	Scenario 2 Results	61
10.7.3	Scenario 3 Results	66
10.7.4	Scenario 4 Results	71
10.7.5	Server Comparison	76
10.8	Web Catalog Server Test Conclusions	77
11.	CONCLUSION	78
	REFERENCES	80
	APPENDIX A: SCREENSHOTS OF THE MANAGER.	83
	APPENDIX B: WEB CATALOG SCREENSHOTS	86
	CURRICULUM VITAE	89

ABBREVIATIONS

XML	: Extensible Markup Language
SOAP	: Simple Object Access Protocol
XSL	: Extensible Stylesheet Language
JVM	: Java Virtual Machine
GUI	: Graphical User Interface
RDMS	: Relational Database Management System
ECMS	: Electronic Catalog Management System
HTTP	: Hypertext Transfer Protocol
WML	: Website Meta Language
GPRS	: General Packet Radio Service
RMI	: Remote Method Invocation
XPATH	: XML Path Language
XUPDATE	: XML Update Language
XQUERY	: XML Query Language
J2EE	: JAVA 2 Platform, Enterprise Edition
J2SDK	: JAVA 2 Platform, Software Development Kit
SKU	: Stock Keeping Unit
SQL	: Structured Query Language
TCP/IP	: Transmission Control Protocol / Internet Protocol
RAM	: Random Access Memory
ERP	: Enterprise Resource Planning
DB	: Database
PDF	: Portable Document Format
SMTP	: Simple Mail Transfer Protocol
ISBN	: The International Standard Book Number

LIST OF TABLES

	<u>Page Number</u>
Table 3.1 Complete Book Example.....	15
Table 4.1 XML - RDMS Comparison.	16
Table 4.2 RDMS Data Example.	20
Table 4.3 Modified RDMS Data Example.....	20
Table 10.1 Scenario 1-A Test Results	45
Table 10.2 Scenario 1-B Test Results	46
Table 10.3 Scenario 1-C Test Results	48
Table 10.4 Scenario 1-C Stress Test Results.....	49
Table 10.5 Scenario 1-D Test Results.....	50
Table 10.6 Scenario 1-D Stress Test Results	50
Table 10.7 Scenario 2 Test Results	51
Table 10.8 Web Catalog Test, Scenario 1, Condition 1 Test Variables.....	56
Table 10.9 Web Catalog Test, Scenario 1, Condition 1 Average Click Time	57
Table 10.10 Web Catalog Test, Scenario 1, Condition 2 Test Variables.....	57
Table 10.11 Web Catalog Test, Scenario 1, Condition 2 Average Click Time	58
Table 10.12 Web Catalog Test, Scenario 1, Condition 3 Test Variables.....	58
Table 10.13 Web Catalog Test, Scenario 1, Condition 3 Average Click Time	59
Table 10.14 Web Catalog Test, Scenario 1, Condition 4 Test Variables.....	59
Table 10.15 Web Catalog Test, Scenario 1, Condition 4 Average Click Time	60
Table 10.16 Web Catalog Test, Scenario 1, Condition 5 Test Variables.....	60
Table 10.17 Web Catalog Test, Scenario 1, Condition 5 Average Click Time	61
Table 10.18 Web Catalog Test, Scenario 2, Condition 1 Test Variables.....	61
Table 10.19 Web Catalog Test, Scenario 2, Condition 1 Average Click Time	62
Table 10.20 Web Catalog Test, Scenario 2, Condition 2 Test Variables.....	62
Table 10.21 Web Catalog Test, Scenario 2, Condition 2 Average Click Time	63
Table 10.22 Web Catalog Test, Scenario 2, Condition 3 Test Variables.....	63
Table 10.23 Web Catalog Test, Scenario 2, Condition 3 Average Click Time	64
Table 10.24 Web Catalog Test, Scenario 2, Condition 4 Test Variables.....	64
Table 10.25 Web Catalog Test, Scenario 2, Condition 4 Average Click Time	65
Table 10.26 Web Catalog Test, Scenario 2, Condition 5 Test Variables.....	65
Table 10.27 Web Catalog Test, Scenario 2, Condition 5 Average Click Time	66
Table 10.28 Web Catalog Test, Scenario 3, Condition 1 Test Variables.....	66
Table 10.29 Web Catalog Test, Scenario 3, Condition 1 Average Click Time	67
Table 10.30 Web Catalog Test, Scenario 3, Condition 2 Test Variables.....	67
Table 10.31 Web Catalog Test, Scenario 3, Condition 2 Average Click Time	68
Table 10.32 Web Catalog Test, Scenario 3, Condition 3 Test Variables.....	68
Table 10.33 Web Catalog Test, Scenario 3, Condition 3 Average Click Time	69
Table 10.34 Web Catalog Test, Scenario 3, Condition 4 Test Variables.....	69
Table 10.35 Web Catalog Test, Scenario 3, Condition 4 Average Click Time	70
Table 10.36 Web Catalog Test, Scenario 3, Condition 5 Test Variables.....	70
Table 10.37 Web Catalog Test, Scenario 3, Condition 5 Average Click Time	71
Table 10.38 Web Catalog Test, Scenario 4, Condition 1 Test Variables.....	71

Table 10.39	Web Catalog Test, Scenario 4, Condition 1 Average Click Time	72
Table 10.40	Web Catalog Test, Scenario 4, Condition 2 Test Variables.....	72
Table 10.41	Web Catalog Test, Scenario 4, Condition 2 Average Click Time	73
Table 10.42	Web Catalog Test, Scenario 4, Condition 3 Test Variables.....	73
Table 10.43	Web Catalog Test, Scenario 4, Condition 3 Average Click Time	74
Table 10.44	Web Catalog Test, Scenario 4, Condition 4 Test Variables.....	74
Table 10.45	Web Catalog Test, Scenario 4, Condition 4 Average Click Time	75
Table 10.46	Web Catalog Test, Scenario 4, Condition 5 Test Variables.....	75
Table 10.47	Web Catalog Test, Scenario 4, Condition 5 Average Click Time	76
Table 10.48	Server Comparison, Test Variables	76
Table 10.49	Server Comparison, Average Click Time	77

LIST OF FIGURES

	<u>Page Number</u>
Figure 3.1 ECMS Data Structure Example.	4
Figure 3.2 Hierarchy of the Data Structure Elements.	7
Figure 3.3 Rule Base Diagram.	9
Figure 3.4 Property Group Mode 1 Example.	10
Figure 3.5 Property Group Mode 2 Example.	10
Figure 3.6 Property Group Mode 3 Example.	11
Figure 3.7 Property Group Mode 4 Example.	11
Figure 3.8 Property Group Mode 5 Example.	12
Figure 3.9 Property Group Mode 6 Example.	12
Figure 3.10 Property Group Mode 7 Example.	13
Figure 3.11 Property Group Mode 8 Example.	13
Figure 3.12 Forkability Example.	14
Figure 5.1 General Architecture.	22
Figure 7.1 Manager Architecture.	25
Figure 7.2 DB-Bridge Locking Mechanism.	27
Figure 7.3 Versioning Mechanism.	28
Figure 7.4 Paging Mechanism.	29
Figure 7.5 Platform Specific Mechanisms	31
Figure 8.1 Web Catalog Architecture.	35
Figure 8.2 Web Catalog Update Workflow.	36
Figure 8.3 An Example of Catalog Viewer Actions.	38
Figure 9.1 CD Catalog Arhitecture.	40
Figure 10.1 Scenario 1-A Process Time Averages	45
Figure 10.2 Scenario 1-B Process Time Averages	46
Figure 10.3 Scenario 1-C Process Time Averages	48
Figure 10.4 Scenario 1-D Process Time Averages	49
Figure 10.5 Scenario 2 Process Time Averages	51
Figure 10.6 Web Catalog Test, Scenario 1, Condition 1 Test Results	56
Figure 10.7 Web Catalog Test, Scenario 1, Condition 2 Test Results	57
Figure 10.8 Web Catalog Test, Scenario 1, Condition 3 Test Results	58
Figure 10.9 Web Catalog Test, Scenario 1, Condition 4 Test Results	59
Figure 10.10 Web Catalog Test, Scenario 1, Condition 5 Test Results	60
Figure 10.11 Web Catalog Test, Scenario 2, Condition 1 Test Results	61
Figure 10.12 Web Catalog Test, Scenario 2, Condition 2 Test Results	62
Figure 10.13 Web Catalog Test, Scenario 2, Condition 3 Test Results	63
Figure 10.14 Web Catalog Test, Scenario 2, Condition 4 Test Results	64
Figure 10.15 Web Catalog Test, Scenario 2, Condition 5 Test Results	65
Figure 10.16 Web Catalog Test, Scenario 3, Condition 1 Test Results	66
Figure 10.17 Web Catalog Test, Scenario 3, Condition 2 Test Results	67
Figure 10.18 Web Catalog Test, Scenario 3, Condition 3 Test Results	68
Figure 10.19 Web Catalog Test, Scenario 3, Condition 4 Test Results	69

Figure 10.20 Web Catalog Test, Scenario 3, Condition 5 Test Results	70
Figure 10.21 Web Catalog Test, Scenario 4, Condition 1 Test Results	71
Figure 10.22 Web Catalog Test, Scenario 4, Condition 2 Test Results	72
Figure 10.23 Web Catalog Test, Scenario 4, Condition 3 Test Results	73
Figure 10.24 Web Catalog Test, Scenario 4, Condition 4 Test Results	74
Figure 10.25 Web Catalog Test, Scenario 4, Condition 5 Test Results	75
Figure 10.26 Server Comparison, Test Results	76
Figure A. 1 Catalog Tree & Product Editing Window.....	83
Figure A. 2 Property Group Editing Menu.....	84
Figure A. 3 Product Family Editing Menu.	84
Figure A. 4 Image Editing Menu.....	85
Figure B. 1 Detailed Product Web Page.	86
Figure B. 2 Product Search Web Page.....	87
Figure B. 3 Search Results Page.	88
Figure B. 4 Basket Module Screenshot.	88

ÖZET

Internet kullanımının yaygınlaşması sonucu, bütün dünyada Elektronik Ticaretin hacminin hızla büyümesi, şirketlerin mevcut pazarlama stratejilerinde yenilikler yapmasını ve elektronik ticarete kullanılacak ürün bilgisinin en etkin olarak ifade edilmesini zorunlu hale getirmiştir. Günümüzde basılı katalogların, şirketlerin ürün bilgilerini yeterince belirtmediği açıktır ve bu konuda Elektronik Kataloglar daha uygun bir çözümdür. İlişkisel veri tabanı kullanan web tabanlı sistemler bir çözüm gibi görünse de, bir elektronik katalogun işlevlerini tam olarak yerine getiremez. Yapılan çalışmada, ürün bilgilerinde bütünlülüğü sağlayıp, elektronik ortamlara aynı tutarlılıkla dağıtılmasını amaçlayan XML tabanlı bir elektronik katalog sistemi sunulmaktadır. Ana fikir, ürüne ait bütün detayları, sınırlara bağlı kalmadan, tek bir platformda toplayıp; web sayfaları, CD gibi elektronik ortamlara aktarılmasını sağlamaktır. JAVA teknolojilerini baz alan bir istemci mimarisi ile, yerel XML veri tabanı arasında oluşturulan dağıtık sistemlerde çalışabilen, RMI tabanlı bir iletişim katmanı oluşturulmuş ve sistemde toplanan ürün bilgisinin, SOAP servisleri kullanılarak, internete aktarılması sağlanmıştır. Sunulan sistemle ilgili analiz, tasarım, ve uygulama safhaları detaylı olarak ele alınmış; çeşitli testlerle, uygulama platformunun performansı incelenmiştir. Sonuç olarak, XML tabanlı; fiyat, ebat, renk gibi sınırsız sayıda ürün bilgisini; resimler, videolar, dosyalar, teknik dokümanlar gibi sınırsız sayıda görsel içerik ile birleştirerek tek bir ortamda saklayabilen, kabul edilebilir bir performansla çalışan bir elektronik katalog sistemi elde edilmiştir.

SUMMARY

Impressive growth in the electronic commerce due to the widespread use of Internet all around the world resulted in reformation in marketing strategies and the effective representation of the product information became a crucial matter for companies. Today it is clearly understood that paper catalogs are not enough to inform every detail about a product and electronic catalogs are the convenient solutions for this problem. Although a web-based system that uses relational database seems to be a solution, it cannot provide all needs of an electronic catalog system. This thesis offers an XML based electronic catalog system that enables construction of boundless product details. The striking idea is to collect the detailed product information in one platform, and distribute them to various platforms such as web site and CD-ROM in order to avoid inconsistency in product integrity. A client architecture based on the JAVA technologies communicates with the native XML database system with the help RMI protocol which enables working on distributed systems. All the product data is published to internet by using the SOAP services. Analyze, design and the application phases of the offered system are described in detail, and the performance analysis of the whole platform is measured by various test cases. As a result, an XML based Electronic Catalog Management System that allows an unrestricted store of product attributes for any product which include an unlimited number of description fields (price, size, or color etc.) or an unlimited number of multimedia attachments such as images, video, design files, marketing brochures or technical specifications working with acceptable performance is achieved.

1. INTRODUCTION

Before Internet and the multimedia environment was common and CD-Rom technology was found, it was impossible to transfer a high quality image with 1.44 diskette or sending the product information via peer-to-peer connection seemed to be nonsensical. These insufficiencies were leading the companies to constitute paper catalogs, but even today they are expensive, they have a limited material for visuality, and changing a detail is nearly impossible with them. Since the production of a print catalog takes too long, it cannot be updated easily and regularly, besides it is hard for a customer to find the appropriate product. Designing a paper catalog template is another problem, which requires a graphic agency or an extra labor force for the companies. Distributing the hard-copy catalog to the person concerned is a hard work as well. With the recent technological developments, these obstacles are removed one by one. Today, the reasonable speed of the Internet and the comfort of the multimedia environment enable us to manage product catalogs much more efficiently by using Electronic Catalogs [1].

Electronic Catalog Management System (ECMS) [2] is an application where a company can store all product information with multimedia files such as images, videos, AutoCAD files etc. and publish this information on media such as Internet, CD for marketing purposes. The catalog content can be updated regularly and all updates are immediately visible on all media. With such a dynamic nature of electronic catalogs, companies can immediately respond to changing market conditions through repackaging, re-pricing etc. In contrast to paper catalogs, electronic catalogs provide a two-way communication channel between the company and its customers. The company needs only one product database, which can be integrated with back-office systems, to be published on different media. Once the product information is published on the web, the company can easily set up an e-commerce mechanism on it. Users can navigate through product catalogs using search engine mechanisms - which should support product specific searching criteria build a basket, place orders on Internet and make payments via secured online transaction mechanisms. The company is also able to provide authentication keys to

its suppliers so that, when a supplier enters the company's web site with this authentication key, he/she can see supplier related private information of products such as pricing, stock etc. and can make online orders. Such a communication ring accelerates order and payment chains of the company.

In most ECMS it is preferable to store data in a Relational Database Management System (RDMS) [3] because of its highly optimized nature. However, in this case, since every product family has a different structure, the data's structure is too dynamic for RDMS; also it needs to be exchanged between different systems and platforms. Thus XML [4, 5] turns up to be a wiser solution. XML can be transported safely over the Internet using a wide variety of protocols, such as HTTP, FTP and SMTP. These qualities indicate why XML should be used in an ECMS.

2. FEATURES OF OUR ECMS

Before offering the XML based ECMS as a solution to the problems given in introduction section, the necessities and the scope of the product representation should be defined. If a customer wants to examine or to buy a product, he/she should find the appropriate product in the catalog, according to specific criteria in his/her mind. After finding the product, the customer achieves the details about the product. Search criteria and final details change according to the type of the customer; if the customer is a supplier, he/she wants to search the product more easily with only using the SKU number and access to business related details like supplier price or stock. Meanwhile, end-users make more general searches and they want to see more multimedia content of products. Therefore, constituting the searching mechanism becomes the primary objective that should be accomplished by the system.

In our system we considered a search engine which supports search by unique product number and detailed search. Also the search system should work with the same principals in every environment. Considering the explanation mentioned above, we defined our final goal as ‘representing maximum product information and providing perfect attainability’.

This excellence is obtained by constructing different concepts that complement each other. First of all, exposition of the product detail should be almost limitless. But the untidiness caused by this freedom should be avoided by providing easiness on the graphical user interface (GUI). The GUI of the software [6] should be user friendly and navigation should be easy. Also processes have to be fast in order to sustain performance. Furthermore, once information is acquired, this should be used several times for different purposes if necessary, which will provide uniqueness and consistency of the information.

Before trying to integrate these features into software application, the data structure should be designed and clarified carefully so that necessities will not turn into restrictions. Only after the data structure is determined, the architecture can fit on it perfectly. Combination of well-designed structure with a powerful architecture, allows us to achieve desirable results in an ECMS.

3. DATA STRUCTURE

In this section, the data structure of the system will be given in detail; firstly the elements of the data structure will be defined which will also be used in the following chapters; after that, the relation between those elements will be given as a rule base with concrete examples.

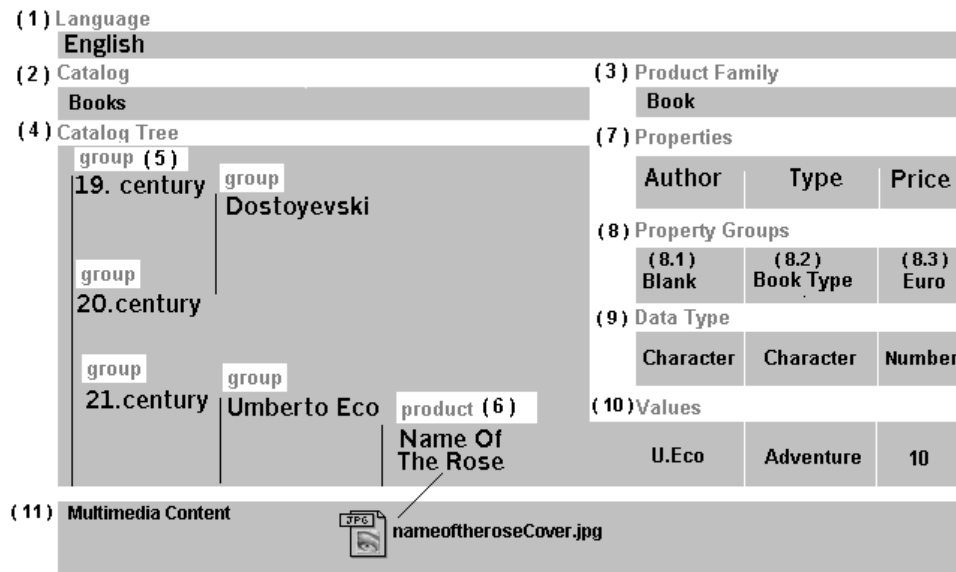


Figure 3.1 ECMS Data Structure Example.

The Data structure is designed according to the features of ECMS. First of all, ECMS supports several languages (Fig. 3.1.1), which is the root component of the system. Every other component takes part under a language except the universal objects like units or numeric values (Fig. 3.1.8.3), which are hierarchically independent. Under a language, products are grouped into catalogs (Fig. 3.1.2) and categorized by product families (Fig. 3.1.3). Catalogs are separated from each other according to the primary distinction between the product species. Each catalog has a product tree (Fig. 3.1.4). This tree holds the groups (Fig. 3.1.5) and the products (Fig. 3.1.6). Groups are branches and the products are leaves of the catalog tree. Catalog tree supplies the general view of the separation among the products in a conventional way. In the same level with the catalog, product families are defined, which are used as interfaces laying down the rules. Every product is an instance of a product-family and obeys the rules of the family, but it contains multimedia content such as pictures,

videos etc. specific to the product itself (Fig. 3.1.11). Each product-family consists of properties (Fig. 3.1.7) and each property defines a detail about the product like price, dimension, color etc.

The system has a complex nature that makes it hard to identify the core elements clearly. In order to resolve the confusion, elements will be defined in the following section.

3.1 Elements of Data Structure

System consists of atomic and non-atomic elements in general. Atomic elements states the final level real data, on the other hand the non-atomic elements states the high level virtual data. Both atomic and non-atomic elements are put together hierarchically in a rule base.

3.1.1 Atomic Elements

3.1.1.1 Text

Represents the data which is formed by a character string. There is no limit for the length of the string.

3.1.1.2 Number

Represents a numeric value.

3.1.1.3 Date

Represents a date, which includes the day, month and the year characteristic for the value.

3.1.2 Non-Atomic Elements

3.1.2.1 Language

The differentiation and the grouping of data start with this element. As a data, every single object has a different meaning in every language, in order to express this principal differentiation; the language element is hierarchically in the top level.

3.1.2.2 Unit

Unit element is used to express the universal units (such as kg, m, l, etc.).

3.1.2.3 Member

Represents a member; each member has special rights and access to specific

elements of the system like price, stock, discount rate etc. Every member obeys the rules that the “member group” defines.

3.1.2.4 Member Group

Defines the access rights of a member.

3.1.2.5 Catalog

Catalog is the primary element that separates the products from each other according to their functional differentiation.

3.1.2.6 Catalog Tree

Catalog tree holds the products and represents the catalog hierarchy in a tree formation.

3.1.2.7 Group

Groups are the branches of the catalog tree that holds products.

3.1.2.8 Property Group

Property group defines the value sets and the restrictions for a property about its value. It has different variations and will be explained in detail.

3.1.2.9 Product Family

Product family is a template for the product. It holds the properties and defines the way of extending the property groups for each property.

3.1.2.10 Property

Property is the representation of the single detail of a product like price, color, dimension etc. It has a value that is constrained by a property group or a unit.

3.1.2.11 Value

This element represents the value of a property. For instance, in this phrase “The price of the book is 10 Euro.” the book is the category, “price” is a property of a “book” and “10” is the value of the property.

3.1.2.12 Product

Product is the pivot element of the system.

3.1.2.13 Picture

Picture is a graphical representation of the product. An image that is related to the product is also counted as picture in the system. This element defines the image properties like image name, size, format etc. and holds the information about the image.

3.1.2.14 File

File is a document that contains information about the product like product manual.

3.1.2.15 XSL View

XSL View defines how the product will be displayed to catalog viewer.

3.2 Hierarchy of the Data Structure Elements

Each element explained in section 3.1 is arranged in a tree formation in order to represent the hierarchy between each other (Fig. 3.2).

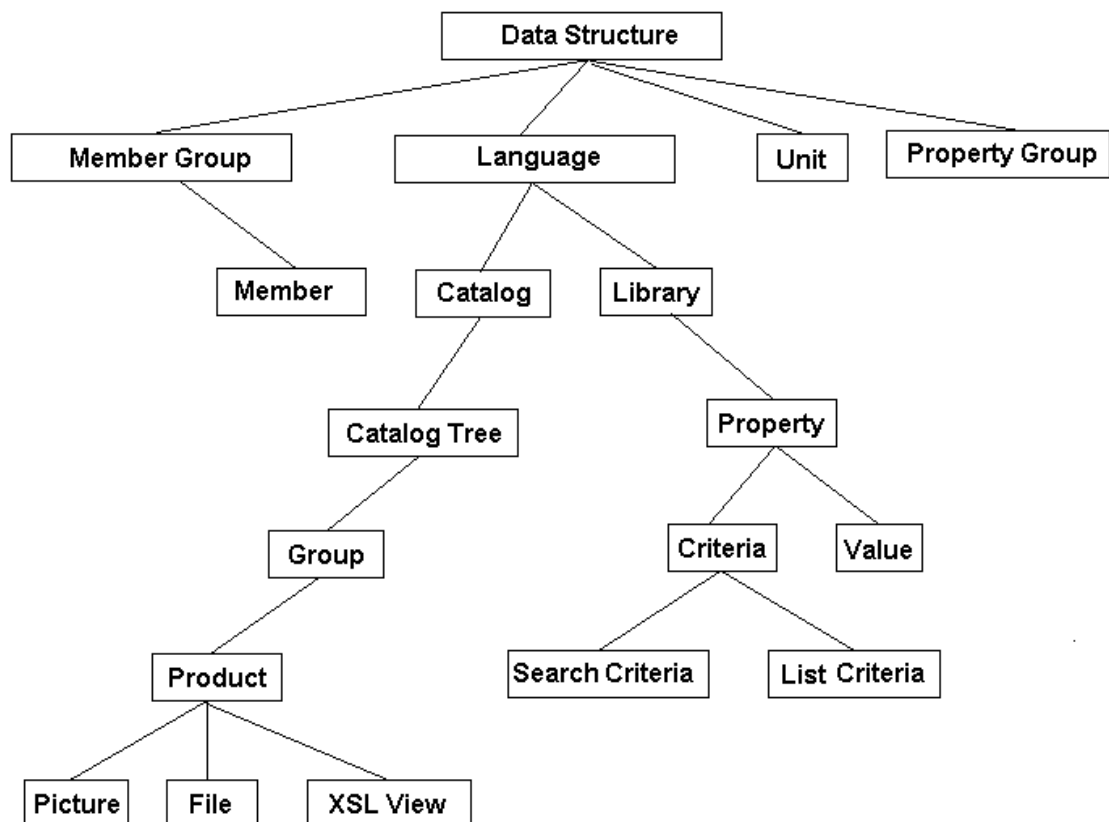


Figure 3.2 Hierarchy of the Data Structure Elements.

3.3 Rule Base

has single: Element has single element.

has: Element has one or more elements.

extends: Element is a template for the other element which attains the value.

- Language has catalogs.
- Language has units.
- Language has property groups.
- Language has product families.
- Member group has members.
- Product Family has properties.
- Property extends unit or property group.
- Property has values.
- Catalog has single catalog tree.
- Catalog tree has groups.
- Group has groups.
- Group has products.
- Product extends product family.
- Product has pictures.
- Product has files.
- Product has single XSL view.

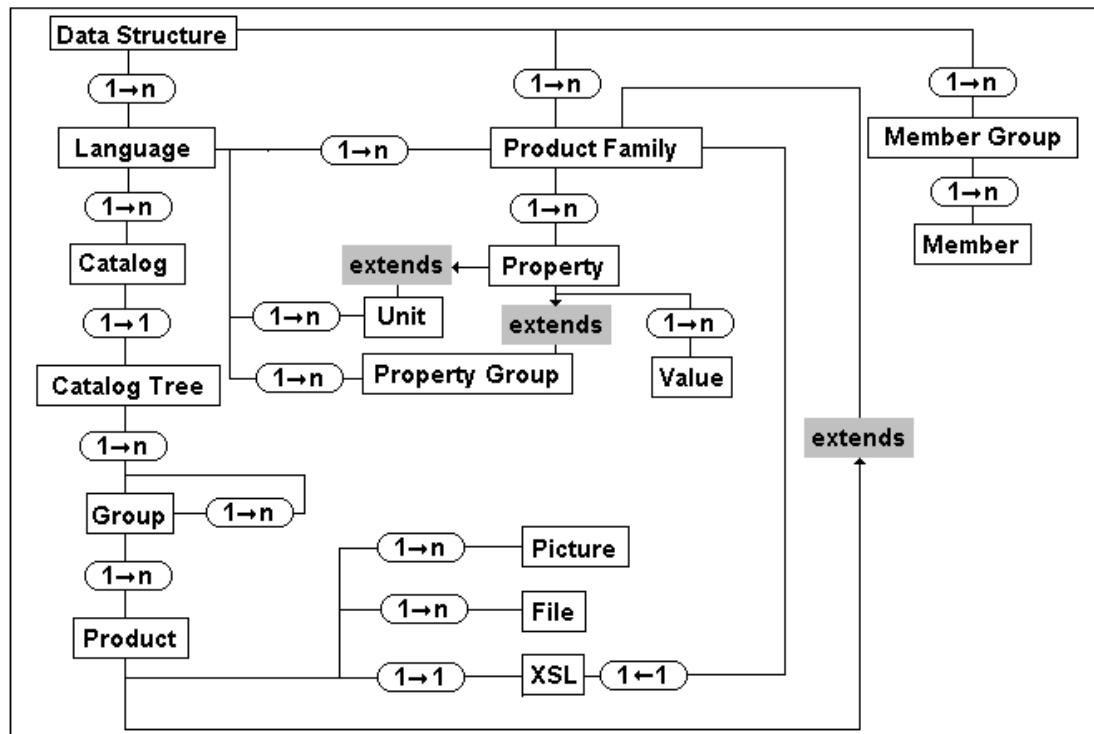


Figure 3.3 Rule Base Diagram.

3.4 Property Groups

The most significant element of the system is the property group element. It determines the restriction rules and how the value will be appointed to a property. In order to emphasize the importance, in this section the details of the property group are given in detail.

In order to define the best value for a property, the property group structure is planned to support various data selection in a simple data structure to reduce the software application's load. Such a structure will lessen the software development time.

The property groups are separated into modes. Each mode defines a different kind of data selection and embodies the data. As a result, the value of the property becomes more expressive.

3.4.1 Property Group Mode 0

There is no constraints or rule for attaining value to a property.

3.4.2 Property Group Mode 1

This mode selects a single element from predefined list, which consists of one or more elements. The property group object has a domain and a result set that selects single element form the domain.

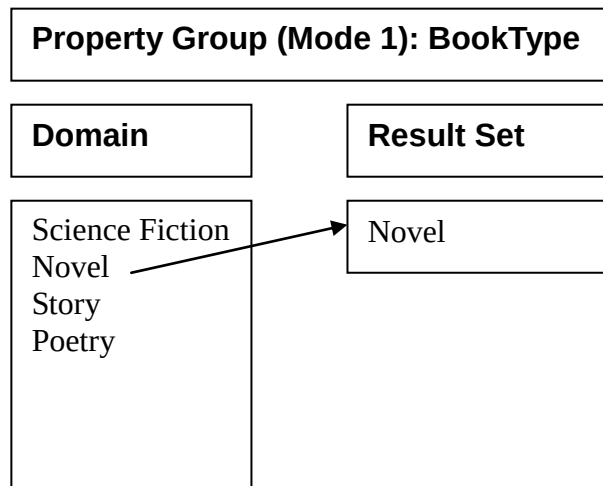


Figure 3.4 Property Group Mode 1 Example.

3.4.3 Property Group Mode 2

This mode selects multiple elements form predefined list, which consists of one or more elements. Mode 2 objects have a domain and a result set that selects multiple elements from the domain.

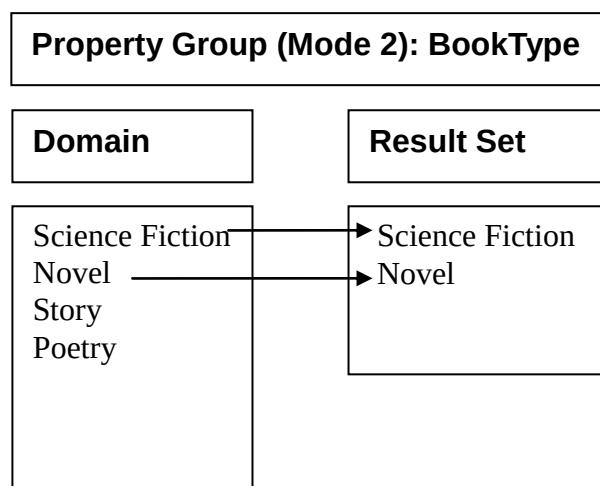


Figure 3.5 Property Group Mode 2 Example.

3.4.4 Property Group Mode 3

This mode selects single element form predefined data tree, which consists of multilevel data hierarchy. Mode 3 object has a domain and a result set that selects single tree component from the domain tree.

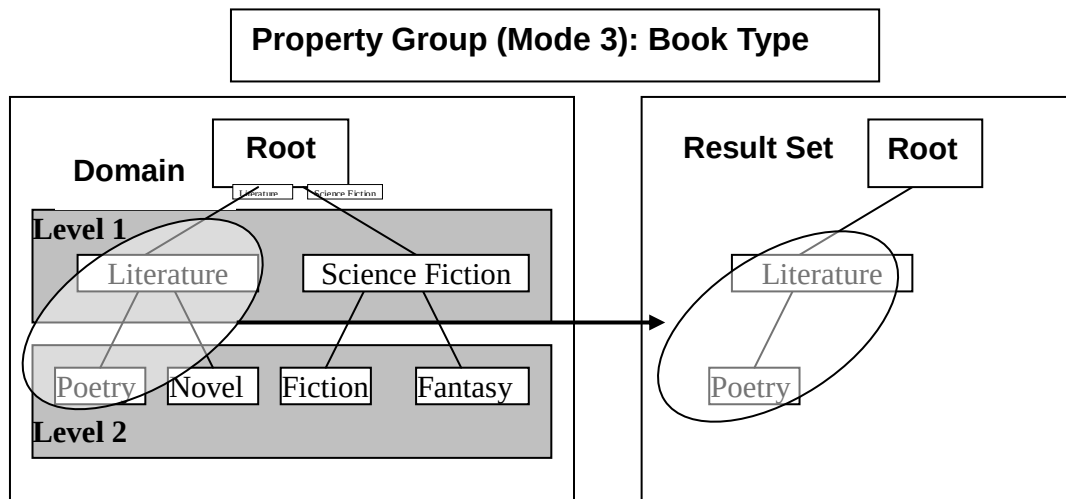


Figure 3.6 Property Group Mode 3 Example.

3.4.5 Property Group Mode 4

This mode selects multiple elements form predefined data tree, which consists of multi-level data hierarchy. Mode 4 objects have a domain and a result set that selects multiple tree components from the domain tree.

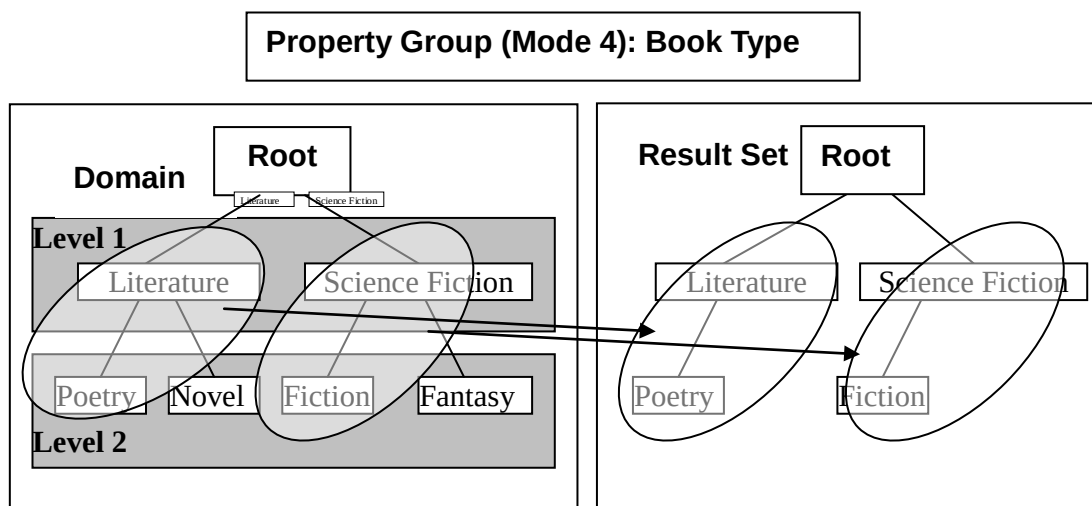


Figure 3.7 Property Group Mode 4 Example.

3.4.6 Property Group Mode 5

This mode type is used in mode 7, and restricted atomically by number, text or date.

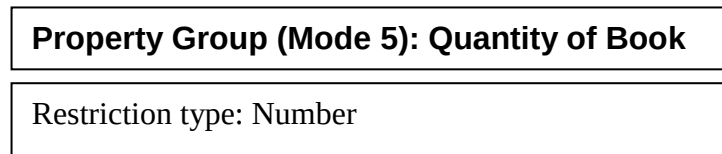


Figure 3.8 Property Group Mode 5 Example.

3.4.7 Property Group Mode 6

This mode is a pointer to another product element in the system. This kind of mode is used to express the sets. For instance, in a book catalog, the books are defined in a single formation but there are some other product types like book sets, which consist of some other books. A single book can be represented by itself, but it can also be a member of a book set which is another product. With this property group, the data structure is able to obtain the ability of “pointing product in another product”. In this property group mode, the domain consists of the all of the products in the system, and the result set consists one of the book in the system.

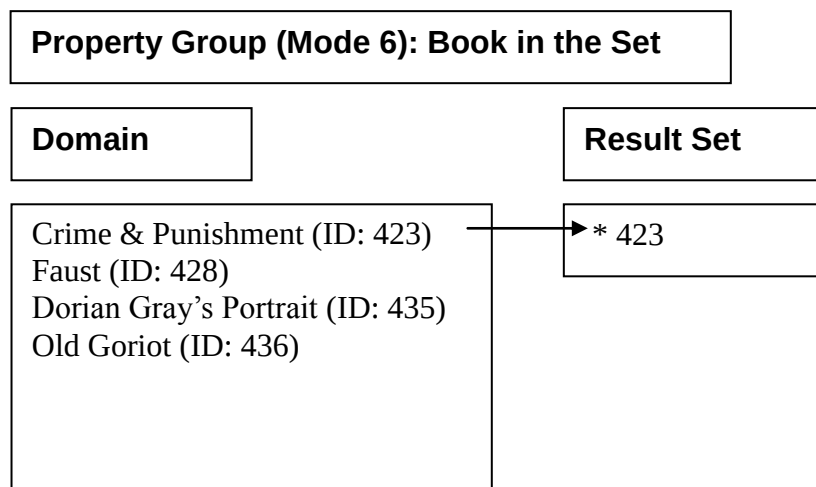


Figure 3.9 Property Group Mode 6 Example.

3.4.8 Property Group Mode 7

This mode widens the data structure horizontally. Mode 7 contains other property groups in itself. Considering the example in the mode 6, pointing the related product in the system by itself is sometimes not enough to express the data effectively. For each product pointer, it can be discussed to add another property which will be meaningful together. In a book set, there can be quantity property for each book to give the full sense.

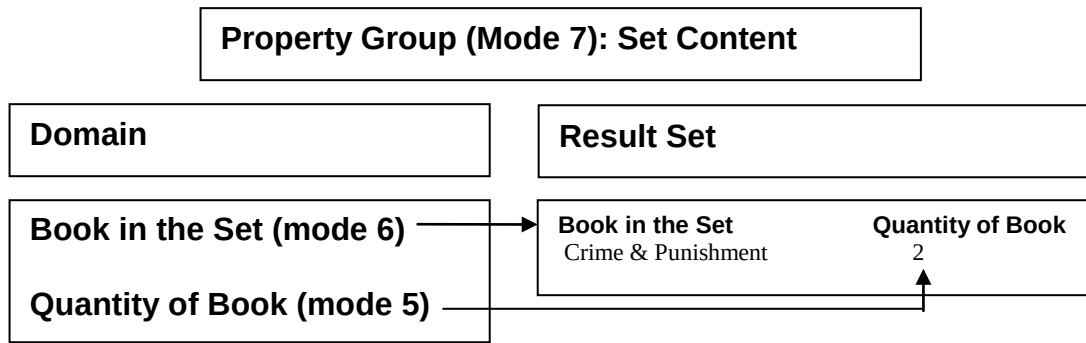


Figure 3.10 Property Group Mode 7 Example.

3.4.9 Property Group Mode 8

This property group is specialized to express the unique product number, which facilitates attainability. But there are some common problems in this subject. A full text search mentality does not solve the problem perfectly, as an alternate the Mode 8 property group offers a different filtering mechanism. The domain is filtered by a function that removes some of the characters that are not numeric or a letter. This filter can be specific to the user.

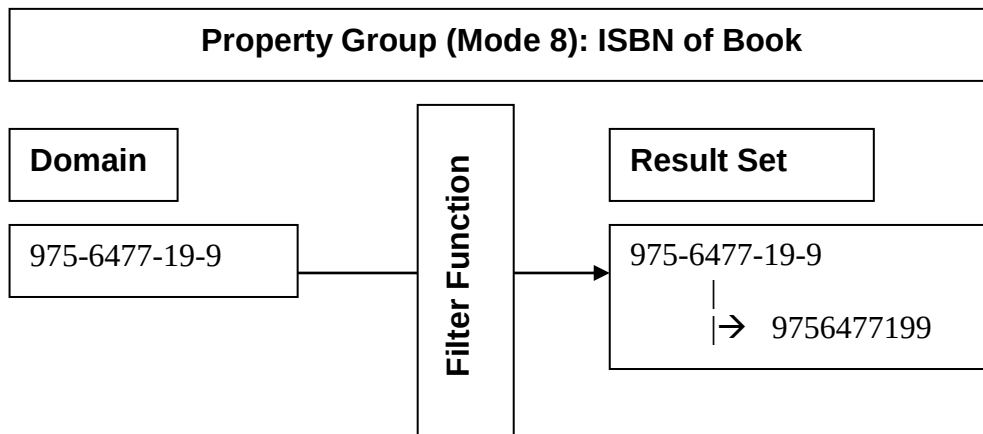


Figure 3.11 Property Group Mode 8 Example.

As a result, if the user enters the search texts like 975-6477-19-9 or 975 6477 19 9 or 975.6477.19.9, generally, 975*6477*19*9 (* represents any character other than numeric or letter based) the result filter will give 9756477199 value so that searching by a product number would be highly effective.

3.5 Forkability

With the property group Mode 7 the system enables widening horizontally. But the system needs widening in vertically, which means attaining multiple values to each property element respectively. According to that need, system offers another virtual

ability called “forkability”. Forkability means cloning the property that extends a property group or a unit element with the same restrictions defined by the property group or by the unit element. Each cloned element may have a different value. In order to express this hierarchy, the system uses the mother-child relations. Each property is signed as mother even if it does not support forkability. Cloning operation means copying the mother element and signing it as a child element. When forkability is applied to the Example give in Fig. 3.9, elements of a book set can be expressed effectively.

Property: Books in the Set <i>extends</i> Property Group (Mode 7): Set Content		
Book in the Set	Quantity of Book	Fork Type
Crime & Punishment	2	Mother
The Brothers Karamazov	4	Child
Notes from Underground	1	Child

Figure 3.12 Forkability Example.

Forkability can be adapted to any kind of property groups. With widening the data representation both horizontally and vertically, an opportunity of constructing a boundless detail is obtained as it is proposed.

3.6 Complete Example

Below there is a book example, taken from Amazon.com, which is a well known ecommerce web site. With this example, we proof that our data structure is capable of defining any king of product successfully.

Mass Market Paperback: 576 pages

Dimensions (in inches): 0.97 x 7.02 x 4.18

Translator: Constance Garnett

Publisher: Bantam

ISBN: 0553211757

Author: Fyodor Dostoyevsky

Price: 6.99 Euro

Other Editions: Hardcover, Paperback, Library Binding

This terminology is translated into our data structure as shown below.

Table 3.1 Complete Book Example

Property Name	Property Group / Unit	Atomic	Forkability	Value
Author	Mode 0	character	false	Fyodor Dostoyevsky
Translator	Mode 0	character	false	Constance Garnett
Publisher	Mode 0	character	false	Bantam
Book Type	Mode 1 (Book Type)	character	false	Novel
ISBN number	Mode 8 (Product Code)	number	false	0553211757
Price	Unit (Euro)	number	false	6.99
Page size	Mode 0	number	false	576
Thickness	Unit (inches)	number	false	0.97
Height	Unit (inches)	number	false	7.02
Width	Unit (inches)	number	false	4.18
Other Editions	Mode 6 (Other Editions)	number	true mother	Hardcover
			child	Paperback
			child	Library Binding

4. XML IMPLEMENTATION

When we consider the abilities of the data structure, which are given in Section 3, we can easily see that all the data structure elements are in tree model. Constructing and modifying tree models on RDMS needs too much workload to provide the data consistency. Therefore XML is chosen to realize the data structure. Below, a table is given that compares RDMS to XML Database System. Analyzing this table clarifies the reasons for choosing XML.

Table 4.1 XML - RDMS Comparison.

XML	RDMS
Data in single hierarchical structure	Data in multiple tables
Nodes have element and/or attribute values	Cells have a single value
Elements can be nested	Atomic cell values
Elements are ordered	Row/column order not defined
Elements can be recursive	Little support for recursive elements
Schema optional	Schema required
Direct storage/retrieval of XML documents	Joins often necessary to retrieve data
Query with XML standards (XQuery[7], XPath[8], XUpdate)	Query with SQL

4.1 Realizing Data Structure Elements in XML Format

In this section, the transformation of data structure elements to XML [9] will be given in detail. Before expanding each component one by one, it is important to explain special attributes that is specific to the XML data structure implementation.

First of all, each element has an ID attribute to become distinct form any other element. Another primary attribute is the type, which defines the atomic elements (Section 3.1.1).

4.1.1 Unit

```
<unit type="number" ID="2815">mm</unit>
```

4.1.2 Property group Mode 1, 2

```
<propertygroup name="Author" ID="5000" mode="1" type="character"
language="English">
```

```
<value><![CDATA[ Fyodor Dostoyevsky]]></value>
```

```
<value><![CDATA[ Tolstoy]]></value>
```

```
<value><![CDATA[ Balzac]]></value>
```

```
<value><![CDATA[ Umberto Eco]]></value>
```

```
</propertygroup>
```

4.1.3 Property group Mode 3, 4

```
<propertygroup name="Book Type" ID="4956" mode="3" type="character"
language="English" levelsize="2" >
```

```
- <value level="0" ID="4957"><![CDATA[Literature]]>
```

```
<subvalue level="1" ID="5028"><![CDATA[Classics]]></subvalue>
```

```
<subvalue level="1" ID="5029"><![CDATA[Novel]]></subvalue>
```

```
<subvalue level="1" ID="5030"><![CDATA[Poetry]]></subvalue>
```

```
<subvalue level="1" ID="5031"><![CDATA[Stories]]></subvalue>
```

```
</value>
```

```
+ <value level="0" ID="4958"><![CDATA[Art]]>
```

```
+ <value level="0" ID="4959"><![CDATA[Entertainment]]>
```

```
+ <value level="0" ID="4960"><![CDATA[Children]]>
```

```
+ <value level="0" ID="4961"><![CDATA[Sport]]>
```

```
+ <value level="0" ID="4962"><![CDATA[Food]]>
```

```
+ <value level="0" ID="4963"><![CDATA[Science]]>
```

```
</propertygroup>
```

4.1.4 Property group mode 6

```
<propertygroup name="Other Editions" ID="5002" mode="6" language="English"
/>
```

4.1.5 Property group mode 8

```
<propertygroup name="Product Code" ID="5001" mode="8" type="character"
language="English" />
```

4.1.6 Property

```
<property type="character" mode="1" ID="5003" order="5" forkable="false">
    <name><![CDATA[Author]]></name>
    <propertyGroup ID="5000"><![CDATA[Author]]></propertyGroup>
</property>
```

4.2 Advantages of XML

Below there is a simplified product structure of our ECMS to give an overview of the structure.

```
<product name = "Crime and Punishment">
    <properties>
        <property>
            <name>Author</name>
            <unit/>
            <value>Fyodor Dostoyevsky </value>
        </property>
        <property>
            <name>Type</name>
            <unit ID='5000'> ( → points to the unique ID of the property
group "Book Type")
            <value>Adventure</value>
            <value>Classics</value>
        </property>
        <property>
            <name>Price</name>
```

```

        <unit>Euro</unit>

        <value>6.99</value>

    </property>

</properties>

<pictures>

    <picture name='cover.jpg'>

        <info>The cover of the book.</info>

    </picture>

</pictures>

<files>

    <file name='summary.pdf'>

        <info>The summary of the book.</info>

    </file>

</files>

</product>

```

According to the structure given above, when a new property is added to a product family, it is automatically included to all existing products. For a relational database, this means that we have to add a new column to our product table, and all SQL statements have to be changed in order to include the new column to the searching mechanism. Changing SQL statements for each modification of the product family may cause an inconsistency of the platform.

XML represents data with a tree structure. Search (XPATH [10, 11]: query language for XML, similar to *SQL select* statements) and update (XUPDATE: update language for XML, similar to *SQL update* statements) mechanisms run on the tree directly (Table 4.1).

Adding a new property to a product family means adding a new leaf to XML document (to the 'properties' element), which is automatically added to searching and updating mechanisms. This structure avoids any inconsistency because XPATH expressions remain the same by this modification.

Example of a full text search case with input text “Adventure” based on the example architecture given above.

By using RDMS:

Table 4.2 RDMS Data Example.

<i>P1 (author)</i>	<i>P2 (type)</i>
<i>Fyodor Dostoyevsky</i>	<i>Adventure</i>

SQL: “Select * from books where P1 = ‘Adventure’ or P2 = ‘Adventure’ ”

Table 4.3 Modified RDMS Data Example.

<i>P1</i>	<i>P2</i>	<i>P3 (price)</i>
<i>Fyodor Dostoyevsky</i>	<i>Adventure</i>	<i>6.99</i>

The SQL statement has to be changed: “Select * from books where P1 = ‘Adventure’ or P2 = ‘Adventure’ or P3 = ‘Adventure’ ”

By using XML:

```
<properties>
  <property name=“P1”>
    <value>Fyodor Dostoyevsky</value>
  </ property>
  <property name=“P2”>
    <value>Adventure</value>
    <value> Classics</value>
  </ property>
</properties>
```

By modification on the document;

```
<properties>
  <property name=“P1”>
    <value> Fyodor Dostoyevsky</value>
  </ property>
  <property name=“P2”>
```

```
        <value>Adventure</value>

        <value> Classics</value>

    </ property>
    <property name="P3">

        <value>10</value>

    </ property>
</properties>
```

The XPATH expression “/product/properties/property/value[text()='Adventure’]” remains the same.

The major advantage of XML is that the representation of the data is separated from the data. XML is transformed via Extended Stylesheet Language (XSL) [12] to HTML, WML etc. in order to display the information to the catalog viewer according to the platform. In other words, for if example we want to change the design of the company’s web site, we neither change the product database nor the software components; we only change the XSL templates.

Another advantage of XML is that product data can be transformed into any structure in order to share it with back-office systems, ERPs etc, or to give reports to the customer in EXCEL, PDF etc. XML can also be sent via a wide variety of protocols such as HTTP, FTP and SMTP etc, which is an alternative way for integrating product data (For ex. with SOAP services [13-19]).

5. GENERAL ARCHITECTURE

Our platform consists of five modules, which are independent in the structure but process same XML documents.

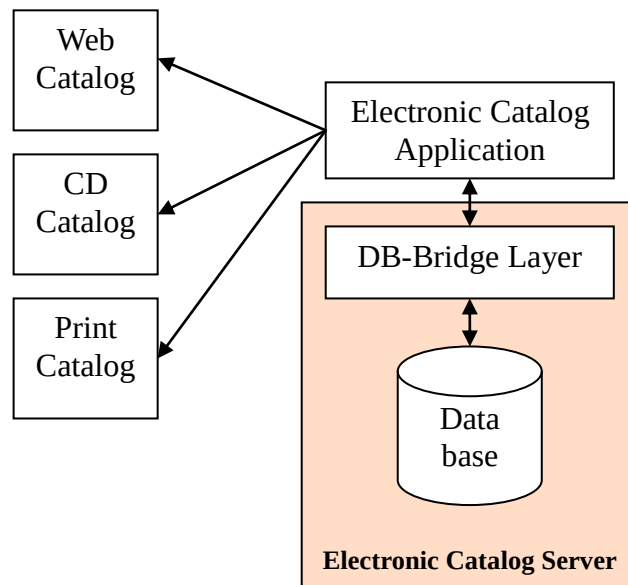


Figure 5.1 General Architecture.

Electronic Catalog Server consists of two modules namely the native XML Database and the DB-Bridge Layer. This server can be installed on any computer and all services can be accessed via TCP/IP protocol. The database holds the documents and the DB-Bridge layer processes XML documents. The architecture of the DB-Bridge is described in next sections.

Electronic Catalog Application will be called as “Manager” for the rest of the thesis. Persons who use the Manager will be called as “user” for the rest of the thesis. This module is the GUI that translates user’s GUI actions to XML processing orders for the DB-Bridge.

Web Catalog, CD Catalog and Print Catalog modules publishes the XML documents – products – on the WEB, CD and paper media. The detailed architecture of these modules will be explained in next sections.

6. DEVELOPMENT ENVIRONMENT

The reasons for choosing XML as data structure are explained in Section 4. For the development of the project, we needed an object oriented programming language with advanced GUI components and support for XML processing.

Considering today's technologies, there were three choices; JAVA, Delphi and Microsoft .NET Platform.

We have chosen JAVA because first of all it is an open source environment, so we didn't need any investment for the development environment of the project. On the other hand, there is a wide support of JAVA APIs for different needs; for example, for generating Excel reports, PDF documents, processing images etc.

On the web server side we have decided to use J2EE technology [20, 21], which can be considered as an extension of JAVA for internet applications, so we had the chance to use same classes and libraries for the development of the web catalog.

For the communication layer between our software layers we have chosen Java Remote Method Invocation (RMI) [22] technology on the application side and SOAP for our web based application. SOAP is an HTTP based protocol and supports XML messages, which fits our needs excellently. Java RMI enables the programmer to create distributed Java technology-based to Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. RMI uses object serialization to marshal and un-marshal parameters and does not truncate types, supporting true object-oriented polymorphism [23].

7. MANAGER ARCHITECTURE

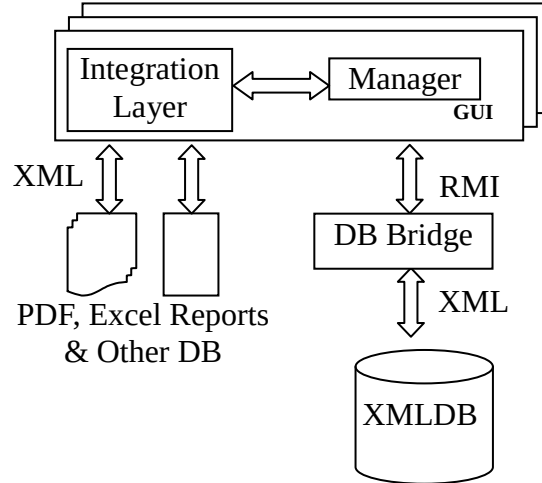


Figure 7.1 Manager Architecture.

7.1 XML DB

Once XML was chosen as data structure, a database system was needed to store XML documents natively (without converting them into relational data), query them by using XPATH and run XUPDATE commands on documents.

Ten database systems were analyzed and tested with our platform for this purpose. Some of these systems were commercial such as Tamino XML Server [24], X-Hive/DB Native XML Storage [25] and some were open-source such as Xindice [26], Exist [27].

Currently, Xindice is being used in our ECMS, as it is free and robust and also very successful in our stress tests.

7.2 DB-Bridge

To be successful on the market it is crucial for an ECMS to support at least some of many existing native XML databases. To do this efficiently, we decided to gather all database features in one layer; namely the DB-Bridge, and to separate the platform from the database. The whole platform can then be switched immediately to another database only by changing the DB-Bridge layer.

DB-Bridge uses JAVA RMI technology in order to talk to Managers. The Manager itself has no connection with the database directly; thus it sends requests to the DB-Bridge for database operations with the user's authentication token. This workflow brings a high security to the platform: Managers cannot request any DB operations from the DB-Bridge that are not 'legal' - which do not contain the user's authentication key or which will harm the database - since they are not defined in DB-Bridge layer.

The workflow of JAVA RMI technology consists of two components: the RMI Server and RMI clients. Clients can connect to the server by using IP and port specific addresses and request operations from them. This means that Managers (each Manager is an RMI client) can connect to our DB-Bridge (RMI Server) from anywhere using the TCP-IP protocol, in other words using the Internet.

7.2.1 Locking & Versioning Mechanism

There are three types of operations, which are critical for platform's data consistency. These are;

- Catalog Operations: Making a change in the catalog tree by adding or deleting a product or a group to/from the catalog.
- "UnitDocument" (UnitDocument contains all unit and property group definitions. Operations on UnitDocument are defined as modifying a unit or property group.
- Product Family Operations.

These operations are tracked and controlled by DB-Bridge. DB-Bridge contains two mechanisms; the versioning mechanism and the locking mechanism. The workflows of these mechanisms are explained by Figure 7.2 and Figure 7.3.

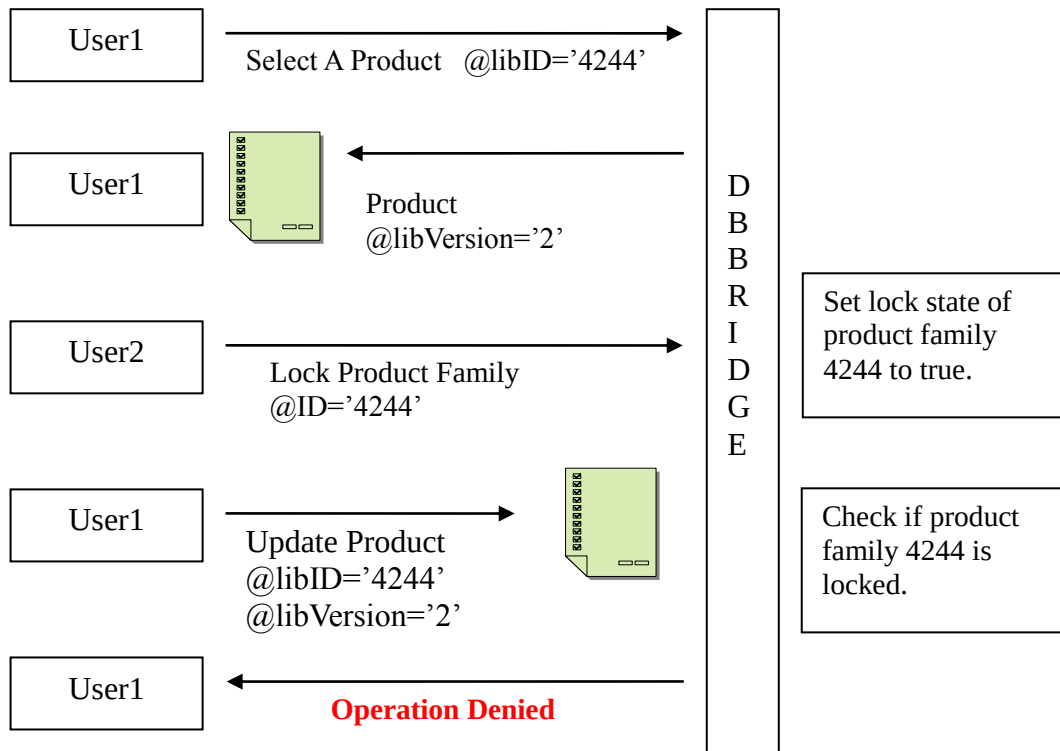


Figure 7.2 DB-Bridge Locking Mechanism.

This figure briefly explains the locking mechanism of the DB-Bridge. In this example, User1 requests a product document from the DB-Bridge and tries to update it after the product's product family is locked by User2 and as a result the update operation is denied by the DB-Bridge.

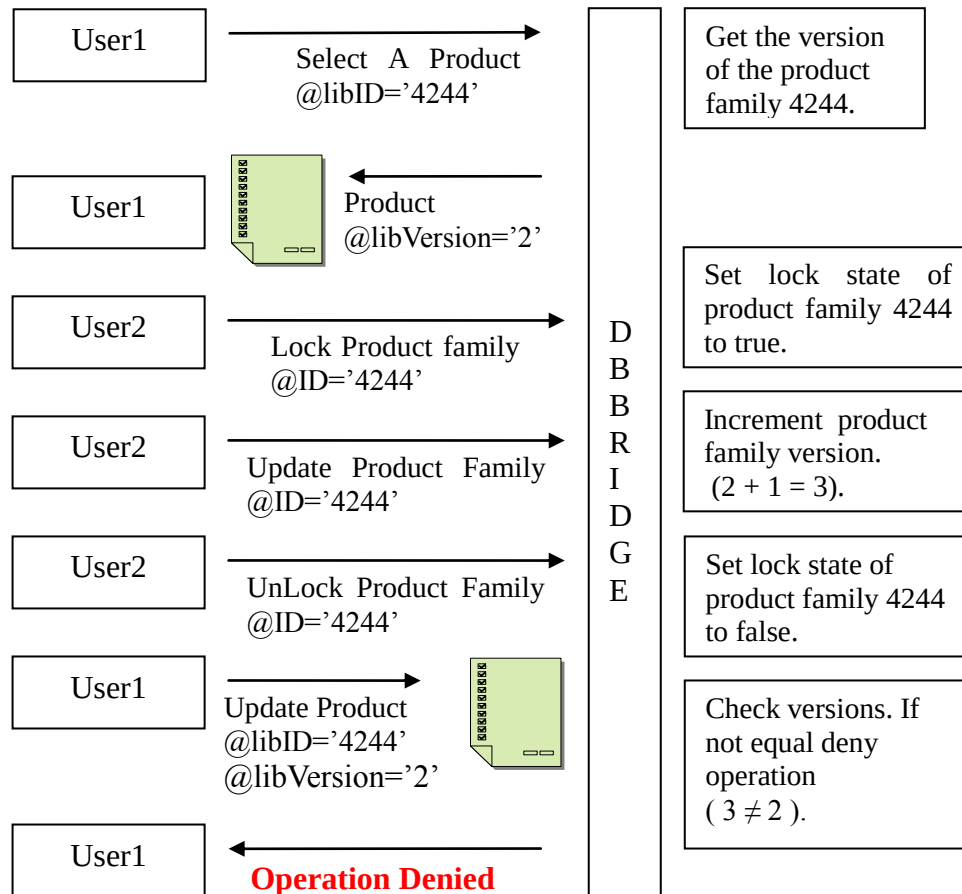


Figure 7.3 Versioning Mechanism.

This figure briefly explains the versioning mechanism of the DB-Bridge. In this example User1 requests a product document from the DB-Bridge and tries to update it after the product's product family is updated by User2. The update of the product family document results as an increment of the product family's version, but User1 has still the old version number attached with the product. As a result the update operation of User2 is denied by the DB-Bridge, since two versions are not equal.

These control mechanisms are used when the user tries;

- To delete a catalog.
- To update a catalog.
- To update unit document.
- To update products.
- To delete products.

- To add products.
- To update categories.

The atomic operations of these mechanisms are synchronized with each other such that only one operation about versions and lock states can be done at a time.

Checking user's authentication, supporting multiple users, backing up and restoring data, terminating sessions of users after a time interval, are some other capabilities of the DB-Bridge layer.

DB-Bridge contains some extra mechanisms in it, which were missing on Xindice. These mechanisms are briefly described below.

7.2.2 Paging Mechanism

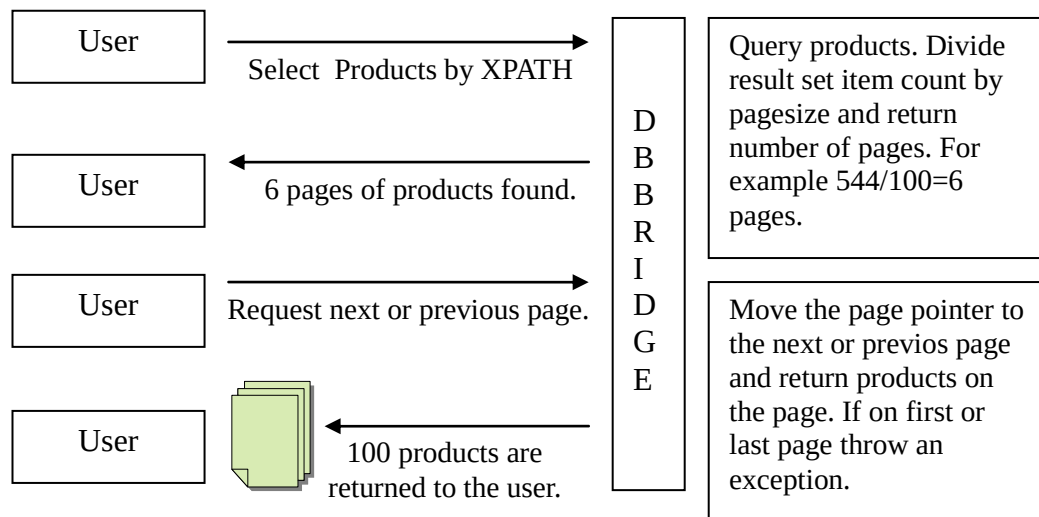


Figure 7.4 Paging Mechanism.

Once an XPATH is run on the product data, a result set is created. When the result set contains too many elements and if all of them are retrieved in a single step by the Manager, this operation leads to out of memory problems. Therefore a paging mechanism is developed for DB-Bridge layer. DB-Bridge divides the result set in pages and the results can only be retrieved page by page.

In this example the "PageSize" is defined as 100 and the XPath statement leads to 544 products, as a result 6 pages of product IDs are stored in a temporary cache in the DB-Bridge layer. In order to avoid memory consumption only unique IDs of products are stored in the cache. If the page pointer is on the first page and the user

requests the previous page or the pointer is on the last page and the user requests next page, an exception is thrown by DB-Bridge. On other cases products are returned to the user page by page.

Commercial database systems like Tamino XML Server or X-Hive/DB have this capability built-in.

7.2.3 Connection Pool Mechanism

The major step of database transactions that requires a long time interval is connecting to the database or to a collection in the database. In order to avoid this latency a connection pool mechanism is developed for the DB-Bridge so that the DB-Bridge holds a specific number of connections to the database and uses these present connections for database transactions.

In our architecture 9 collections are used to store our data. These are:

- files: Stores files and images of products.
- config: Stores configuration documents.
- catalogs: Stores product catalogs.
- deleteditems: Stores the track of deleted documents since last update of the web catalog.
- xsl: Stores XSL templates.
- members: Stores members and member groups documents.
- libraries: Stores product family documents.
- units: Stores units and property groups.
- products: Stores products.

7.2.4 Platform Specific Mechanism

DB-Bridge is also responsible for tracking updated and added documents, give unique ID to newly added documents and to track all delete operations since last successful web catalog update. Figure 7.5 briefly describes these mechanisms.

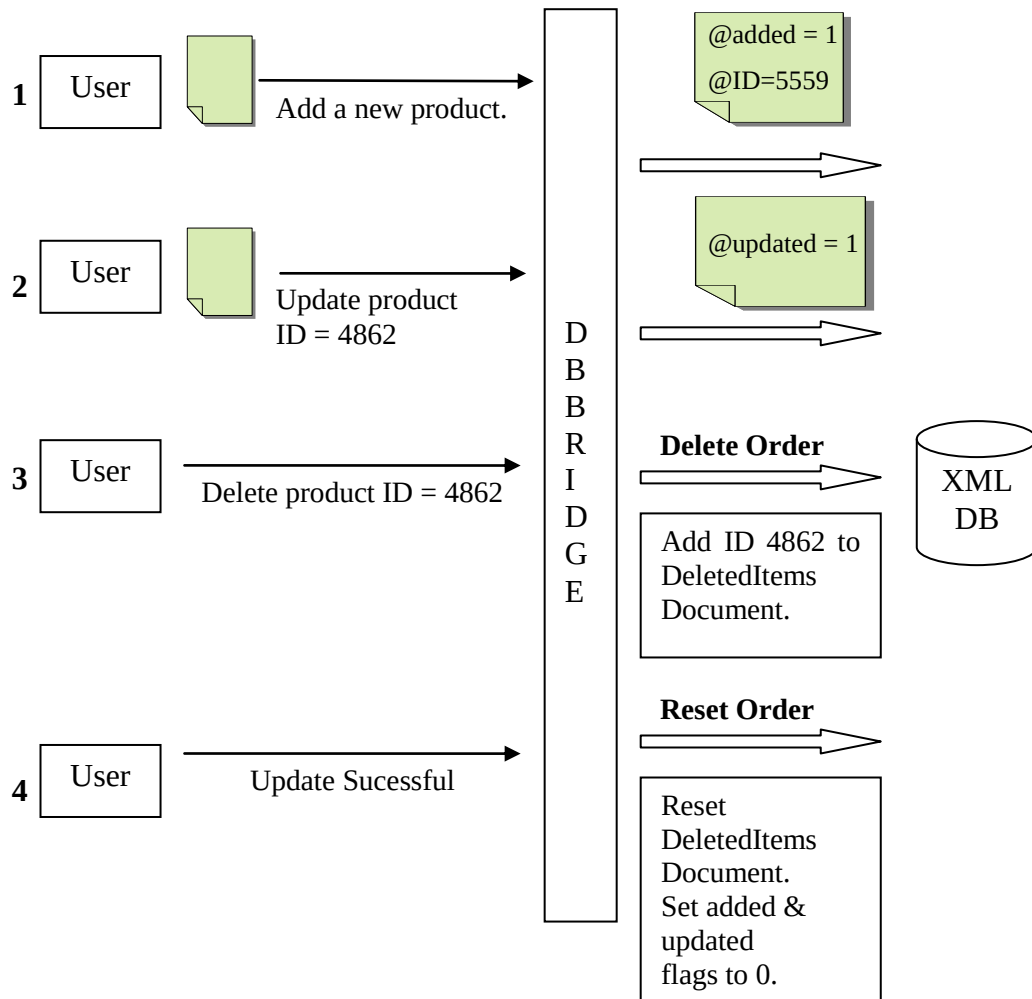


Figure 7.5 Platform Specific Mechanisms

Platform specific options are described in Fig. 7.5 with for major possible actions of the user. These are:

1. In the 'config' collection, there are two documents; 'IDDocument' and 'DeletedItemsDocument', which have a critical role in the platforms architecture.

The IDDocument holds the recently used maximum and minimum integers, which were used as unique IDs for documents in the system. When a user requests to store a new product document from the DB-Bridge, firstly the DB-Bridge increments the maximum positive integer value in the IDDocument, updates the IDDocument, assigns this integer value as the ID attribute to the product and then stores the product to the database. With this workflow every document has a unique ID in the system, except configuration documents which have their own IDs.

The DB-Bridge also marks the newly added product documents with the 'added' attribute and assigns the value '1'. When a user requests an update of the web catalog, the Manager requests all updated and added content from the DB-Bridge and prepares an update package. With this workflow the DB-Bridge can easily access all newly added content with a single XPATH query: */product[@added='1']*.

2. When the user requests an update of an existing document, DB-Bridge marks the document with 'updated' attribute and assigns the value '1'. This workflow is also constructed in order to access the updated content easily with a single XPATH query: */product[@updated='1']*.
3. Unique IDs of removed documents are stored in DeletedItemsDocument. When the user requests to delete an existing document, the DB-Bridge removes the document with the given ID, in this case 4862, and updates the DeletedItemsDocument. When a web catalog update is requested, this document will also be sent to Web Catalog Module and same documents will be deleted on the web server to achieve data consistency.
4. When a web update is successfully done, the DB-Bridge will be informed automatically by the Manager. In this case all 'added' and 'updates' attributes of all documents will be set to '0' and the content of the DeletedItemsDocument will be erased.

With such a workflow each modification of catalog content is controlled and is applied to both parts of the platform; the Electronic Catalog Server and the Web Catalog.

7.3 Manager

As described above, one or more Managers can connect to the DB-Bridge using TCP-IP and request database operations according to the user's actions on the GUI. In other words the Manager is only a GUI that translates user's actions for the DB-Bridge as database operations. All actions can be grouped as follows:

7.3.1 Navigation Related Operations

The user can navigate through different languages and their catalogs using the GUI. The user can add/delete groups on the catalog tree, add/delete or change the order of

products under a group. The constructed catalog tree will also be used in WEB and CD Catalog for easy navigation.

7.3.2 Product Family Related Operations

The user can add, delete a product family or modify (add/delete properties, change properties order, add/delete tables, select an XSL scheme for products, select search and listing criteria, select supplier specific properties, add/remove fork-ability to properties etc.) an existing one, which also modify all products that are children of the mentioned family, accordingly.

7.3.3 Product Related Operations

The user can add, delete or modify a product by changing property values, add or delete multimedia files and technical data. The user can also backup and restore current product database.

7.3.4 Supplier Related Operations

The user can add/remove a supplier or modify an existing supplier. Each supplier has a group and properties can be associated to these groups so that the property is only visible to the visitor if he/she has an authentication key.

7.3.5 Data Sharing Operations

The user can update the web site of the company, create a new CD catalog.

7.3.6 Administrative Operations

The administrator can add/delete and modify property groups, add/delete and change user accounts and has the right to modify every component in the system such as adding and deleting a language.

7.4 Integration

The integration layer uses the Manager's graphical components but it maintains its independency. Such a structure is chosen to construct a scheduling mechanism on it. The user can define integration tasks on the system, for example we can assume a situation such that our system gets the stock information of the products from an ERP system and updates the web every night at 02.00 am.

When the user schedules such a task, the operation takes place automatically, avoiding human mistakes, reducing workload and maintaining the up to dateness and uniqueness of data.

The user can also get reports in Excel or PDF format.

8. WEB CATALOG

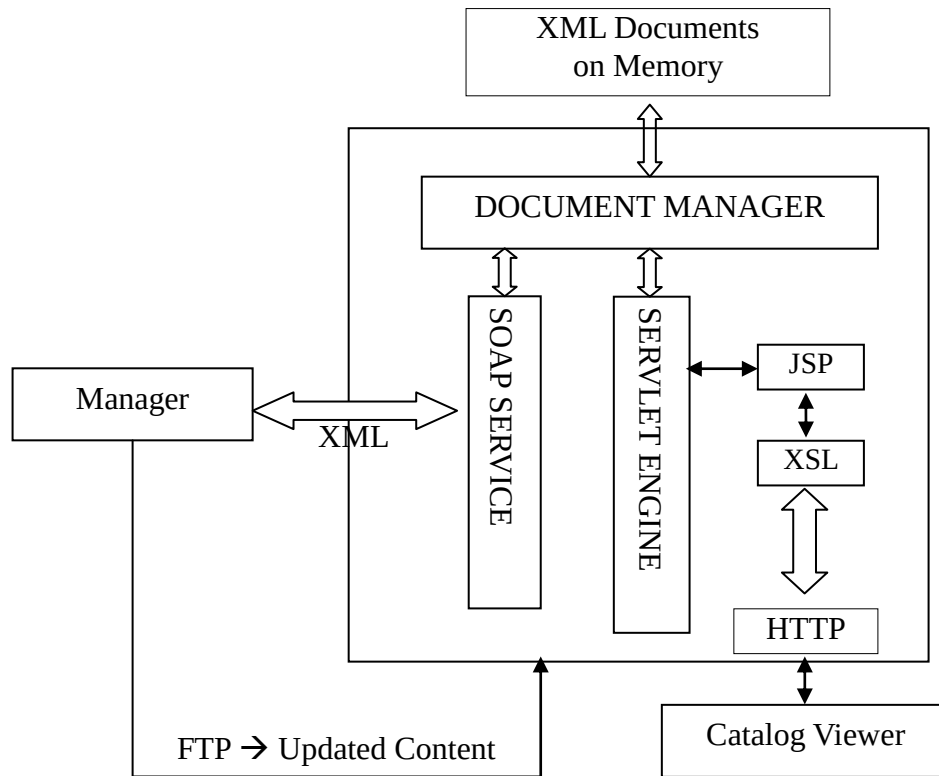


Figure 8.1 Web Catalog Architecture.

The Web Server consists of three major components:

- **SOAP Service:** SOAP is a lightweight XML based protocol for exchange of information in a decentralized, distributed environment. The manager talks to the SOAP server by using HTTP protocol and XML messages. SOAP service is responsible for processing updated content and makes it visible to catalog viewers on the internet.
- **Servlet Engine:** Servlet Engine is developed with J2EE platform. It consists of Servlets and JSPs, which are responsible to publish HTML pages to visitors by processing XML files.
- **Document Manager:** Document Manager also uses J2EE architecture. All XML documents are controlled by this layer. Document Manager is responsible to serve as a data resource for Servlets and JSPs and it is also

responsible to update the catalog content on the hard disk and reload catalog documents from hard disk to memory if necessary.

An innovative point of the WEB architecture is that XML data is kept in memory. Hard copy of all products and configuration files are stored on hard disk, but copies of these documents are loaded into memory. We can achieve such a workflow successfully due to the XML data structure. This workflow brings two advantages; firstly we do not need a database, which lowers the expenses, and secondly the performance of the server increases enormously, since every operation is done on RAMs. With the improvement in hardware technology, the cost of hardware components – RAM in this case – becomes much lower than database software, a big advantage for our system.

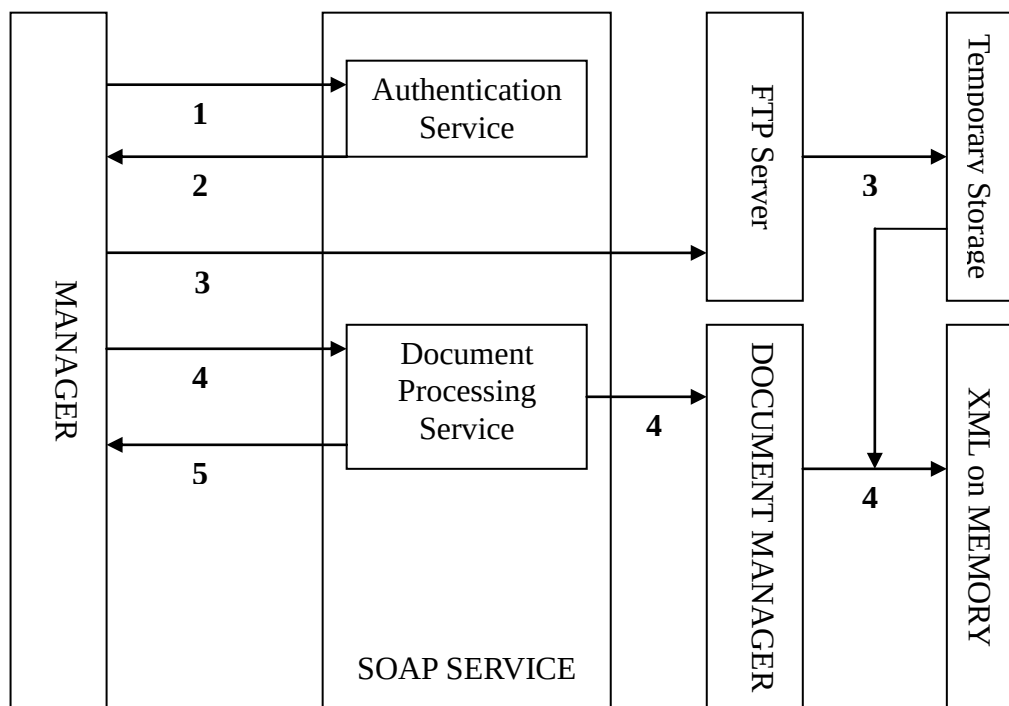


Figure 8.2 Web Catalog Update Workflow.

In Figure 8.2 the update procedure of the Web Catalog is described. Update procedure runs as follows.

The Manager requests the updated content from the DB-Bridge and prepares an update package. The DB-Bridge keeps track of the updated and newly added data after every successful update of the web. Then it sends the authentication key to the SOAP Authentication service. The message is like:

```
<request>
  <username>demo</username>
  <password>demo</password>
</request>
```

If the key is valid, a message including information of the currently idle FTP Server address and FTP authentication key will be returned in a response message:

```
<response>
  <ftp>160.75.38.152</ftp>
  <ftpusername>ftpu</ftpusername>
  <ftppassword>ftpp</ftppassword>
</response>
```

Until this message the Manager does not have any information about the address to send the updated content.

The Manager connects to the FTP server and sends the update package, which is stored to a temporary storage.

As a final step the SOAP Document Processing Service will be informed by the Manager that the upload process is complete with the authentication key. The SOAP service requests from the Document Manager to process the update content, which means that the Document Manager reads the content from the temporary storage and updated necessary files on the memory. As a final step documents on the hard disk will also be updated.

Then the Manager is informed that the update process is completed. If successful XML documents, multimedia content of products and configuration files are updated, search engine is reconstructed etc one the Web Catalog.

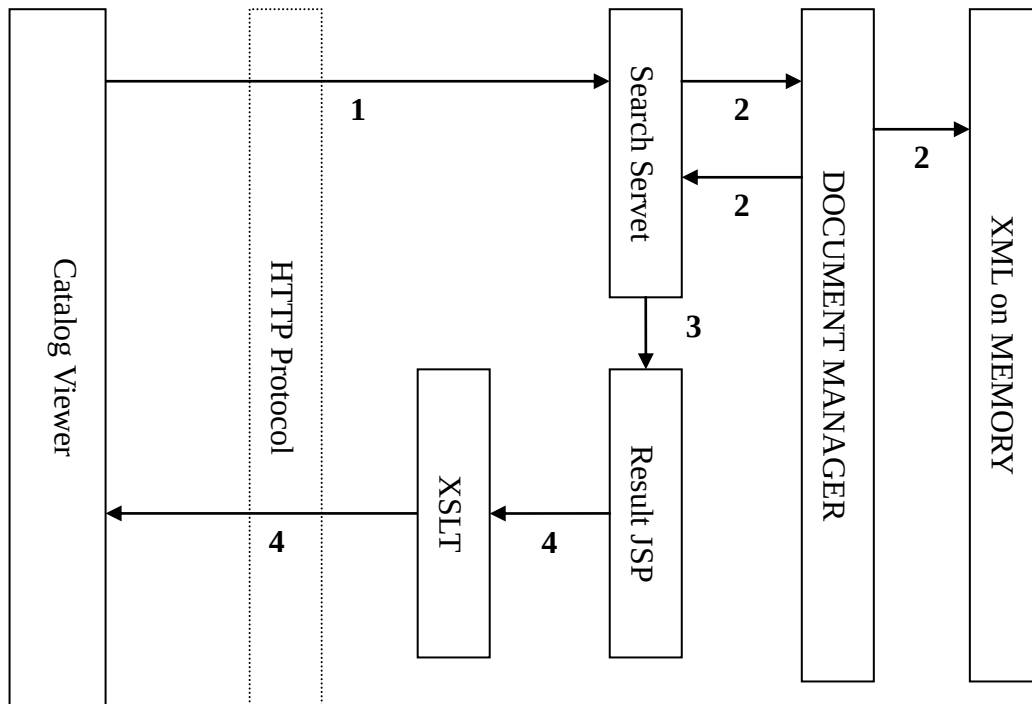


Figure 8.3 An Example of Catalog Viewer Actions.

Figure 8.3 demonstrates a sample catalog viewer who performs a search within the products in a catalog using Internet. The figure shows actions performed by the Web Catalog during processing XML documents and publishing a search result page to the catalog viewer.

Catalog viewer performs a search by filling a form on the browser and this request is sent to the Search Servlet.

First of all Search Servlet has to access related XML document which are kept on memory. Since Document Manager Layer is responsible for serving these documents, Search Servlet requests these documents from the Document Manager.

After accessing related documents, a search is performed on these documents using XPATH query, which is constructed according to the selections of the catalog viewer. The result set is send to the Search Result JSP.

Search Result JSP processes the result set if necessary and sends the to XSLT (XSL Transformation) Layer, which transforms XML documents to HTML content and publishes them to the catalog viewer as search results.

An advantage of this workflow is as mentioned above; if for example we want to change the design of the company's web site, we neither change the product database nor the software components; we only change the XSL templates, because only

XSLT layer is responsible for the visualization of HTML content.

The WEB server has an authentication mechanism for company's suppliers that are defined by the Manager. If a property of a product family is assigned to a supplier group, unauthorized standard users cannot see this information. An example for such information can be the discounted price or the current stock of a product.

Another mechanism, supported by the WEB is electronic shopping system. While navigating through products, the visitor or the supplier can build up a basket and can either request for a quotation or buy the products using secure socket layer and common transaction mechanisms.

9. CD CATALOG

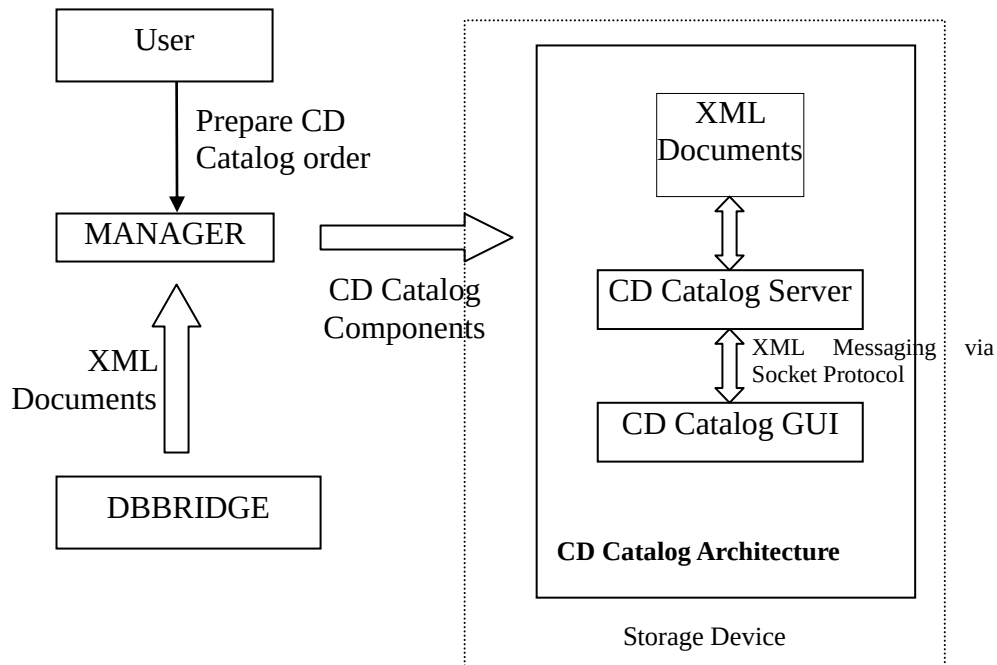


Figure 9.1 CD Catalog Arhitecture.

When the user requests a new CD catalog from the Manager, first of all the Manager retrieves the catalog content from the DB-Bridge and processes this content for the CD Catalog. XML documents are the dynamic components of the CD Catalog architecture. On the next step the Manager copies the static pre-compiled modules to the storage device such that a master CD is built.

Considering main features, the CD Catalog is very similar to the WEB Catalog.

As for the architecture, instead of the Servlet Engine of the WEB Catalog, server-client architecture is used. The server side is written with JAVA and is responsible for processing the XML documents. XML documents are being hold in memory like on the WEB. The reason for choosing JAVA on the server side is the advantage of using same classes that are used in DB-Bridge and Manager.

The client is written with Macromedia Flash [28] and is responsible for the GUI of the CD Catalog. The requests and responses are sent to the server via a socket using Macromedia's XML Messaging Architecture.

The reason for choosing Flash for the GUI is its dynamic structure, and friendly and interactive graphical components it has.

The major advantage of this architecture is that there is no need for an installation to run the CD. All files remain on the CD, for optimal resource usage.

10. TEST RESULTS

In this thesis, the idea of using XML as a data source is presented starting from its data structure, implementation and constitution of the architecture respectively. Since now, the system answers all needs of desired ECMS features. By measuring the performance, according to the architecture, the sufficiency of the whole platform will be proved. In order to claim this sufficiency, the platform needs to be tested extensively. It is needed to show the stability of the architecture when the components and variables are changed. The solidity of the system should be forced by some stress tests.

10.1 ECMS Performance Test

In the manager architecture, the manager accesses to the native XML DB by using the DB-Bridge layer. There are several subjects to measure the performance between the layers that forms the manager architecture.

Firstly, the whole platform should be stable, when the size of data that kept in the XML-DB increases. This is completely related to the chosen XML-DB architecture and the indexing mechanism.

Secondly, when the manager requests huge numbers of data at a time, the total size of the response of XML DB can exceed the memory limit of the system. The DB-Bridge uses a paging mechanism in order to solve the out of memory problems, but sending the product page by page causes a reasonable performance loss. The optimum page size should be used for the paging mechanism that ensures the memory sufficiency and minimum delay.

The main goal of these tests will be to show the compatibility of the manager architecture layers, by finding the suitable page size according to the number of the products kept in the system and to prove that the number of the products in the database does not affect the performance. In all tests, the Manager uses a total RAM of 64 Mbytes.

Three variables are measured during the tests;

XPATH Time: The time that XML DB spends for querying product documents.

Connection Time: The time that Manager needs to connect to the DB-Bridge layer.

Process Time: The time that Manager needs to get product documents page by page and to process them. This time interval also includes XPATH times.

Total Time: Is the sum of Connection and Process Time.

Variant (σ^2), standard deviation (σ): All calculations are done five times; the variation and the standard deviation of these measurements are also calculated to show the accuracy of test results.

10.2 ECMS Performance Test Variables

10.2.1 Test Scenarios

Defines the scenarios used in the test described below:

10.2.1.1 Scenario 1

Manager requests all product documents in the system from the DB-Bridge and performs xml operations on these documents.

Such an action can be reconstructing web server content, publishing a CD catalog; publishing PDF & Excel documents or integrate the content with another system such as databases, ERPs etc.

Since xml operations are done on every product document, the latency related to these operations is not calculated.

10.2.1.2 Scenario 2

Manager requests a single product document from the DB-Bridge and performs xml operations on this document. Such an action can be changing a property of a product. Since xml operations are done on every product document, the latency related to these operations is not calculated.

10.2.2 Number of Products

Defines the number of products in the system.

10.2.3 PageSize Value

Defines the page size value, which is used by the DB-Bridge.

10.2.4 DB-Bridge Server

The server, which runs the DB-Bridge application. This test is run with a single variation.

Server Configuration:

INTEL PENTIUM 4 2400 MHz.

Cash Size: L1 8KB, L2 512 KB.

Bus Speed: 400 MHz.

Ram Type, Size: DDR Ram, 768 Mbytes.

Main Board: Desktop PC Architecture

Operating System: Windows 2000 Professional

10.2.5 Manager Client

The client, which runs the Manager application. This test is run with a single variation.

Client Configuration:

INTEL PENTIUM 4 2000 MHz.

Cash Size: L1 8KB, L2 512 KB.

Bus Speed: 333 MHz.

Ram Type, Size: RD Ram, 256 Mbytes

Main Board: Desktop PC Architecture

Operating System: Windows 2000 Professional

10.2.6 JVM

Two different JVM are used in the tests, SUN JVM 1.4.2_03 and BEA JROCKIT 1.4.2_03.

BEA WebLogic JRockit is a Java Virtual Machine (JVM), developed uniquely for server-side applications and optimized for Intel architectures. JRockit ensures reliability, scalability, manageability, and flexibility for Java applications and provides seamless interoperability across multiple hardware and operating system configurations [29, 30].

10.2.7 Network Connection

The network connection is 100 Mbits/s.

10.3 Test Results

10.3.1 Scenario 1-A

The product size is constant and equals to 250 products. This test shows the architecture's product processing speed according to the PageSize value by using SUN's JVM.

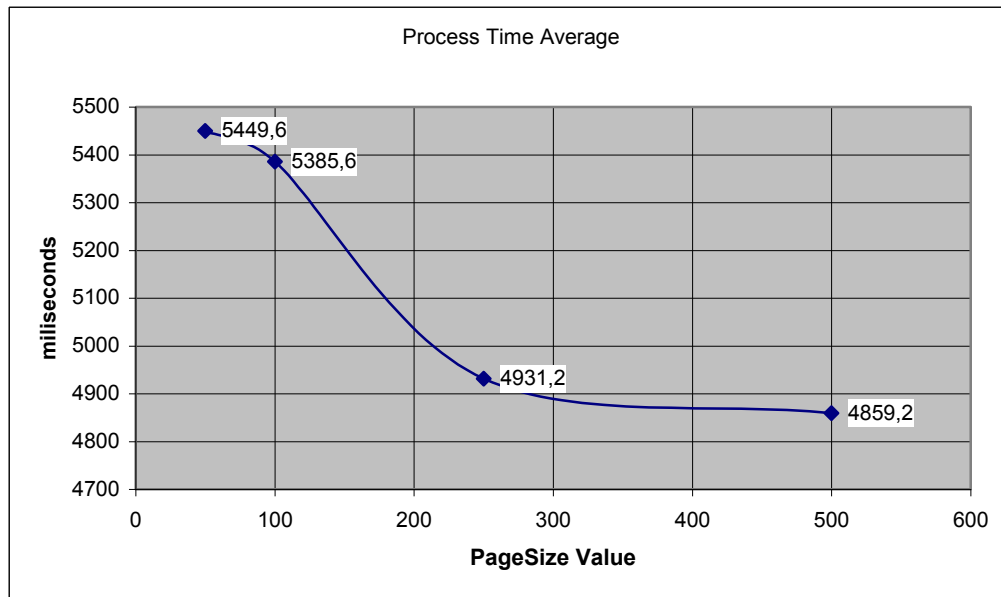


Figure 10.1 Scenario 1-A Process Time Averages

Table 10.1 Scenario 1-A Test Results

	XPATH (ms)	Connection (ms)	Process (ms)	Total (ms)
PageSize = 50				
Average	781	991	4416	5407
σ^2	91,66667	513,5833	356,9167	1642
σ	9,574271	22,66238	18,89224	40,5216
PageSize = 100				
Average	727	1033,2	4352,4	5385,6
σ^2	2291,667	2223,583	35566,92	24642
σ	47,87136	47,15489	188,5919	156,9777

	XPATH (ms)	Connection (ms)	Process (ms)	Total (ms)
PageSize = 250				
Average	647	999,4	3931,8	4931,2
σ^2	2291,667	295,3333	14862	16333,33
σ	47,87136	17,18527	121,9098	127,8019
PageSize = 500				
Average	643,2	1001,6	3857,6	4859,2
σ^2	1236,917	132,9167	22351,58	24208,33
σ	35,16983	11,52895	149,5045	155,5903

As the results show, with the increase of PageSize variable, the performance of the platform increases, because the product documents are transported between DB-Bridge and Manager in fewer steps.

10.3.2 Scenario 1-B

The product size is constant and equals to 250 products. This test shows the architecture's product processing speed according to the PageSize value by using SUN's JVM.

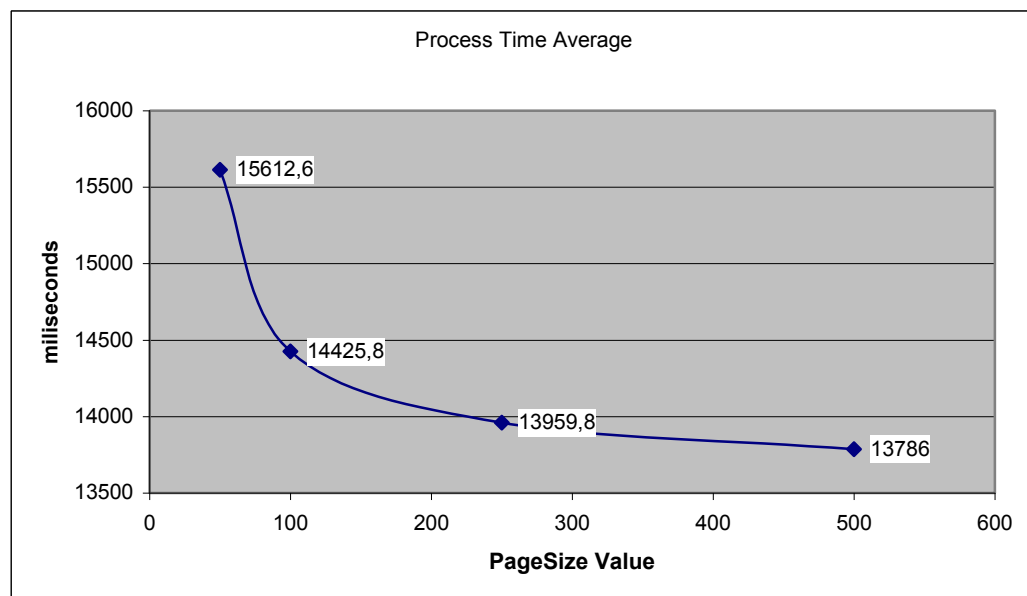


Figure 10.2 Scenario 1-B Process Time Averages

Table 10.2 Scenario 1-B Test Results

	XPATH (ms)	Connection (ms)	Process (ms)	Total (ms)
PageSize = 50				
Average	3090,2	995,4	14617,2	15612,6
σ^2	54530,25	36,91667	157126,9	160361
σ	233,5171	6,075909	396,3924	400,451
PageSize = 100				
Average	2759,6	1016	13409,8	14425,8
σ^2	100115	100	213262,9	218592,9
σ	316,4095	10	461,804	467,5392
	XPATH (ms)	Connection (ms)	Process (ms)	Total (ms)
PageSize = 250				
Average	2752	995,2	12964,6	13959,8
σ^2	3691,667	200	58028,67	56355,33
σ	60,75909	14,14214	240,8914	237,3928
	XPATH (ms)	Connection (ms)	Process (ms)	Total (ms)
PageSize = 500				
Average	2515,6	1131,8	12654,2	13786
σ^2	35133,58	85767,33	212450,3	63486,92
σ	187,4395	292,8606	460,9233	251,9661

As the results show, with the increase of PageSize variable, the performance of the platform increases, because the product documents are transported between DB-Bridge and Manager in fewer steps.

10.3.3 Scenario 1-C

The product size is constant and equals to 5750 products. This test shows the architecture's product processing speed according to the PageSize value by using SUN's JVM. In Scenario 3 there is also a stress test included, which measures the platform's responses by high PageSize values.

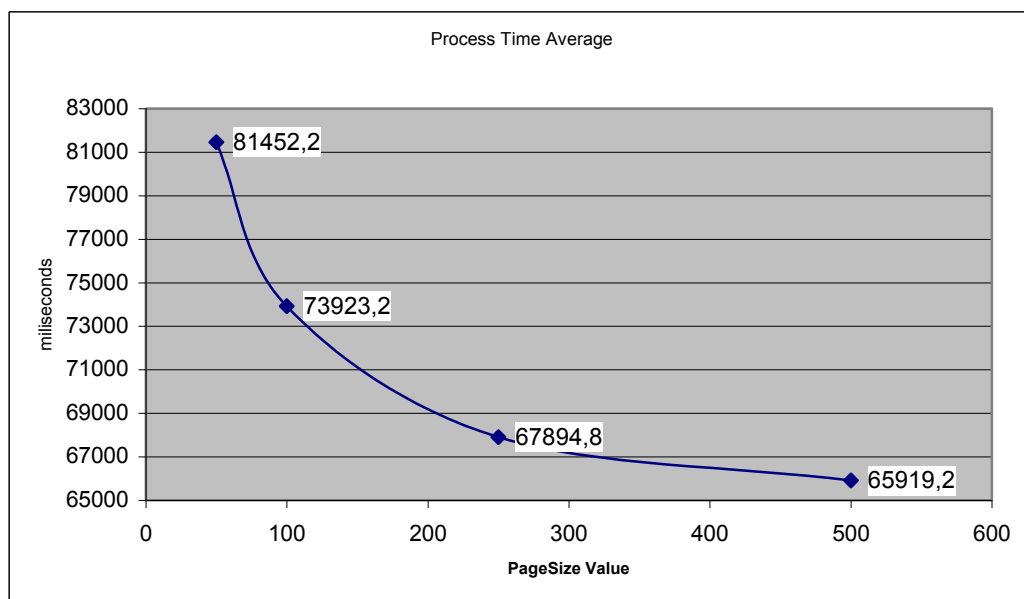


Figure 10.3 Scenario 1-C Process Time Averages

Table 10.3 Scenario 1-C Test Results

	XPATH (ms)	Connection (ms)	Process (ms)	Total (ms)
PageSize = 50				
Average	17431	1373,6	80078,6	81452,2
σ^2	62233,33	0	2586763	2586763
	XPATH (ms)	Connection (ms)	Process (ms)	Total (ms)
PageSize = 50				
σ	249,4661	0	1608,342	1608,342
PageSize = 100				
Average	16621,2	1004,8	72918,4	73923,2
σ^2	3978,25	294	218330,9	234032,3
σ	63,07337	17,14643	467,2589	483,7688
PageSize = 250				
Average	15652,6	993,2	66901,6	67894,8
σ^2	84437,67	66,66667	473945,7	467979
σ	290,5816	8,164966	688,4371	684,0899
	XPATH (ms)	Connection (ms)	Process (ms)	Total (ms)
PageSize = 500				
Average	14731,6	987	64932,2	65919,2
σ^2	30026,25	33,33333	88,91667	212,25
σ	173,2808	5,773503	9,429563	14,5688

Table 10.4 Scenario 1-C Stress Test Results

PageSize	XPATH (ms)	Connection (ms)	Process (ms)	Total (ms)
2500	14611	1032	OutOfMemoryException	N/A
2000	14571	1001	OutOfMemoryException	N/A
1500	14620	1603	OutOfMemoryException	N/A
1000	14581	982	63821	79384

As the results show, with the increase of PageSize variable, the performance of the platform increases, but when the PageSize value exceeds 1000, the Manager throws java.lang.OutOfMemory exception, as shown in Table 10.4, because the product's size climbs over the JVM's memory size.

10.3.4 Scenario 1-D

The product size is constant and equals to 5750 products. This test shows the architecture's product processing speed according to the PageSize value by using BEA JRockit JVM. In Scenario 3 there is also a stress test included, which measures the platform's responses by high PageSize values.

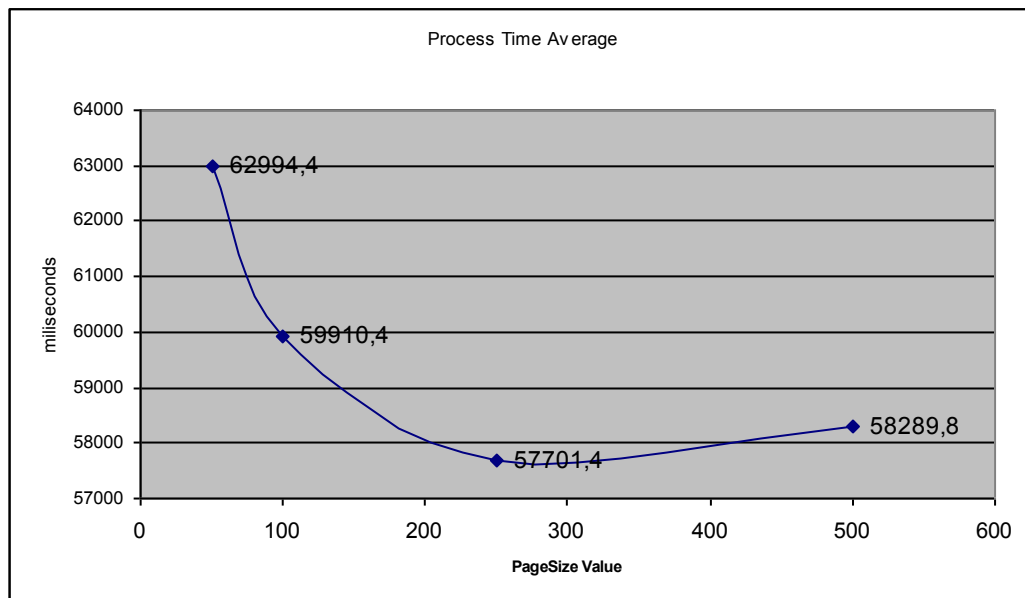


Figure 10.4 Scenario 1-D Process Time Averages

Table 10.5 Scenario 1-D Test Results

	XPATH (ms)	Connection (ms)	Process (ms)	Total (ms)
PageSize = 50				
Average	12179,4	1017,4	61977	62994,4
σ^2	234467,6667	1248,666667	304153,6667	321152,3333
σ	484,2186145	35,33647785	551,5012844	566,703038
PageSize = 100				
Average	11929,2	1013,4	58897	59910,4
σ^2	502	28,66666667	299210,25	301894,25
σ	22,4053565	5,354126135	547,0011426	549,4490422
PageSize = 250				
Average	12304,4	1047,6	56653,8	57701,4
σ^2	33856,91667	95,33333333	212999	212147,6667
σ	184,0024909	9,763879011	461,518147	460,5949052
PageSize = 500				
Average	12888,8	1139,6	57150,2	58289,8
σ^2	1183813,667	87196,91667	860296,25	830016,6667
σ	1088,032015	295,2912404	927,5215631	911,0525049

Table 10.6 Scenario 1-D Stress Test Results

PageSize	XPATH (ms)	Connection (ms)	Process (ms)	Total (ms)
2500	14611	1032	Exception	N/A
2000	14571	1001	Exception	N/A
1500	26268	1041	Exception	N/A
1000	11981,2	1129,8	58582,4	59712,2

As the results show, with the increase of PageSize variable, the performance of the platform increases, but when the PageSize value exceeds 1000, the Manager throws java.lang.OutOfMemory exception, as shown in Table 10.6, because the product's size climbs over the JVM's memory size. On the other hand, by using BEA JRockit JVM, the performance of the platform increases. But this is a virtual increase, because the results before JVM caches classes and documents are higher than the one of SUN's JVM.

10.3.5 Scenario 2

This test shows the architecture's product processing speed according to the PageSize value by using SUN's JVM and BEA JRockit JVM.

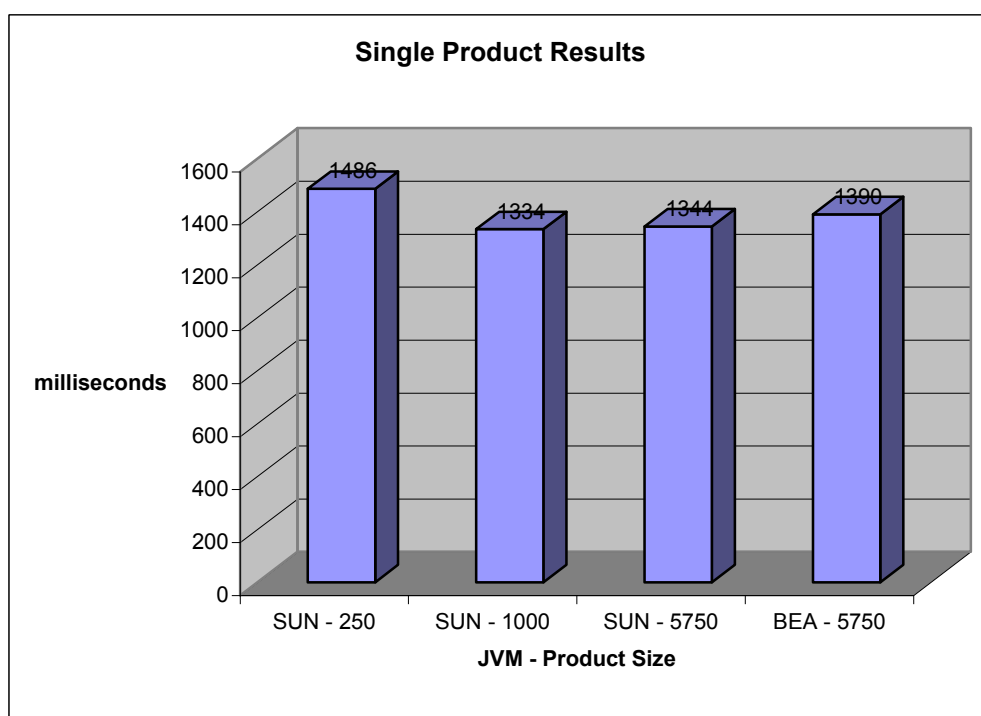


Figure 10.5 Scenario 2 Process Time Averages

Table 10.7 Scenario 2 Test Results

	XPATH (ms)	Connection (ms)	Process(ms)	Total (ms)
SUN – 250 Products	6	1155,6	324,4	1486
SUN – 1000 Products	18	985,2	348,8	1334
SUN – 5750 Products	14	989,4	354,6	1344
BEA – 5750 Products	4	1053,4	336,6	1390

The platform's product processing time is independent form the product count on the system. In every condition approximately same results are measured.

10.4 ECMS Performance Test Summary

If the product count is constant and PageSize variable is changed, average XPATH query times remains the same as expected, because XPATH time depends on only product count.

The results also show that connection time remains the same on every condition.

Graphics show clearly that process time is direct proportional to product count and process time decreases logarithmically to PageSize variable.

In this case we have the best performance when PageSize is set to maximum available value. With stress tests we have proven that the architecture becomes instable for PageSize variables ≥ 1500 . By using IBM's DOM benchmark test classes

[31], total size of 1000 products are measured as approximately 23 Mbytes when loaded into memory. These results show that the system crashes when 34.5 (1500*0.023) Mbytes limit is exceeded.

The formula to calculate the maximum consistent value for PageSize is:

A: Memory size of the product, which occupies maximum memory among all other products in the system.

B: Available memory on the system for paging mechanism. B = 34.5.

Optimum PageSize = B/A

In order to decrease the possibility of a system failure and system performance, we use this formula:

$\text{PageSize} \leq (B/A) * 0.80$

10.5 Web Catalog Server Test

In the web architecture, the system holds the catalog content in XML format, and uses the document manager system, which loads all catalog content into memory. These tests are based on the product count of the catalog, because product count varies from customer to customer and is the essence variable that affects the system performance. Another critical variable that will affect the performance is the size of the physical memory of the operating system.

The primary goal of these tests is to show the constancy of the average response time (time difference between the catalog viewers request and response actions using a standard web browser) for catalog viewer who performs operations over the HTTP protocol, which are defined as the unique product number search, advanced search and view single product.

10.5.1 Test Group 1

Constant: Product number

Conditions: 1- Physical memory is enough to hold all products

2- Physical memory is not enough to hold all products

The analysis of the test according to the separation defined above, we expect a decrease in the system performance. Since the system is based on “holding the

catalog content in memory”, system behavior is expected to be under the estimated performance after the test is completed for condition 2.

Constant: Sufficient Physical memory.

Conditions: 1- Number of product is 1000

2- Number of product is 10000

As a result of the analysis according to the separation defined above, we expect the system performance to be stable, if there is physical memory for the products; the response time for the user should be nearly the same. Otherwise the system behavior is expected to be under the estimated performance. As a result we can propose that; with enough memory in the system, the number of the product will not affect the performance of the system.

By using IBM’s DOM benchmark test classes [31], total size of 1000 products are measured as approximately 23 Mbytes when loaded into memory. As expected total size of 10000 products is 230 Mbytes when loaded into memory.

10.5.2 Test Group 2

There are some other important factors that will effects the performance which are independent from the ECMS architecture, such as the network traffic and the server’s hardware properties. By changing these independent variables, the performance change in the system will be examined.

10.5.3 Test Group 3

In order to find the perfect server configuration, the JVM used in the web server acts a very important role for the performance result because, the JVM itself determines the usage of the memory, and uses its own caching mechanism in itself. The servlet container TOMCAT [32] is a java based application and runs under the JVM. Since there are several JVMs for different purposes, we have to find the perfect JVM that is concordant to the ECMS architecture.

10.6 Web Catalog Server Test Variables

10.6.1 Test Scenario 1

SKU Search Test; catalog viewer selects single product element from document manager by running an XPATH like: /product/property[@mode = 8 and value() =

‘125-544-785’]

Catalog Viewer enters the SKU number of the product as a parameter in a HTML form. The SearchBySKU servlet queries all documents on the memory with the XPATH, until related product is found.

10.6.2 Test Scenario 2

Advanced search test; catalog viewer selects a single element from document manager by running an XPATH.

The user enters search criteria in several HTML forms. A session is assigned to the user by Tomcat for tracking user’s actions and holding search criteria.

As a final step the PerformSearch servlet is invoked, which queries all documents with XPATH queries and displays the results.

For this scenario, user enters specific criteria to find only one product.

The search engine forces the user to select one specific category at the first step. Therefore the document count to query is lower than in Scenario 1. If there are 10.000 documents divided into 5 categories, PerformSearch servlet has to query 2000 products. The complexity of Scenario 2 is lower than the complexity of Scenario 1.

10.6.3 Test Scenario 3

Advanced search test; catalog viewer selects all elements from document manager by running an XPATH.

The workflow is the same as in Scenario 2. As a difference the search result set consists of multiple documents instead of a single document. Therefore the complexity of the XPATH expression is lower than the one in Scenario 3. On the other hand the result set data is higher than the one of Scenario 2.

The complexity of Scenario 3 is approximately the same of Scenario 1.

10.6.4 Test Scenario 4

Catalog viewer selects single element by giving its unique ID.

The viewer selects a product from the catalog tree and the details of the product are displayed. Considering system architecture this Scenario has the least complexity.

10.6.5 Number of Products

Defines the number of products in the document manager.

10.6.6 Catalog Web Server

Two different web servers are used to test the system. The detailed configuration is given below:

10.6.6.1 Server 1 Configuration

INTEL CELERON 2000 MHz.

Cash Size: L1 8KB, L2 128 KB.

Bus Speed: 400 MHz.

Ram Type: DDR Ram

Main Board: Desktop PC Architecture

Operating System: REDHAT LINUX 9.0 with standard kernel

J2EE Server: Apache Tomcat Servlet Container 4.5

Web Server: Apache HTTP Server 2.1.1

10.6.6.2 Server 2 Configuration

INTEL PENTIUM 2400 MHz.

Cash Size: L1 8 KB, L2 512 KB.

Bus Speed: 400 MHz.

Ram Type: 1024 Mbytes DDR Ram

Main Board: Server Architecture

Operating System: REDHAT LINUX 9.0 with standard kernel

J2EE Server: Apache Tomcat Servlet Container 4.5

Web Server: Apache HTTP Server 2.1.1

10.6.7 Memory

Total size of the physical memory used in the server.

10.6.8 JVM

Two different JVM are used in the tests, SUN JVM 1.4.2_03 and BEA JROCKIT 1.4.2_03.

10.6.9 Network Connection

100 Mbits /s or 52 Kbits / s (Dial Up Connection) connections are used by the test.

10.7 Web Catalog Server Test Results

10.7.1 Scenario 1 Results

Table 10.8 Web Catalog Test, Scenario 1, Condition 1 Test Variables

Product Count :	1.000
Server Type:	Server 1
RAM (Mbytes):	512
JVM:	BEA JRockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	10 viewers click randomly.
Duration:	60 sec.

Expected Result:

We expect acceptable results, since there is enough memory on the system.

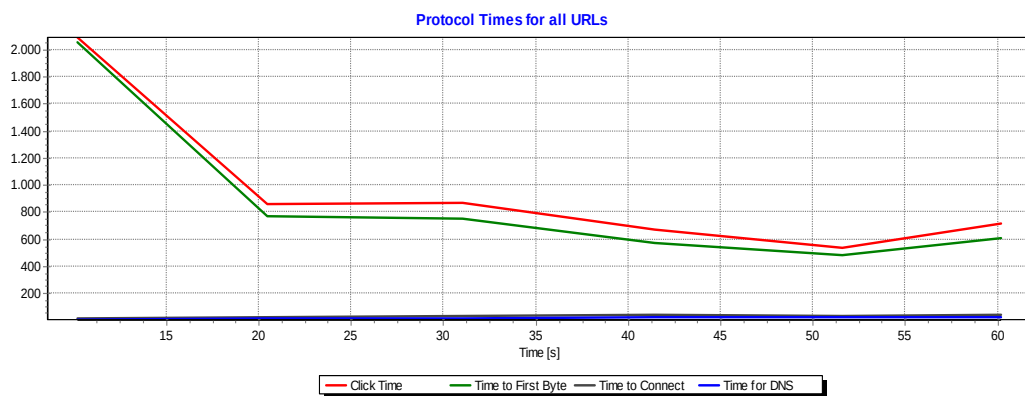


Figure 10.6 Web Catalog Test, Scenario 1, Condition 1 Test Results

Table 10.9 Web Catalog Test, Scenario 1, Condition 1 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	205	0	0	184928	902

Comments:

This test is successful because the Web Catalog Server responses in average 902 ms to the user, measured for 60 seconds long.

Table 10.10 Web Catalog Test, Scenario 1, Condition 2 Test Variables

Product Count :	1.000
Server Type:	Server 1
RAM (Mbytes):	256
JVM:	BEA JRockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	10 viewers click randomly.
Duration:	60 sec.

Expected Result:

We expect same results as in Condition 1, because there is still enough memory on the system, although physical memory is decreased by 50%.

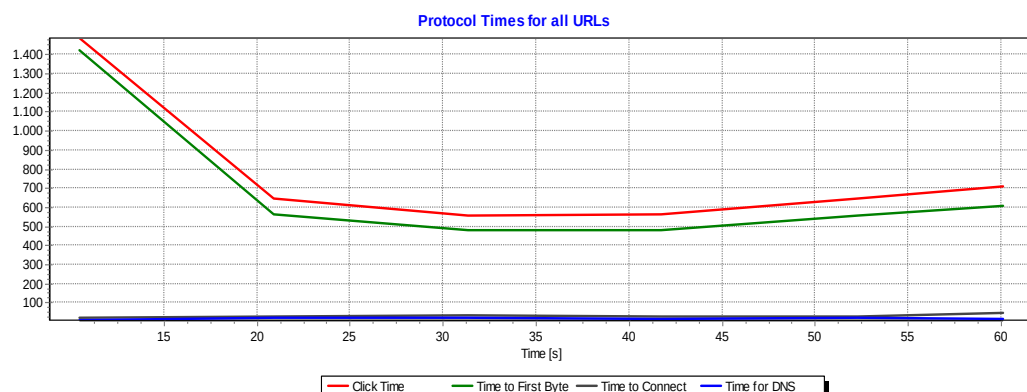


Figure 10.7 Web Catalog Test, Scenario 1, Condition 2 Test Results

Table 10.11 Web Catalog Test, Scenario 1, Condition 2 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	216	0	0	160141	741

Comments:

Although physical memory size is decreased by 50 %, the average response time of the Web Catalog Server is approximately the same. The test is successful.

Table 10.12 Web Catalog Test, Scenario 1, Condition 3 Test Variables

Product Count :	10.000
Server Type:	Server 1
RAM (Mbytes):	512
JVM:	BEA JRockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	10 viewers click randomly.
Duration:	60 sec.

Expected Result:

The product count is increased by 1000 %. But there is enough memory on the system to hold the catalog content on physical memory. A small increase is expected because the document count to query is 10 times more than in Condition 1 and Condition 2.

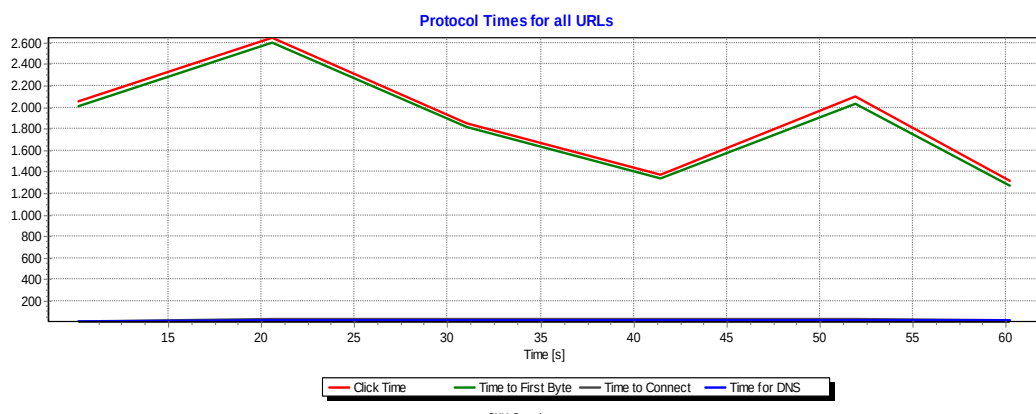


Figure 10.8 Web Catalog Test, Scenario 1, Condition 3 Test Results

Table 10.13 Web Catalog Test, Scenario 1, Condition 3 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	101	0	0	190918	1907

Comments:

There is an increase on the average click time, but the system is still healthy, because document size is increased by 1000% but average click time is only increased by 100%. Test is successful.

Table 10.14 Web Catalog Test, Scenario 1, Condition 4 Test Variables

Product Count :	10.000
Server Type:	Server 1
RAM (Mbytes):	256
JVM:	BEA JRockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	1 viewer clicks randomly.
Duration:	120 sec.

Expected Result:

The system will collapse down since all documents cannot be loaded into physical memory and the system will use virtual memory, which is much slower than physical memory. The catalog viewer count is decreased to 1 because otherwise the system stops responding and no calculations can be made.

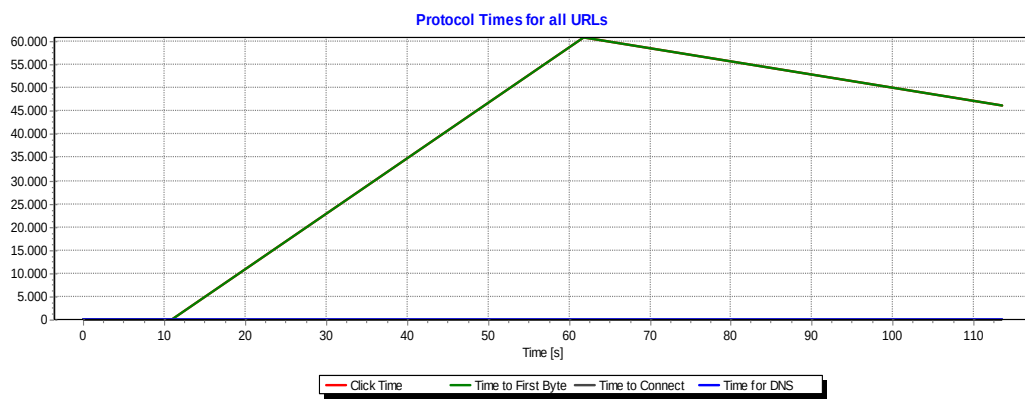


Figure 10.9 Web Catalog Test, Scenario 1, Condition 4 Test Results

Table 10.15 Web Catalog Test, Scenario 1, Condition 4 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	2	0	0	106829	53415

Comments:

The result 53415 ms is not acceptable. The system slowed down by 2700%. Test is successful.

Table 10.16 Web Catalog Test, Scenario 1, Condition 5 Test Variables

Product Count :	3.000
Server Type:	Server 1
RAM (Mbytes):	512
JVM:	SUN JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	6 viewers click randomly.
Duration:	60 sec.

Expected Result:

The system will slow down because SUN JVM is not optimized for server side applications. The document size is decreased to 3000 and the catalog viewer size is decreased to 6 in order to avoid OutOfMemory exceptions.

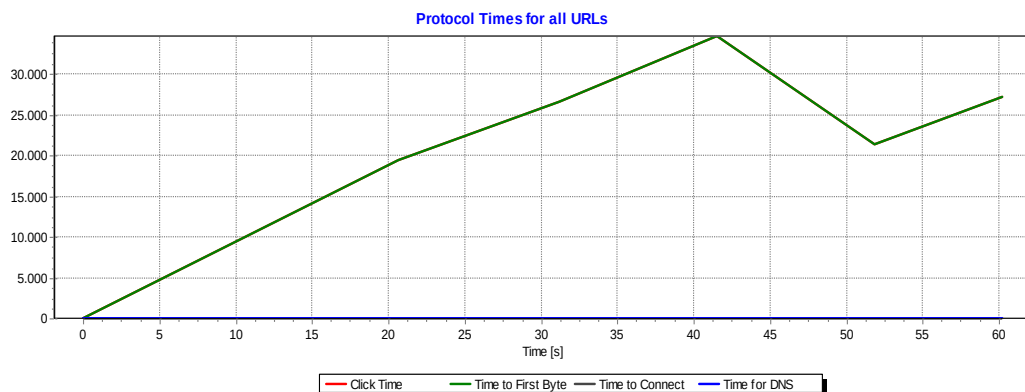


Figure 10.10 Web Catalog Test, Scenario 1, Condition 5 Test Results

Table 10.17 Web Catalog Test, Scenario 1, Condition 5 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	8	0	0	217777	27222

Comments:

A 1300% increase of average click time is measured. Sun JVM is not suitable for our application. Test is successful.

10.7.2 Scenario 2 Results

Catalog Viewer selects a single product using advanced search engine.

Table 10.18 Web Catalog Test, Scenario 2, Condition 1 Test Variables

Product Count :	1.000
Server Type:	Server 1
RAM (Mbytes):	512
JVM:	BEA Jrockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	10 viewers click randomly.
Duration:	60 sec.

Expected Result:

We expect acceptable results, since there is enough memory on the system.

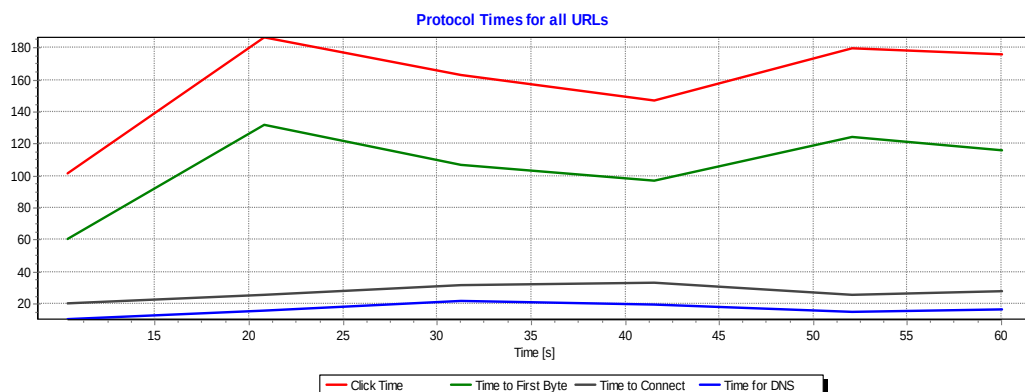


Figure 10.11 Web Catalog Test, Scenario 2, Condition 1 Test Results

Table 10.19 Web Catalog Test, Scenario 2, Condition 1 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	59	0	0	46820	822

Comments:

This test is successful because the Web Catalog Server responses in average 822 ms to the user, measured for 60 seconds long.

Table 10.20 Web Catalog Test, Scenario 2, Condition 2 Test Variables

Product Count :	1.000
Server Type:	Server 1
RAM (Mbytes):	256
JVM:	BEA JRockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	10 viewers click randomly.
Duration:	60 sec.

Expected Result:

We expect same results as in Condition 1, because there is still enough memory on the system, although physical memory is decreased by 50%.

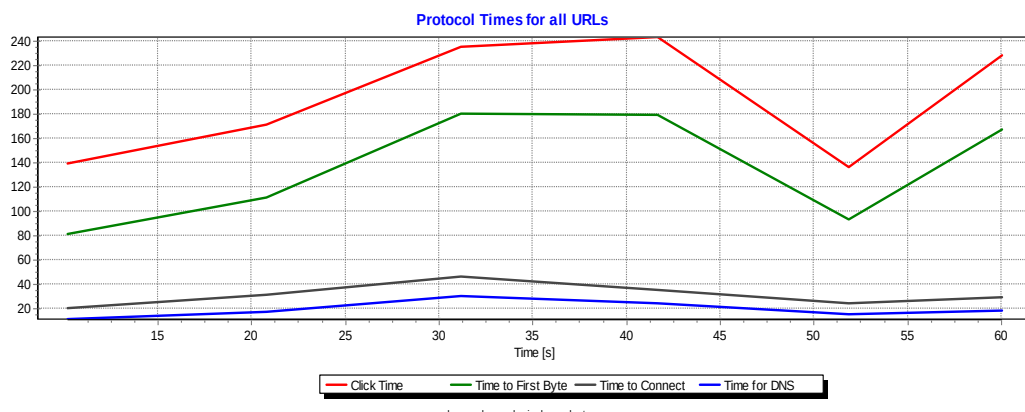


Figure 10.12 Web Catalog Test, Scenario 2, Condition 2 Test Results

Table 10.21 Web Catalog Test, Scenario 2, Condition 2 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	52	0	0	50626	972

Comments:

Although physical memory size is decreased by 50 %, the average response time of the Web Catalog Server is approximately the same. The test is successful.

Table 10.22 Web Catalog Test, Scenario 2, Condition 3 Test Variables

Product Count :	10.000
Server Type:	Server 1
RAM (Mbytes):	512
JVM:	BEA JRockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	10 viewers click randomly.
Duration:	60 sec.

Expected Result:

The product count is increased by 1000 %. But there is enough memory on the system to hold the catalog content on physical memory. A small increase is expected because the document count to query is 10 times more than in Condition 1 and Condition 2.

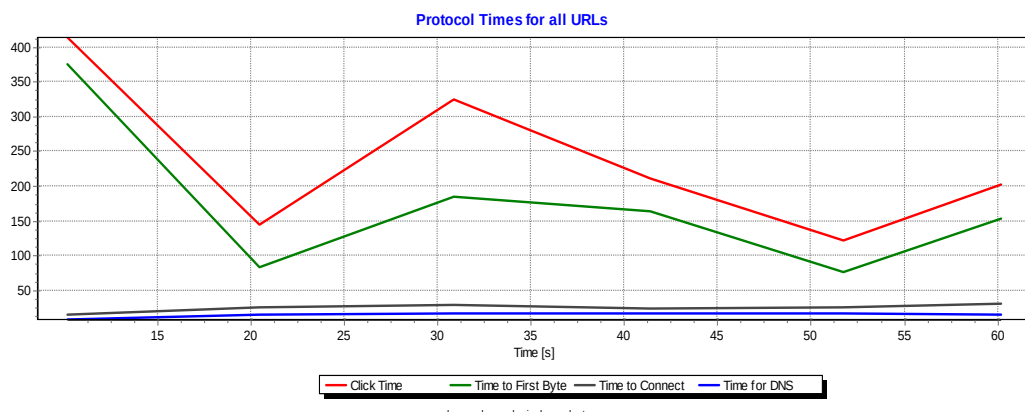


Figure 10.13 Web Catalog Test, Scenario 2, Condition 3 Test Results

Table 10.23 Web Catalog Test, Scenario 2, Condition 3 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	48	0	0	57645	1240

Comments:

There is an increase on the average click time, but the system is still healthy, because document size is increased by 1000% but average click time is only increased by 50%. Test is successful.

Table 10.24 Web Catalog Test, Scenario 2, Condition 4 Test Variables

Product Count :	10.000
Server Type:	Server 1
RAM	256
JVM	BEA JRockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	1 viewer clicks randomly.
Duration:	120 sec.

Expected Result:

The system will collapse down since all documents cannot be loaded into physical memory and the system will use virtual memory, which is much slower than physical memory. The catalog viewer count is decreased to 1 because otherwise the system stops responding and no calculations can be made.

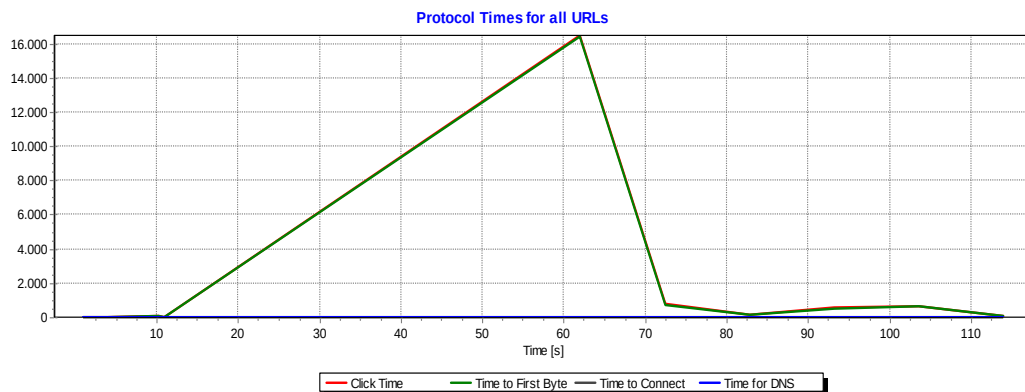


Figure 10.14 Web Catalog Test, Scenario 2, Condition 4 Test Results

Table 10.25 Web Catalog Test, Scenario 2, Condition 4 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	7	0	0	112465	14053

Comments:

The result 14053 ms is not acceptable for a single catalog viewer. The system slowed down by 1000%. Test is successful.

Table 10.26 Web Catalog Test, Scenario 2, Condition 5 Test Variables

Product Count :	3.000
Server Type:	Server 1
RAM (Mbytes):	512
JVM:	SUN JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	6 viewers click randomly.
Duration:	60 sec.

Expected Result:

The system will slow down because SUN JVM is not optimized for server side applications. The document size is decreased to 3000 and the catalog viewer size is decreased to 6 in order to avoid OutOfMemory exceptions.

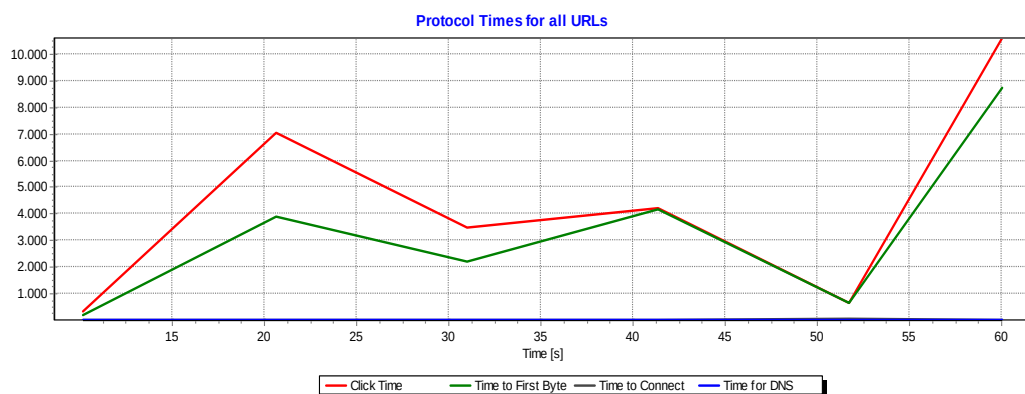


Figure 10.15 Web Catalog Test, Scenario 2, Condition 5 Test Results

Table 10.27 Web Catalog Test, Scenario 2, Condition 5 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	41	0	<u>79,17</u>	99070	21734

Comments:

A 1300% increase of average click time is measured and an error rate (responses over 25s are calculated as errors) of 79% is achieved, which is unacceptable. Test is successful.

10.7.3 Scenario 3 Results

Catalog Viewer selects all elements using advanced search.

Table 10.28 Web Catalog Test, Scenario 3, Condition 1 Test Variables

Product Count :	1.000
Server Type:	Server 1
RAM (Mbytes):	512
JVM:	BEA Jrockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	10 viewers click randomly.
Duration:	60 sec.

Expected Result:

We expect acceptable results, since there is enough memory on the system.

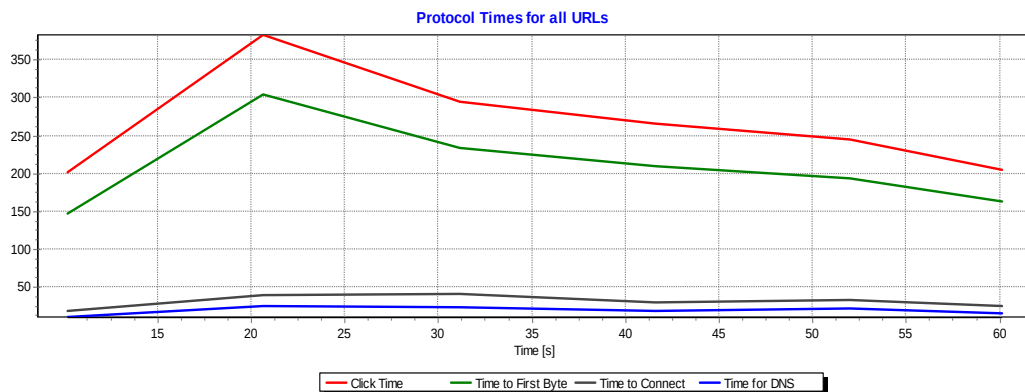


Figure 10.16 Web Catalog Test, Scenario 3, Condition 1 Test Results

Table 10.29 Web Catalog Test, Scenario 3, Condition 1 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	63	0	0	67616	1109

Comments:

This test is successful because the Web Catalog Server responses in average 1109 ms to the user, measured for 60 seconds long.

Table 10.30 Web Catalog Test, Scenario 3, Condition 2 Test Variables

Product Count :	1.000
Server Type:	Server 1
RAM (Mbytes):	256
JVM:	BEA JRockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	10 viewers click randomly.
Duration:	60 sec.

Expected Result:

We expect same results as in Condition 1, because there is still enough memory on the system, although physical memory is decreased by 50%.

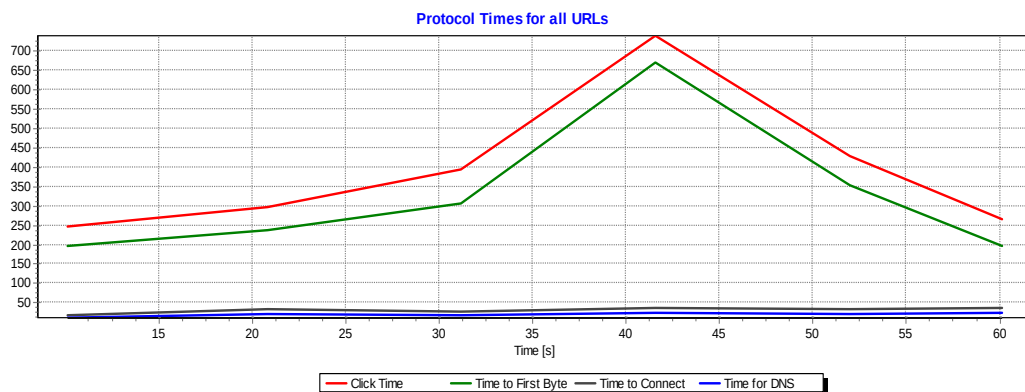


Figure 10.17 Web Catalog Test, Scenario 3, Condition 2 Test Results

Table 10.31 Web Catalog Test, Scenario 3, Condition 2 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	63	0	0	94873	1597

Comments:

Although physical memory size is decreased by 50 %, the average response time of the Web Catalog Server is approximately the same. The test is successful.

Table 10.32 Web Catalog Test, Scenario 3, Condition 3 Test Variables

Product Count :	10.000
Server Type:	Server 1
RAM (Mbytes):	512
JVM:	BEA Jrockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	10 viewers click randomly.
Duration:	60 sec.

Expected Result:

The product count is increased by 1000 %. But there is enough memory on the system to hold the catalog content on physical memory. A small increase is expected because the document count to query is 10 times more than in Condition 1 and Condition 2.

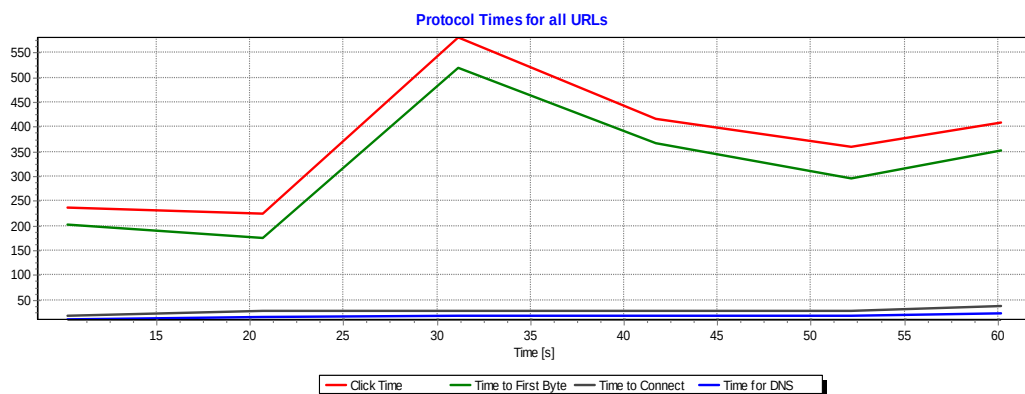


Figure 10.18 Web Catalog Test, Scenario 3, Condition 3 Test Results

Table 10.33 Web Catalog Test, Scenario 3, Condition 3 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	63	0	0	96205	1528

Comments:

There is an increase on the average click time, but the system is still healthy, because document size is increased by 1000% but average click time is only increased by 50%. Test is successful.

Table 10.34 Web Catalog Test, Scenario 3, Condition 4 Test Variables

Product Count :	10.000
Server Type:	Server 1
RAM (Mbytes):	256
JVM:	BEA JRockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	1 viewer clicks randomly.
Duration:	120 sec.

Expected Result:

The system will collapse down since all documents cannot be loaded into physical memory and the system will use virtual memory, which is much slower than physical memory. The catalog viewer count is decreased to 1 because otherwise the system stops responding and no calculations can be made.

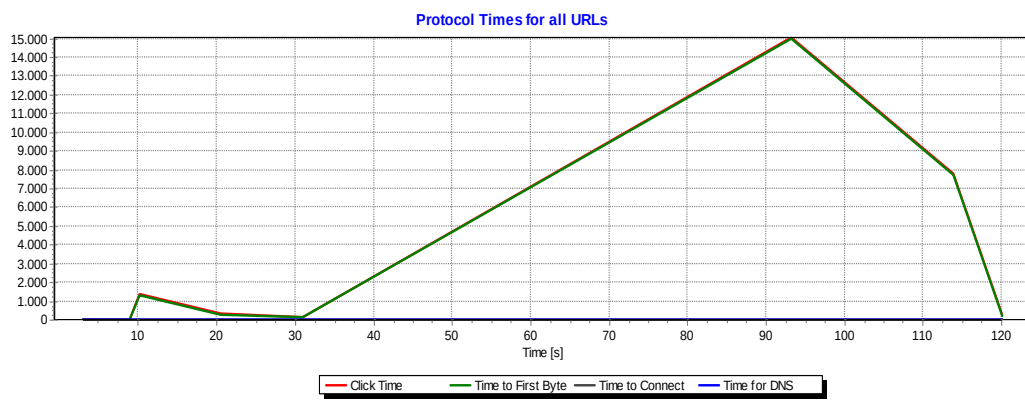


Figure 10.19 Web Catalog Test, Scenario 3, Condition 4 Test Results

Table 10.35 Web Catalog Test, Scenario 3, Condition 4 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	4	0	0	82506	27188

Comments:

The result 27188 ms is not acceptable for a single catalog viewer. The system slowed down by 1600%. Test is successful.

Table 10.36 Web Catalog Test, Scenario 3, Condition 5 Test Variables

Product Count :	3.000
Server Type:	Server 1
RAM (Mbytes):	512
JVM:	SUN JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	6 viewers click randomly.
Duration:	60 sec.

Expected Result:

The system will slow down because SUN JVM is not optimized for server side applications. The document size is decreased to 3000 and the catalog viewer size is decreased to 6 in order to avoid OutOfMemory exceptions.

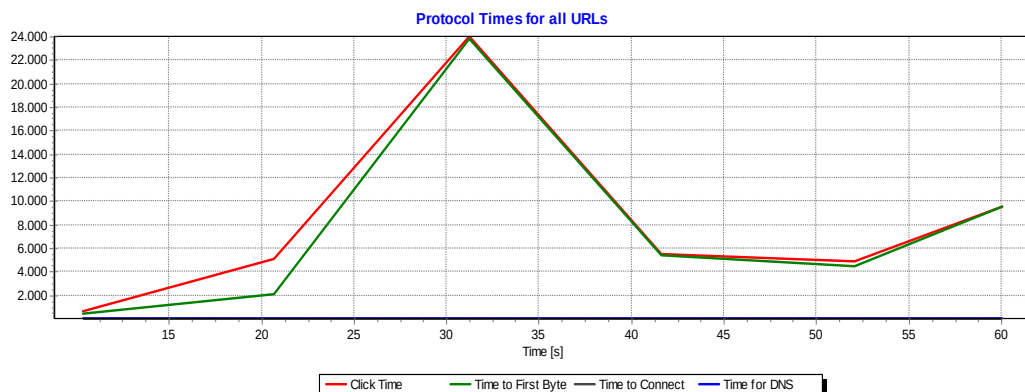


Figure 10.20 Web Catalog Test, Scenario 3, Condition 5 Test Results

Table 10.37 Web Catalog Test, Scenario 3, Condition 5 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	37	31	%83	159867	23610

Comments:

A 1300% increase of average click time is measured and the result 23160 ms is unacceptable. Test is successful.

10.7.4 Scenario 4 Results

Catalog viewer selects one product with unique ID.

Table 10.38 Web Catalog Test, Scenario 4, Condition 1 Test Variables

Product Count :	1.000
Server Type:	Server 1
RAM (Mbytes):	512
JVM:	BEA Jrockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	10 viewers click randomly.
Duration:	60 sec.

Expected Result:

We expect acceptable results, since there is enough memory on the system.

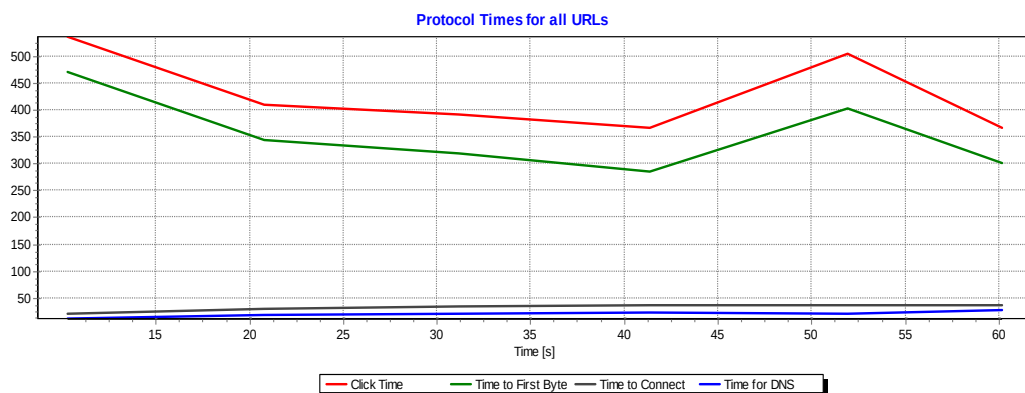


Figure 10.21 Web Catalog Test, Scenario 4, Condition 1 Test Results

Table 10.39 Web Catalog Test, Scenario 4, Condition 1 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	233	0	0	101826	437

Comments:

This test is successful because the Web Catalog Server responses in average 437 ms to the user, measured for 60 seconds long.

Table 10.40 Web Catalog Test, Scenario 4, Condition 2 Test Variables

Product Count :	1.000
Server Type:	Server 1
RAM (Mbytes):	256
JVM:	BEA Jrockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	10 viewers click randomly.
Duration:	60 sec.

Expected Result:

We expect same results as in Condition 1, because there is still enough memory on the system, although physical memory is decreased by 50%.

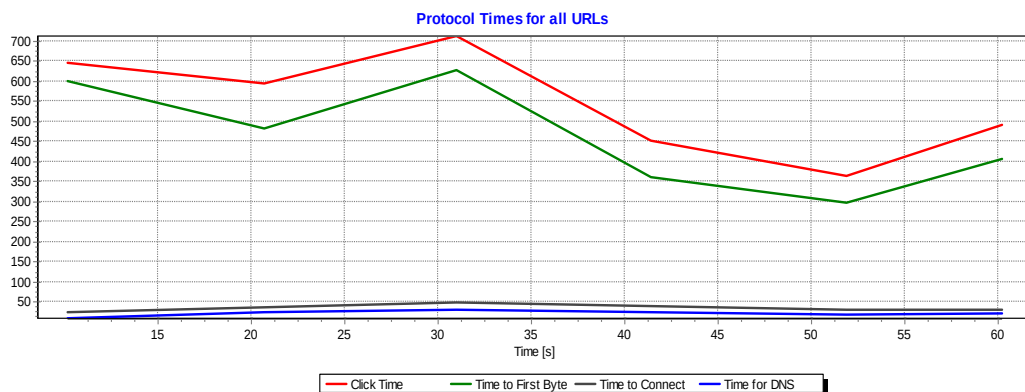


Figure 10.22 Web Catalog Test, Scenario 4, Condition 2 Test Results

Table 10.41 Web Catalog Test, Scenario 4, Condition 2 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	228	0	0	123240	541

Comments:

Although physical memory size is decreased by 50 %, the average response time of the Web Catalog Server is approximately the same. The test is successful.

Table 10.42 Web Catalog Test, Scenario 4, Condition 3 Test Variables

Product Count :	10.000
Server Type:	Server 1
RAM (Mbytes):	512
JVM:	BEA JRockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	10 viewers click randomly.
Duration:	60 sec.

Expected Result:

The product count is increased by 1000 %. But there is enough memory on the system to hold the catalog content on physical memory. A small increase is expected because the document count to query is 10 times more than in Condition 1 and Condition 2.

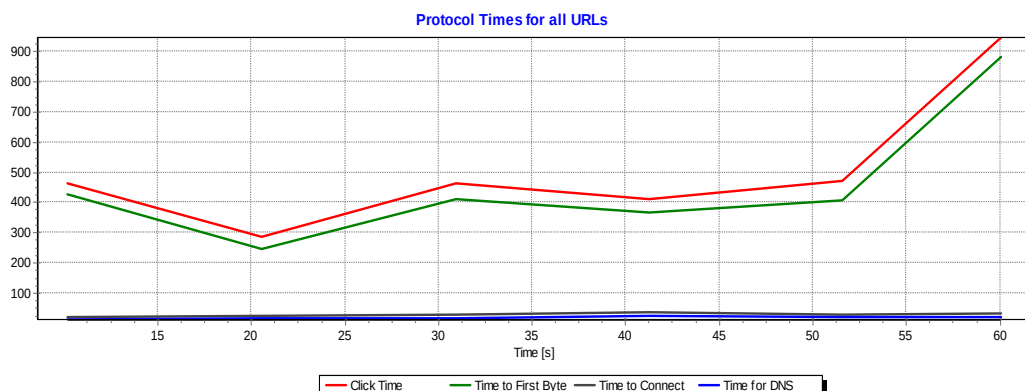


Figure 10.23 Web Catalog Test, Scenario 4, Condition 3 Test Results

Table 10.43 Web Catalog Test, Scenario 4, Condition 3 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	231	0	0	120198	516

Comments:

There is an increase on the average click time, but the system is still healthy, because document size is increased by 1000% but average click time is only increased by 20%. Test is successful.

Table 10.44 Web Catalog Test, Scenario 4, Condition 4 Test Variables

Product Count :	10.000
Server Type:	Server 1
RAM (Mbytes):	256
JVM:	BEA JRockit JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	1 viewer clicks randomly.
Duration:	120 sec.

Expected Result:

The system will collapse down since all documents cannot be loaded into physical memory and the system will use virtual memory, which is much slower than physical memory. The catalog viewer count is decreased to 1 because otherwise the system stops responding and no calculations can be made.

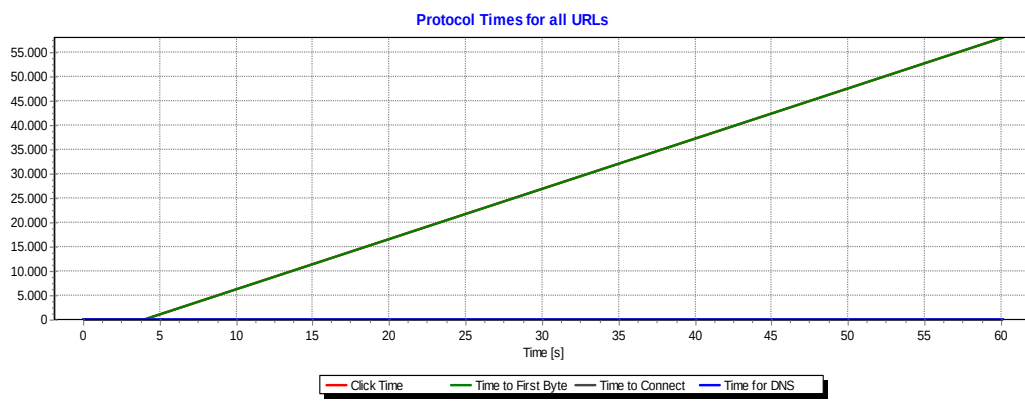


Figure 10.24 Web Catalog Test, Scenario 4, Condition 4 Test Results

Table 10.45 Web Catalog Test, Scenario 4, Condition 4 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	1	0	0	58107	58107

Comments:

The result 58107 ms is not acceptable for a single catalog viewer. The system slowed down by 11100%. Test is successful.

Table 10.46 Web Catalog Test, Scenario 4, Condition 5 Test Variables

Product Count :	3.000
Server Type:	Server 1
RAM (Mbytes):	512
JVM:	SUN JVM
Network Speed:	100 Mbits /s
Catalog Viewers:	6 viewers click randomly.
Duration:	60 sec.

Expected Result:

The system will slow down because SUN JVM is not optimized for server side applications. The document size is decreased to 3000 and the catalog viewer size is decreased to 6 in order to avoid OutOfMemory exceptions.

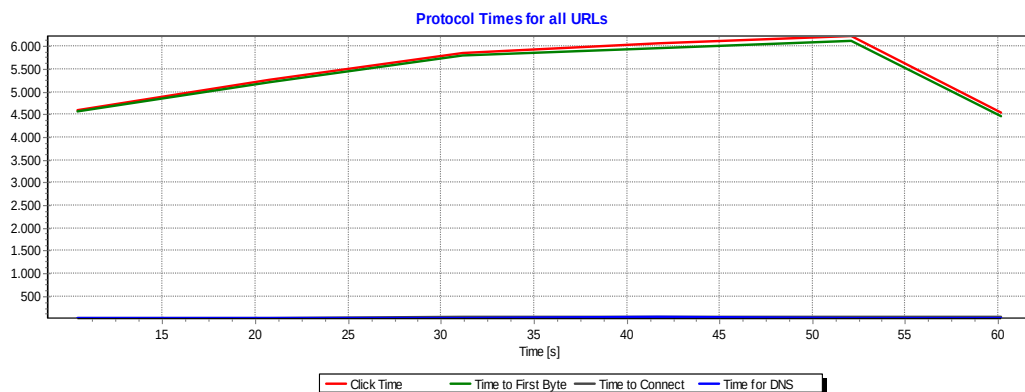


Figure 10.25 Web Catalog Test, Scenario 4, Condition 5 Test Results

Table 10.47 Web Catalog Test, Scenario 4, Condition 5 Average Click Time

URL No.	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1	1	231	99.5	120198	120198

Comments:

A 1300% increase of average click time is measured and an error rate (responses over 25s are calculated as errors) of 99.5% is achieved, which is unacceptable. Test is successful.

10.7.5 Server Comparison

Expected Result:

Since Server2's architecture is more advanced than Server1's architecture, we expect better results. This benchmark test is done only for Scenario 1.

Table 10.48 Server Comparison, Test Variables

Product Count :	10.000
Server Type:	Server 2
RAM (Mbytes):	1024
JVM:	BEA Jrockit JVM
Network Speed:	52 Kbits /s
Catalog Viewers:	10 viewers click randomly.
Duration:	60 sec.

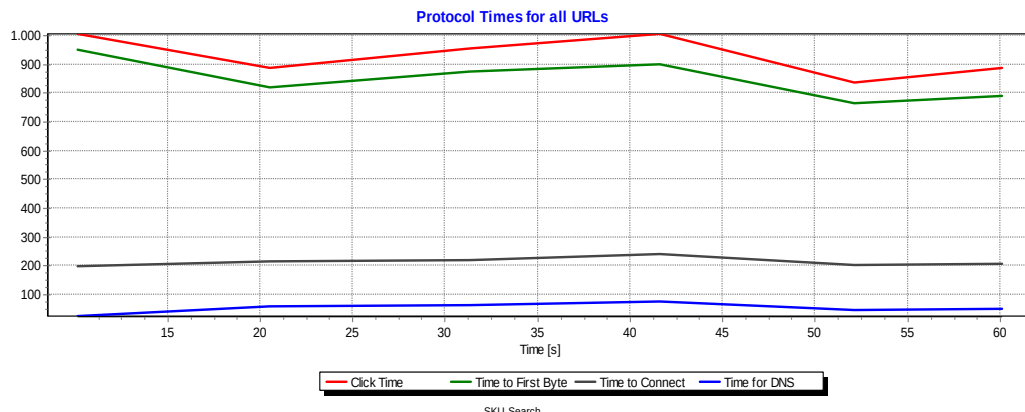


Figure 10.26 Server Comparison, Test Results

Table 10.49 Server Comparison, Average Click Time

URL No.	Name	Clicks	Errors	Errors [%]	Time Spent [ms]	Avg. Click Time [ms]
1		201	0	0	187017	930

Comments:

Under same conditions the average click time for Server 1 is measured as 1907 ms whereas Server 2 responses to the catalog viewer in 930 milliseconds.

10.8 Web Catalog Server Test Conclusions

Test results show that the platform is stable, when there is enough physical memory to hold all catalog content.

For the stability, the JVM has a critical role and with these test we have proven that BEA JRockit JVM is more suitable than SUN's JVM for our platform.

Average response times are approximately the same and are independent from the catalog content's size as long as there is enough physical memory. Small differences in the test are caused by the architecture of several components like operating system, servlet container, web server etc.

11. CONCLUSION

In this thesis, we purposed to establish a new approach to an XML based Electronic Catalog Management System.

Firstly, main features of desired ECMS are determined:

- Limitless and boundless product detail.
- Unique information.
- User friendly management GUI with easy navigation.
- Distribution of information to various platforms with the minimum work and cost for the user.
- Working with an acceptable performance.

Secondly, the data structure of the system is designed according to the features of desired ECMS. In order to achieve boundless product details and unique information, structures of different kinds of products are analyzed. As a conclusion products are separated into elements called as properties, which contains either atomic or non-atomic elements. The hierarchy and the restrictions of these atomic and non-atomic elements are clearly defined in order to form a rule base system and a semantic network. Finally, we constructed an architecture, in which product element consists of property elements, which represents a feature of the product; and the value set of the related property is restricted and ruled by the property group and unit elements. Property group elements are divided into “modes” according to their functionality and have a capability of widening the data horizontally which means, for each property of the product, a combination of the multiple property groups can be attached together. Also by using the “forkability”, each property can have multiple values with the same restrictions and rules, which means widening the data vertically.

Many of the data structure components are constructed in tree formation and as a result XML turns up to be a wise solution, because XML is in tree formation and is more capable than the relational data structure, as explained in Section 4.2.

The data structure elements are transformed into XML documents and the whole data is kept on a native XML database. A Java based client software application is developed which is called “Manager” in order to work with the XML files. The communication layer DB-Bridge is constructed and adapted between the Manager and the XML database, in order to facilitate the DB operations like connection pool, paging mechanisms with JAVA RMI support. Generally, the Manager is a user friendly GUI that translates user’s actions to database operations for the DB-Bridge. Java is chosen as the programming language because of its reasonable support for XML. As a result, a system is constructed, which collects and organizes the product information in an XML DB and which is capable of publishing this information to different media.

In order to represent the product data on the internet, the Manager prepares the XML content and sends it to the WEB Catalog by using the SOAP service. Web Catalog is a J2EE based software application, which consists of JAVA servlets and JSPs. XSL transformation is used to produce HTML pages from XML documents.

The same information, created by the manager is also used in the CD Catalog. CD Catalog is a Macromedia FLASH application that works with a JAVA service by using XML communication via socket protocol.

Furthermore, by using the Integration layer, the Manager communicates with other applications such as back-office systems, ERPs etc, in order to import and export product information from/to various data systems.

The performance analysis of the system showed us that whole platform is quite sufficient and stable. And we achieved some keys of adjustment in order to sustain the performance, such as the page size that is used in the DB-Bridge layer and the memory balance in the WEB Catalog system.

As a result, we achieved an Electronic Catalog Management System, which is adaptable for any kind of industry.

REFERENCES

- [1] **Zygon**, Enterprise Catalog Management Executive White Paper.
http://www.zygon.com/files/Enterprise%20Catalog%20Mngt_43.pdf
- [2] **Segev, A., Wan, D., Beam, C.**, 1995. Electronic Catalogs: a Technology Overview and Survey Results, *CIKM'95, Baltimore MD USA*.
- [3] **Danish, S.**, 1998. Building Database-driven Electronic Catalogs, *SIGMOD Record*, Vol. 7, No.4, December 1998.
- [4] **Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F.**, 2004. Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation. <http://www.w3.org/TR/2004/REC-xml-20040204/>
- [5] **Boss, B.**, 2003. XML in 10 points, W3C Communications team, Created 27 March 1999, revised 13- 11- 200, last update 02-06-2003.
<http://www.w3.org/XML/1999/XML-in-10-points.html>
- [6] **Stolze, M., Koenemann, J.**, 1999. User interfaces for electronic product catalogs, *CHI '99 extended abstracts on Human factors in computing systems*.
- [7] **Boag, S., Chamberlin, D., Fernández, M. F., Florescu, D., Robie, J., Siméon, J.**, 2003. XQuery 1.0: An XML Query Language, W3C Working Draft, 12 November 2003.
- [8] **Clark, J., DeRose, S.** 1999. XML Path Language (XPath) Version 1.0, W3C Recommendation, 16 November 1999.
- [9] A special DTD for Electronic Catalogs. <http://www.ecx-xml.org/>
- [10] XML Tutorials. <http://www.w3schools.com/xml/>
- [11] XPATH Tutorials. <http://www.zvon.org>
- [12] **Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.**, 2001. Fine grained access control for SOAP E-services, *Proceedings of the tenth international conference on World Wide Web*.
- [13] **Reuven, M. L.**, 2001. At the Forge: Introducing SOAP, *Linux Journal*.
- [14] **Conan, C. A.**, 2004. How clean is the future of SOAP?, *Communications of the ACM*, Volume 47, Issue 2.

- [15] **Hogg, K., Chilcott, P., Nolan, M., and Srinivasan B.**, 2004. An evaluation of Web services in the design of a B2B application, *Proceedings of the 27th conference on Australasian computer science, Volume 26*.
- [16] **Kamalsinh, F. C.**, 2004. Anatomy of a Web service, *The Journal of Computing in Small Colleges, Volume 19, Issue 3*.
- [17] Apache Soap Web Site <http://ws.apache.org/soap/>
- [18] **Martin Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.J., and Nielsen, H. F.** 2003. SOAP Version 1.2 Part 1: Messaging Framework, *W3C Recommendation*.
- [19] **Berglund, A.**, 2003. Extensible Stylesheet Language (XSL) Version 1.1, *W3C Working Draft 17 December 2003*.
- [20] **Williams, J.**, 2003. E-services: The Web services debate: J2EE vs. .NET, *Communications of the ACM, Volume 46, Issue 6*.
- [21] J2EE Technology Official Web Site. <http://java.sun.com/j2ee/>
- [22] **Matjaz, B. J., Rozman, I., Nash, S.**, Java 2 distributed object middleware performance analysis and optimization, *ACM SIGPLAN Notices, Volume 35, Issue 8*.
- [23] JAVA RMI Technology. <http://java.sun.com/products/jdk/rmi/>
- [24] Tamino XML Server.
<http://www2.softwareag.com/corporate/products/tamino/default.asp>
- [25] X-Hive/DB Native XML Storage. <http://www.x-hive.com/>
- [26] Xindice XML Database. <http://xml.apache.org/xindice/>
- [27] eXist XML Database. <http://exist.sf.net/>
- [28] Macromedia Official Web Site. <http://www.macromedia.com>
- [29] BEA JROCKIT 1.4.2_03 Java Virtual Machine.
<http://e-docs.bea.com/wljrockit/docs142/>
- [30] **Neffenger, J.**, 2003. The Volano Report.
<http://www.volano.com/report/index.html>
- [31] **Sosnoski, M.D.**, 2001. A Look At features and performance of XML document models in JAVA, *XML and JAVA Technologies: Document Models, Part1 : Performance*.
<http://www-106.ibm.com/developerworks/xml/library/x->

injava/index.html

[32] Apache Tomcat Official Web Site. <http://jakarta.apache.org/tomcat/>

APPENDIX A: SCREENSHOTS OF THE MANAGER.

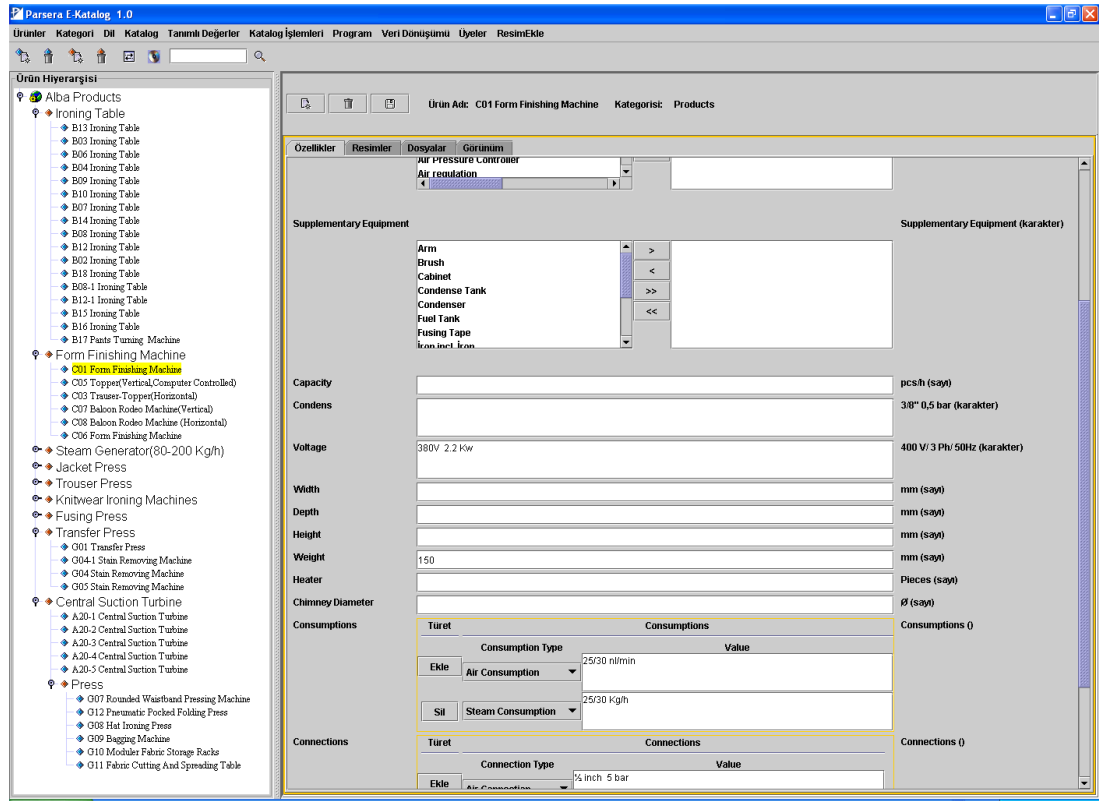


Figure A. 1 Catalog Tree & Product Editing Window.

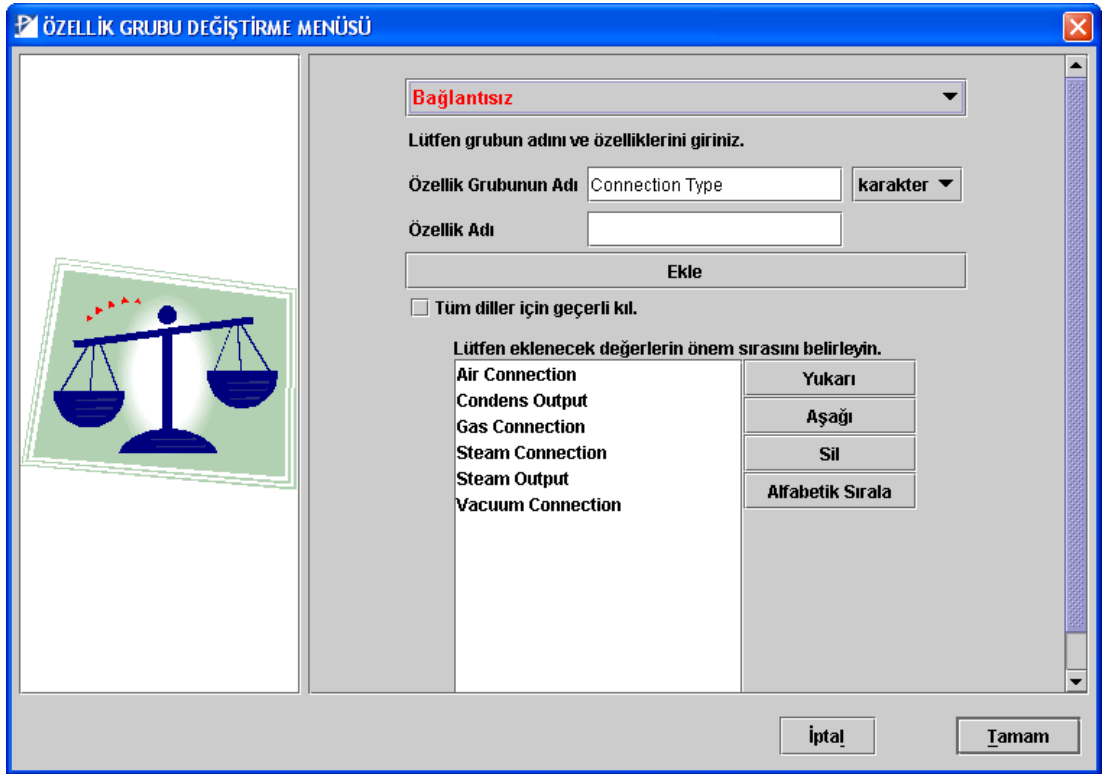


Figure A. 2 Property Group Editing Menu.

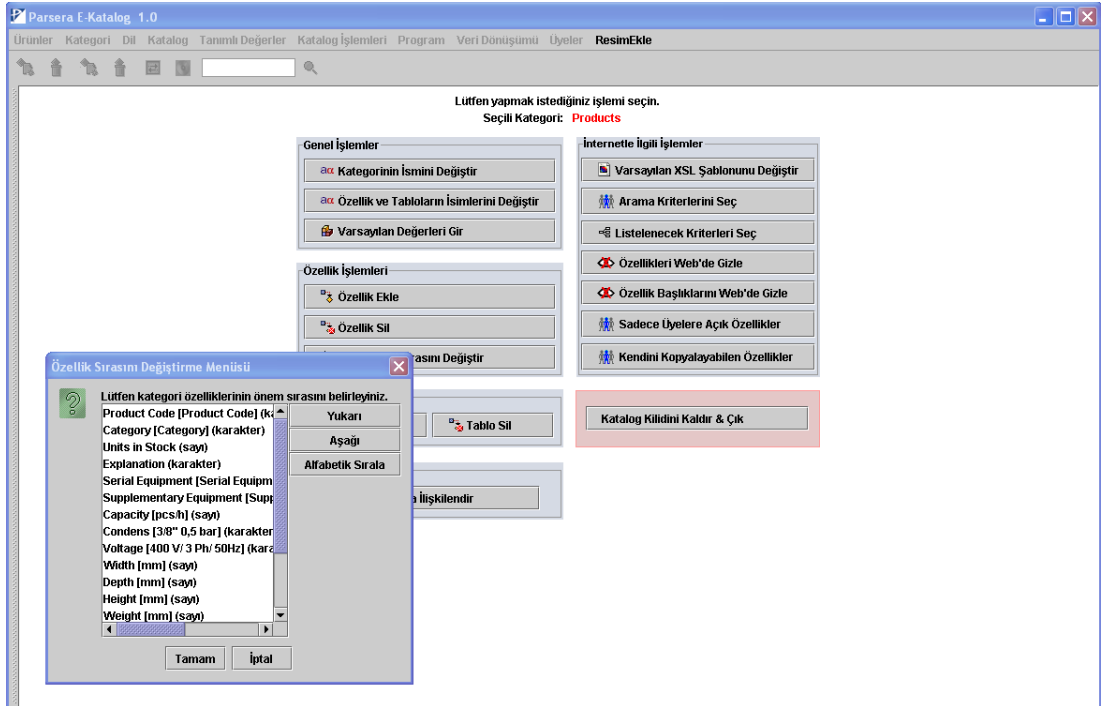


Figure A. 3 Product Family Editing Menu.

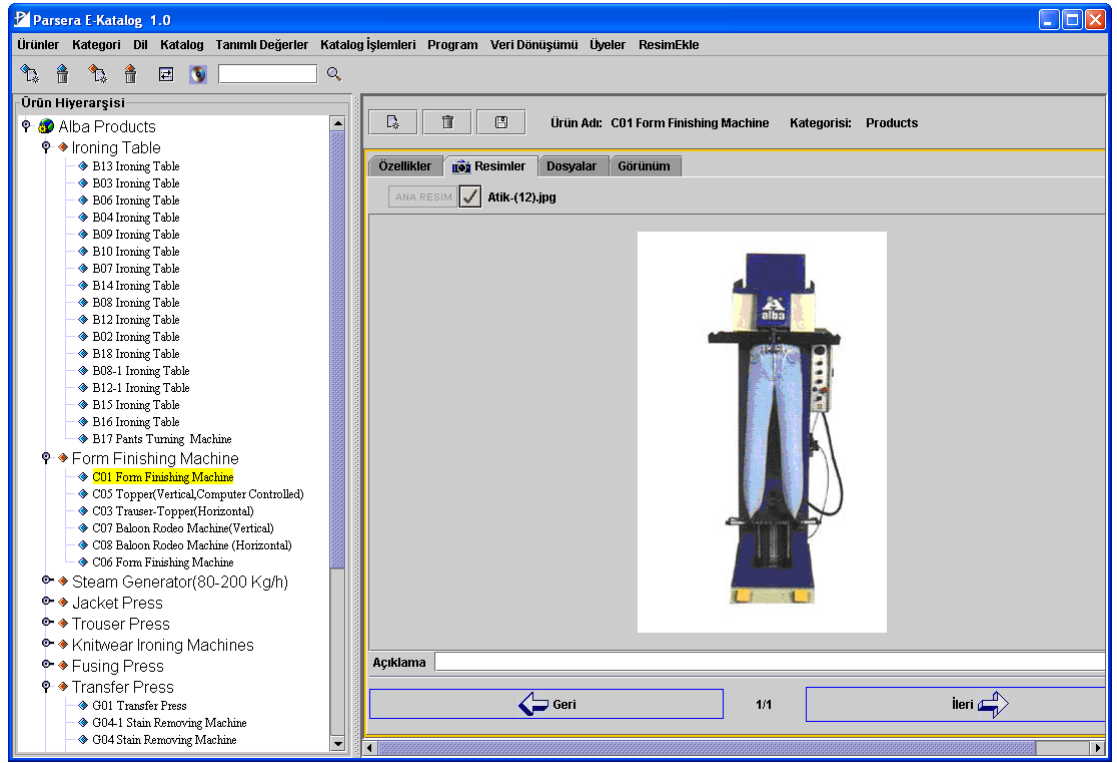


Figure A. 4 Image Editing Menu.

APPENDIX B: WEB CATALOG SCREENSHOTS

The screenshot displays the ALBA Machine Industry website. The header features the ALBA logo and a welcome message. A navigation bar includes links for Home, Search, Products, About Us, and Contact Us. The left sidebar shows a tree view of products, with 'Form Finishing Machine' selected. The main content area, titled 'Properties', displays the details for the 'C01 Form Finishing Machine'. It includes an 'Add To List' button, a product image, and a list of specifications such as Product Code, Category, Explanation, Serial Equipment, Voltage, Weight, Consumptions, Connections, and Motors. The footer contains the company name, copyright information, and additional links like FAQ, Site Map, Extranet, Brochure, and Contacts.

Properties

C01 Form Finishing Machine

[Add To List](#)

Product Code C01

Category Form Finishing Machine

Explanation It Provides Lower Costs and High Productivity With The Peculiarity of Time Adjustable Blowing and Steaming System

Serial Equipment 1st Class Strong Sheet Metal Body
Powerful Fan
Pneumatic-Controlled height adjustmen with
Stretching Device
Blocking Device

Voltage 380V 2.2 Kw 400 V/ 3 Ph/ 50Hz

Weight 150 mm

Consumptions

Consumption Type	Value
Steam Consumption	25/30 Kg/h
Air Consumption	25/30 nl/min

Connections

Connection Type	Value
Steam Connection	½ inch 5 bar
Air Connection	½ inch 5 bar

Motors

Motor Types	Value
Vacuum Motor	-

ALBA Machine Industry
Copyright2003-2004, All rights reserved

FAQ | SITE MAP | EXTRANET | BROCHURE | CONTACTS

Figure B. 1 Detailed Product Web Page.



PREMIUM QUALITY IN IRONING

Welcome

ALBA MACHINE INDUSTRY, which has been set in the year 1969, is a leading firm in Turkey about the production of ironing sets, steam boilers, steam generators etc. used in garment industries.

[Home](#) | [Search](#) | [Products](#) | [About Us](#) | [Contact Us](#)

Search By Part Number

In order to use search by part no option, you have to know the product ID. Please enter the product ID and click on the Proceed button. The product having the entered code will be listed with all its properties.

Part No:

[Proceed >>](#)

Advanced Search

By using the advanced search option, you can filter the products. Please select desired criterias and click on the List button.

Category

Form Pressing Shapes

Capacity (pcs/h)

[List](#)

ALBA Machine Industry
Copyright 2003-2004, All rights reserved

[FAQ](#) | [SITE MAP](#) | [EXTRANET](#) | [BROCHURE](#) | [CONTACTS](#)

Figure B. 2 Product Search Web Page.

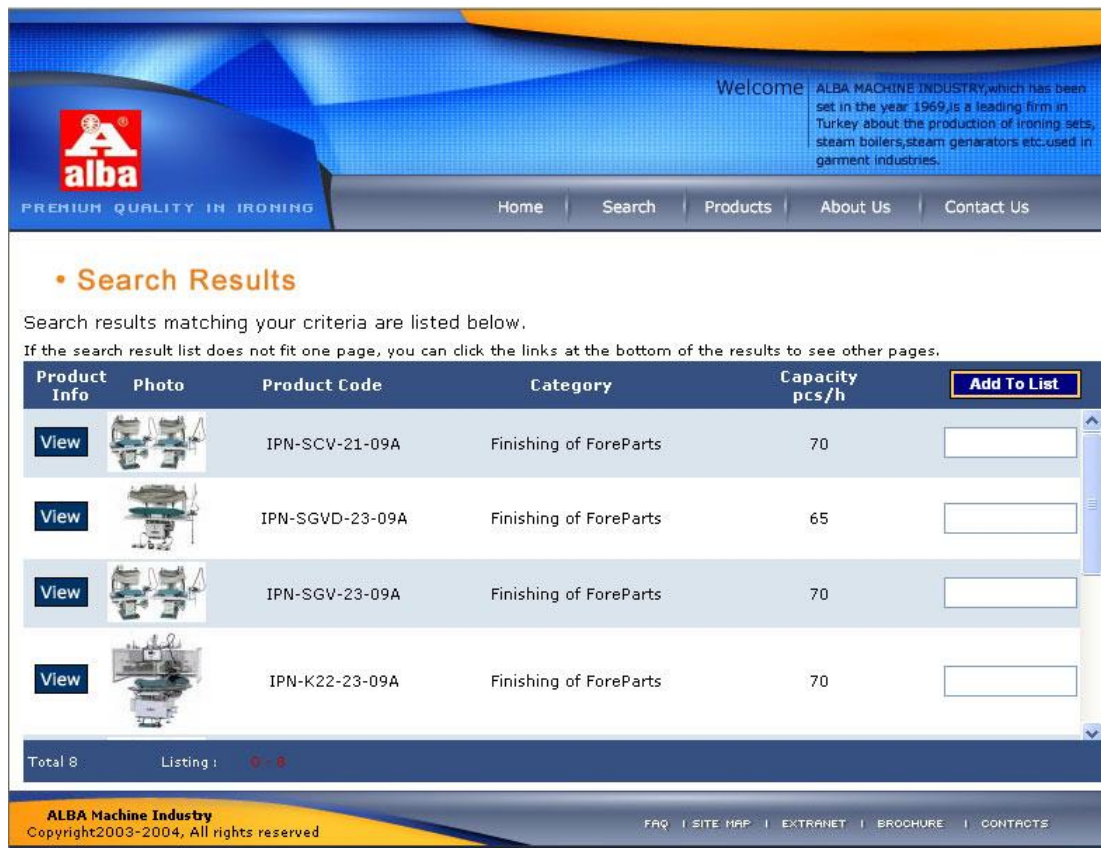


Figure B. 3 Search Results Page.

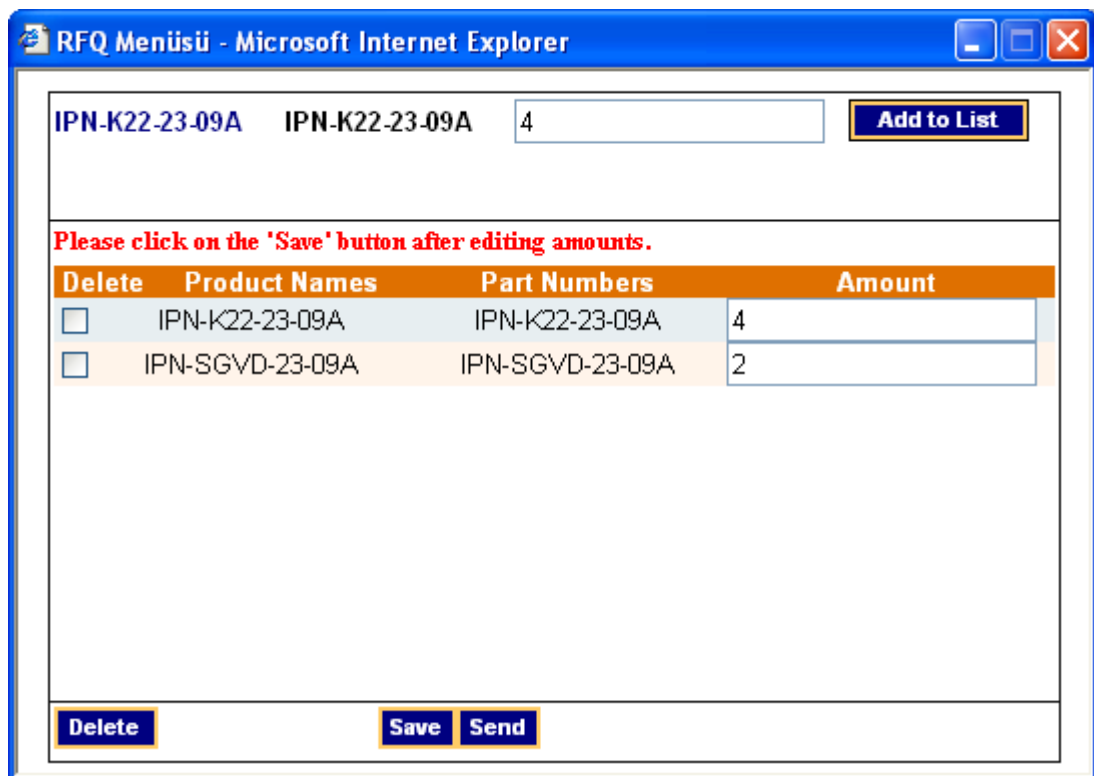


Figure B. 4 Basket Module Screenshot.

CURRICULUM VITAE

Born in 1978 in Istanbul, Nurettin Burak Bari attended high school at Austrian College and then graduated from the Istanbul Technical University (ITU), Faculty of Electric and Electronic Engineering, Control and Computer Engineering in 2001. He continued his education in the Institute of Science and Technology Computer Engineering Department in Istanbul Technical University. He is the founder of Parsera Information Technologies and is still working on XML and software development technologies in ITU KOSGEB Technology Development Center.