# DEVELOPING AN AUTOMATIC METADATA EXTRACTION (METEX) SYSTEM FROM ELECTRONIC DOCUMENTS

**A MASTER'S THESIS**

**in**

**Software Engineering**

**Atılım University**

**by**

**MURAT ÖZMERT**

**JUNE 2007**

# DEVELOPING AN AUTOMATIC METADATA EXTRACTION (METEX) SYSTEM FROM ELECTRONIC DOCUMENTS

**A THESIS SUBMITTED TO**
**THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**
**OF**
**ATILIM UNIVERSITY**
**BY**

**MURAT ÖZMERT**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE**
**DEGREE OF**
**MASTER OF SCIENCE**
**IN**
**THE DEPARTMENT OF SOFTWARE ENGINEERING**
**JUNE 2007**

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Selçuk Soyupak

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Asst. Prof. Dr. Çiğdem Turhan

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Asst. Prof. Dr. Nergiz Ercil Çağıltay

Supervisor

Examining Committee Members

Prof.Dr. İbrahim Akman _____

Asst.Prof.Dr. Çiğdem Turhan _____

Asst.Prof.Dr. Nergiz Ercil Çağıltay _____

Instructor Gül Tokdemir _____

Computer Engineer (MS) Arzu Serpen _____

## ABSTRACT

**DEVELOPING AN AUTOMATIC METADATA EXTRACTION (METEX) SYSTEM FROM ELECTRONIC DOCUMENTS**

Özmert, Murat

M.S., Software Engineering Department

Supervisor: Asst. Prof. Dr. Nergiz Ercil Çağıltay

June 2007, 96 pages

Today, the companies make big investments on Enterprise Resource Planning (ERP) solutions to manage their enterprise activities more effectively, easily and productively. Accordingly, they try to update their information systems. While companies make big investments, they try to retrieve significant data from hard copy documents to their information systems using man-power. These tedious chain of processes cause several losses. One of the most common problems met in the production oriented companies is the time loss due to effort on data extraction from publications/printed documents. Manual data input into ERP software slows down the work processes of the company and may cause entering incorrect data into the system because of high mistyping rate. This causes time and productivity loss in a company.

Data retrieval from massive amounts of technical content can be a challenge for data input operators. Moreover, every technical publications and their subgroups having its own structure, the difference of the data extracted on every technical publication and belonging to different object group in the target information system create several challenges.

When all these issues are considered, automatic metadata extraction processes gain more importance. Data collection activity should be completed in a short period of time for an information system whose software development phase is completed to begin serving for users as soon as possible (e.g. A dam must be filled with water to produce electricity).

This study is a descriptive case study which analyze metadata extraction processes to support information systems. This case study is conducted in a real-world logistic domain that has predefined (standard) structural technical documentation to feed its information system. This study aims to guide other studies to better organise their infrastructure on the way of supporting their information system in a reliable domain.

In this thesis, a framework that extracts metadata from massive electronic technical documents and transforms into XML, is presented. In this regard, aspects such as the basic structures of developed system, development processes, basic services provided and the similar systems are also elaborated. To better show the gains of the developed system, the durations of the processes in classical(manual) system and the developed system are also evaluated and compared.

**Keywords:** Data Extraction, Metadata, Electronic document, XML transformation, ERP data input.

# ÖZ


**ELEKTRONİK DOKÜMANLARDAN OTOMATİK VERİ AYRIŞTIRMA (METEX) ARACININ GELİŞTİRİLMESİ**

Özmert, Murat

Yüksek Lisans, Yazılım Mühendisliği Bölümü

Tez Yöneticisi: Yrd.Doç.Dr. Nergiz Ercil Çağıltay

Haziran 2007, 96 sayfa

Günümüzde, şirketler kurumsal faaliyetlerini daha etkin, kolay ve verimli bir şekilde gerçekleştirebilmek amacıyla Kurumsal Kaynak Planlama (ERP) otomasyonu ile çözümlere büyük kaynaklar ayırmakta ve büyük yatırımlar yapmaktadırlar. Bu şekilde kullandıkları teknolojiyi her zaman yükseltmeye çalışmaktadırlar. Ancak şirketler bu sistemlere bu kadar yatırım yaparken çeşitli teknik dokümanlar üzerindeki bilgileri insan gücü kullanarak zahmetli bir biçimde bilgi sistemlerine aktarmaya çalışırlar. Bu durum çeşitli kayıplara yol açmaktadır. Üretime yönelik şirketlerde en çok karşılaşılan ortak problemlerden biri de yüksek volümlü teknik dokümanlardan verilerin hedef sisteme yüklemek amacıyla çıkartılmasında harcanan kayıplardır.

İnsan gücü kullanarak ERP sistemine veri girişi, süreçlerin yavaşlamasına ve hata oranının yüsek olmasından dolayı yanlış bilgilerin sisteme aktarılmasına neden olabilmektedir. Bir organizasyon için, bu da verimlilik ve zaman kaybı anlamına gelmektedir.

Yüksek hacimli teknik dokümanlardan verilerin bulunup hedef sisteme yüklenmesi amacıyla elde edilmesi, veri giriş operatörleri için oldukça zahmetli bir iş olabilmektedir.

Ayrıca, her teknik dokümanın ve alt gruplarının kendine özgü bir yapısının olması, her dokümandan temin edilen verilerin farklılığı ve hedef bilgi sisteminde verilerin farklı nesne gruplarına ait olması da bazı güçlükler oluşturmaktadır.

Tüm bu konular değerlendirildiğinde, otomatik veri ayrıştırma süreçleri önem kazanmaktadır. Yazılımı tamamlanmış ve kullanıcılarına hizmet vermeye hazır bir bilgi sisteminin veri toplama süreçlerinin mümkün olan en kısa sürede tamamlanması gerekmektedir. (bir baraj göletinin su ile dolmaması halinde elektrik üretilememesi gibi).

Bu çalışma bilgi sistemlerini desteklemek amacıyla veri ayrıştırma işlemlerini analiz eden tanımlayıcı bir çalışmadır. Bu çalışma bilgi sistemlerini beslemek amacıyla standart teknik doküman yapısına sahip olan gerçek bir lojistik faaliyet alanında yürütülmüştür. Çalışmanın amacı, güvenilir bir alanda bilgi sistemlerini desteklemede, altyapılarını daha iyi bir şekilde organize etmek isteyenlere rehberlik etmektir.

Bu tezde, yüksek volümlü teknik dokümanlardan verileri ayrıştıran ve bunları XML formatına dönüştüren bir yapı sunulmaktadır. Bu kapsamda, geliştirilen sistemin temel yapısı, tasarım/geliştirme aşamaları, sağlanan temel fonksiyonlar ve benzer sistemler özetlenmiştir. Geliştirilen sistemin kazançlarını daha iyi ifade edebilmek için sonuç bölümünde, elle veri giriş ve geliştirilen sistem ile veri giriş süreleri değerlendirilmiş ve karşılaştırılmıştır.

**Anahtar kelimeler:** Veri ayrıştırma, metadata, elektronik doküman, XML dönüşümü, ERP veri girişi.

To My Wife, Sonsoles

&

To My Daughter, Elif

**ACKNOWLEDGEMENTS**

I am grateful to my advisor, Asst. Prof. Dr. Nergiz Ercil Çağıltay, for her guidance, motivation, patience, and the support that she has provided throughout the course of this study.

I should also express my appreciation to Asst. Prof. Dr. Çiğdem Turhan for her continuous encouragement since 1992.

I would like to thank to my family. Without their support and love, I could never taste any achievement in my life.

## TABLE OF CONTENTS

ABSTRACT ..... iii

ÖZ ..... v

ACKNOWLEDGEMENTS ..... viii

TABLE OF CONTENTS ..... ix

LIST OF TABLES ..... xii

LIST OF FIGURES ..... xiii

LIST OF ABBREVIATIONS ..... xv

CHAPTER

1. INTRODUCTION ..... 1

1.1 Printed Documents & ERP Systems ..... 1

1.2 Manual Data Entry ..... 2

1.3 Data Collection/Input Methods ..... 4

1.4 The Aim of the Case Study ..... 5

2. BACKGROUND AND RELATED WORKS ..... 7

2.1 General Concepts and Terms ..... 7

2.2 Previous Works and Commercially Available Products ..... 17

2.3 Evaluation and Comparison of the Products ..... 23

3. IMPLEMENTATION ..... 28

3.1. Project Definition ..... 28

3.2 System Overview ..... 28

3.2.1 Provided Functionalities ..... 29

3.2.2 Requirements & Expectations ..... 29

3.3 General System Architecture ..... 30

3.3.1 Source Files to be Processed ..... 31

3.3.2 Description Argument and Description File Used ..... 33

3.3.3 PDF to Text Conversion ..... 35

3.3.4 Text to Excel Conversion ..... 36

3.3.5 Man-In-The-Loop ..... 39

3.3.6 Excel to XML Conversion ..... 39

3.4 How Does METEX Work? ..... 41

3.5. Technology Used ..... 57

4. EXPERIMENTAL RESULTS ..... 62

4.1 Document Structure ..... 62

4.2 Manual Data Input Transactions in The Target ERP System ..... 64

4.3 Total Duration Calculation ..... 68

5. CONCLUSION ..... 70

5.1 Future Work ..... 71

REFERENCES ..... 72

APPENDICES

A. ENTITY RELATIONSHIP DIAGRAMS RELATIVE TO CHAPTER 3 ..... 76

B. DATAFLOW DIAGRAMS RELATIVE TO CHAPTER 3 ..... 79

C. SEQUENCE DIAGRAMS RELATIVE TO CHAPTER 3..... 85

D. REGULAR EXPRESSIONS RELATIVE TO CHAPTER 3 ..... 90

E. PSEUDO CODES RELATIVE TO CHAPTER 3 ..... 91

## LIST OF TABLES

### TABLES

Table 2.1 An Example of A Simple Metadata Record ..... 13
Table 2.2 Updated Tool Categories ..... 24
Table 2.3 Batch-Oriented Tool Categories ..... 26
Table 2.4 Updated On-line/Real-Time/Interactive Tool Categories..... 27
Table 3.1 Description Argument/File And Their Functions..... 33
Table 4.1 Time Portions for Manual Data Input Steps in ERP System..... 68

## LIST OF FIGURES

### FIGURES

Figure 1.1 Possible Data Collection Methods..... 4
Figure 2.1 Information Extraction in Context [12]..... 8
Figure 2.2 Business-to-Business Communications with XML/EDI [16]..... 9
Figure 2.3 An Example of XML Code [16]..... 10
Figure 2.4 The Classical Approaches to Search for Regular Expressions in a Text . 16
Figure 2.5 The Filtering Approach to Search for Regular Expressions in a Text ..... 16
Figure 2.6 A Sample Web Page..... 18
Figure 2.7 PDF Files Retrieved From Web Page Using Web2DB Software ..... 19
Figure 2.8 DataMite Software Screenshot..... 20
Figure 2.9 Pattern Editor Software Screenshot..... 21
Figure 3.1 General System Architecture..... 31
Figure 3.2 Sample Portion of PDF Source File ..... 32
Figure 3.3 Sample Portion of Excel Source File..... 33
Figure 3.4 Sample FSA (Finite State Automata) for a Sample Regular Expression . 34
Figure 3.5 Portion of the Access File Contains ERP's XML Id-Tags..... 35
Figure 3.6 Sample PDF File That Contains Date Information ..... 36
Figure 3.7 Extraction of Date Information From Text Body..... 37
Figure 3.8 The Source Code that Executes the Parsing Process..... 37
Figure 3.9 The FSA for Above Given Regular Expression..... 38
Figure 3.10 The Excel File Contains Extracted Data ..... 38
Figure 3.11 Description File Contains XML-Tags Correspond to Excel Data..... 40
Figure 3.12 Specific Data in the XML File ..... 41
Figure 3.13 PDF To Excel Converter Main Window..... 43
Figure 3.14 Screenshot of Open-File-Dialog-Box..... 44
Figure 3.15 Presentation of PDF To Text Conversion Result ..... 45
Figure 3.16 Results of Text Processing Using Regular Expression ..... 46
Figure 3.17 Screenshot of Excel-File-Creation Window..... 47
Figure 3.18 Save-Work-Area Functionality ..... 48
Figure 3.19 Excel To XML Converter Main Window ..... 49
Figure 3.20 Load Excel and Select Sheet ..... 50

Figure 3.21 Successfully Loaded Excel File..... 51
Figure 3.22 Failure on Loading Excel File ..... 52
Figure 3.23 Load Access File ..... 53
Figure 3.24 Result of Column Pairing of Loaded Access and Excel Files..... 54
Figure 3.25 Failure on Column Pairing of Loaded Access and Excel Files..... 55
Figure 3.26 Creation of XML File..... 56
Figure 3.27 Saving an XML File ..... 57
Figure 3.28 Sample PDF File..... 58
Figure 4.1 Experimental Document Breakdowns..... 62
Figure 4.2 Sample Data Set and Data Locations in the Printed Technical Publication. 63
Figure 4.3 Sample Sub Data Set and Data Locations in the Printed Technical Publication ..... 64
Figure 4.4 Manual Data Input in the Target ERP System ..... 65
Figure 4.5 Manual Data Input for the Subsystem (1) in the Target ERP System..... 66
Figure 4.6 Manual Data Input for the Subsystem (2) in the Target ERP System..... 66
Figure 4.7 Manual Data Input for the Subsystem(3) in the Target ERP System..... 67
Figure 4.8 Manual Data Input for the Subsystem(4) in the Target ERP System..... 67

## LIST OF ABBREVIATIONS

| | |
|---|---|
| CSV: | Comma Separated Values |
| DOM: | Document Object Model |
| DTD: | The Document Type Definition |
| ERP: | Enterprise Resource Planning |
| GUI: | Graphical User Interface |
| HTML: | Hypertext Markup Language |
| METEX: | Metadata Extraction System |
| ODBC: | Open Database Connectivity |
| SCM: | Supply Chain Management |
| SQL: | Structured Query Language |
| TCP/IP: | Transmission Control Protocol/Internet Protocol |
| XML: | Extensible Markup Language |
| XSL: | The eXtensible Style Language |
| W3C: | World Wide Web Consortium |
| NFA: | Nondeterministic Finite Automaton |
| DFA: | Deterministic Finite Automaton |
| IE: | Information Extraction |

# CHAPTER 1

## INTRODUCTION

In this chapter a general background is provided to better understand the concepts studied in this thesis.

### 1.1 Printed Documents & ERP Systems

Despite the growing use of data capture, the business world still relies heavily on paper for account applications and other transactions. An estimated $15 billion dollars is spent each year in the United States to key in and validate data. Considerable savings could be achieved by automating this process. There is an increase in the amount of data captured electronically. The number of documents generated both electronically and on paper will increase to 20 trillion over the next five years [1]. Although the percentage of printed documents will decline from 90 percent of the total today to about 40 percent, this volume still represents a fourfold increase in the total number of document pages, to 4 trillion pages [2]. Information drives business and 80% of all corporate content is unstructured in nature. The volume of information produced is growing at 50% annually. On average, a knowledge worker generates about 800MB of content each year [3].

Nowadays companies need to analyze the technical publications produced by the manufacturers and extract the data needed from those publications and then input them into their information systems.

Similarly, companies make big investments on Enterprise Resource Planning (ERP) solutions to manage their enterprise activities more effectively, easily and productively. Accordingly, they try to update their information systems regularly. However while companies make big investments, they try to migrate big amount of information from hard copy documents to their information systems by using manpower [4]. This causes time and productivity loss.

Installations of new, highly integrated systems, such as ERP systems, may compound the problem of human error. Internal controls may be lacking due to

deadline pressures and complexity of the installation. In fact, approximately 40% of participants in a recent set of interviews pointed out that the internal controls for ERP systems are often inadequate [5]. The data entry errors occur frequently. One explanation may be that humans lapse into 'habits of mind' during data entry because they mistakenly assume that the computerized systems have controls to catch most errors when modern systems may not [5].

### 1.2 Manual Data Entry

Manual data entry is the slowest and most error prone way to record data [6]. Human error in manual data entry is a concern, as is the potential for technical problems or equipment failure in automated methods of data collection [7].

One of the most common problems met at the companies is that the time loss due to effort on data extraction from publications slows down work process of the companies. Manual data entry into ERP system slows down the work processes of the company and may cause entering incorrect data into the system. Entering data into the information system manually takes considerable time and causes loss of manpower. Another concern is the rate of human errors in such a repetitive task. When a part number is entered incorrectly into an accounting information system, what is the impact? If good preventive, detective, and corrective controls are in place, the impact will be little, if any. However, if the appropriate controls are not in place, human error can have big ramifications. For example, when an Army clerk mis-keyed a part number, the product that was ordered was a seven-ton marine anchor instead of a headlight for a jeep. As a result, the huge anchor was delivered as a result of the wrong order [5].

The process of detecting, monitoring and correcting of wrong data is another concern. When considering the loss of time and man power on this process, the total cost of entering data gets very high [4].

Data retrieval from massive amounts of technical content can be a challenge for data input operators. Multiple users and departments need to collaborate to input data to the target information systems. Regarding logistics processes, the extraction of data in high volume technical publications by evaluating and analyzing them is very hard and tedious process. Due to large volume of data, the job requires many people working parallel. Not only time consuming, manual data entry also means a high

potential for errors [3]. The data read on technical publications and entered manually by system users may cause invalid data on the system. Invalid data entered into the system due to user errors cause financial loss and time consumption and sometimes system's security and safety violation.

When information gets captured, it's often a manual process to validate it, separate it from other content, and classify it based on content type or sensitivity [32]. This type of manual process is inefficient, insecure, and error prone. There are many potential points of failure, no built-in security measures, and no built-in compliance management. Mistakes made in these areas can destroy a business' credibility and reputation and result in costly legal fees [3].

*The drawbacks of the manual-data-entry can be grouped under the following topics*;

Error-Prone:

- Manual data entry is error-prone. Even the best workers rarely maintain the vigilance and consistency required to achieve 100% data accuracy for significant periods of time [8].

Cost Driven:

- It is not always cost effective solution. The ERP system data entered, usually dictates that users should follow a certain path for entering data not to violate the data relations and data integrity. Moreover it requires the users to know the ERP system requirements well. This requirement brings increasing cost due to user training, consuming resources such as time and money.

Source Definition & Data Preparation:

- Documents published in foreign languages requires person who is responsible for data entry to know that foreign language at least at the level of understanding the documents.
- It takes a long time to find the data required on the technical documents. The users dealing with data entry need to be an expert on functionality of the work processes.
- Every technical publication and their subgroups having its own structure, the difference of the data extracted on every technical publication and belonging to different object group in the information system create several challenges.

- Data entered into the target ERP system can be in various media and in several standards.

Update:

- Addition to the primary data migration, according to the technological advances, the technical data changes published by the manufacturers must be reflected into the system so often during product the life cycle. For each new product entering the system, the technical publications and their detailed levels need to be evaluated and then data collecting process should be repeated.

### 1.3 Data Collection/Input Methods

The possible data input methods into the ERP System can be listed as follows:

- Manual data entry
- Automatically loading data obtained from electronic documents into the ERP system
- Hybrid solutions

Figure 1.1 illustrates the possible methods for data collection.

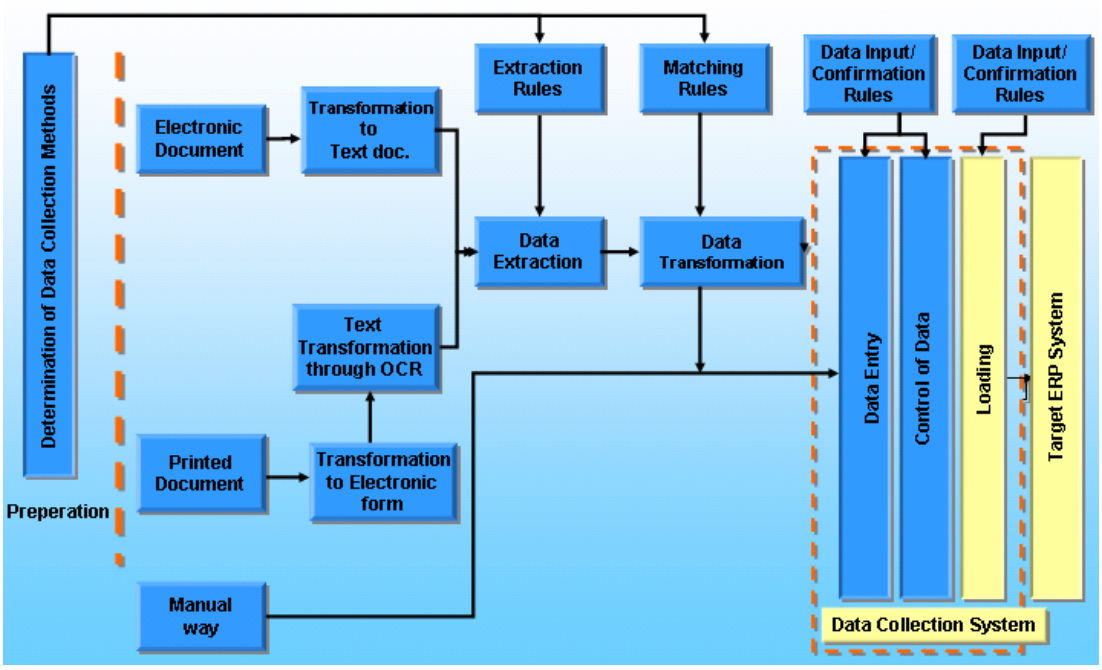



**Figure 1.1 Possible Data Collection Methods**

There is no "one-size-fits-all" process for data conversion and capture. The most appropriate method should be selected for the specific project, whether it is keyboarding, optical character recognition, automated processes or double key entry with verification [9].

### 1.4 The Aim of the Case Study

To address the problems associated with manual data entry, an automated chain of processes was needed. The large volume of data on printed or electronic technical publications makes it difficult to enter the data into the system manually. To break the information in data content up automatically into the information system objects was a reasonable solution. Traditionally, information involved in knowledge-based systems has been manually acquired in collaboration with domain experts. However, both the high cost of such a process and the existence of textual sources containing the required information have led to the use of automatic acquisition approaches [10].

The aim of this study is to develop a software tool that extracts the metadata from unstructured/structured electronic documents and transform them into XML file compatible with target ERP system. The developed system will be used in the logistics activity area. Using minimum man power, it will automate the manual data collection processes.

By analyzing ERP software's table structures, data hierarchy, data format and data authoring mechanisms are solved and XML file as a final product is formed for this purpose. Considering relational data integrity, the automatically generated XML file is accepted by ERP software tool. The structure of the XML file compatible with ERP has came out after analyzing ERP's database tables. Briefly, the proposed system will extract the metadata and data from electronic documents and transform them into XML file compatible with the target ERP system. This thesis primarily concentrates on contents, functions and the automation processes of data extraction from electronic documents. In this regard, to meet the requirements of a data extraction system, a case study with custom development solution is also provided.

This thesis is organised as; Chapter 2 presents the terminology and the related works providing comparison chart concerning existing products. A classification of different products is presented from different perspectives in this section as well.

Chapter 3 explains the implementation details and usage of proposed framework. Within this framework, the general architecture of the proposed system is described in this section. Chapter 4 contains the experimental results, and finally, chapter 5 gives conclusions and possible future extensions.

# CHAPTER 2

## BACKGROUND AND RELATED WORKS

In this chapter, a brief description of some basic concepts and terms that used frequently in this study will be explained. Descriptions gathered from several sources will be explained in a summarized manner.

### 2.1 General Concepts and Terms

The basic concepts and terms that used frequently is explained in the sub paragraphs.

Information Extraction: Information extraction is the ability of convert text from a document captured in digital form and export that text to a file or a database [5]. Information extraction is the extraction by computer of selected types of information from written natural language texts, for instance information about joint ventures or new products from financial newswires. The pressure to develop such a technology has been steadily growing with the swelling volume of electronic texts in such forms as scientific journal abstracts, financial newswires, patents and patent abstracts, corporate and government technical documentation, electronic mail and electronic bulletin boards threatens to engulf us [5]. These texts contain a wealth of information of vital scientific, economic and technical significance. But the sheer volume of these sources is increasingly preventing the timely finding of relevant information [11].

Information extraction is the process of filling the fields and records of a database from unstructured or loosely formatted text. Thus (as shown in Figure 2.1), it can be seen as a precursor to data mining. Information extraction populates a database from unstructured or loosely structured text, data mining then discovers patterns in that database [12].

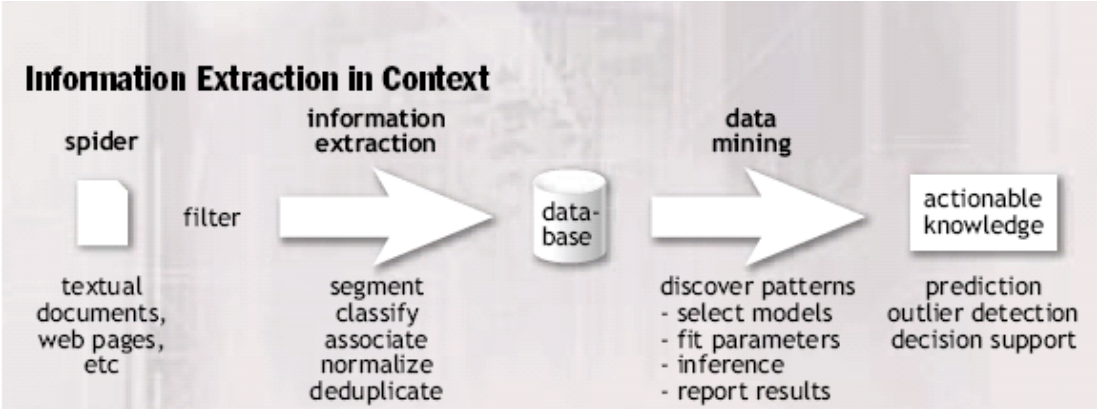



**Figure 2.1 Information Extraction in Context [12]**

The accuracy of automated extraction methods varies drastically depending on the regularity of the text input and the strength of the extraction method used. Extraction from formatted, highly regular database-generated Web pages (such as those on Amazon.com) can be done with perfect accuracy. Extraction from other somewhat regular text, such as postal address blocks or research paper citations, usually has a percentage accuracy in the mid- to high-90s. Accuracies in the mid-90s are now standard for extracting names of people, companies, and locations from standard news articles (Extracting these entity names from Web pages, however, is more difficult, yielding accuracies in the 80s). Extraction of protein names is more difficult, since their naming scheme is more irregular; the accuracies in a recent competition were in the 80s. Success in the association stage of extraction is generally more difficult because a correct final answer also requires correct segmentation and classification of all of the fields that should be associated [12].

Information extraction (IE) is a type of information retrieval whose goal is to automatically extract structured information, i.e. categorized and contextually and semantically well-defined data from a certain domain, from unstructured machine-readable documents [13]. The significance of IE is determined by the growing amount of information available in unstructured (i.e. without metadata) form, for instance on the Internet. This knowledge can be made more accessible by means of transformation into relational form, or by marking-up with XML tags [13].

<u>What is XML:</u> The number of applications currently being developed that are based on, or make use of, XML documents is truly amazing . For our purposes, the word "document" refers not only to traditional documents, like the thesis itself, but also to

the myriad of other XML "data formats". These include vector graphics, e-commerce transactions, mathematical equations, object meta-data, server APIs, and a thousand other kinds of structured information. XML specifies neither semantics nor a tag set. In fact XML is really a meta-language for describing markup languages. In other words, XML provides a facility to define tags and the structural relationships between them. Since there's no predefined tag set, there can't be any preconceived semantics. All of the semantics of an XML document will either be defined by the applications that process them or by style sheets [14].

XML stands for “Extensible Markup Language”. It is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. Extensible Markup Language is the emerging Internet standard for data [15]. XML is a set of tags that can be used to define the structure of a hypertext document.

Figure 2.2 illustrates a typical Electronic Data Interchange Platforms via XML[16].

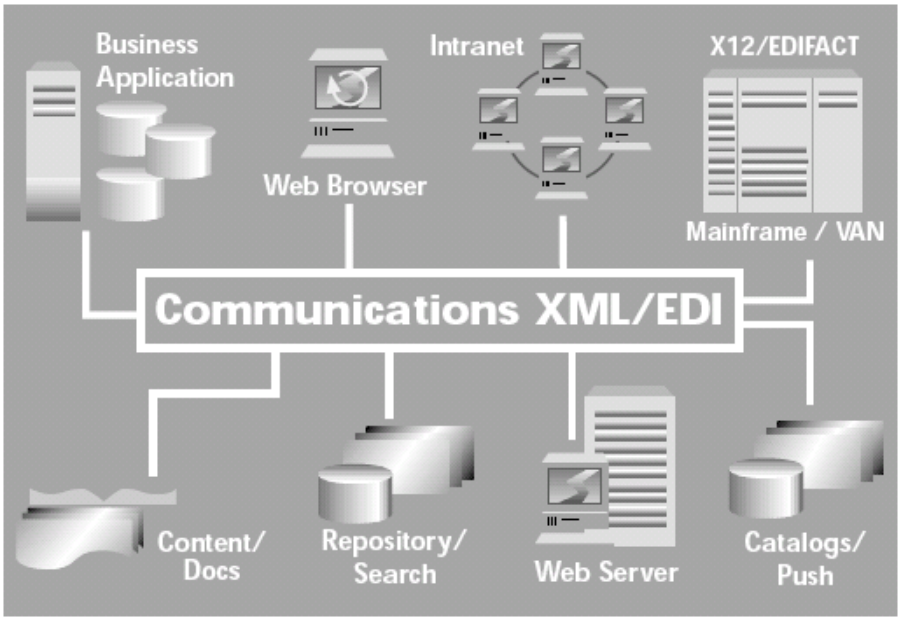



**Figure 2.2 Business-to-Business Communications with XML/EDI [16]**

Figure 2.3 illustrates an example of XML code by comparing it with plain text and HTML formats[16].

| Plain text | HTML | XML |
|---|---|---|
| Act One<br>SCENE ONE. Helsingör. A terrace in front of the castle. Francisco is on sentinel duty. Enter Bernardo.<br><br>BERNARDO: Who's there?<br>FRANCISCO: Nay, answer me. Stand and unfold yourself.<br>BERNARDO: Long live the King! | <H1>ACT ONE</H1><br><P><I>SCENE ONE. Helsingör. A terrace in front of the castle.</I></P><br><P><I>FRANCISCO is on sentinel duty. Enter BERNARDO.</I></P><br><P><B>BERNARDO:</B>Who's there?</P><br><P><B>FRANCISCO:</B>Nay, answer me. Stand and unfold yourself.</P><br><P><B>BERNARDO:</B>Long live the King!</P> | <ACT><TITLE>ACT ONE</TITLE><br><SCENE><br><TITLE>SCENE ONE.<br>Helsingör. A terrace in front of the castle.</TITLE><br><STAGEDIR>FRANCISCO is on sentinel duty. Enter BERNARDO.</STAGEDIR><br><SPEECH><br><SPEAKER>BERNARDO</SPEAKER><br><LINE>Who's there?</LINE><br></SPEECH><br><SPEECH><br><SPEAKER>FRANCISCO</SPEAKER><br><LINE>Nay, answer me. Stand and unfold yourself.</LINE><br></SPEECH><br><SPEECH><br><SPEAKER>BERNARDO</SPEAKER><br><LINE>Long live the King!</LINE><br></SPEECH><br></SCENE> </ACT> |
| Human beings don't need any text codes or tags in order to be able to read this excerpt from Shakespeare's Hamlet. Thanks to what they learned in school many moons ago, they are able to identify or impute the meaning of individual text passages. | The same text in HTML format. The HTML tags merely define how the text is represented in a browser. They do not add any semantic value to it and are thus unable to provide any information with regard to content (for example: "How much text does Bernardo have to recite?"). | This time, XML tags have been added to the text. A computer is not able to derive meaning from a context, just to analyze character strings on the basis of defined rules. It therefore needs text codes to infer information about the content. The XML text could now be loaded in a voice computer, for instance, or used to draw up a cast list. |

**Figure 2.3 An Example of XML Code [16]**

The benefits of XML can be described as follows [17];

*Simplicity*: Information coded in XML is easy to read and understand, plus it can be processed easily by computers.

*Openness*: XML is a W3C standard, endorsed by software industry market leaders.

*Extensibility*: There is no fixed set of tags. New tags can be created as they are needed. XML's one-of-a-kind open structure allows you to add other state-of-the-art elements when needed. This means that you can always adapt your system to embrace industry-specific vocabulary [18].

*Self-description*: In traditional databases, data records require schemas set up by the database administrator. XML documents can be stored without such definitions, because they contain meta data in the form of tags and attributes. XML Provides a basis for author identification and versioning at the element level. Any XML tag can possess an unlimited number of attributes such as author or version.

*Contains machine-readable context information*: Tags, attributes and element structure provide context information that can be used to interpret the meaning of content, opening up new possibilities for highly efficient search engines, intelligent data mining and agents. This is a major advantage over HTML or plain text, where context information is difficult or impossible to evaluate. Separates content from presentation : XML tags describe meaning not presentation. Multiple views or presentations of the same content are easily rendered by XSL style sheets.

*Facilitates the comparison and aggregation of data*: The tree structure of XML documents allows documents to be compared and aggregated efficiently element by element.

*Can embed multiple data types*: XML documents can contain any possible data type - from multimedia data (image, sound, video) to active components (Java applets, ActiveX).

*Can embed existing data*: Mapping existing data structures like file systems or relational databases to XML is simple. XML supports multiple data formats and can cover all existing data structures.

*Complete integration of all traditional databases and formats*: XML documents can contain any imaginable data type - from classical data like text and numbers, or multimedia objects such as sounds, to active formats like Java applets or ActiveX components [18].

*Self-describing data:* Unlike records in traditional data base systems, XML data does not require relational schemata, file description tables and external data type definitions. Because the data itself contains this information. In contrast to the

widely used Web format, HTML, which only ensures the correct presentation of the formatted data, XML also guarantees total usability of data. This is imperative for business applications whose tasks extend beyond the mere presentation of content [18].

*Modifications to data presentation (no reprogramming required):* You can change the look and feel of documents or even entire websites with XSL Style Sheets without manipulating the data itself [18].

*One-server view of distributed data*: XML documents can consist of data from many different databases distributed over multiple servers. In other words: With XML the entire World Wide Web is transformed into a single all-encompassing database [18]. XML documents can consist of nested elements that are distributed over multiple remote servers. XML is currently the most sophisticated format for distributed data - the World Wide Web can be seen as one huge XML database [17].

*Internationalization:* Internationalization is of utmost importance for electronic business applications. XML supports multilingual documents and the Unicode standard [18].

*Future-oriented technology:* XML is the endorsed industry standard of the World Wide Web Consortium (W3C) and is supported by all leading software providers. Furthermore, XML is also the standard today in an increasing number of other industries, for example health care [18]. Rapid adoption by industry. Software AG, IBM, Sun, Microsoft, Netscape, Data Channel, SAP and many others have already announced support for XML [17].

*Metadata:* It is simply “Data about data” or “Data explain data”. Metadata is structured data which describes the characteristics of a resource. It shares many similar characteristics to the cataloguing that takes place in libraries, museums and archives. The term "meta" derives from the Greek word denoting a nature of a higher order or more fundamental kind. A metadata record consists of a number of pre-defined elements representing specific attributes of a resource, and each element can have one or more values [19]. Below is an example of a simple metadata record [19]:

**Table 2.1 An Example of A Simple Metadata Record**

| **Element name** | **Value** |
|---|---|
| Title | Web catalogue |
| Creator | Dagnija McAuliffe |
| Publisher | University of Queensland Library |
| Identifier | http://www.library.uq.edu.au/iad/mainmenu.html |
| Format | Text/html |
| Relation | Library Web site |

*Each metadata schema will usually have the following characteristics:*

- A limited number of elements
- The name of each element
- The meaning of each element

Typically, the semantics is descriptive of the content, location, physical attributes, type (e.g. text or image, map or model) and form (e.g. print copy, electronic file). Key metadata elements supporting access to published documents include the originator of a work, its title, when and where it was published and the subject areas it covers. Where the information is issued in analog form, such as print material, additional metadata is provided to assist in the location of the information, e.g. call numbers used in libraries. The resource community may also define some logical grouping of the elements or leave it to the encoding scheme. For example, "Dublin Core" may provide the core to which extensions may be added.

*Some of the most popular metadata schemas include:*

- Dublin Core [20]
- AACR2 (Anglo-American Cataloging Rules) [21]
- GILS (Government Information Locator Service) [22]
- EAD (Encoded Archives Description) [23]
- AGLS (Australian Government Locator Service) [24]

While the syntax is not strictly part of the metadata schema, the data will be unusable, unless the encoding scheme understands the semantics of the metadata schema. The encoding allows the metadata to be processed by a computer program.

*Important schemes include:*

- HTML (Hyper-Text Markup Language)
- SGML (Standard Generalized Markup Language)
- XML (eXtensible Markup Language)
- RDF (Resource Description Framework)
- MARC (MAchine Readable Cataloging)
- MIME (Multipurpose Internet Mail Extensions)

*Metadata may be deployed in a number of ways:*

- Embedding the metadata in the Web page by the creator or their agent using META tags in the HTML coding of the page
- As a separate HTML document linked to the resource it describes
- In a database linked to the resource. The records may either have been directly created within the database or extracted from another source, such as Web pages.

The simplest method is for Web page creators to add the metadata as part of creating the page. Creating metadata directly in a database and linking it to the resource, is growing in popularity as an independent activity to the creation of the resources themselves. Increasingly, it is being created by an agent or third party, particularly to develop subject-based gateways. Metadata is data associated with objects which relieves their potential users of having full advance knowledge of their existence or characteristics. Information resources must be made visible in a way that allows people to tell whether the resources are likely to be useful to them. This is no less important in the online world, and in particular, the World Wide Web. Metadata is a systematic method for describing resources and thereby improving access to them. If a resource is worth making available, then it is worth describing it with metadata, so as to maximize the ability to locate it. Metadata provides the essential link between the information creator and the information user [19].

While the primary aim of metadata is to improve resource discovery, metadata sets are also being developed for other reasons, including [19]:

- Administrative control
- Security
- Personal information
- Management information
- Content rating
- Rights management
- Preservation

Below is an example of a Dublin Core record for a short poem, encoded as part of a Web page using the <META> tag [19]:

```
<HTML> !4.0!
<HEAD>
<TITLE>Song of the Open Road</TITLE>
<META NAME="DC.Title" CONTENT="Song of the Open Road">
<META NAME="DC.Creator" CONTENT="Nash, Ogden">
<META NAME="DC.Type" CONTENT="text">
<META NAME="DC.Date" CONTENT="1939">
<META NAME="DC.Format" CONTENT="text/html">
<META NAME="DC.Identifier"
CONTENT="http://www.poetry.com/nash/open.html">
</HEAD>
<BODY><PRE>
I think that I shall never see
A billboard lovely as a tree.
Indeed, unless the billboards fall
I'll never see a tree at all.
</PRE></BODY>
</HTML>
```

The <META> tag is not normally displayed by Web browsers, but can be viewed by selecting "Page Source" [19].

<u>Regular Expressions:</u> Regular Expressions give an extremely powerful way to express a set of search patterns, containing all the previous types of problems. A

regular expression specifies simple strings and concatenations, unions, and repetitions of other sub expressions [25]. The algorithms addressing them are more complex and should be used only when the problem cannot be expressed as a simpler one. Searching for a regular expression is a multistage process. First we need to parse it to obtain a more workable tree representation. The second stage is to build a nondeterministic finite Automation (NFA) [25][46].

*A regular expression RE is a string on the set of symbols;*

$\sum \cup \{\varepsilon, |, ., *, (,)\}$, which is recursively defined as the empty character ε; a character $\alpha \in \sum$; and (RE1), (RE1˙ RE2), (RE1|RE2), and (RE1*), where RE1 and RE2 are regular expressions. Figure 2.4 illustrates the classical approaches to searching for regular expressions in a text [25].

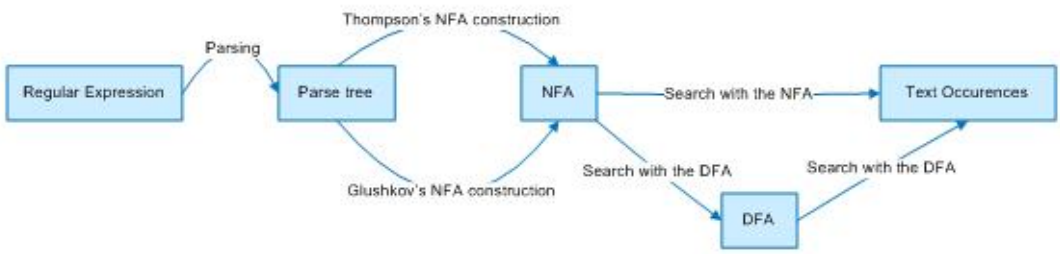



**Figure 2.4 The Classical Approaches to Search for Regular Expressions in a Text**

Figure 2.5 illustrates the filtering approach to search for regular expressions in a text [25].

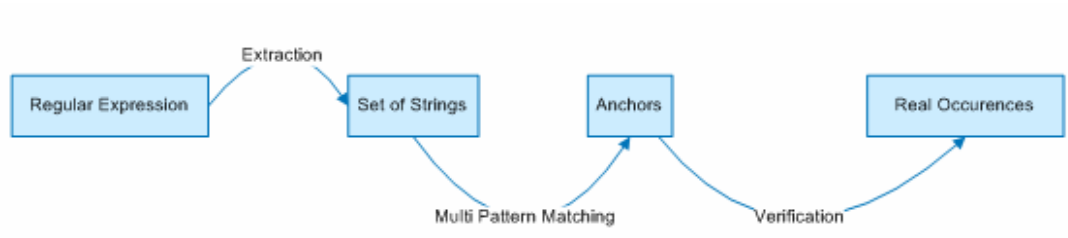



**Figure 2.5 The Filtering Approach to Search for Regular Expressions in a Text**

## 2.2 Previous Works and Commercially Available Products

Today in the market, there are different types of commercially available data extraction products (running on different platforms) ranging from small business to large scale business requirements. Below, some of the represented works are listed and their functionalities are mentioned briefly.

*FormMagic:* It is a kind of data collection tool from the Paper, Fax, Invoice and electronic forms [4].

*Automation Anywhere:* Extracts web data, screen scrape from web pages or use it for web mining. Extracts web data into database, spreadsheet or any other application [26].

*Ficstar Web Grabber .NET:* Collects data from anywhere on the Internet and extract any content from the targeted web pages [27].

*Data Extractor:* Extract emails, URLs or data from websites or files. Easily extract Email addresses or URLs from text files or the web [28].

*Data Manager V2.8.2:* Allows to process and manipulate data in a easy and logical manner using a graphical interface. DataManager reads and writes deliminated files such as comma separated files (CSV) and also can read data from ODBC Data Sources [29].

*Web Data Scraper:* Retrieve website information directly into DOC, XLS, or other file that supports ActiveX objects [30].

*Web2DB:* It is a web data extraction service. It takes unstructured data from web html pages and converts into structured records [31].
Figure 2.6 illustrates a sample web page.

**Figure 2.6 A Sample Web Page**

Figure 2.7 illustrates the retrieved-data using Web2DB software.

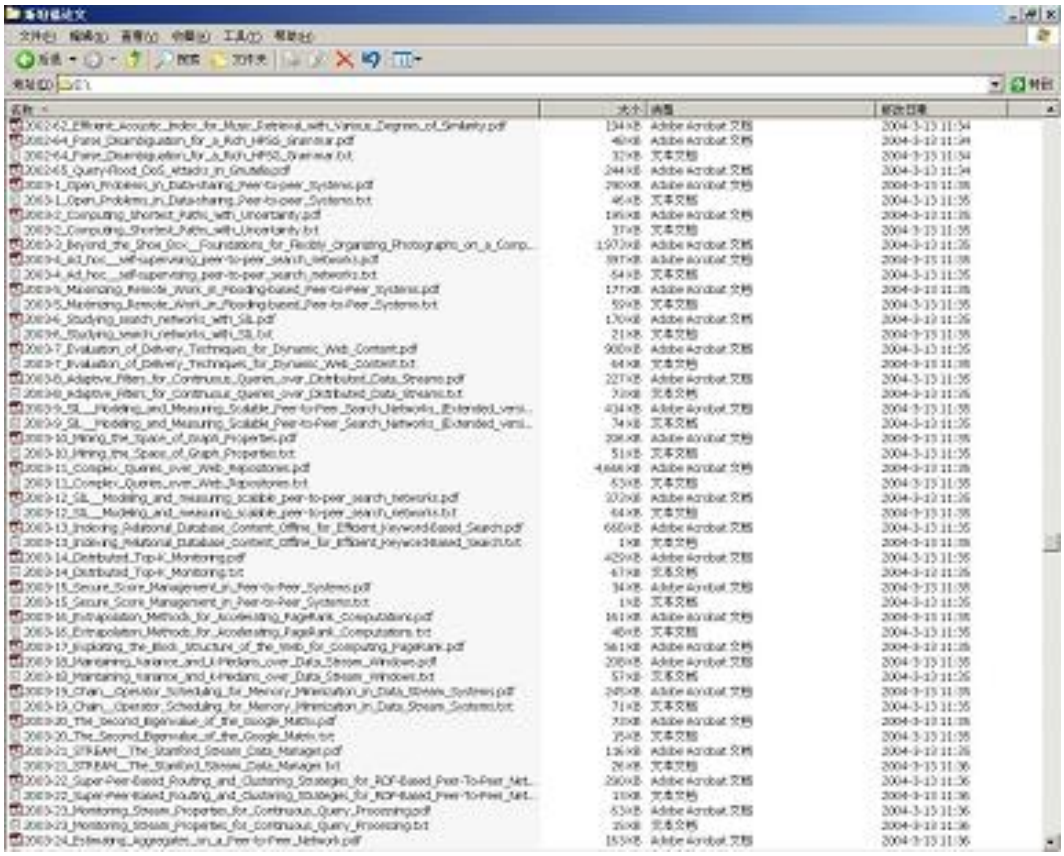


**Figure 2.7 PDF Files Retrieved From Web Page Using Web2DB Software**

*DataMite:* Provides solution to extract data from report files with an all-in-one tool that is easy to use. Its report-mining and data translation visual tools extract data from files without any programming. Imports data from any sources, extract the data needed, builds a visual query, creates report and an exports files visually [32]. Figure 2.8 illustrates screenshot of DataMite software.

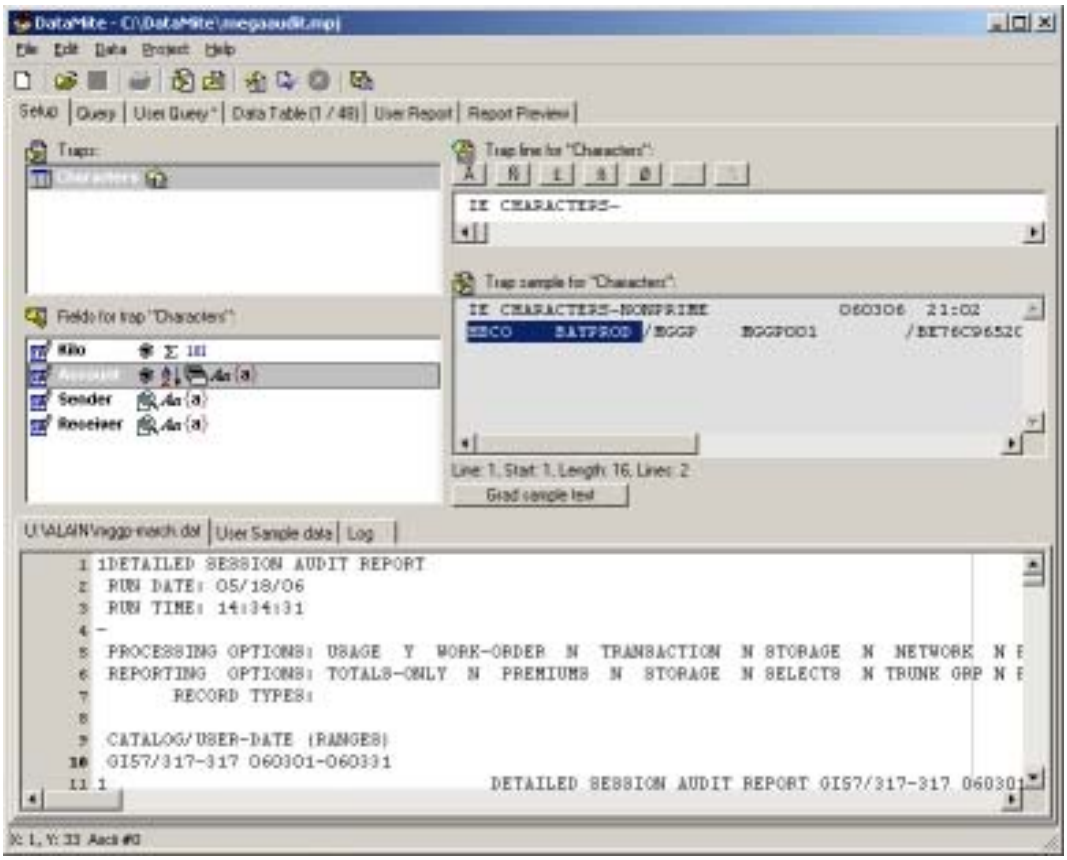



**Figure 2.8 DataMite Software Screenshot**

*EZ-Pickin's Report Mining & Data Extraction Tool:* Provides customized data extraction - Extracts reports, transfers data into Excel Spreadsheets [33].

*ETI Extract:* ETI describes Extract as a tool suite that "selectively retrieves, transforms, and moves high volumes of data from any database or file format to any other, regardless of the hardware/software." [34].

*Pattern Editor:* A tool for debugging Perl5 regular expressions [35]. Figure 2.9 illustrates the screenshot of Pattern Editor software.

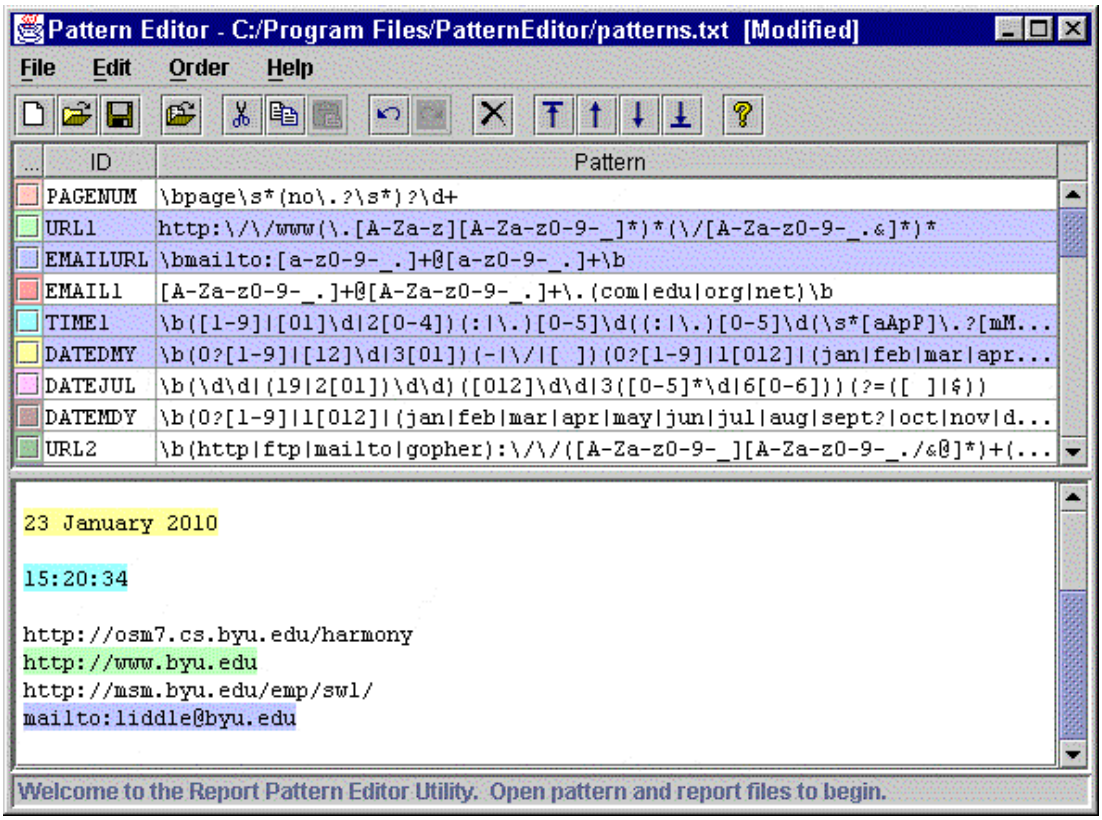



**Figure 2.9 Pattern Editor Software Screenshot**

*Web-Harwest:* It is open-source Web data extraction tool written in Java. It offers a way to collect desired Web pages and extract useful data from them. In order to do that, it leverages well established techniques and technologies for text/xml manipulation such as XSLT, XQuery and Regular Expressions**.**

*Apertus Technologies, Enterprise Integrator Tool:* Users write transformation rules. Data is filtered against domains and ranges of legal values and compared to other data structures [36].

*Bachman:* Modeling and reverse engineering tool. The reverse engineering tool derives a consistent set of metadata from several potential source system's metadata [36].

*Carleton Passport:* Users enter extraction and transformation parameters, and data is filtered against domains and ranges of legal values [37].

*Embarcado, ER1:* Departmental modeling and reverse engineering tool. The reverse engineering tool derives a consistent set of metadata from several potential source system's metadata [36].

*Evolutionary Technology (ETI), EXTRACT:* Users write transformation rules. Data is filtered against domains and ranges of legal values and compared to other data structures [36].

*Kismet Analytic, KisMeta Workbench:* Enterprise data modeling and reverse engineering tool. The tool derives a unified conceptual schema, and rationalizes and cross-indexes all of the enterprise's metadata [36].

*LogicWorks, ERWIN ERX:* The reverse engineering tool derives a consistent set of metadata from several potential source system's metadata [36].

*Oracle, Symmetric Replicator:* A data replication product designed to extract data from several platforms, perform some filtering and transformation, and distribute and load to another database or databases [36].

*Platinum, InfoRefiner:* A data pump product designed to extract data from several mainframe platforms, perform some filtering and transformation, and distribute the load to another mainframe platform database [36].

*Platinum, InfoHub:* A middleware product designed to establish a permanent relationship (including filtering and transformation) between source systems and a logical model [36].

*Praxis, Omni Replicator:* A data replication product designed to extract data from several platforms [36].

*Prism Solutions, Warehouse Manager:* Users enter extraction and transformation parameters, and data is filtered against domains and ranges of legal values [36]. Prism states that the product "automatically generates database creation and load control statements for the definition of data warehouse structures to the target DBMS. It also generates refresh, update, and append statements to support ongoing maintenance." [37].

*PostalSoft, ACE:* Specialty Address correction tool. Uses a "brute force" strategy of comparing each address with a USPS database with over one hundred million valid US addresses [36].

*PostalSoft, Library:* A class library of callable routines that developers user to create "address-smart" data entry screens that automatically edit most address characteristics [36].

### 2.3 Evaluation and Comparison of the Products

Each of the above mentioned product/work serves to specific purpose such as data collection from web resources, data collection from various forms and reports and data migration/extraction among databases.

In general, there are several works partially related to our subject. However, since our work focuses on an uncommon problem (data extraction from technical document which belongs to complex product in the logistics domain, and conversion of these data into XML format that target ERP system can recognise), there are no directly related and customizable previous work, in our knowledge.

"The highest level of perfection in a given extraction can be achieved only with custom programs. But custom programs are expensive, and dozens of custom programs in different environments are almost impossible to manage at the Enterprise level." [36A]. The table explaining the tool categories is modified within the light of the recent products investigated [36]. Table 2.2 illustrates the updated tool categories.

**Table 2.2 Updated Tool Categories**

| Category | Name | Functionality |
|---|---|---|
| Data Reverse Engineering<br><br>Metadata based | Bachman, LogicWorks, ERWIN/ERX, Embarcadero ER/1, Kismet KisMeta | Process metadata to document systems and abstract business rules and relationships. (High-end tools also derive some logical and conceptual schema information) |
| Data Reverse Engineering<br><br>Data content based | Vality Integrity, QDB Analyze, Data Star, WizRule | Process data content in conjunction with metadata to abstract business rules and relationships, automatically. |
| Batch Export/Transport<br><br>Parameter-based extraction code generators | Carleton Passport, ETI Extract, Prism, Warehouse Manager | Extraction is centrally controlled by parameters; extraction code programs are automatically generated. The tools accommodate data conversion and abstraction, as well as transport. Conversion and abstraction assumes a known and consistent format for source data. Tools offer sophisticated tracking and management of replication, a Prism strong point. |
| Batch Export/Transport<br><br>User develops extraction code | 3GL/4GL, Platinum, InfoRefiner, Platinum InfoPump, Praxis, OmniReplicator, | Container for extraction/conversion code, interfaces with databases. Modest replication tracking capabilities. (Transformation and range of import/export targets more capable than Replication tools, below) |
| Replication | IBM DataPropagator, Sybase, Replication Server | Designed specifically to manage replicated data (distributed database or warehouse). May include export/transport functions, or may use other tools. (Limited transformation and import/export capabilities) |
| Middleware<br><br>On-line/interactive extract/transport | CA Enterprise, Access, Platinum InfoHub, Praxis Omni Replicator, Sybase Enterprise Connect, IBM DataJoiner, Intersolv Sequelink, | Similar to batch export transport tools in concept, but supports queries, including on-line queries, by creating a "virtual database" and automating the interface between several source system platforms and the user's query tools. |
| Data Content<br><br>Quality - Filter based | Apertus,<br><br>Trillium, | Positioned between export and import, these tools support parameter-based data filtering. These tools specialize in and are more capable at managing relationships and transformation than tools such as Prism. They can be used to keep data with different keys in alignment, to avoid misinterpretation during queries. |

| | | |
|---|---|---|
| Data Content Quality<br><br>Relationship based | Vality,<br>DB Star,<br>WizRule,<br>IDI, | Data quality is evaluated based on data content. Data patterns, rules and relationships discovered assist analysts determine data quality problem areas. |
| Special Purpose<br><br>Data Quality | PostalSoft ACE,<br>Group 1 Nadis,<br>SSA | Data quality for special purposes, such as name/address correction, and pharmaceutical naming conformity. |
| Data Quality<br><br>Special purpose data entry support | PostalSoft Library,<br>Mailers +4 | Automatic editing of address data as it is being entered on-line (including street name, address range, Zip). Tools are incorporated into data entry screens as class libraries, either client server or client solutions. |
| Data translation | Data Junction,<br>Cambio | Data format translation aids (for use in conjunction with other processes.) |
| Web Data Extraction | Automation Anywhere<br>Ficstar Web Grabber .NET<br>Data Extractor<br>Web Data Scraper<br>Web2DB<br>Web-Harwest | - Web data extraction into database, spreadsheet or any other application.<br><br>- Data collection from anywhere on the Internet and extraction any content from the targeted web pages.<br><br>- Extraction of emails, URLs or data from websites.<br><br>- Conversion of unstructured data captured from web html pages into structured records. |
| Data Extraction From Printed Media & Reports<br><br>Report-mining and Data translation<br><br>Parameter-based | FormMagic<br>DataMite<br>Pattern Editor<br>EZ-Pickin's Report Mining & Data Extraction Tool | - Data collection from the paper, fax, invoice and electronic forms.<br>- Report-mining and data translation visual tools extract data from files without any programming. Imports data from any sources, extracts the data needed, builds a visual query, creates report and exports files visually.<br>- Debugging Perl5 regular expressions<br>- Customized data extraction |

Table 2.3 illustrates Batch-Oriented tool categories [36A].

**Table 2.3 Batch-Oriented Tool Categories**

| Batch-Oriented Data Extraction, Cleansing and Migration Tools | |
|---|---|
| **Tool Category** | **Tool Name** |
| Data Reverse Engineering<br><br>- Metadata based | Bachman |
| | ERWIN/ERX |
| | KisMeta |
| | PowerSoft |
| | Embarcadero |
| Data Reverse Engineering<br><br>- Data content based | Vality Integrity |
| | QDB Analyze |
| | WizRule |
| Batch Export/Transport<br><br>- Parameter-based extraction code generators | Carleton Passport |
| | ETI Extract |
| | Prism |
| Batch Export/Transport<br><br>- User develops extraction code | 3GL/4GL |
| | Platinum InfoRefiner |
| | Platinum InfoPump |
| Replication | Praxis OmniReplicator |
| | IBM DataPropagator |
| | Oracle 7 Symmetric Replication |
| Data Content Quality - relationship based | Vality Integrity |
| | WizRule |
| | IDI |
| Special Purpose Data Quality | PostalSoft ACE |
| | SSA |
| | Group 1 Nadis |

The table [36] explaining the On-line/Real-Time/Interactive tool categories is modified within the light of the recent products investigated. Table 2.4 illustrates the updated On-line/Real-Time/Interactive tool categories [36A].

**Table 2.4 Updated On-line/Real-Time/Interactive Tool Categories**

| On-Line/Real-time/Interactive Data Extraction, Cleansing and Migration Tools | |
|---|---|
| **Tool Category** | **Tool Name** |
| Middleware<br><br>On-line/interactive extract/transport | Platinum InfoHub |
| | Praxis Omni Replicator |
| | Sybase Enterprise Connect |
| | IBM DataJoiner |
| | Intersolv Sequelink |
| Data Quality - Data Entry support | PostalSoft Library |
| | Mailer +4 |
| Web Data Extraction<br><br>On-line/interactive extraction | Automation Anywhere<br>Ficstar Web Grabber .NET<br>Data Extractor<br>Web Data Scraper<br>Web2DB<br>Web-Harwest |
| Data Extraction From Printed Media & Reports<br>Report-mining and Data translation<br><br>On-line/interactive extraction | FormMagic<br>DataMite<br>Pattern Editor<br>EZ-Pickin's Report Mining & Data Extraction Tool |

Due to their limited functional capabilities, being not appropriate for customization, not having their source code, the cost of their license, not serving for our specific need, it is believed that above explained solutions for metadata extraction and data collection does not bring much help on our specific work. Although no single method is currently available that meets all of these requirements, we should be aware of existing and emerging technologies that allow data collection to be conducted in the most accurate and efficient manner possible. To overcome the problems mentioned above, a decision was made on developing of a software to convert the metadata extracted out of a complex technical publication prepared in English into XML format.

# CHAPTER 3

## IMPLEMENTATION

In this chapter, implementation details of the proposed framework will be explained.

### 3.1. Project Definition

Data extraction from massive amounts of technical content can be a challenge for data input operators. Multiple users and departments need to collaborate to input data to the target information systems. Regarding logistics domain, the extraction of data in high volume technical publications by evaluating and analyzing them is a time consuming and tedious process. To overcome above described problems, METEX software was developed.

### 3.2 System Overview

METEX software was developed by a team consists of three persons. Team members and their roles are as follows;

As researcher my role : To perform requirement analysis, feasibility study, risk assessment, to prepare use-cases, search literature, perform document-anatomy study for the source documents, prepare regular expressions for the source documents, prepare the target ERP system's XML template, develop a prototype and to perform system tests.

Developer: In .NET Platform, using C#, coding the METEX which is previously partly-developed by the researcher using Delphi integrated development environment. To perform conceptual design, and prepare regular expressions for the source documents and perform module tests.

Functional expertise: To perform the man-in-the-loop function and make necessary enhancements for the regular expressionsdepending the result of visual inspections, perform system tests and form XML-ID tags file in MS Access format.

METEX application consists of 3-tier architecture such as interface layer, application layer and database layer. The application runs on a stand-alone configuration without any network connection. The software is implemented on .NET platform and Microsoft Access used for the purpose of storage. User interface and the application layers were developed using Microsoft Visual C# Object Oriented Programming Language. Software Packages used within the framework of METEX are Adobe Acrobat Reader, Microsoft Access and Microsoft Excel. Thanks to its graphical user interface, METEX handles all the processes from accepting a document to producing XML document and storing it in its local database. Through its graphical user interface it is possible to monitor matching fields between Excel document and Access structure. After that matching process, all extraction processes are implemented in the application layer and, finally XML document produced in accordance with ERP XML template. The final product can be checked through application's user interface. In the database layer only read process is executed. Access files in the database contain the target ERP system's XML Id-tags. These values are retrieved for the purpose of processing in the application layer. The details of the proposed framework is presented in the following parts of this section.

### 3.2.1 Provided Functionalities

METEX is a kind of data-extraction-tool that performs the following functionalities;

- To extract data from existing source files (From PDF or from Excel)
- To transform extracted data into the XML file in accordance with target ERP system's templates.

### 3.2.2 Requirements & Expectations

- Reliable, efficient and fast data input mechanism for the target ERP system will be established.
- The ideal data-collection method would be inexpensive, easy to use, and applicable to widely varying types of studies.
- It would allow end-users who have received minimal training and who do not have a significant prior level of technical proficiency or education to use

the method to enter data efficiently and in a consistent way that minimizes input errors and loss of data.

- The ideal method should be able to make use of relatively inexpensive equipment, or equipment that is already in use in a department, such as computer workstations.
- The data collection activity should be completed in a short time for an information system of which software development phase completed to begin serving for users as soon as possible (e.g. A dam must be filled with water to produce electricity).
- Delayed data collecting activities will postpone the use of information system of which software development phase completed and the end-users will begin to have negative feelings against the system. Briefly, the investments should turn us back in short time.
- Extraction of metadata out of technical publications and loading it into the information system as master data will bring many benefits.
- The proposed system should be capable of extracting the metadata from the massive volume of technical publication.

### 3.3 General System Architecture

METEX accepts PDF or Excel document types as a source and produces output in XML document type. By doing this, regular expressions and the ERP system's XML templates are taken into the consideration. In case, the provided input file format is in PDF, METEX's all sub processes will be used respectively. In the first stage, PDF source file is converted into MS Excel file format. The output of the first stage is used as an input to the second stage. In the second stage, MS-Excel-data-rows are matched to data in Access database (which represents the id-tags of target XML file). And finally XML file in accordance with ERP XML template is generated. The general system architecture of the proposed framework is presented in Figure 3.1.

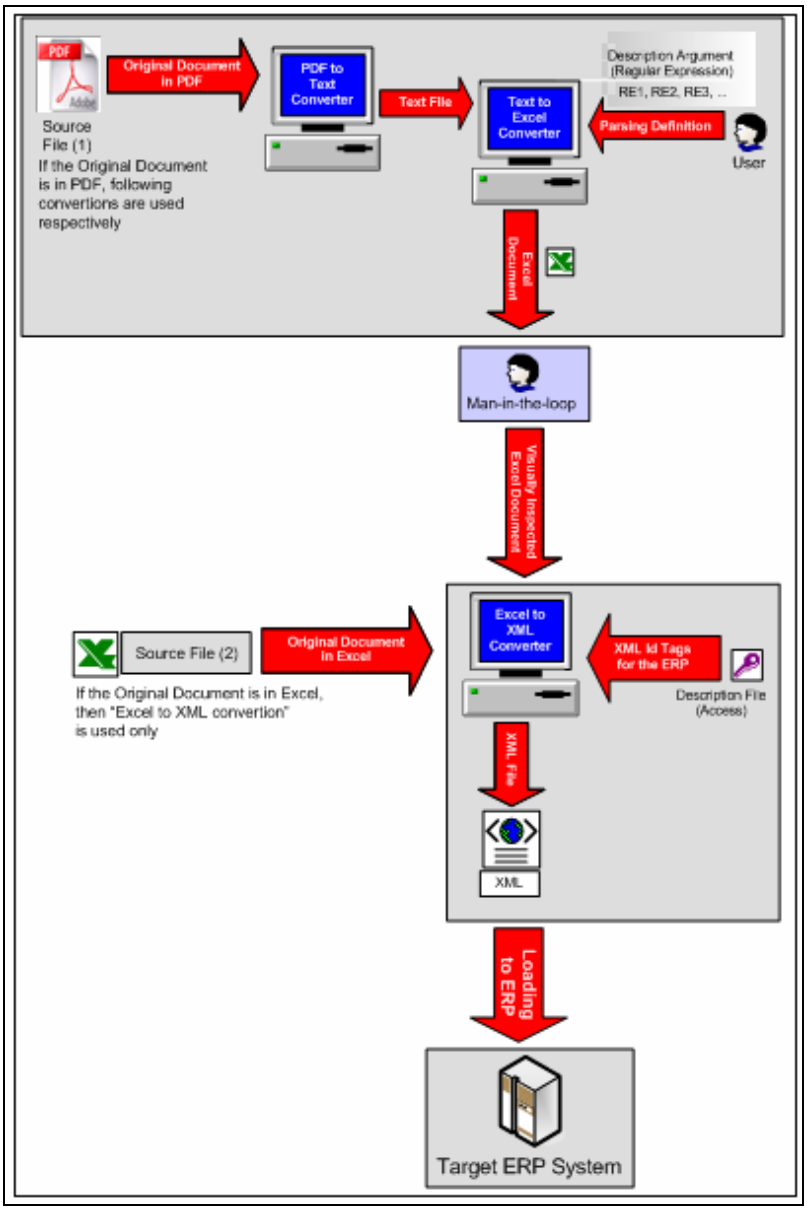


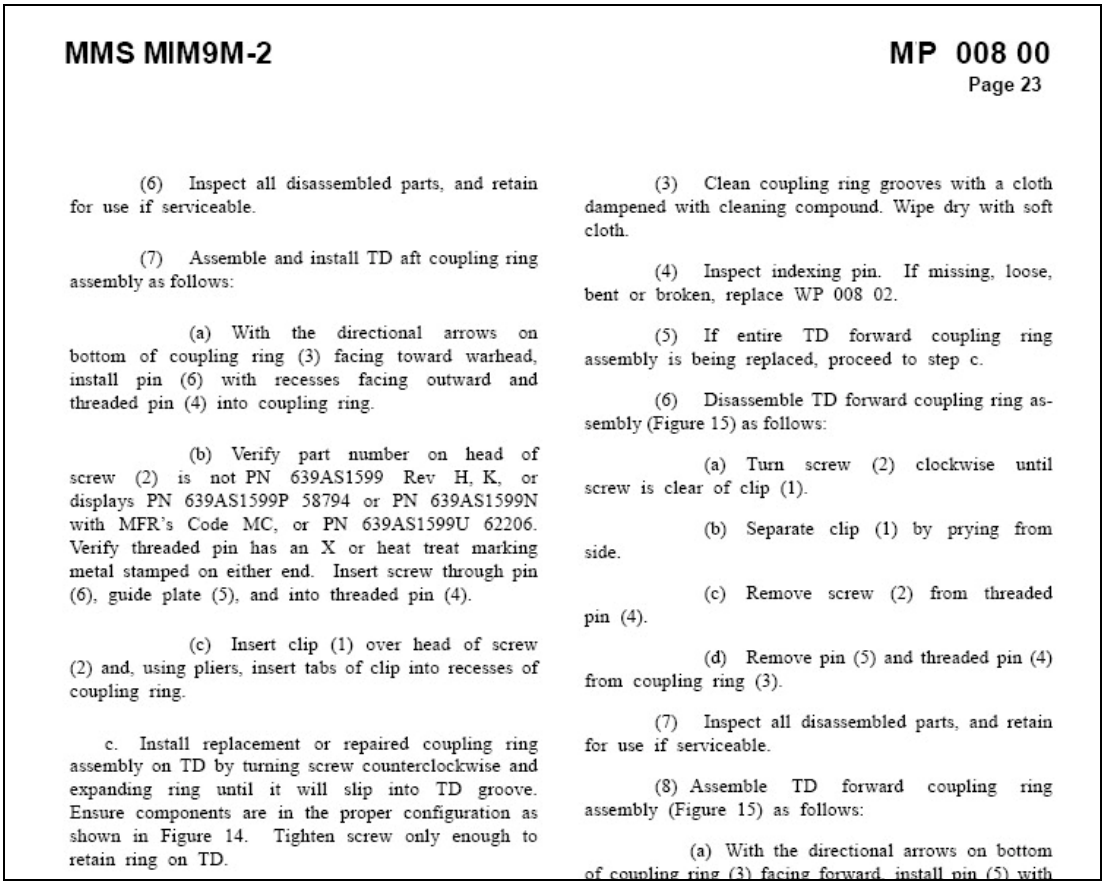

**Figure 3.1 General System Architecture**

In order to establish a basis for the creation of the software design, Entity Relationship Diagrams, Data Flow Diagrams and Sequence Diagrams were created. Concerning METEX design, Entity Relationship Diagrams in Appendix-A, Data Flow Diagrams in Appendix-B, Sequence Diagrams in Appendix-C and Finite State Automata diagram of selected Regular Expressions in Appendix-D were presented.

### 3.3.1 Source Files to be Processed

It is observed that the technical documents in the logistics domain exist in several file formats such as PDF, Word or Excel. In the current problem domain the source files exist either in PDF or Excel file formats. METEX consists of three sub

functions for the file format provided; Data extraction from PDF, Data extraction from Excel, Converting extracted data to XML. Figure 3.2 illustrates the sample PDF source file.



MMS MIM9M-2 MP 008 00
Page 23

(6) Inspect all disassembled parts, and retain for use if serviceable.

(7) Assemble and install TD aft coupling ring assembly as follows:

(a) With the directional arrows on bottom of coupling ring (3) facing toward warhead, install pin (6) with recesses facing outward and threaded pin (4) into coupling ring.

(b) Verify part number on head of screw (2) is not PN 639AS1599 Rev H, K, or displays PN 639AS1599P 58794 or PN 639AS1599N with MFR's Code MC, or PN 639AS1599U 62206. Verify threaded pin has an X or heat treat marking metal stamped on either end. Insert screw through pin (6), guide plate (5), and into threaded pin (4).

(c) Insert clip (1) over head of screw (2) and, using pliers, insert tabs of clip into recesses of coupling ring.

c. Install replacement or repaired coupling ring assembly on TD by turning screw counterclockwise and expanding ring until it will slip into TD groove. Ensure components are in the proper configuration as shown in Figure 14. Tighten screw only enough to retain ring on TD.

(3) Clean coupling ring grooves with a cloth dampened with cleaning compound. Wipe dry with soft cloth.

(4) Inspect indexing pin. If missing, loose, bent or broken, replace WP 008 02.

(5) If entire TD forward coupling ring assembly is being replaced, proceed to step c.

(6) Disassemble TD forward coupling ring assembly (Figure 15) as follows:

(a) Turn screw (2) clockwise until screw is clear of clip (1).

(b) Separate clip (1) by prying from side.

(c) Remove screw (2) from threaded pin (4).

(d) Remove pin (5) and threaded pin (4) from coupling ring (3).

(7) Inspect all disassembled parts, and retain for use if serviceable.

(8) Assemble TD forward coupling ring assembly (Figure 15) as follows:

(a) With the directional arrows on bottom of coupling ring (3) facing forward, install pin (5) with

**Figure 3.2 Sample Portion of PDF Source File**

Figure 3.3 illustrates a portion of a sample Excel source file.

Microsoft Excel - sample_excel_file.xls

File Edit View Insert Format Tools Data FlashPaper Window Help

F66 '4

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | A1 | A2 | A3 | A4 | A5 |
| 2 | PLKOD-KTEXT | PLKO-PLNNR | VT_S_PLKO-AENNR | PLKOD-SLWBEZ | VT_S_PLKO-TO_KEY |
| 3 | 1.23.2.1.1 | 1.23.2.1.1 | 1.23.2.1.12 | 1.23.2.1.1 | 1.23.2.1.1 |
| 4 | SEQUENCE OF OPERATIONS. | M9FH0G01 | | 300 | M9F0800GLI001 |
| 5 | SEQUENCE OF OPERATIONS. | M9FH0G01 | | 300 | M9FH0G01 |
| 6 | SAFETY AND ACCIDENT PREVENTION. | M9FH0G03 | | 300 | M9F0800GLI003 |
| 7 | SAFETY AND ACCIDENT PREVENTION. | M9FH0G03 | | 300 | M9FH0G03 |
| 8 | INCIDENTS/ACCIDENTS. | M9FH0G05 | | 300 | M9F0800GLI005 |
| 9 | INCIDENTS/ACCIDENTS. | M9FH0G05 | | 300 | M9FH0G05 |
| 10 | TEST RECORDS/DOCUMENTATION. | M9FH0G09 | | 300 | M9F0800GLI009 |
| 11 | TEST RECORDS/DOCUMENTATION. | M9FH0G09 | | 300 | M9FH0G09 |
| 12 | HANDLING AND TEST EQUIPMENT PREPARA- TION. | M9FH0G11 | | 300 | M9F0800GLI11 |
| 13 | HANDLING AND TEST EQUIPMENT PREPARA- TION. | M9FH0G11 | | 300 | M9FH0G11 |
| 14 | TORQUE REQUIREMENTS. | M9FH0G13 | | 300 | M9F0800GLI13 |
| 15 | MAINTENANCE. | M9FH0G15 | | 300 | M9F0800GLI15 |
| 16 | CLEANING AND CORROSION TREATMENT. | M9FH0G18 | | 300 | M9F0800GLI18 |
| 17 | CLEANING AND CORROSION TREATMENT. | M9FH0G18 | | 300 | M9FH0G18 |
| 18 | UMBILICAL BLOCK SPRING PIN REPLACE- MENT. | M9FH0G22 | | 300 | M9F0800GLI22 |
| 19 | UMBILICAL CABLE BREAKAWAY SCREW LOCK WIRE INSTALLA | M9FH0G23 | | 300 | M9F0800GLI23 |
| 20 | UMBILICAL CABLE PROTECTIVE CAP PRE- FORMED PACKING | M9FH0G24 | | 300 | M9F0800GLI24 |
| 21 | UMBILICAL CABLE PROTECTIVE CAP RE- PLACMENT/TEST. | M9FH0G25 | | 300 | M9F0800GLI25 |

**Figure 3.3 Sample Portion of Excel Source File**

### 3.3.2 Description Argument and Description File Used

During the conversion such as ; "PDF to Excel" and "Excel to XML" description argument and description files are used to describe how conversion will be done. Table 3.1 explains the description argument/file and their functions.

**Table 3.1 Description Argument/File And Their Functions**

| Description File | Phase Used | Purpose | Format |
|---|---|---|---|
| Regular Expressions | During the conversion from PDF to Excel | To define how the PDF will be parsed. | As an argument in the METEX GUI. |
| XML template for the target ERP system. | During the conversion from Excel to XML | To define how the XML will be formed for the target ERP system. | MS Access (*.mdb) |

Figure 3.4 illustrates the sample FSA (Finite State Automata) for the regular expression.

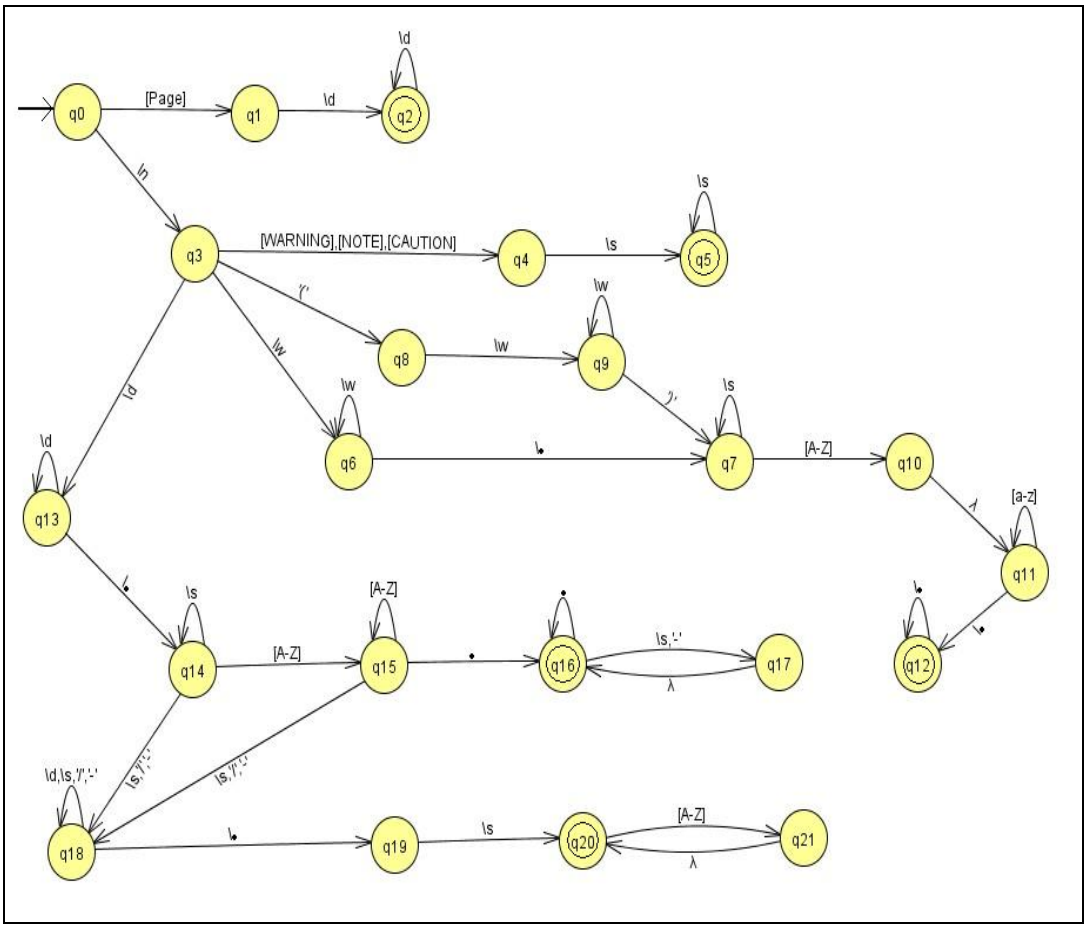



**Figure 3.4 Sample FSA (Finite State Automata) for a Sample Regular Expression**

In the Finite State Automata diagram, the used symbols and their meanings are as follows;

'\d' indicates a digit [0-9]

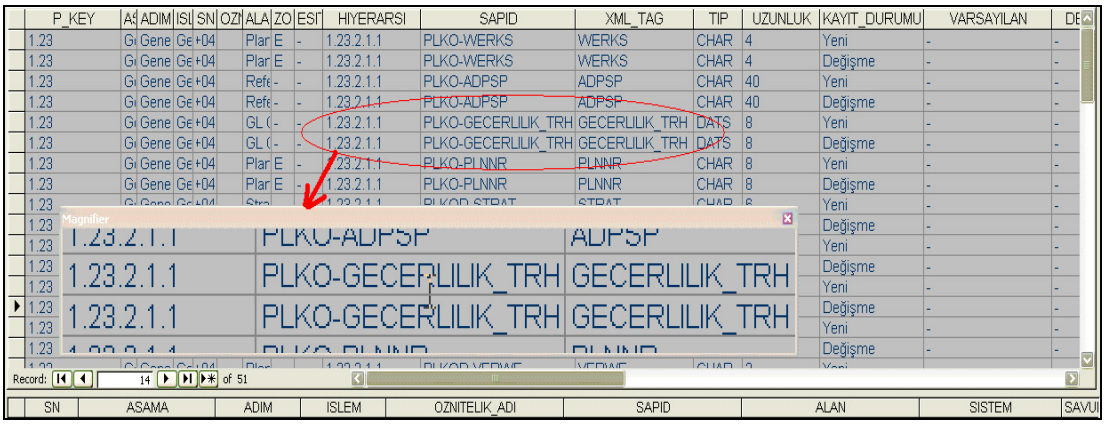
'\s' indicates a white space char [\t\n\r\f]

'\w' indicates a word char [a-zA-Z0-9_]

'\n' indicates new line char

'.' indicates any character

Figure 3.5 illustrates the portion of an Access file that contains XML Id-Tags for the target ERP system.



| P_KEY | AS | ADIM | ISL | SN | OZN | ALA | ZO | EST | HIYERARSI | SAPID | XML_TAG | TIP | UZUNLUK | KAYIT_DURUMU | VARSAYILAN | DE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.23 | G | Gene | Ge | +04 | | Plar | E | - | 1.23.2.1.1 | PLKO-WERKS | WERKS | CHAR | 4 | Yeni | - | - |
| 1.23 | G | Gene | Ge | +04 | | Plar | E | - | 1.23.2.1.1 | PLKO-WERKS | WERKS | CHAR | 4 | Değişme | - | - |
| 1.23 | G | Gene | Ge | +04 | | Refe | - | - | 1.23.2.1.1 | PLKO-ADPSP | ADPSP | CHAR | 40 | Yeni | - | - |
| 1.23 | G | Gene | Ge | +04 | | Refe | - | - | 1.23.2.1.1 | PLKO-ADPSP | ADPSP | CHAR | 40 | Değişme | - | - |
| 1.23 | G | Gene | Ge | +04 | | GL ( | - | - | 1.23.2.1.1 | PLKO-GECERLILIK_TRH | GECERLILIK_TRH | DATS | 8 | Yeni | - | - |
| 1.23 | G | Gene | Ge | +04 | | GL ( | - | - | 1.23.2.1.1 | PLKO-GECERLILIK_TRH | GECERLILIK_TRH | DATS | 8 | Değişme | - | - |
| 1.23 | G | Gene | Ge | +04 | | Plar | E | - | 1.23.2.1.1 | PLKO-PLNNR | PLNNR | CHAR | 8 | Yeni | - | - |
| 1.23 | G | Gene | Ge | +04 | | Plar | E | - | 1.23.2.1.1 | PLKO-PLNNR | PLNNR | CHAR | 8 | Değişme | - | - |
| 1.23 | | | | | | | | | | | | | | Yeni | - | - |
| 1.23 | | | | | | | | | | | | | | Değişme | - | - |
| 1.23 | | | | | | | | | | | | | | Yeni | - | - |
| 1.23 | | | | | | | | | | | | | | Değişme | - | - |
| 1.23 | | | | | | | | | | | | | | Yeni | - | - |
| 1.23 | | | | | | | | | | | | | | Değişme | - | - |
| 1.23 | | | | | | | | | | | | | | Yeni | - | - |
| 1.23 | | | | | | | | | | | | | | Değişme | - | - |

Magnifier

| | | |
|---|---|---|
| 1.23.2.1.1 | PLKO-ADPSP | ADPSP |
| 1.23.2.1.1 | PLKO-GECERLILIK_TRH | GECERLILIK_TRH |
| 1.23.2.1.1 | PLKO-GECERLILIK_TRH | GECERLILIK_TRH |

Record: 14 of 51

| SN | ASAMA | ADIM | ISLEM | OZNITELIK_ADI | SAPID | ALAN | SISTEM | SAVU |
|---|---|---|---|---|---|---|---|---|

**Figure 3.5 Portion of the Access File Contains ERP's XML Id-Tags**

In order to provide better understanding for the conversion processes, the formation a single metadata will be explained step by step in the following parts.

### 3.3.3 PDF to Text Conversion

This conversion is made implicitly within the PDF to Excel conversion process by using DLL file provided by Adobe. In Figure 3.6, a sample PDF file that contains data *"31 May 2005"* is shown. This is the validation date of the technical document.

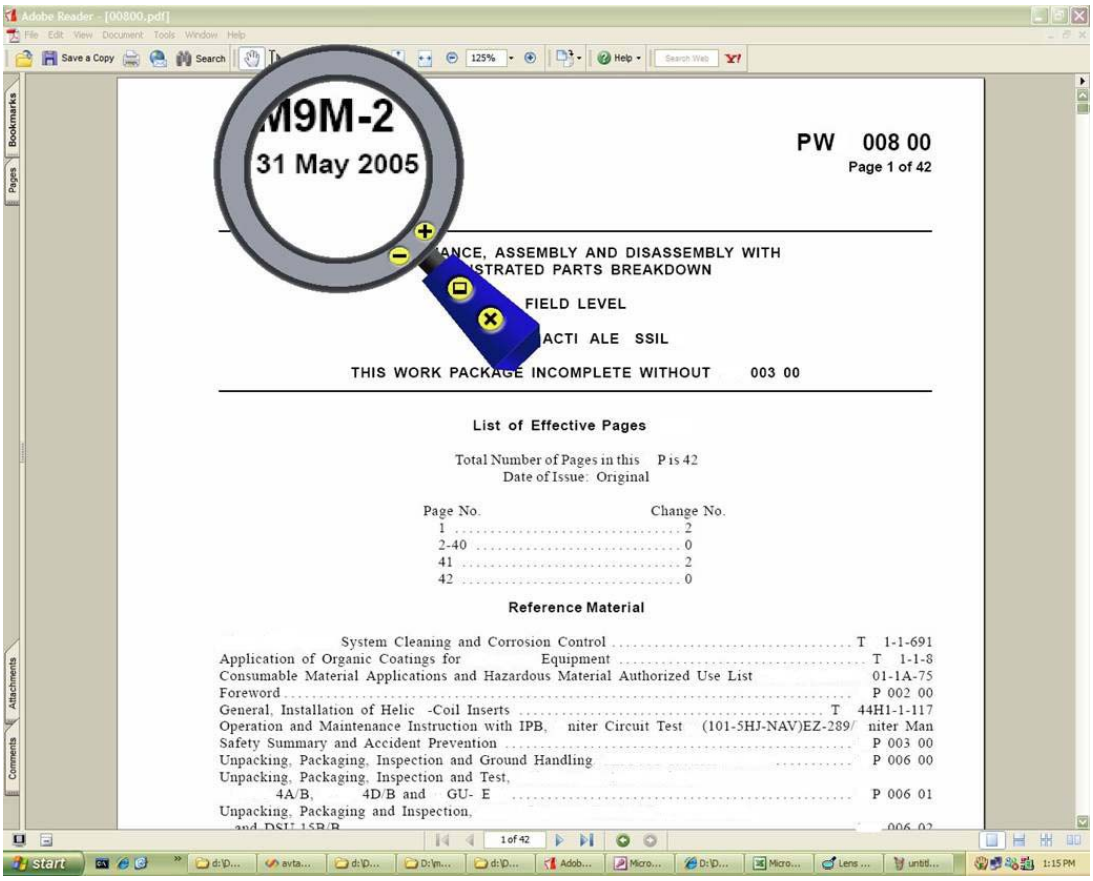



**Figure 3.6 Sample PDF File That Contains Date Information**

In the first stage of METEX (PDF to Excel Conversion), the selected source file in PDF is loaded by the user. By pressing "Text İşleme" button, the selected PDF document is converted to text, and the text-based-data is presented in the middle layout of Excel conversion dialog box.

### 3.3.4 Text to Excel Conversion

After the conversion of PDF document into the text file and browsing it in the user interface, text file processing is started. In the second layout of the excel conversion dialog-box, rest of the data are presented. In the first lens of Figure 3.7, the sample data *"31 May 2005"* is shown. In the light of regular expressions, this data is extracted from the text-based data. After regular expression processing it is formatted into the format shown in second lens.

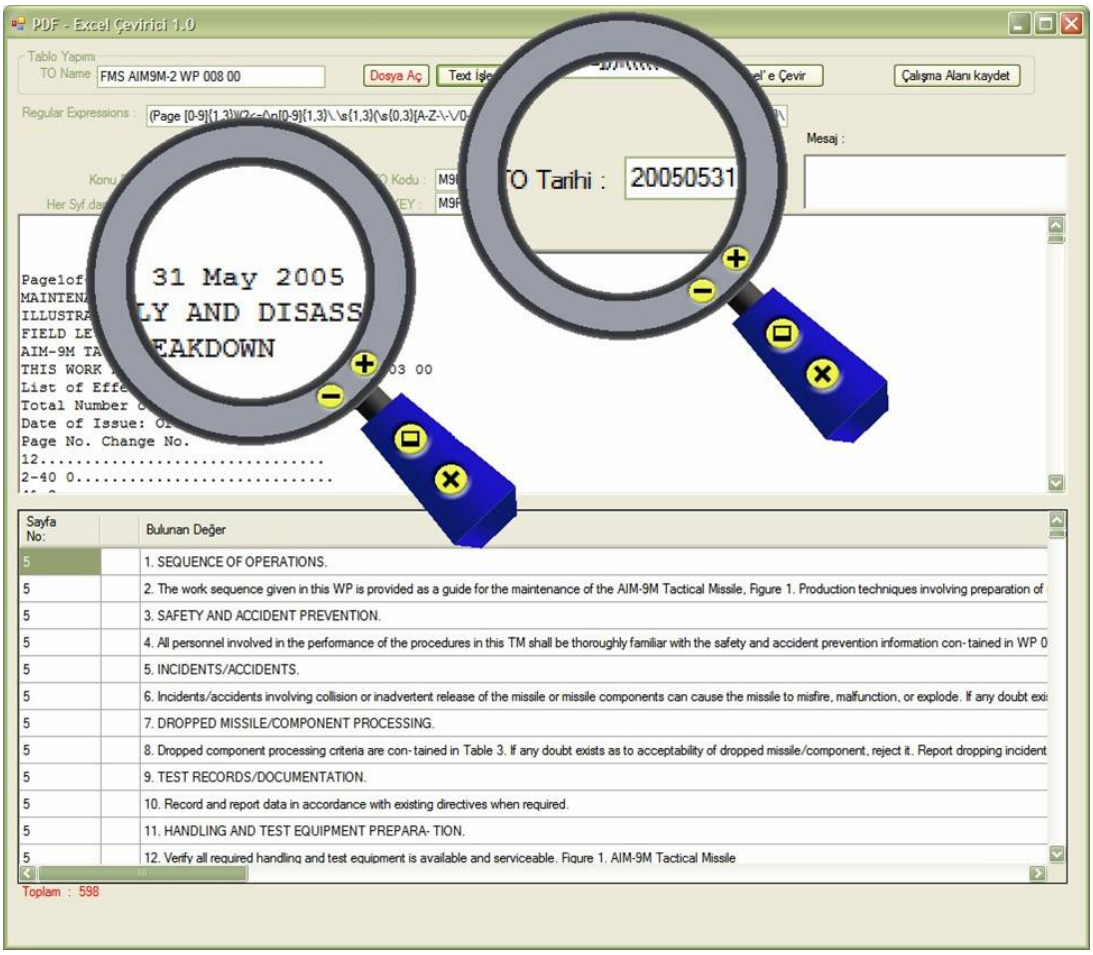



**Figure 3.7 Extraction of Date Information From Text Body**

Figure 3.8 shows the source code that executes the above mentioned process.

```
private string TO_tarih()
{
    Regex tarih_regex = new Regex(@"[0-9]{1,2}\s[A-Z][a-z]{2,15}\s[1-2][0-9]{3}");
    Match bulunan_tarih = tarih_regex.Match(txt_display.Text);
    if (bulunan_tarih.Success) return Convert.ToDateTime(bulunan_tarih.Value).ToString("yyyyMMdd");
    else return "";

}
```

**Figure 3.8 The Source Code that Executes the Parsing Process**

Figure 3.9 illustrates the Finite State Automata (FSA) for the above given regular expression.

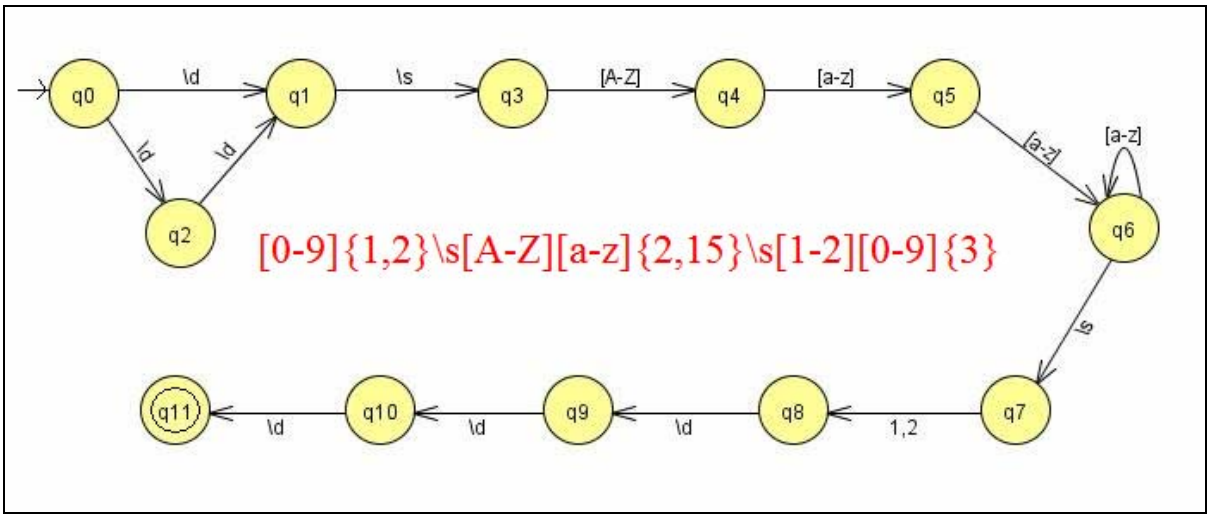



**Figure 3.9 The FSA for Above Given Regular Expression**

The next step is the conversion to Excel. After "Excel'e Çevir" button is clicked, the extracted data are written to an Excel file. The formatted date info *"20050531"* is shown in A12 column of Figure 3.10. In the lens there are three types of data are shown; ERPID which is *"PLKO-GECERLILIK_TRH"*, The hierarchy number of that data *"1.23.2.1.1"*, The formatted data for the Date.

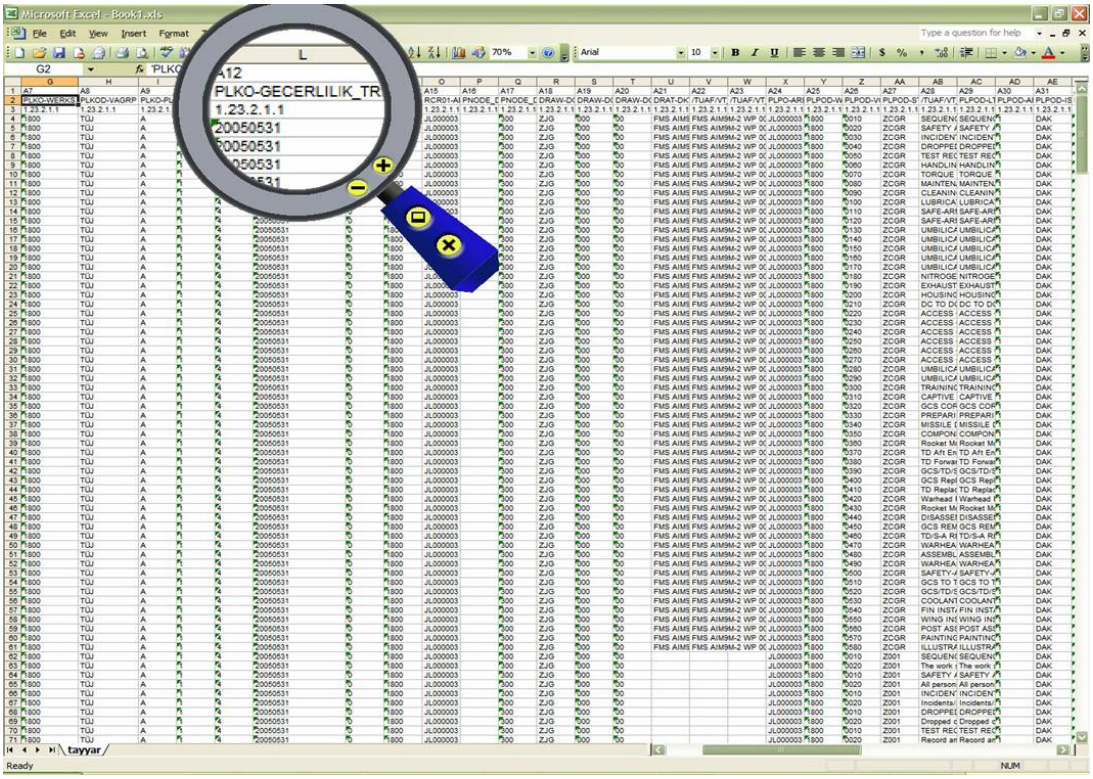



**Figure 3.10 The Excel File Contains Extracted Data**

Finally the Excel conversion is completed. The document validation date is inserted into Excel. After the generation of Excel file, the visual inspection is made by the user (Man-in-the-loop).

### 3.3.5 Man-In-The-Loop

As produced data will be used by the troubleshooters (in the ERP system), it is considered that it is convenient to insert human-control called “man-in-the-loop” after conversion process. After performing document anatomy study, the document hierarchy is recognized and a valid regular expression is prepared in order to extract data from the source document. As source documents are in standard structure, conversion process (text to excel converter) is ready to handle expected document. But there are rare documents that mismatch that expected document structure. In case of mismatch, the excel file will not be generated properly. For instance, a document that contains publication name in the wrong location of the document will cause the wrong excel data sheet to be produced. As a result of mislocated data in the document, there will be failure in production of excel document. In such case, there may be date information in the publication-name-location of excel file. In this point, it is considered that visual inspection of the produced excel document is required. In this case there are some alternatives to handle unexpected source document. These are ; to change the regular expression in a way that handle this unexpected document, or to change the structure of unexpected source document to expected document hierarchy. Where applicable, one of the above mentioned alternatives can be used to handle this unexpected document. The inspected Excel document is now ready to conversion to XML.

### 3.3.6 Excel to XML Conversion

In the XML conversion of METEX, the description file (in Access format) holding the XML tags of ERP system is used. This database file is used to access the data in the Excel file. Figure 3.11 illustrates the hierarchy number, the ERPID and the XML tag name of “Document Date” data.

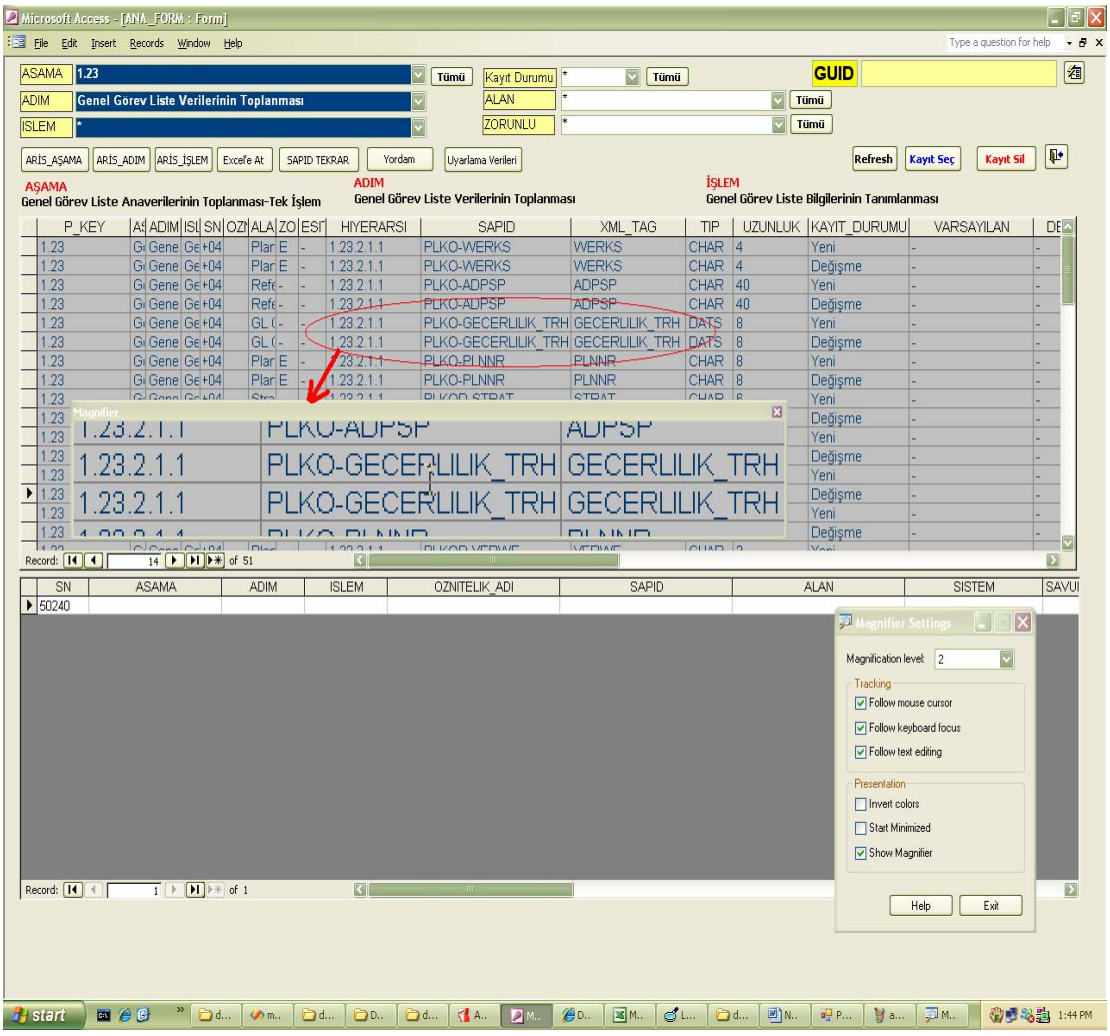



**Figure 3.11 Description File Contains XML-Tags Correspond to Excel Data**

In accordance with the relation-definition given in the description file (Access), the Excel data and, the XML-Tags in the Access file are paired. And the relevant data are inserted between the corresponding ERP-Tags. Finally they are written in to the XML file. As illustrated in Figure 3.12, the XML-tag data of "document validation date" named "*GECERLILIK_TRH*" is presented in the XML body.

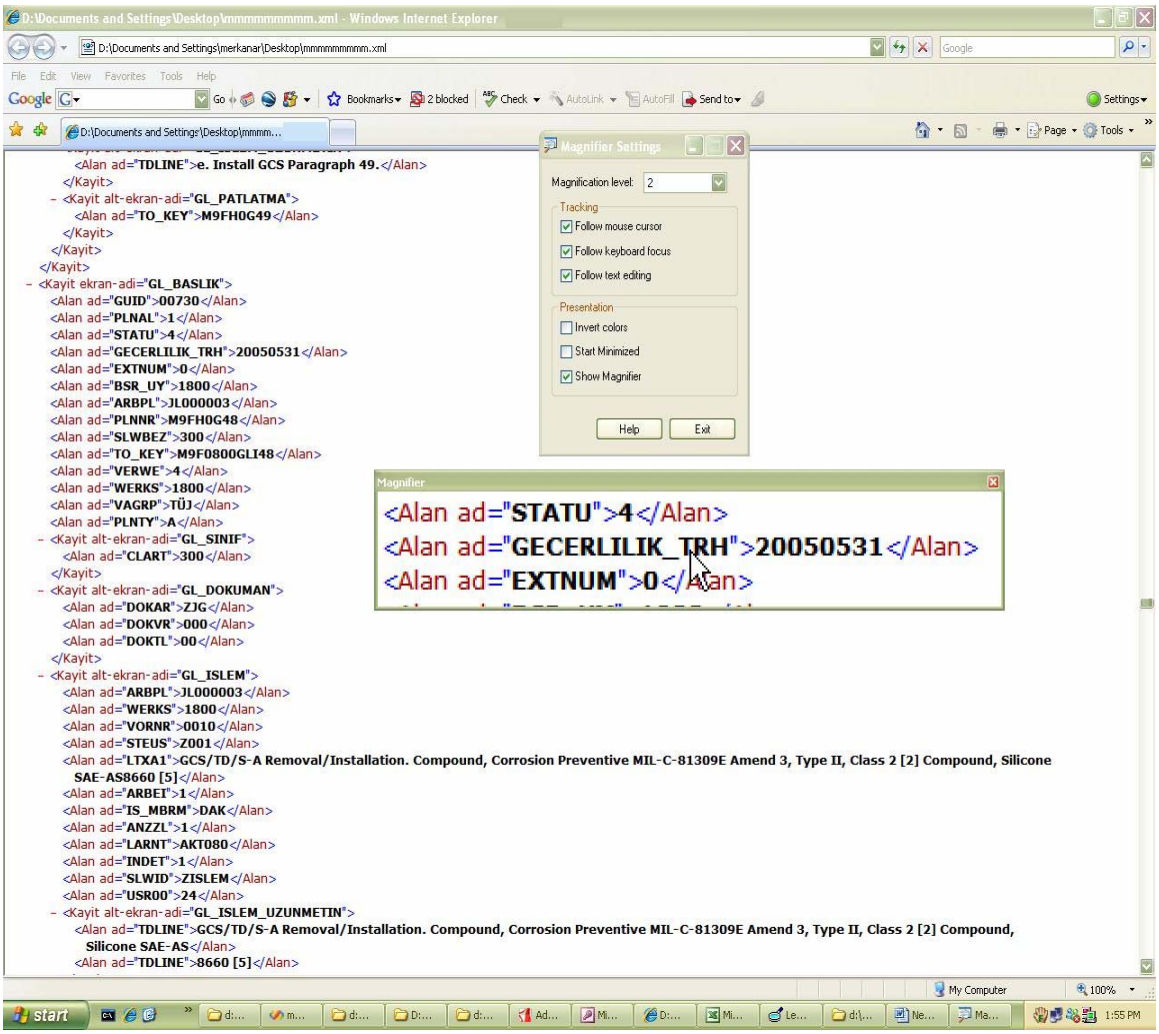



**Figure 3.12 Specific Data in the XML File**

The XML file generated by METEX will be injected in to the ERP system by Batch-Input-Method. Doing so, it is considered that a significant amount of time and money will be saved. Moreover manual data input error rate will be minimized as well. Shortening the data entry activity, will enable target ERP system to use as early as possible and make use of it as expected.

### 3.4 How Does METEX Work?

Within the framework of proposed case study there are some basic processes such as Conversion Processes, Loading Processes and XML creation processes. Conversion processes handle the source files provided. Depending on the source file (Original Document) provided, related conversion processes are used. For instance, if provided source file is in Excel format, then “Excel to XML” convertor is used. If provided source file is in PDF, then “PDF to Excel” converter is used first, then

"Excel to XML" converter is used. In "PDF to Excel" converter, there is another implicit conversion process called "PDF to Text". Within the conversion processes there are different loading processes such as "Loading PDF file", "Loading Excel", "Loading Access". As it is explained in earlier section of this chapter, METEX functionality is divided into two stages. Depending on the document type, the related conversion processes will be used in order to process these source documents. The detailed sub functions of METEX will be explained in the following part.

Handling PDF Source Documents:

Most of Technical Publications in the logistic material domain are in PDF. And processing of this digital data storage is quite difficult. As it doesn't contain structural data, the data transfer from PDF documents is not easy and fast. Besides, logistic units use Excel documents as data storage for non-digital data to collect, manage and process efficiently, easily and fast. As a result of this situation, and due to advantages of Excel documents, MS Excel document is chosen as default data storage format. Excel documents which are created by users in the logistic problem domain are accepted as required standard format. To convert PDF files into Excel files, a program named "PDF to Excel converter" is developed. This program is the first stage of METEX and it produces input for the second program (to convert data to XML). This program discussed below in details.

"PDF to Excel Converter" accepts a PDF document as an input, and it exhibits the text representation transformed with the help of "dll" (dynamic link library) to the user. In this stage of the program, user is expected to enter "regular expression" (as an argument) and the PDF related "chapter", "publication code" and "publication key" information for the purpose of accurate data extraction. This information such as "regular expression" are prepared at the end of document anatomy studies.

After the conversion of PDF document into the text file and after the browsing it in the user interface, text file processing is started by pressing "process text"(text işleme) button. After data values preparation, this information is presented to the user together with their page numbers. In order to store these values in Excel document, "table name" field that corresponds to "page name" should be entered by the user. In the next step, "convert to Excel"(Excel'e Çevir) button is pressed and an excel document named 'Book1.xls' is created and stored in the hard disk.(in the same folder where application software is installed). So, desired Excel document is

produced. Above mentioned “regular expression” and the PDF related “chapter”, “publication code” and “publication key” information are not expected to change. For that reason “save work area” button can be used to save the arguments and to use them on the later repeated processes. Figure 3.13 illustrates the screen shoot of “PDF to Excel Converter” main window.

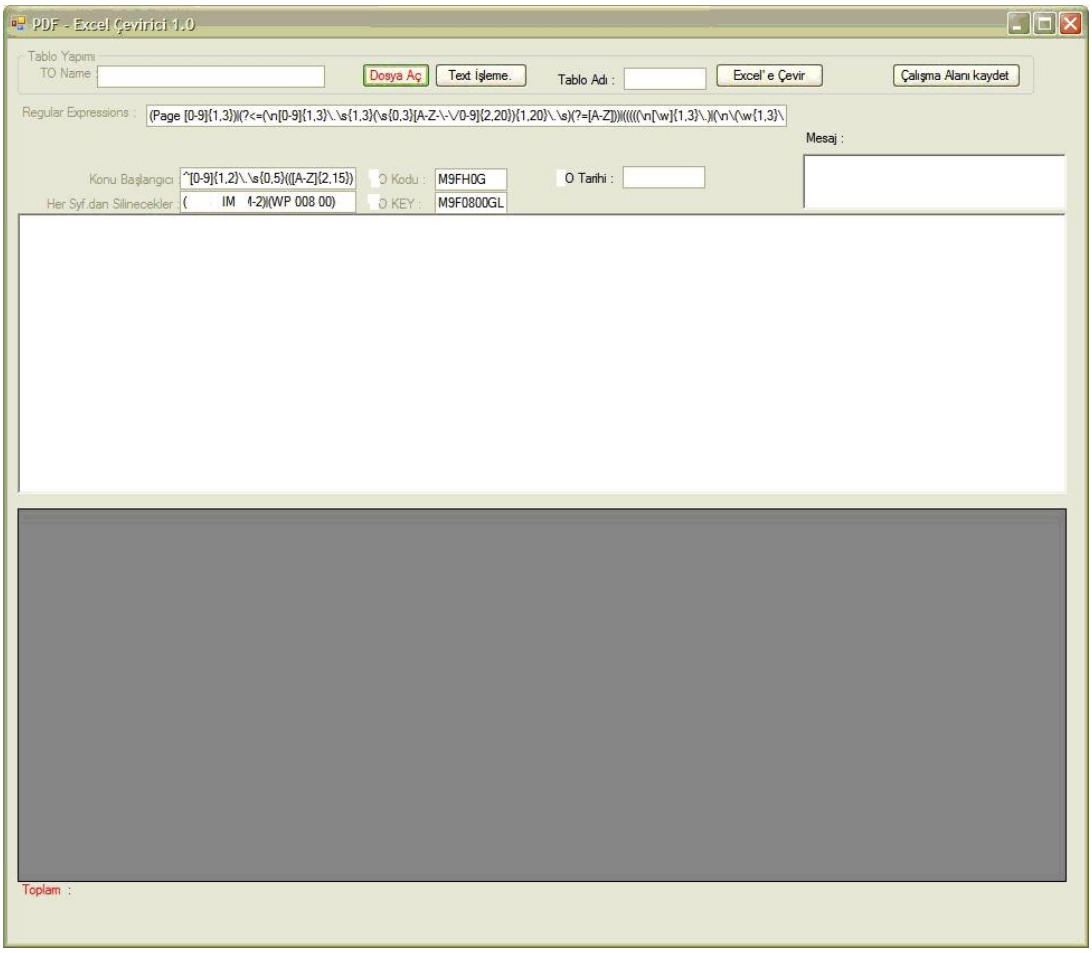



**Figure 3.13 PDF To Excel Converter Main Window**

Loading PDF File:

After starting METEX’s first stage “PDF to Excel”, an open-file-dialog-box pops up and user is asked to choose PDF file to be processed. At the end of this action target-PDF file is loaded into the system. Figure 3.14 illustrates a sample Open-file-dialog-box.

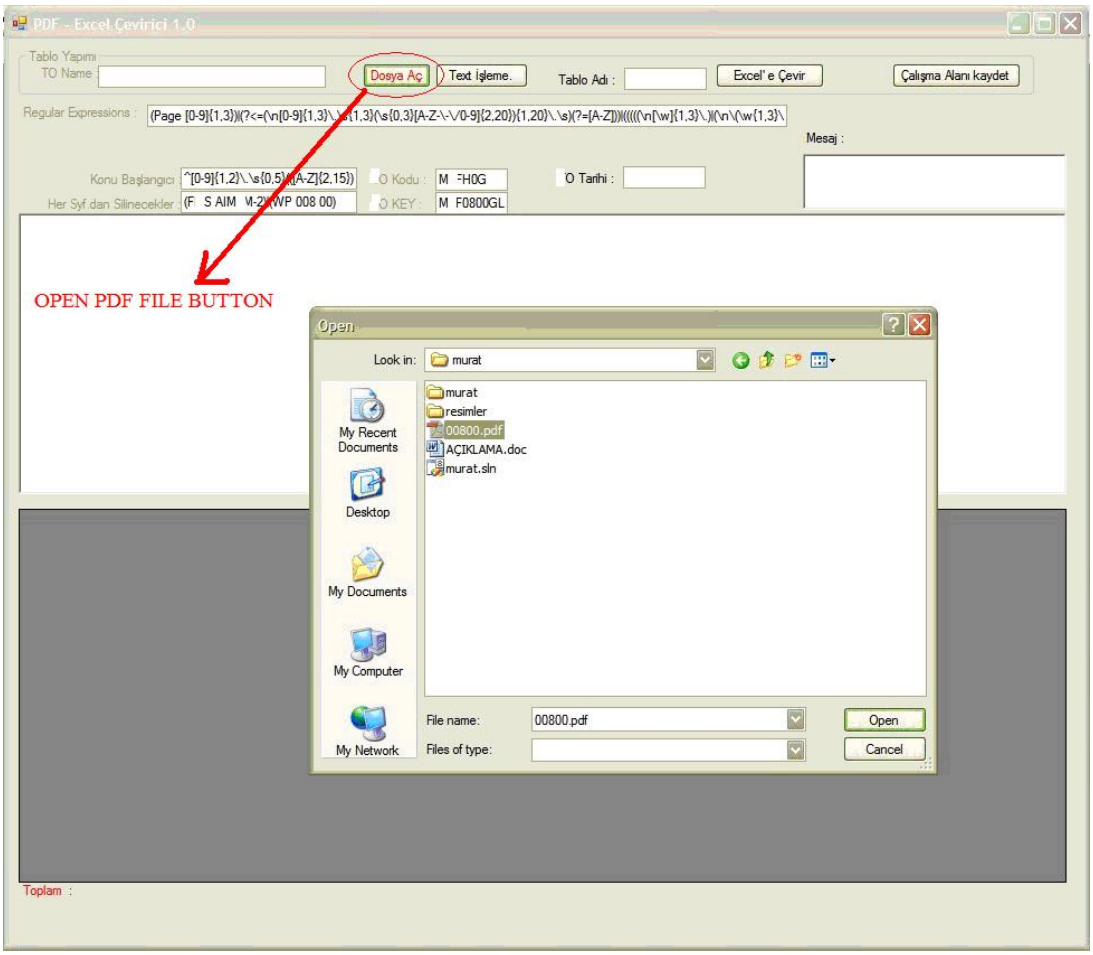



**Figure 3.14 Screenshot of Open-File-Dialog-Box**

Converting To Text File:

After selection of PDF document to be converted into Excel document, text file conversion process is executed in the background and it is presented to the user through the user interface's layout. Figure 3.15 illustrates a sample PDF to Text conversion result window.

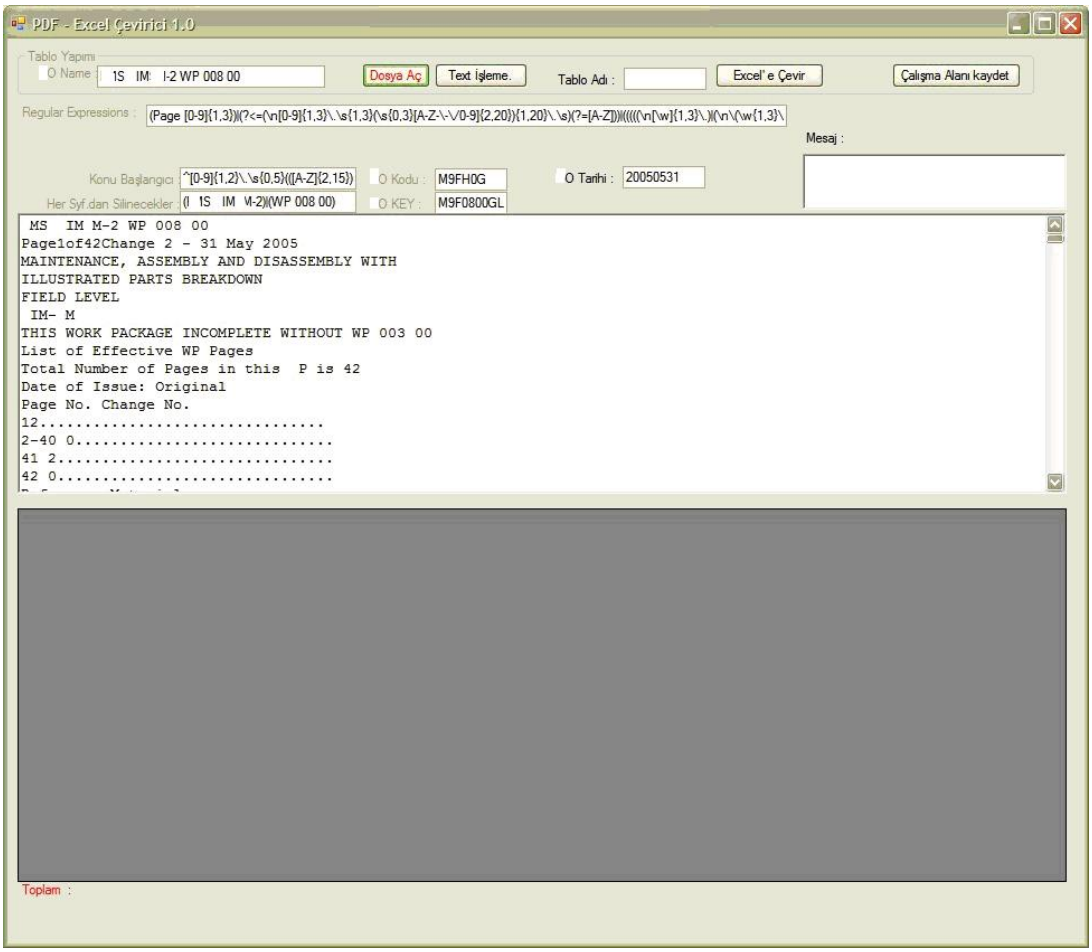



**Figure 3.15 Presentation of PDF To Text Conversion Result**

Formation of Data Cells, Customizing Regular Expression Rules:

In the light of regular expression, text file processing is commenced by pressing "process text"(text işleme) button. After data extraction, these extracted data are presented to the user together with their page numbers in the user interface's layout. Figure 3.16 Illustrates a sample Text Processing window.

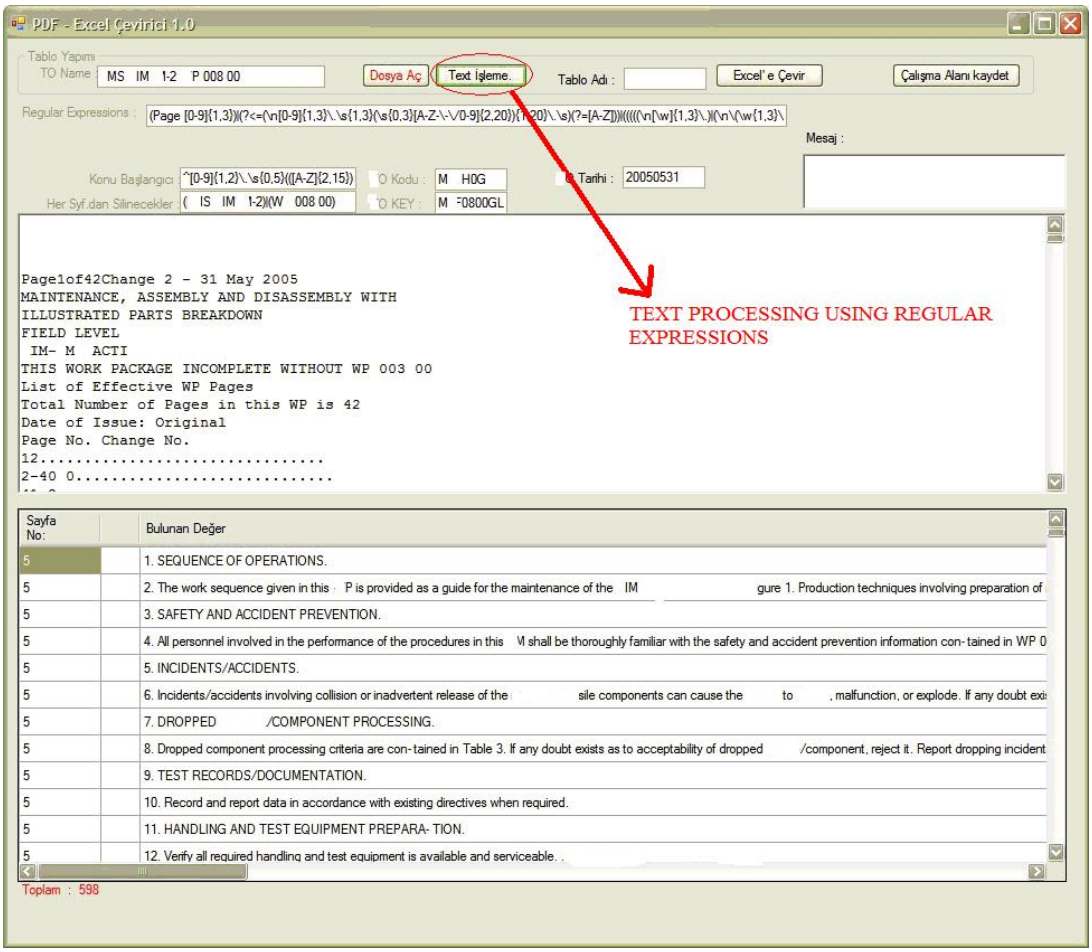



**Figure 3.16 Results of Text Processing Using Regular Expression**

<u>Storing the Results in Excel Document:</u>

The next step after extracting the needed data is to store them in Excel document. In order to perform this action, "convert to excel"(Excel'e Çevir) is clicked. The name of the excel sheet should be entered as well. After the completion of this process an Excel document named 'Book1.xls'(default name) is created and stored in the hard disk where application is installed. Figure 3.17 Illustrates the Excel-File-Creation window.

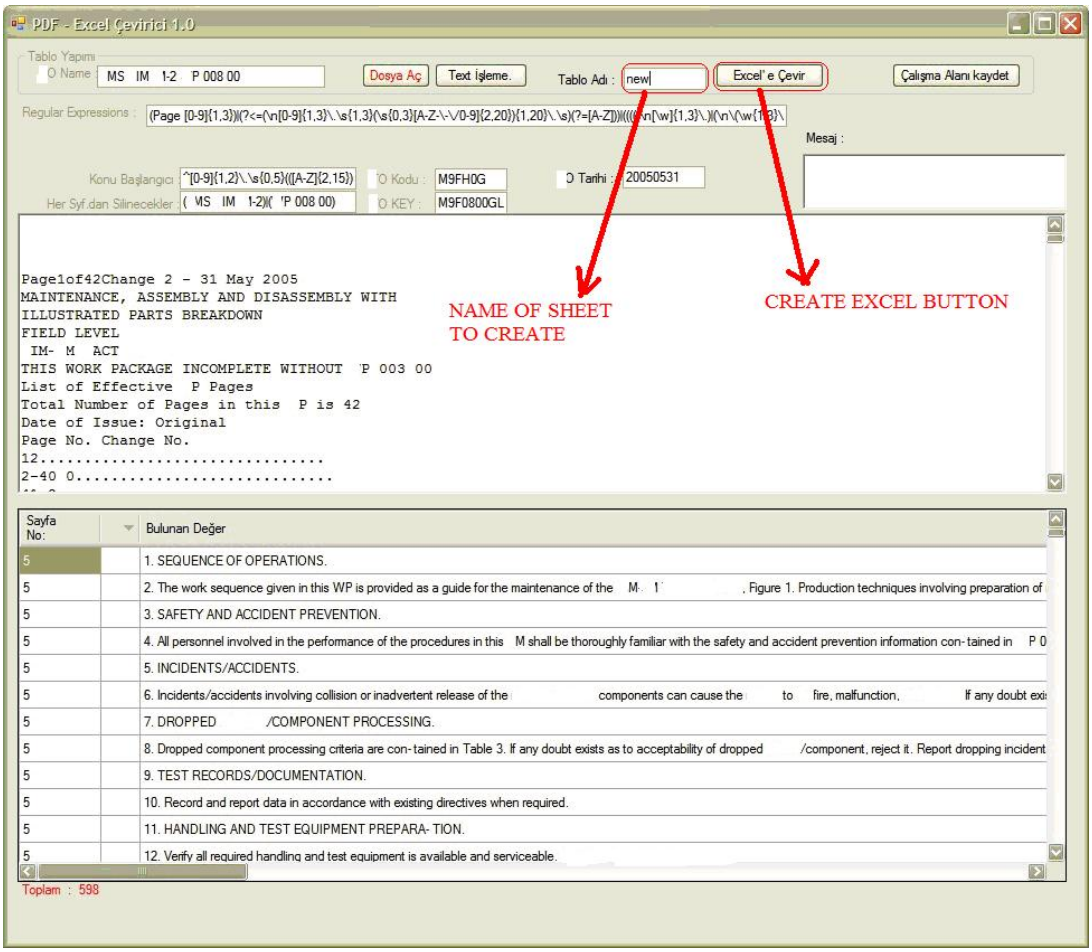



**Figure 3.17 Screenshot of Excel-File-Creation Window**

<u>Saving the Work Area:</u>

Above entered parameters such as "regular expression" and the PDF related "chapter", "publication code" and "publication key" arguments are not expected to change. For this reason, "save-work-area" button can be used to save the arguments and to use them on the later repeating processes. Figure 3.18 illustrates a Save-Work-Area functionality.

.

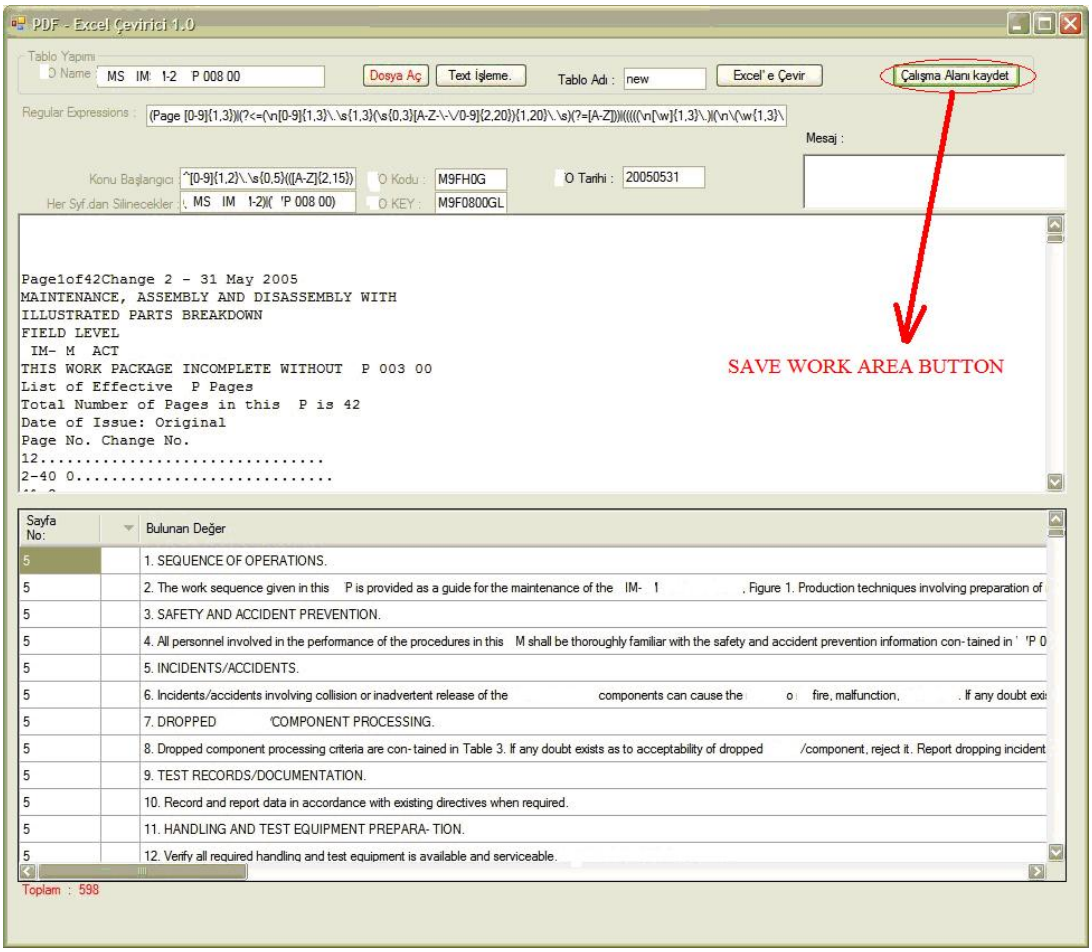



**Figure 3.18 Save-Work-Area Functionality**

Handling Excel Source Documents:

Conversion from Excel document to XML is performed by METEX's second portion named "Excel to XML converter". This function of the program accepts/uses an Excel document that is previously created by the first portion of METEX. Later on, using MS-Access file that stores needed process steps, the data coupled with data in the Excel document. Below, the screenshot of "Excel to XML converter" main window is shown (Figure 3.19). In this main window Excel file to be loaded is selected by a button named 'Excel Dosyasını Yükle'. Under this button there is a combo box which includes sheets in loaded Excel file. User selects sheet to process from this box. In the middle of main window a layout exists which shows column numbers of values to be taken from Excel file. Access file, which stores database's process step data, is loaded to the software by clicking a button named 'Ekran ve Alan İsimlerini Yükle' which is under the layout of window. In the second column of layout, paired column data, which are taken from loaded Access file, are shown. After a successful pairing process, by clicking button 'XML

Oluştur' an XML file, which is created from data in the layout, is shown in a new window. To save this XML file in new window, 'Kaydet' button can be clicked. After XML-file is formed, second part of METEX Excel to XML Converting is ended. In the main window at the bottom, there is a status bar named 'Durum:'. This bar shows the completing percentage of forming XML. 'Kapat' button at the bottom is used to terminate the program. The second part of METEX, which is discussed shortly above, is detailed below in a functional look. Figure 3.19 illustrates a sample Excel To XML Converter main window.

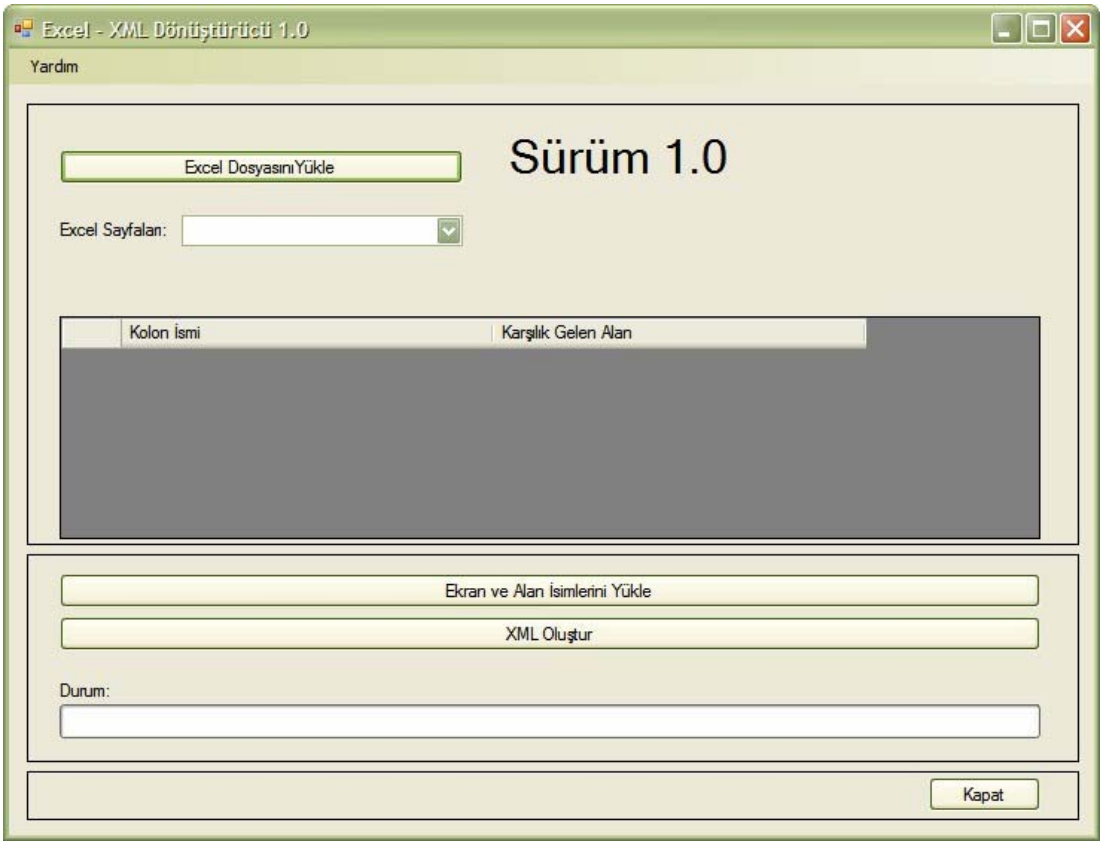



**Figure 3.19 Excel To XML Converter Main Window**

<u>Loading Excel File and Selecting Concerned Sheet:</u>

To load Excel file, which is the data storage and output of first program of METEX, the button 'Excel Dosyasını Yükle' is clicked. The Excel sheets are loaded automatically to combo box that is under the loading button. The sheet that includes our data storage would be selected using this combo box. Figure 3.20 illustrates a sample Load Excel and Select Sheet window.

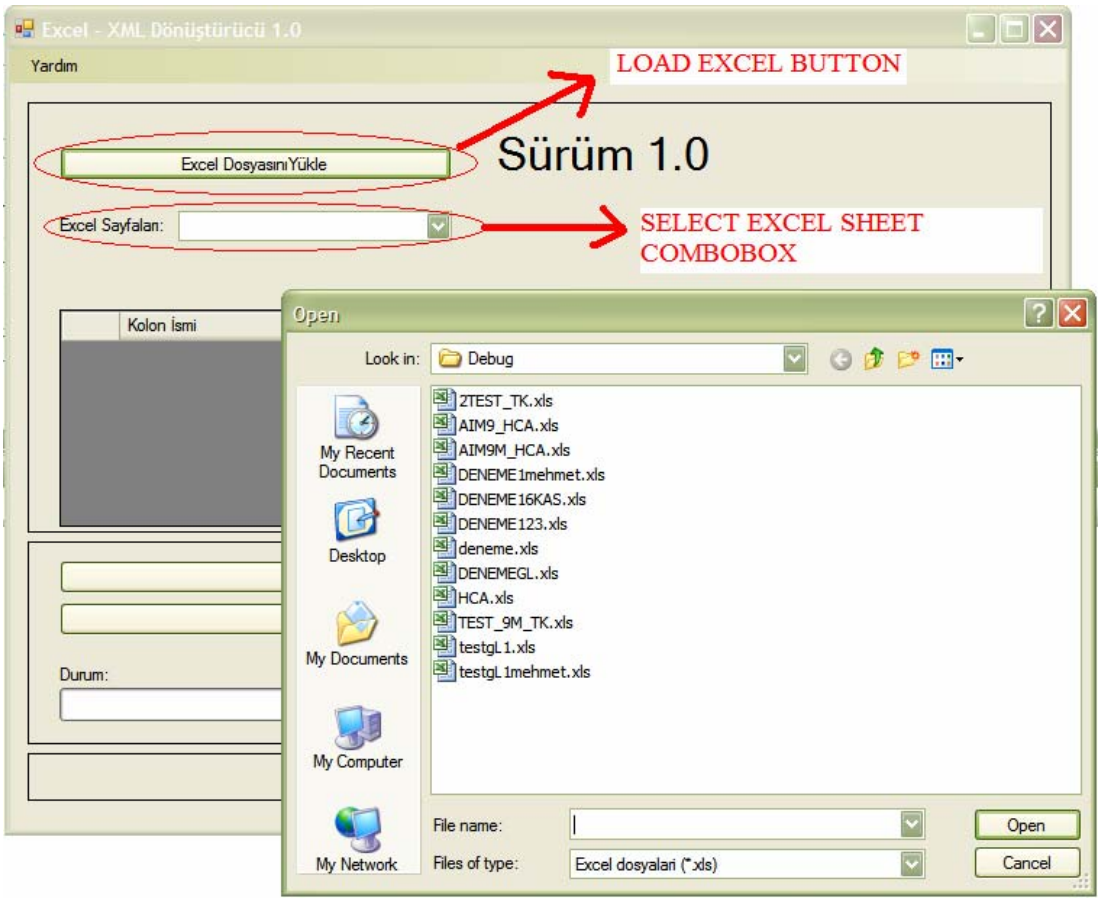



**Figure 3.20 Load Excel and Select Sheet**

Result of Loading Excel File:

After loading the correct Excel file and its concerned sheet, column numbers of required data of Excel sheet are loaded successfully to first column of layout in the middle of main window. If the loading process fails, an error message apperas indicating that there is an error while loading Excel file.

Successful Loading Excel File:

In the situation of correct and standard Excel file loaded, our layout's left column named 'Kolon İsmi' is filled. Figure 3.21 illustrates a Successful Loading Excel File window.

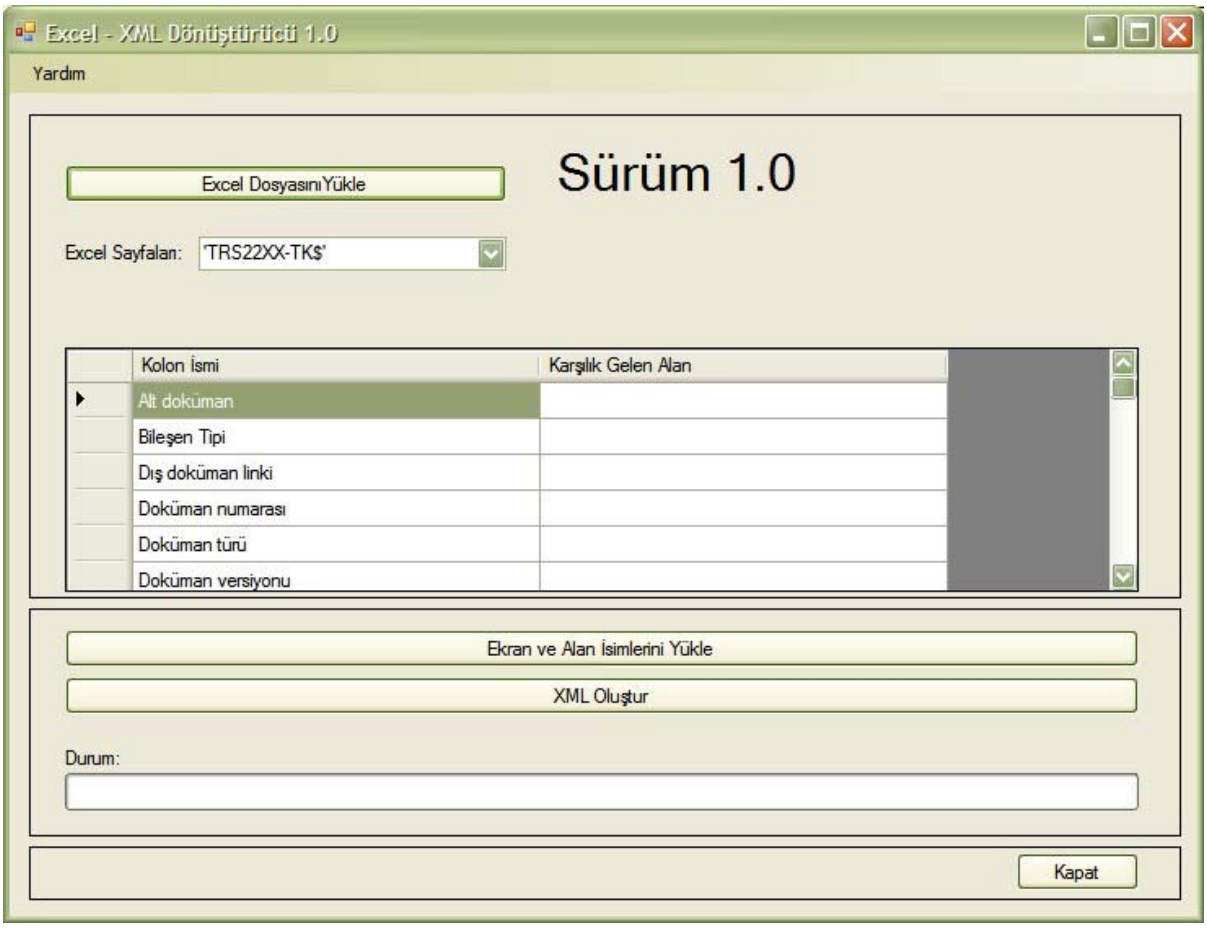



**Figure 3.21 Successfully Loaded Excel File**

Failure on Loading Excel File:

An explanatory error message is shown, if user loads a wrong Excel file. By the help of this message, user can restart loading Excel file process in the same way. Figure 3.22 illustrates a Fail on Loading Excel File window.

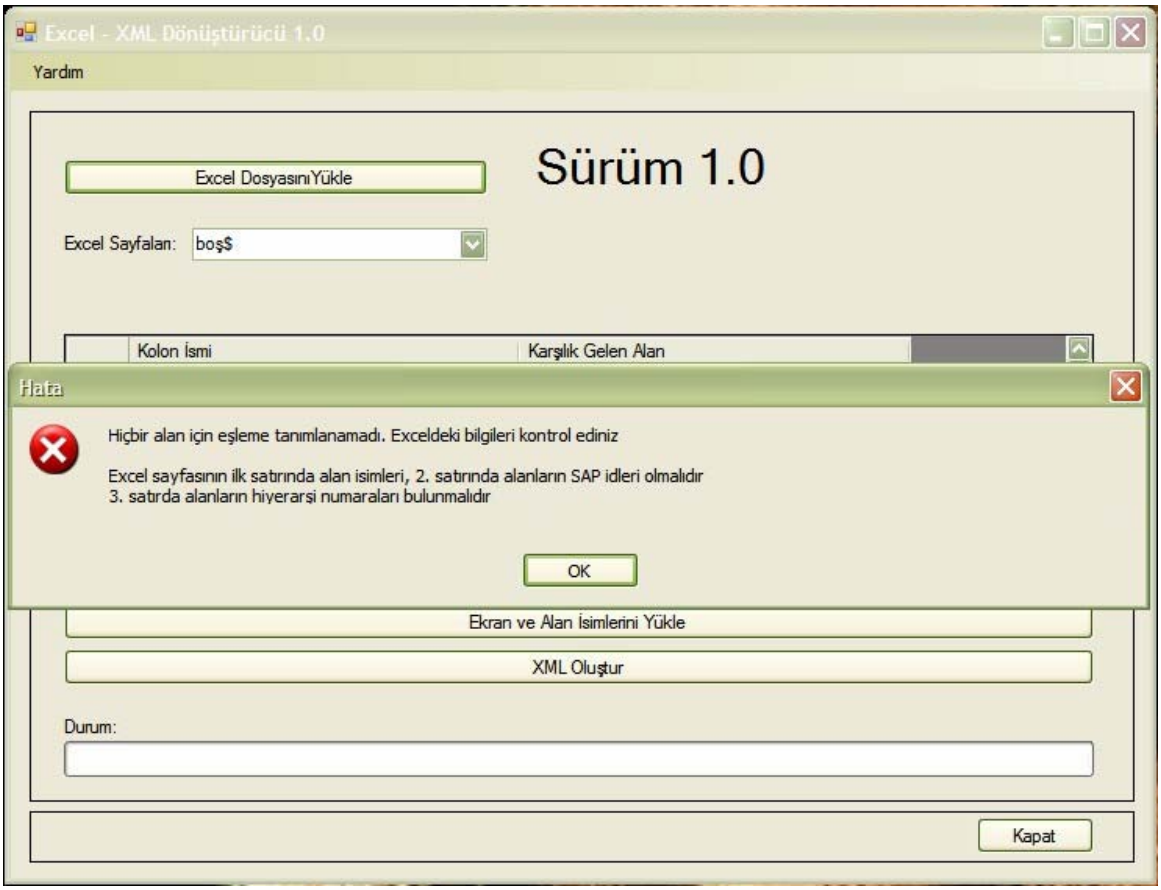



**Figure 3.22 Failure on Loading Excel File**

<u>Loading Access Database File:</u>

After loading Excel file successfully, an Access file, which includes required step process data, is loaded by clicking button 'Ekran ve Alan İsmilerini Yükle' to fill second column of our layout with data named *ERPID.* At the end of pairing process, the result will be presented in a new popup window. Figure 3.23 illustrates a sample Load Access window.

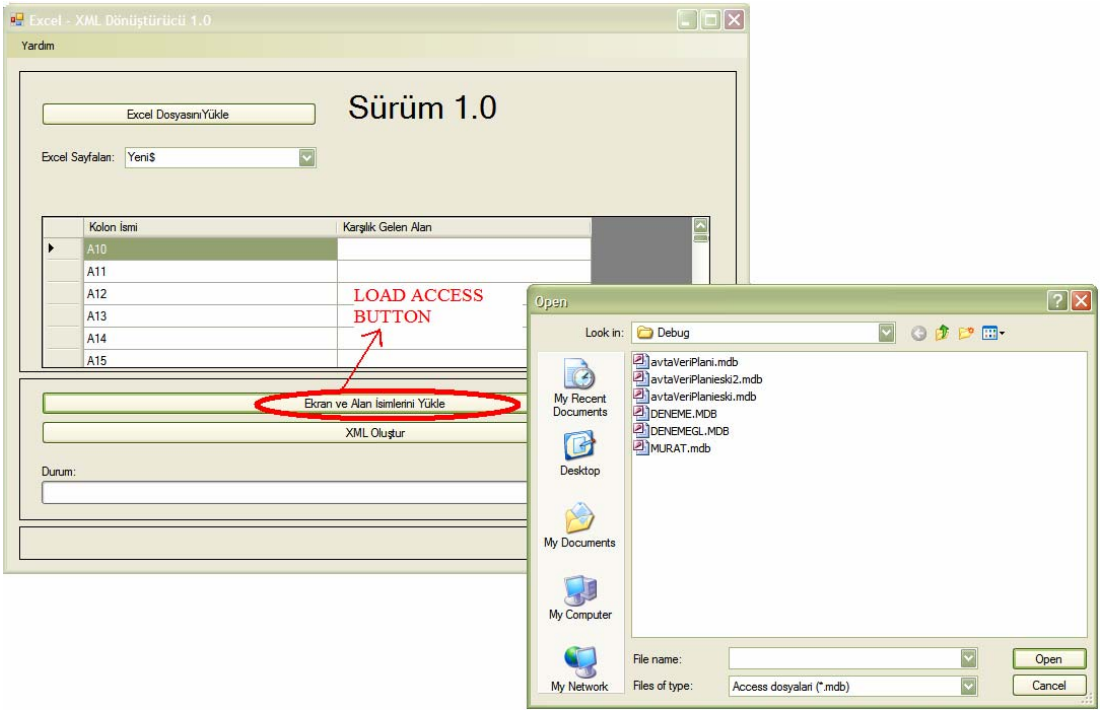



**Figure 3.23 Load Access File**

Result of Column Pairing of Loaded Access and Excel Files:

The result of column pairing process of loaded Access file with loaded Excel file is shown in a new popup window. If the process successful, the second column named 'Karşılık Gelen Alan' would be filled and an information popup window would be seen. If not, that column would be empty and an error-popup-window would be seen.

Successful Column Pairing of Loaded Access and Excel Files:

The second column of our layout 'Karşılık Gelen Alan' would be filled and an information of successfully column pairing would be seen after the correct and concerned Access file is loaded. Figure 3.24 illustrates a Successful Column Pairing of Loaded Access and Excel Files window.

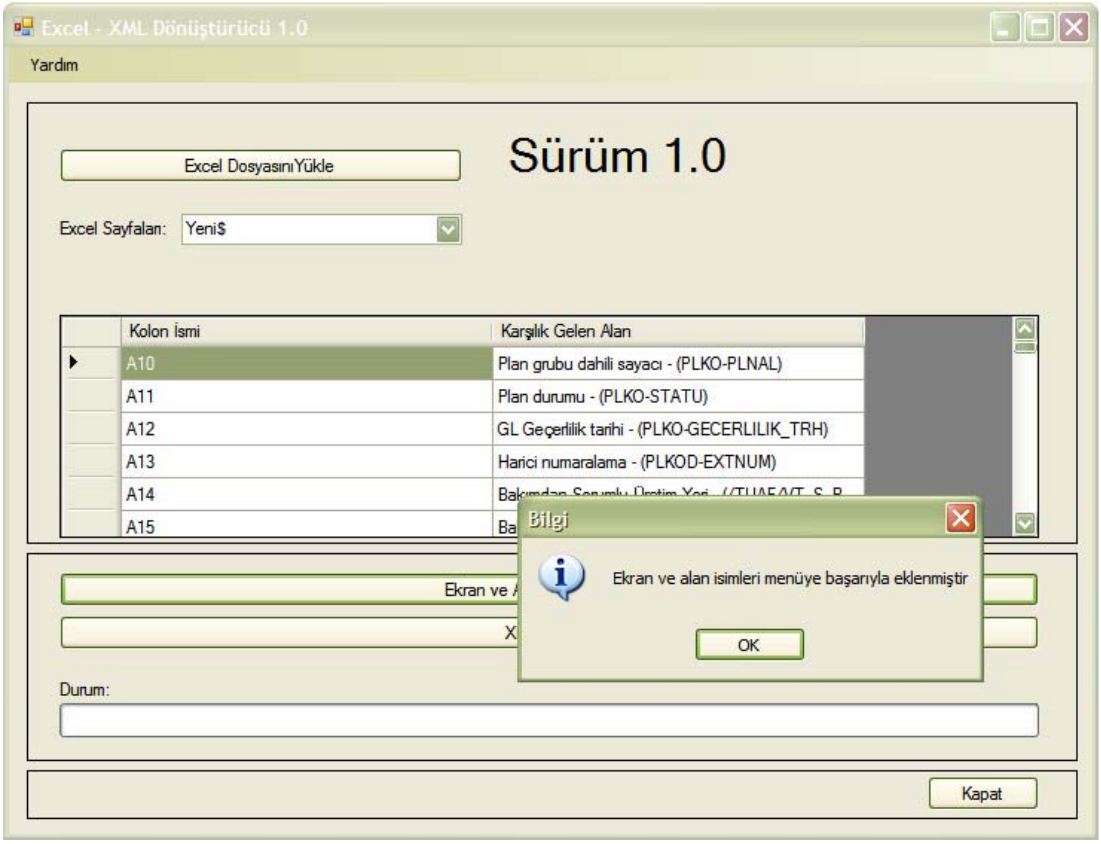



**Figure 3.24 Result of Column Pairing of Loaded Access and Excel Files**

<u>Failed Column Pairing of Loaded Access and Excel Files:</u>

If selected Access file does not include data in the standard format or a wrong Access file is selected, an error message, which explains that the selected Access file does not include required data, would be shown. According to this, the second column 'Karşılık Gelen Alan' would not be filled. Figure 3.25 illustrates a Failed Column Pairing of Loaded Access and Excel Files window.

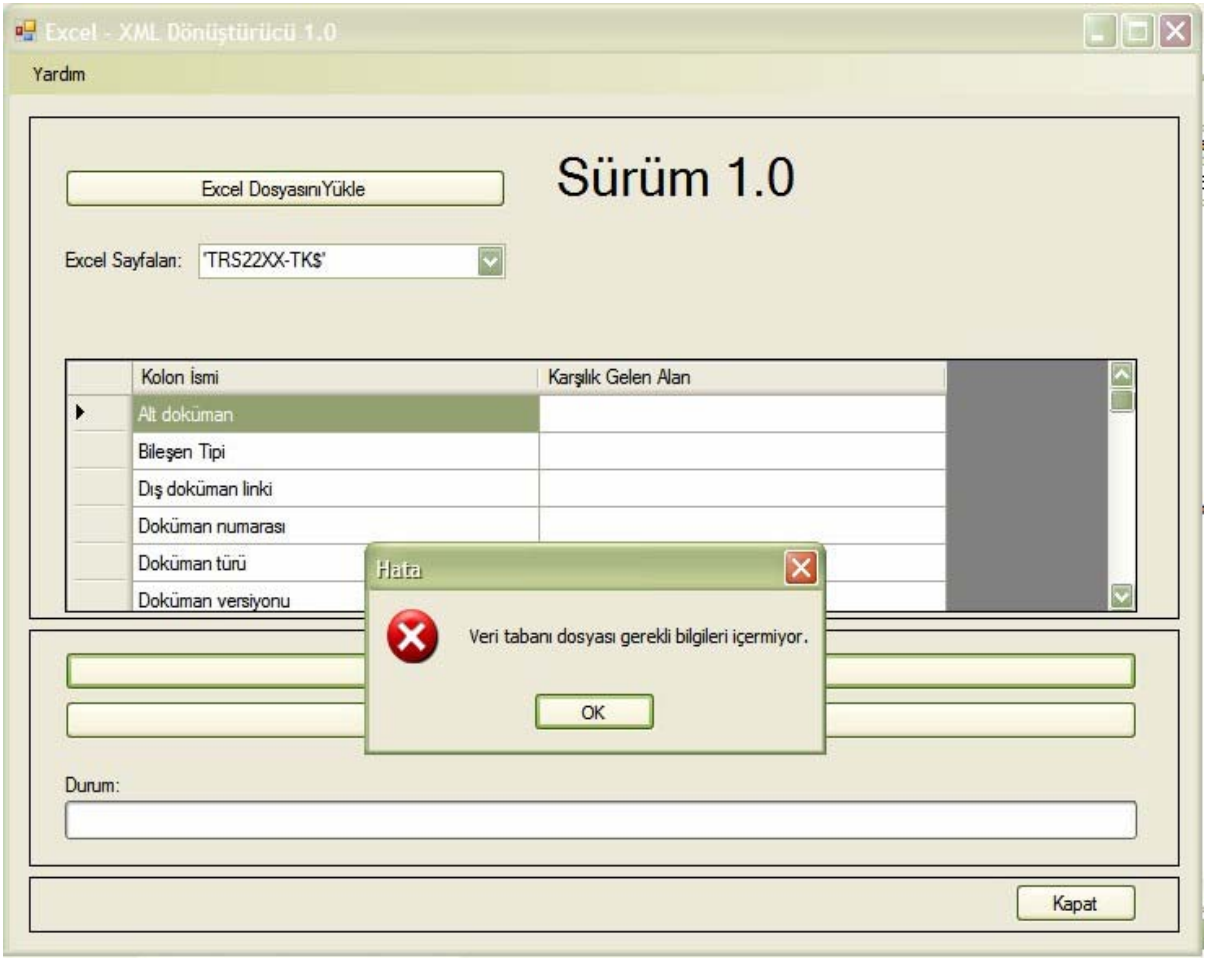



**Figure 3.25 Failure on Column Pairing of Loaded Access and Excel Files**

Creation of XML File:

After a successful pairing of loaded Access file with loaded Excel file's selected concerned sheet, last step forming XML is left behind. To do this, a button named 'XML Oluştur' is clicked. The status bar named 'Durum:', which is at the bottom of main window, shows the completing percentage of forming XML process. By filling of status bar, formed XML file is shown in a new window. Figure 3.26 illustrates a Create XML File window.

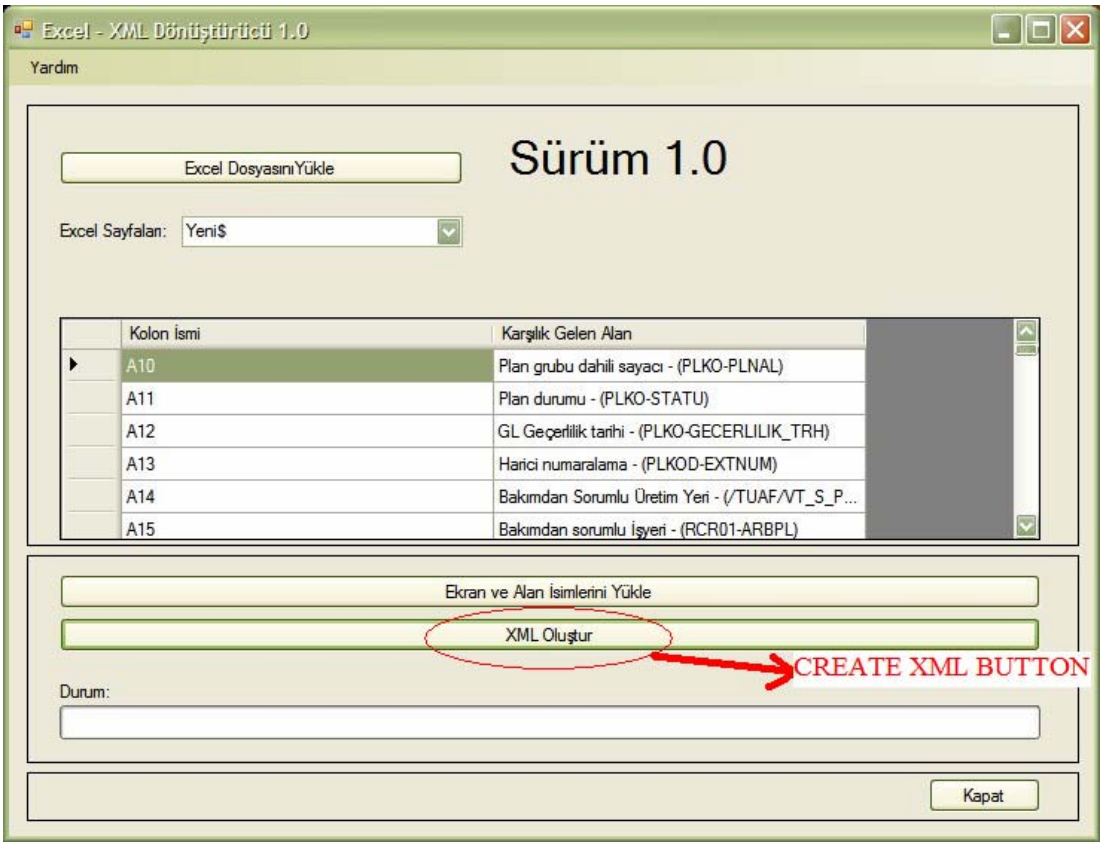



**Figure 3.26 Creation of XML File**

Created XML file window contains two buttons named 'Kaydet' and 'Kapat'. To close the XML window without saving button 'Kapat' button is clicked. To save formed XML file the button 'Kaydet' button is clicked. By clicking saving button, a save-file-dialog-popup is shown to browse where the file would be saved. After file name and the directory is defined, the XML file is saved. Figure 3.27 illustrates how to save a created XML file.

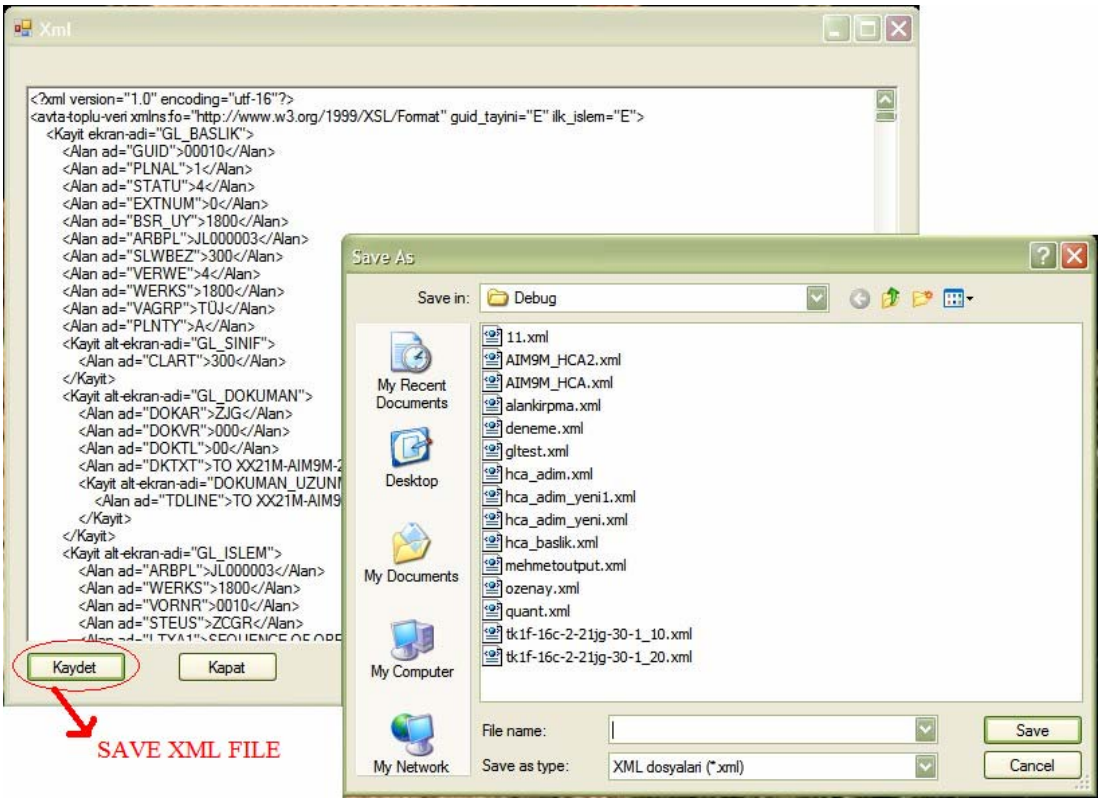



**Figure 3.27 Saving an XML File**

## 3.5. Technology Used

PDF (Portable Document Format):

Portable Document Format (PDF), sometimes mistaken for "Printable Document Format", is an open file format created by Adobe Systems in 1993 and is now being prepared for submission as an ISO standard. It is used for representing two-dimensional documents in a device independent and resolution independent fixed-layout document format. Each PDF file encapsulates a complete description of a 2D document (and, with the advent of Acrobat 3D, embedded 3D documents) that includes the text, fonts, images, and 2D vector graphics that compose the document. While PDF can describe very simple one page documents, it may also be used for many pages, complex documents that use a variety of fonts, graphics, colors, and images[38].

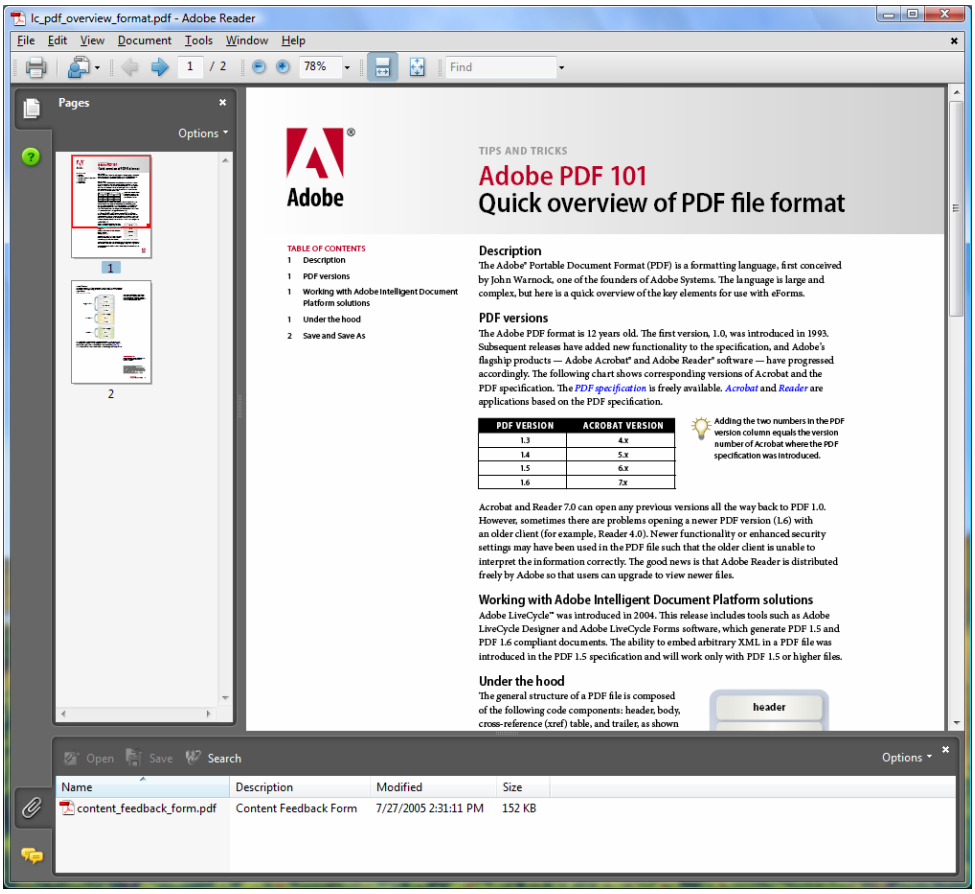



**Figure 3.28 Sample PDF File**

In real world logistic domain, data storage are held in PDF format. PDF files are input values of METEX software. Today's technology PDF document is one of the best in data reliability in documentation technology. So it is used in our program to hold data storage in a truth way.

Microsoft Excel:

Microsoft Excel (full name Microsoft Office Excel) is a spreadsheet program written and distributed by Microsoft for computers using the Microsoft Windows operating system and for Apple Macintosh computers. It features an intuitive interface and capable calculation and graphing tools which, along with aggressive marketing, have made Excel one of the most popular microcomputer applications to date. It is overwhelmingly the dominant spreadsheet application available for these platforms and has been so since version 5 in 1993 and its bundling as part of Microsoft Office [39]. We discussed that our input document for METEX is PDF in previous section. The documents and non-digital data storages used as input, which are not in PDF format, must be formatted in a digital format. So MS Excel is chosen

to store these types of data because of its flexibility and ease of usage. Thus, both a standard format is provided and a fast data transfer process has started.

Microsoft Access:

Microsoft Office Access, previously known as Microsoft Access, is a relational database management system from Microsoft which combines the relational Microsoft Jet Database Engine with a graphical user interface. It is a member of the 2007 Microsoft Office system. Access can use data stored in Access/Jet, Microsoft SQL Server, Oracle, or any ODBC-compliant data container. Skilled software developers and data architects use it to develop application software. Relatively unskilled programmers and non-programmer "power users" can use it to build simple applications. It supports some object-oriented (OO) techniques but falls short of being a fully OO development tool. Access was also the name of a communications program from Microsoft, meant to compete with ProComm and other programs. There is standard steps for data in real world logistics domain. Collected data which are in Excel files, must be paired with these steps. To hold step data, MS Access is used as a database. It is used because it is a very simple database tool for managing and loading data [40].

XML (Extensible Markup Language):

The Extensible Markup Language (XML) is a general-purpose markup language. Its primary purpose is to facilitate the sharing of data across different information systems, particularly via the Internet. It is a simplified subset of the Standard Generalized Markup Language (SGML), and is designed to be relatively human-legible. By adding semantic constraints, application languages can be implemented in XML. These include XHTML, RSS, MathML, GraphML, Scalable Vector Graphics, MusicXML, and thousands of others. Moreover, XML is sometimes used as the specification language for such application languages. XML is recommended by the World Wide Web Consortium. It is a fee-free open standard. The W3C recommendation specifies both the lexical grammar, and the requirements for parsing [41]. For loading data to our system, a standard format is needed. This standard format is XML because XML is a tool to create easy documents to be read

by both human and data process systems and besides hiding data it is used as a middle interpreter format for data transfer.

DLL (Dynamic-Link Library):

Dynamic-link library (also written without the hyphen), or DLL, is Microsoft's implementation of the shared library concept in the Microsoft Windows and OS/2 operating systems. These libraries usually have the file extension **DLL, OCX** (for libraries containing ActiveX controls), or DRV (for legacy system drivers). The file formats for DLLs are the same as for Windows EXE files — that is, Portable Executable (PE) for 32-bit Windows, and New Executable (NE) for 16-bit Windows. As with EXEs, DLLs can contain code, data, and resources, in any combination. In the broader sense of the term, any data file with the same file format can be called a resource DLL. Examples of such DLLs include icon libraries, sometimes having the extension **ICL**, and font files, having the extensions **FON** and **FOT**. [42][43]. In METEX software, for processing PDF document typed digital data, initially to convert PDF files to text files is needed because of easy processing of text documents, after PDF documents are taken as input, they are converted to text using 'DLL' technology.

Microsoft .NET Framework:

The Microsoft .NET Framework is a software component that can be added to the Microsoft Windows operating system. It provides a large body of pre-coded solutions to common program requirements, and manages the execution of programs written specifically for the framework. The .NET Framework is a key Microsoft offering, and is intended to be used by most new applications created for the Windows platform. The pre-coded solutions that form the framework's class library cover a large range of programming needs in areas including: user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. The functions of the class library are used by programmers who combine them with their own code to produce applications. Programs written for the .NET Framework execute in a software environment that manages the program's runtime requirements. This runtime environment, which is also a part of the .NET Framework, is known as the Common Language Runtime

(CLR). The CLR provides the appearance of an application virtual machine, so that programmers need not consider the capabilities of the specific CPU that will execute the program. The CLR also provides other important services such as security mechanisms, memory management, and exception handling. The class library and the CLR together compose the .NET Framework. The framework is intended to make it easier to develop computer applications and to reduce the vulnerability of applications and computers to security threats. First released in 2002, it is included with Windows Server 2003 and Windows Vista, and can be installed on most older versions of Windows. METEX software is created in Microsoft .NET Framework. All programs of METEX are coded in C# (see 5.4.7 C# section) programming language of .NET platform. Therefore .NET is used because it is platform independent.

C# Object-Oriented Programming Language:

C# is an object-oriented programming language developed by Microsoft as part of their .NET initiative, and later approved as a standard by ECMA and ISO. C# has a procedural, object-oriented syntax based on C++ that includes aspects of several other programming languages (most notably Delphi and Java) with a particular emphasis on simplification (fewer symbolic requirements than C++, fewer declarative requirements than Java) [44]. METEX software is coded in C# programming language. C# is chosen because it provides easier ways for some properties, though it is sampled by Java. METEX is coded in C# because especially C# is the one of most active programming language for developing concept of object-oriented programming.

# CHAPTER 4

## EXPERIMENTAL RESULTS

In this chapter, to better show the gains of the developed system, the durations of the processes in classical(manual) system and the developed system are also evaluated and compared.

### 4.1 Document Structure

For the purpose of testing developed system, a document set in a real world logistic domain was used. Figure 4.1 illustrates the document breakdown.

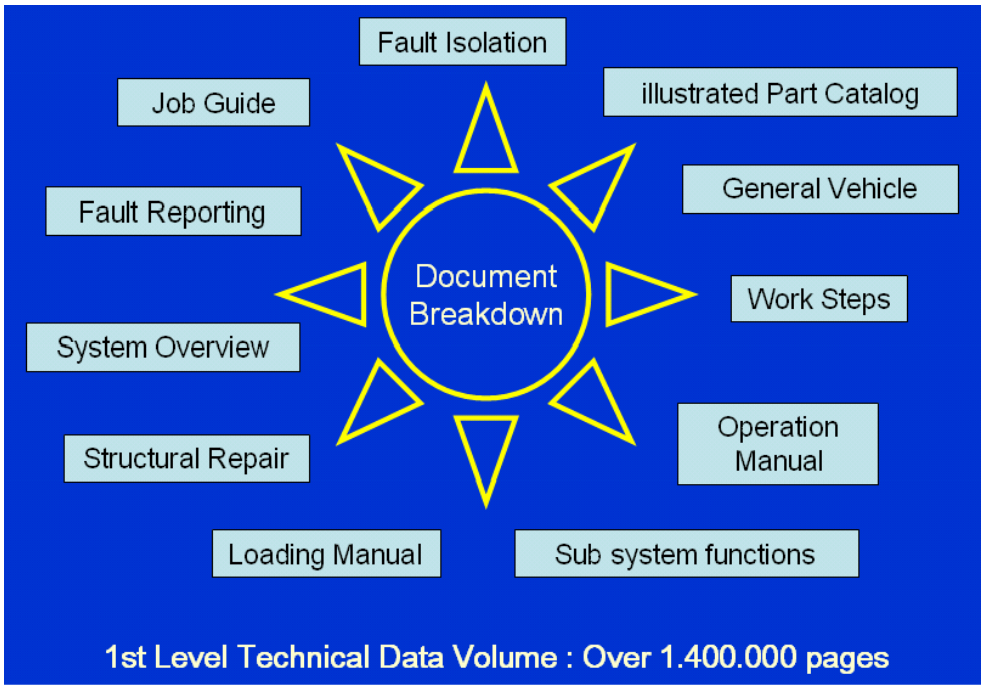



**Figure 4.1 Experimental Document Breakdowns**

<u>Features of Original Source Documents:</u> The source documents (technical publications) in the current logistics problem domain have following characteristics;

- The documents are either in unstructured (hardcopy or PDF) or structured (MS Excel) document type.
- The documents were prepared in foreign language.
- Total volume of source documents is around 1.4 million pages.
- The documents are categorized in eleven different levels.
- The document hierarchy and the document structure is standard for almost all the documents.
- The documents belong to a complex product in the logistics activity area.

Figure 4.2 illustrates the sample Data Set and Data locations in the printed technical Publication.

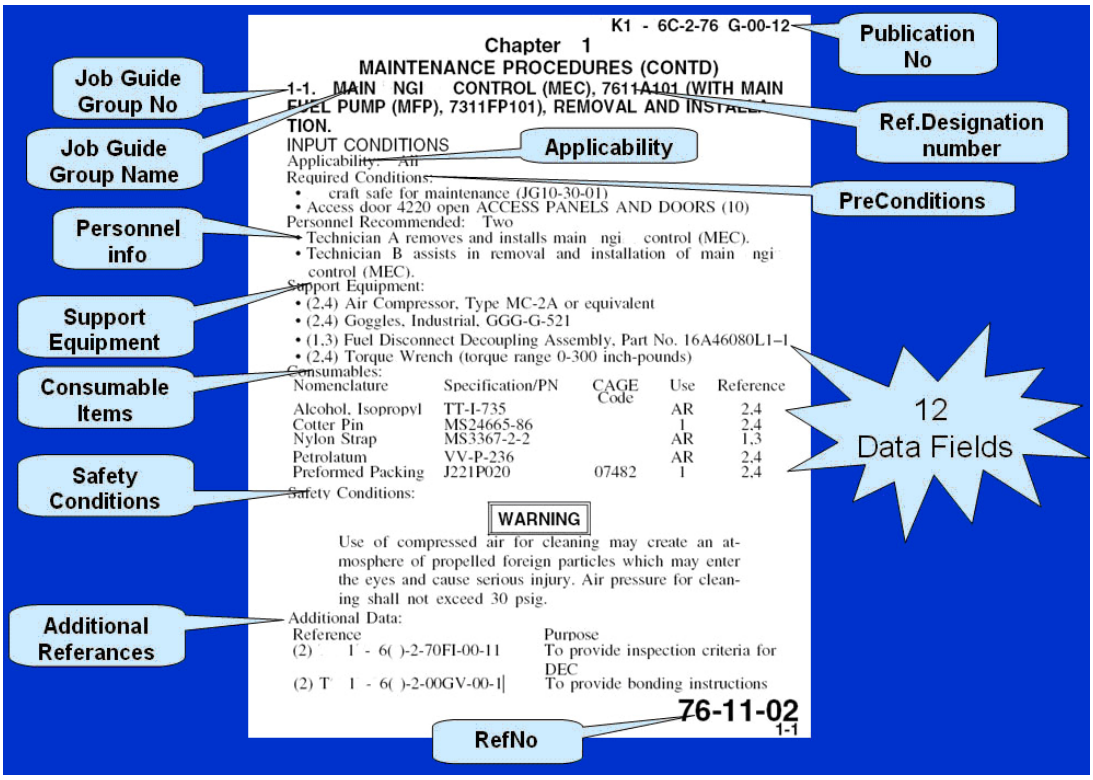



**Figure 4.2 Sample Data Set and Data Locations in the Printed Technical Publication**

Figure 4.3 illustrates the sample subdata set and data locations in the printed technical publication.

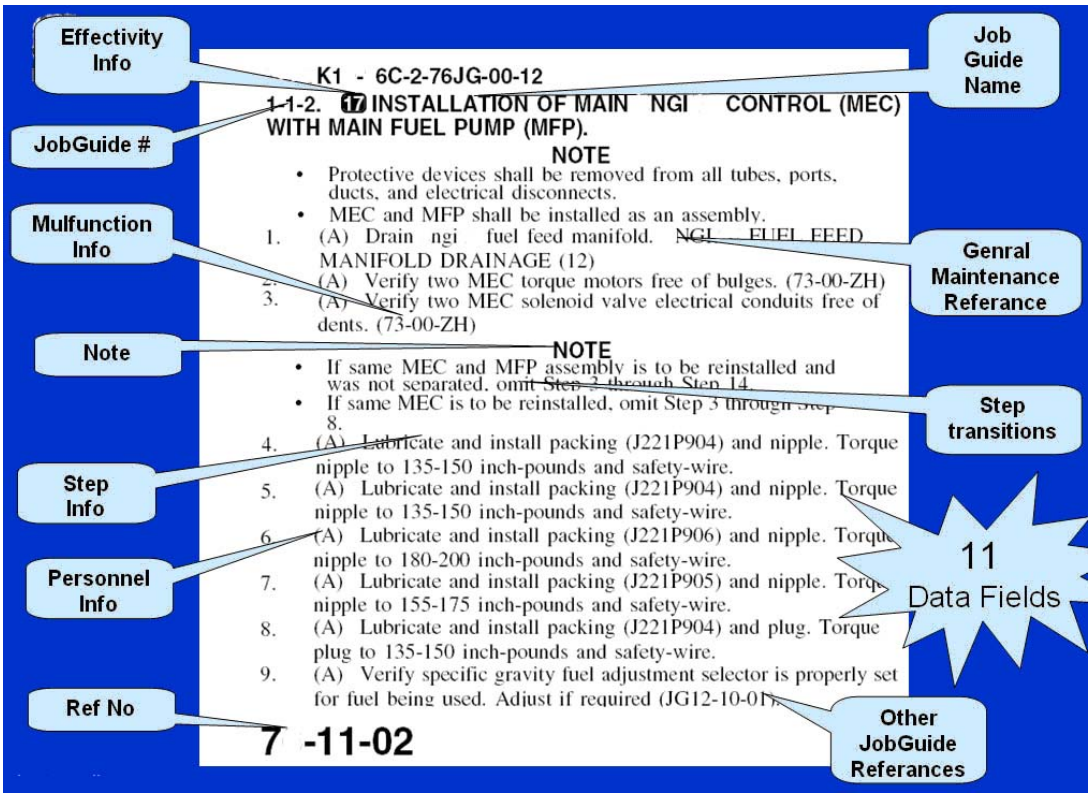



**Figure 4.3 Sample Sub Data Set and Data Locations in the Printed Technical Publication**

## 4.2 Manual Data Input Transactions in The Target ERP System

Manual data input period for each step is almost 4 minutes. Figure 4.4 illustrates the manual data Input in the target ERP system.

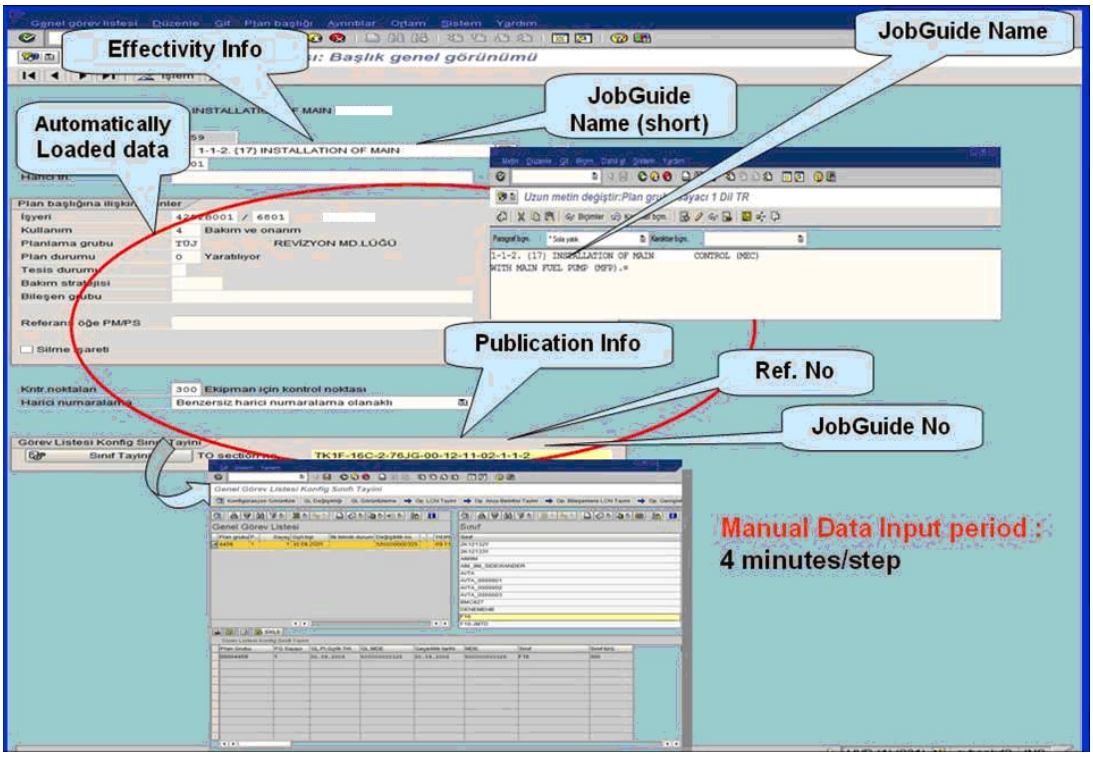



**Figure 4.4 Manual Data Input in the Target ERP System**

Manual data input period for each sub step are 1, 3.5, 4 and 3 minutes respectively. Figure 4.5, Figure 4.6, Figure 4.7 and Figure 4.8 illustrate the manual data Input for the subsystem in the target ERP system.

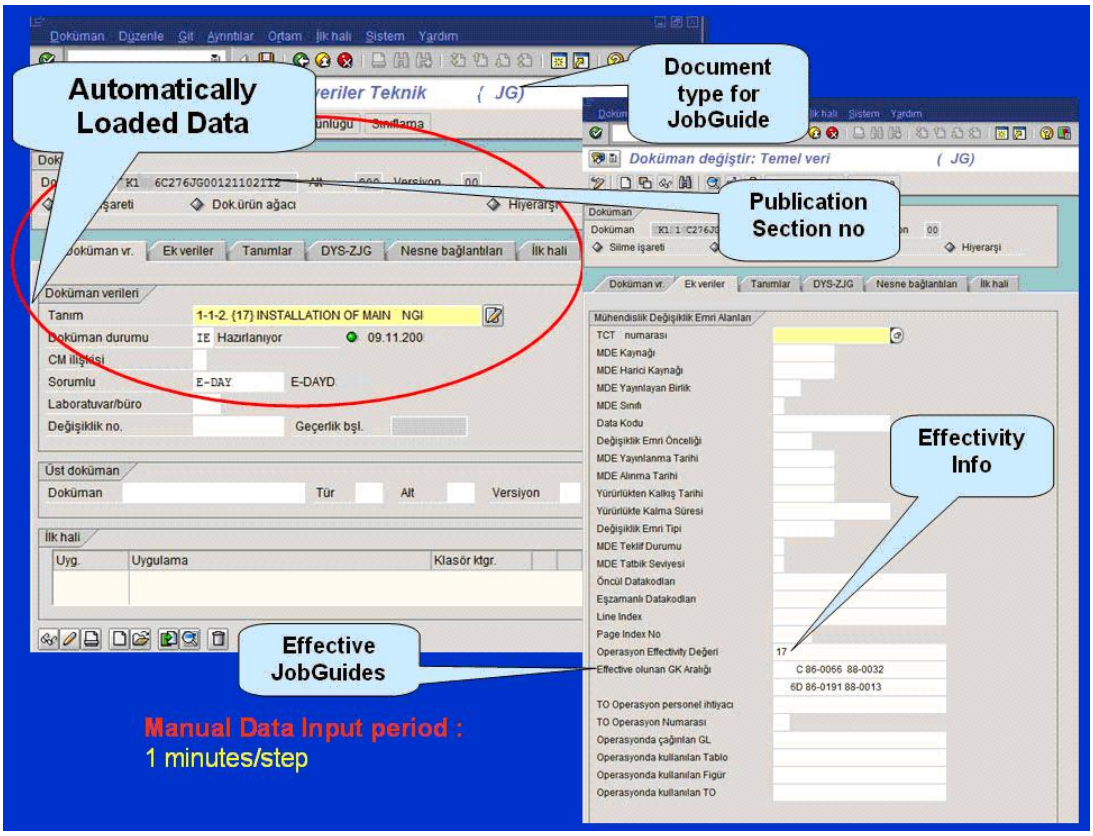



**Figure 4.5 Manual Data Input for the Subsystem (1) in the Target ERP System**

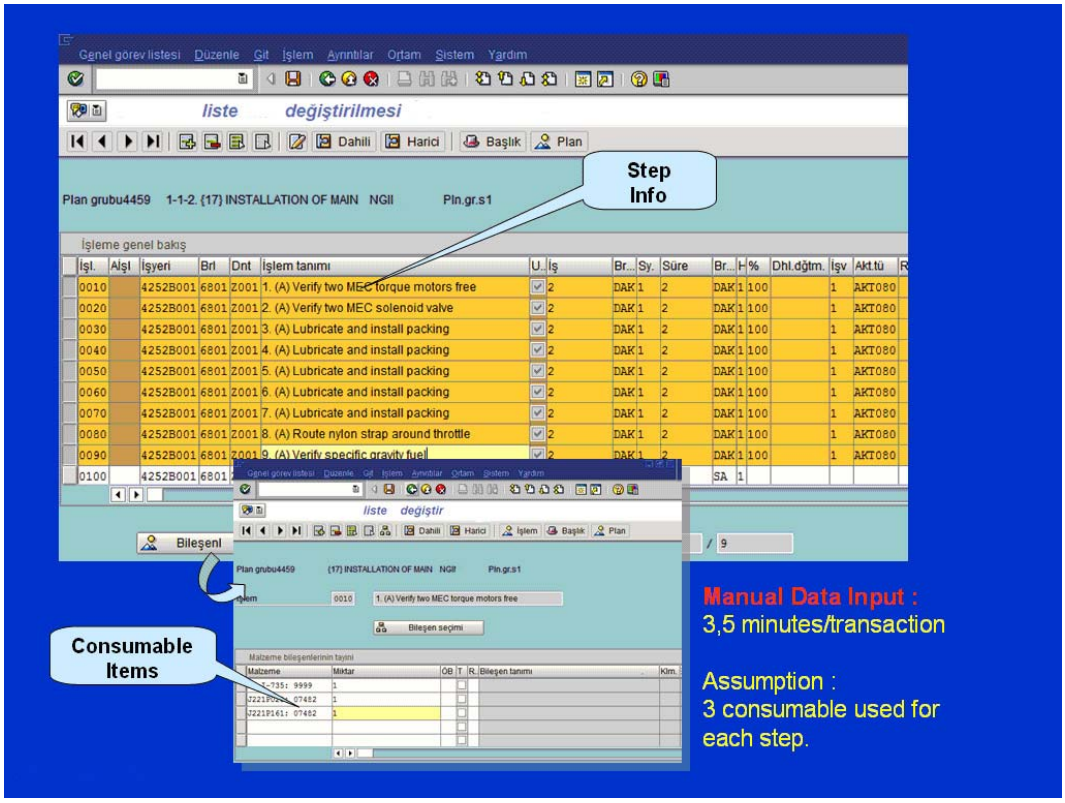



**Figure 4.6 Manual Data Input for the Subsystem (2) in the Target ERP System**

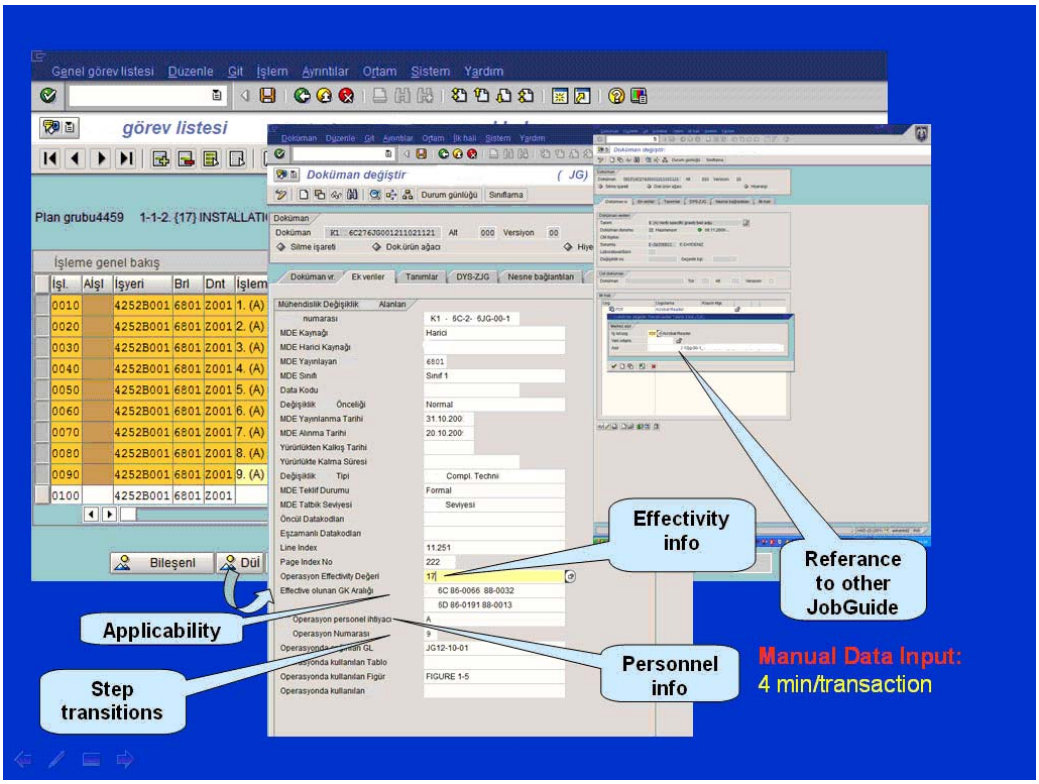



**Figure 4.7 Manual Data Input for the Subsystem(3) in the Target ERP System**

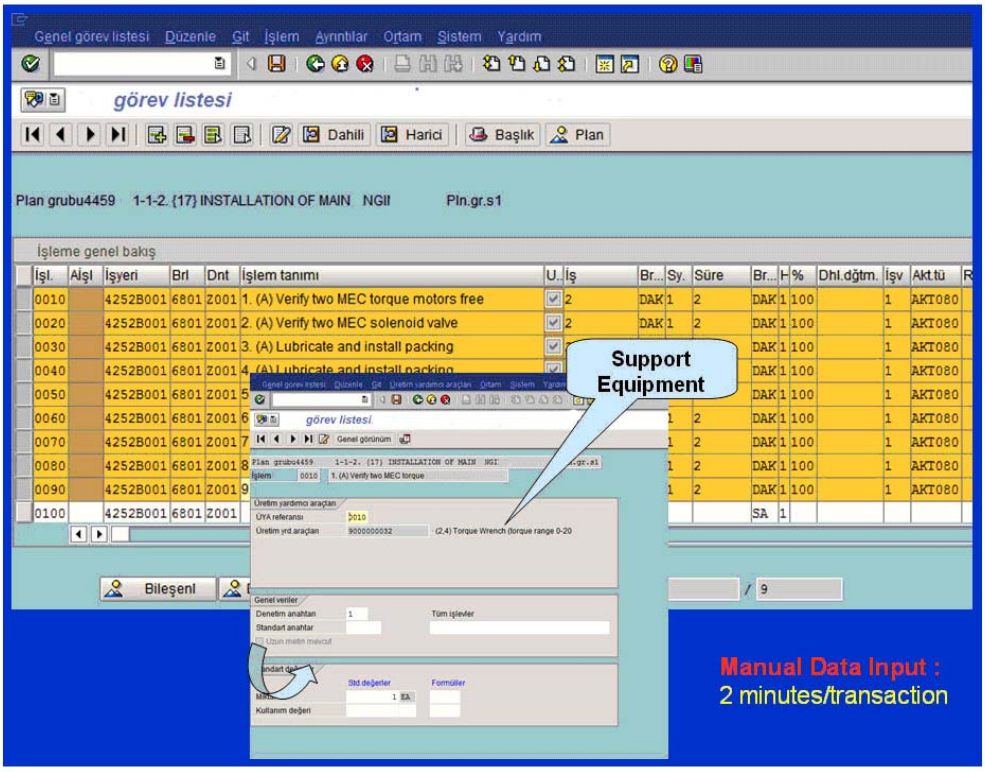



**Figure 4.8 Manual Data Input for the Subsystem(4) in the Target ERP System**

### 4.3 Total Duration Calculation

**Table 4.1 Time Portions for Manual Data Input Steps in ERP System**

| No | Step | Duration (minutes/transaction) |
|---|---|---|
| 1 | Job Guide-header data input | 4 |
| 2 | Creation of electronic-document-address-label related to Job Guide. | 3 |
| 3 | Establishment of "required condition" linkage for the Job guide. | 1,5 |
| 4 | Establishment of "safety info" linkage for the Job guide. | 1,8 |
| 5 | Input of "procedure explanations" and "material info" for each procedures. | 3,5 |
| 6 | Input of "Job Guide Ref", "effectivity info", "applicability", "personel Ref" and "step transitions" data. | 4 |
| 7 | Input of "support Equipment" | 2 |
| 8 | Definition of relation between Reference number and Logistics Control Number. | 1 |
| 9 | Definition of malfunction code. | 1,5 |
| 10 | Assignment of materials for installation/removal procedures. | 2 |
| 11 | Assignment of other job guides resulted previous procedure. | 1,2 |
| 12 | Establishment of linkage between procedure/figure and table. | 0,8 |
| 13 | Definition of relation between procedures. | 0,1 |
| 14 | Definition of relations between operation steps and technical publication page number. | 0,5 |
| 15 | Assignment of related next Job Guides | 1,2 |
| 16 | Matching "Reference number" to "Logistics Control Number" for the operations. | 1,5 |
| 17 | Changes in the operational steps. | 0,1 |
| 18 | Repetition of step 16. | 1,5 |
| TOTAL TIME | | 29,7 minutes |

- Each Job Guide technical publication contains ~250 pages
- Each page contains ~10 steps
- Each equipment system contains ~100 technical publications
- Monthly Working Hours : 21 X 8 =168 hours/month
- Needed total time duration for a single step : 29.7 minutes

- Needed total time for 10 steps : 297 minutes
- Needed total time for each technical publication : 297 X 250 = 74.250 min.
- Needed total time for each equipment system : 100 X 74.250 = 7.425.000 min.
- Needed total time for 3 equipment systems : 7.425.000 X 3 = 22.275.000 min.
- Needed total duration : 21.975.000 /60 = 371.250 hours

Total Months needed : 371.250 / 168 = 2210 months/person or 184 year/person or 184 person should work during one year, to be able to collect above mentioned data and to enter to the target ERP system.

In the calculation, it is not taken into the consideration personnel learning-curves and, it is assumed that some electronic data are entered to the target system using Cut/Paste method. It is considered that following above mentioned manual data entry procedures, total data entry efforts needed will be around 184 year/person with 80 percentage accuracy. Using METEX tool, applying above mentioned operations (for all the documents in the selected logistics domain) it will be possible to perform the same task within 15 days (including document anatomy preparation) with the 94 percentage accuracy.

# CHAPTER 5

## CONCLUSION

One of the main drawbacks of IE technology, however, refers to the difficulty of adapting IE systems to new domains. Classically, this task involves the manual tuning of domain-dependent knowledge such as terminological dictionaries, domain specific lexico-semantics, extraction patterns, and so on [10]. During the case study, "A well-known fact in the community is that simpler ideas work better in practice." [25] principle is followed. The aim of this work is to extract metadata from technical documents. Automating metadata extracting from the massive technical document has great importance because it shortens the data collection time. This work provides a tool that allows the people who involve in logistics domain to collect the master catalog data and procedural troubleshooting information. Since the subject is not quite common, there were no directly (and covering all functionality) related previous work. Accordingly, a new framework is proposed. Some approaches that are used in other domains such as matching and pattern recognition are adopted and to our problem. Resulting system can extract the data of a selected PDF document in less than 12 minutes with the 95 percentage of accuracy. Our experimental results, suggest that the system scales well with the increasing number of search patterns, both in terms of computation time, and storage. In contrast to robust parsing methods, in our parsing schema, longer documents do not affect the computation time dramatically, but only cause linear increment.

Automatic extraction accuracy was usually very high, but in cases of low-confidence extraction, human verifiers were used to improve accuracy. Further evaluation is needed to prove the quality of the system. There were several challenges for proposing the framework. The most significant of them were; to challenge to obtain accurate results while achieving reasonable runtime performance, to handle erroneous input documents and to prepare document anatomy.

Several critical decisions, such as limiting input document type and adding control mechanism are taken for this purpose. In order to handle erroneous input documents,

fine-tuning is made by using cots packages such as abby-fine reader. This study is tested with several documents which contains different information and which has different document breakdown. Experimental results and error analysis show that proposed framework is capable of giving desired results in real world logistic domain. Technical publications containing almost 4.500 pages (almost 1/200 of total document) were processed using this tool and extracted data was successfully loaded into the target ERP system.

### 5.1 Future Work

In this study, a tool which extracts metadata from regular technical publications is presented. The results were quite satisfactory with ideal data. On the other hand, small errors were observed with the irregular breakdown in the document structure. Observing that irregulatory, new regular expressions were added. The errors observed were related to mislocation of the tabular data. Since the levels of errors were smaller than expected, METEX is considered successful.

Even though the levels of errors were acceptable, the result can be improved. Some of the methods mentioned in the pattern matching and pattern recognition books [45], [25] were partly used in the implementation in order to achieve good runtime performance. Nevertheless, those techniques can be used to improve the quality of the results. This study can also be used in preparation of document breakdown structure for the different purposes such as checklist for safety critical troubleshooting.

Even though most of the required features are included in this tool, it still can be improved. Currently the parameter entrance part of this tool, can be considered as a practical tool that performs required operations in a simple way. However, various options, combobox choices, better user friendly graphical user interfaces can improve the functionality of this tool. Integration of SVG functionality (for rendering picture objects) to this tool can be quite useful for those who work in logistics domain. Such integration could be quite helpful in terms of adding pictures and figures into the master data catalogues.

Together with the improvements described above, a very useful data extraction /collection tool can be obtained.

# REFERENCES

[1] Improving Customer Care in the Financial Services Industry through Automated Forms Capture, Captiva Software Corporation. Accessed in May 2007 at http://aiimne.org/library/FinancialServices.pdf

[2] Xeros Sees Explosion of Digital, Paper Documents, Work-Group Computing Report: Issue. Jan, 24 2000.

[3] Weinberger David, Business is a Converstaion, November 2006
Accessed in May 2007 at http://www.kofax.com/g/whitepapers/ice_english.pdf

[4] PaperWork, FormMagic form işleme, http://www.paperwork.com.tr

[5] Burns Mary, Quilliam William, (2007) Risk Of Material Misstatements: Exploring Causes Of Inadvertent Error. ABR & TLC Conference Proceedings, Hawaii, USA., University of South Florida.

[6] Manual Data Entry, Office of Information Technology Research Services, The University of Tennessee, Accessed in May 2007 at http://itc.utk.edu/research/oit_single_entry.php?rid=109.

[7] Shapiro Jason, Bessette Michael, Baumlin Kevin, Ragin Deborah Fish, Richardson Lynne, Automating Research Data Collection.

[8] What are the Disadvantages of Manual Data Entry?, Orbit, Industrial Automation Software, Accessed in March 2007 at [http://www.otc.net/realtime.htm]

[9] Anything Paper to Anything Electronic, Data Entry India, Accessed in May 2007 at http://www.data-entry-india.com.

[10] Jordi Turmo, Alicia Ageno, Neus Catal`A, MS Thesis "Adaptive Information Extraction", Universitat Polit`ecnica de Catalunya, Spain

[11] ECRAN Language Engineering Project,
http://www.dcs.shef.ac.uk/research/ilash/Ecran/overview.html

[12] Mccallum Andrew, Distilling Structured Data from Unstructured Text, University Of Massachusetts, Amherst

[13] Information extraction, From Wikipedia, the free encyclopedia. Accessed in March 2007 at http://en.wikipedia.org/wiki/Information_extraction

[14] Walsh Norman, A Technical Introduction to XML, O'REILLY XML.COM, Accessed in March 2007 at http://xml.com/pub/a/98/10/guide0.html?page=2

[15] Extensible Markup Language (XML), "http://www.w3.org/XML"

[16] Business-to-Business Communication with XML/EDI, "http://www.softwareag.com/xml/about/e-XML_Backgrounder_WP03E0700.pdf"

[17] XML - The Benefits, "http://www.softwareag.com/xml/about/xml_ben.htm"

[18] Why XML?, "http:// www.softwareag.com/xml/about/xml_why.htm"

[19] Taylor Chris, An Introduction to Metada, University of Queensland Library, Australia / 29 July 2003.

[20] The Dublin Core Metadata Initiative, http://dublincore.org/

[21] Anglo-American Cataloging Rules, http://www.ala.org/ala/aasl/aaslproftools/resourceguides/catalogingclassification.cfm

[22] Government Information Locator Service, http://www.usgs.gov/gils/

[23] Encoded Archives Description, http://www.loc.gov/ead/

[24] The National Archives of Australia, AGLS Metadata Standard http://www.naa.gov.au/recordkeeping/gov_online/agls/summary.html

[25] Navarro Gonzalo & Raffinot Mathieu, Flexible Pattern Matching in String, May 2002, Cambridge University Press.

[26] Tethys Solutions, Automation Anywhere, http://www.tethyssolutions.com/automation-software.htm

[27] Ficstar Software, Web Extractor/Crawler, http://www.ficstar.com/web_grabber_net.html

[28] Iconico, DataExtractor, http://www.iconico.com/DataExtractor

[29] Data Manager Software, http://www.ggmate.com/DataManagerSoftware

[30] PCSOURCECODE, WebDataScraper, http://www.webdatascraper.com

[31] KnowleSys Software Inc., http://www.knowlesys.com/examples.htm

[32] DataMite ArianeSoft, Release Notes, http://arianesoft.ca/page.php?12

[33] Business Computer Design Int'l Inc., Report Mining & Data Extraction Tool, http://www.bcdsoftware.com/iseries400solutions/ez-pickins/index.htm

[34] ETI-EXTRACT ETML Solution http://www.taborcommunications.com/dsstar/00/0718/101927.html

[35] Data-Extraction Tools, http://www.deg.byu.edu/tools

[36] [36A] Orli Richard, Data Extraction, Transformation, and Migration Tools, http://www.kismeta.com/ex2.html#Summary

[37] Kimball Ralph, Automating Data Extraction, http://www.dbmsmag.com/9607d05.html

[38] Ted Padova, Working Effectively With Acrobat PDF Files.

[39] Walkenbach John, Excel 2003 Bible Release, September 01, 2003

[40] Jerke Noel, Microsoft Office Access 2003 Professional Results.

[41] Professional XML, WROX Programmer To Programmer, WROX Press Ltd, 1 May 2001

[42] Johnson Hart, Windows System Programming Third Edition, Addison-Wesley, 2005.

[43] Rector, Brent et al. Win32 Programming, Addison-Wesley Developers Press, 1997.

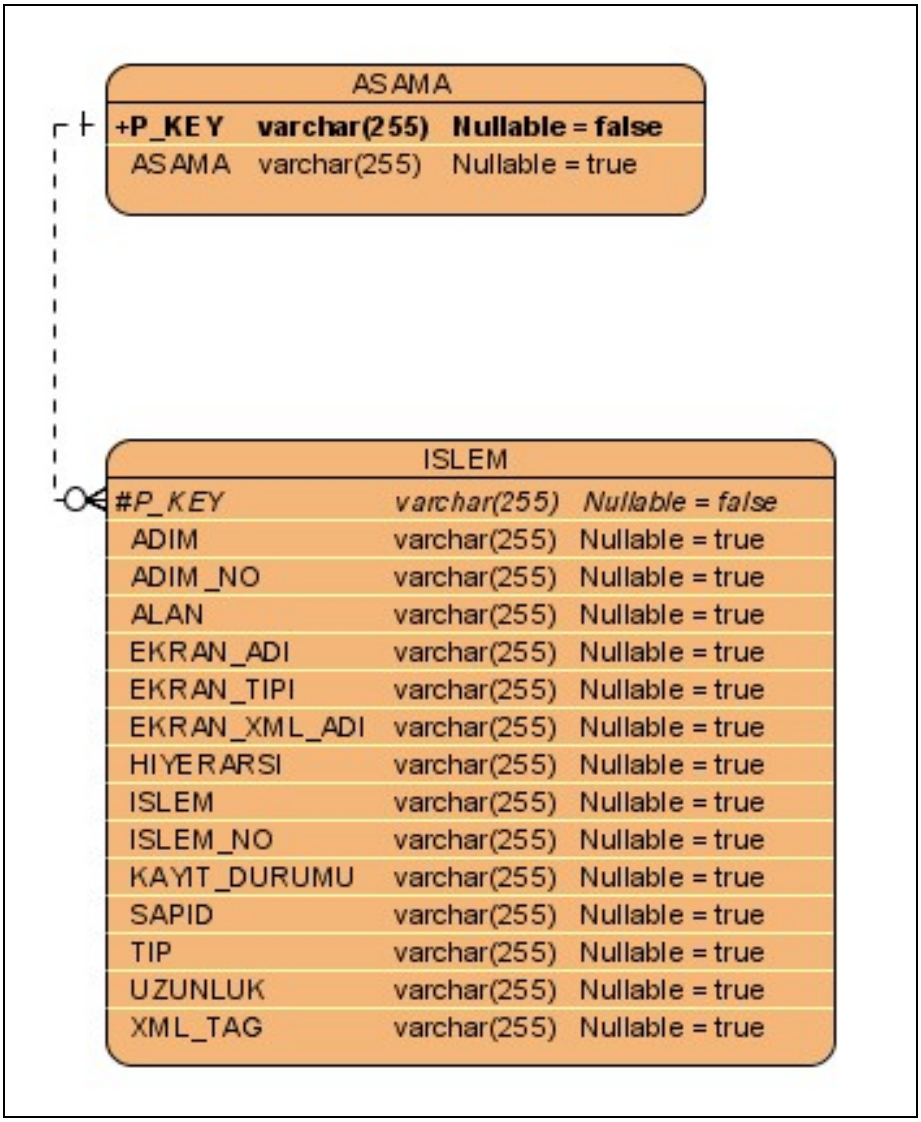
[44] Anderson Tim, C# pulling ahead of Java, November 14, 2006.

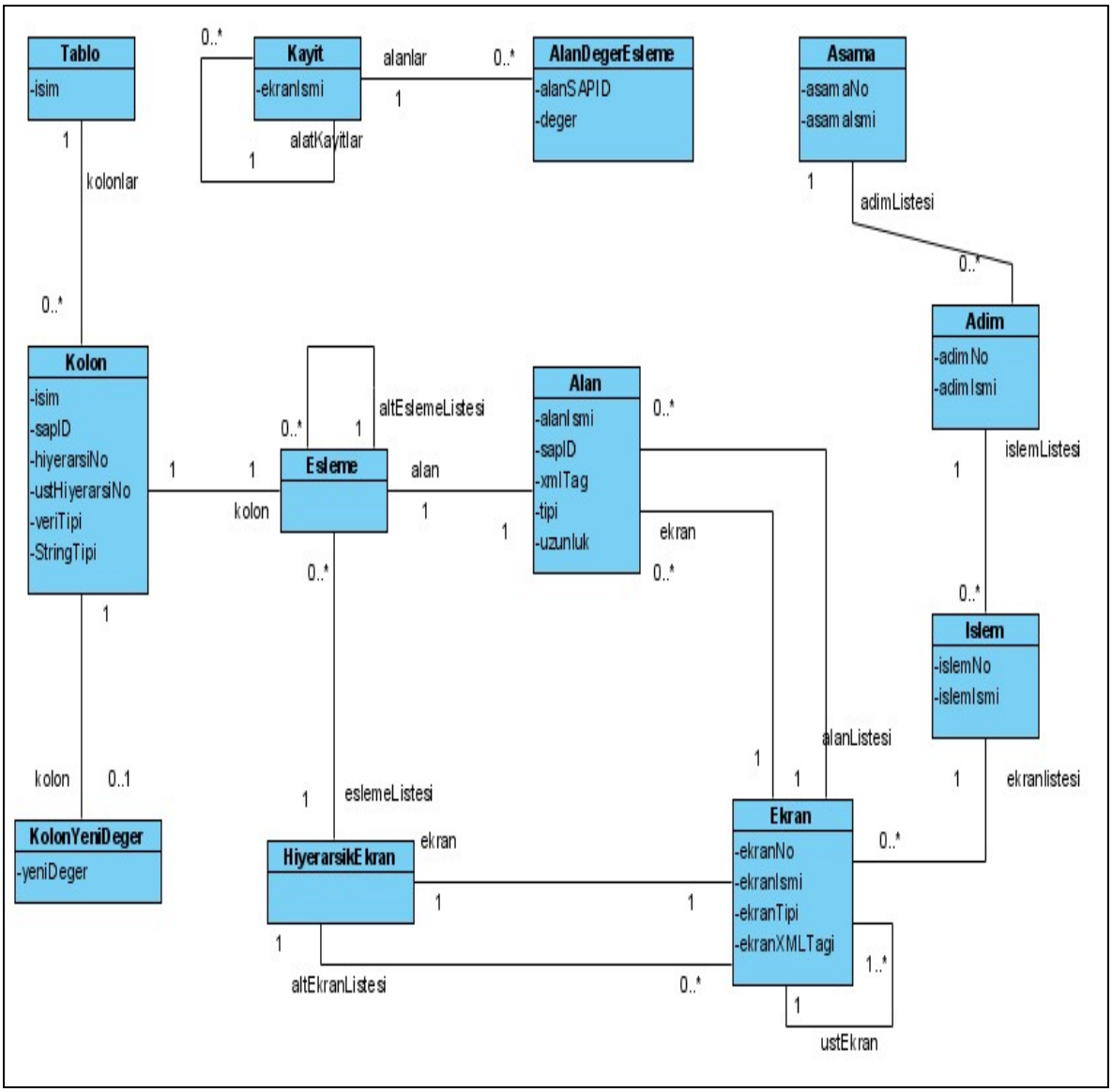
[45] Pavel Monique, Fundamentals of Pattern Recognition.

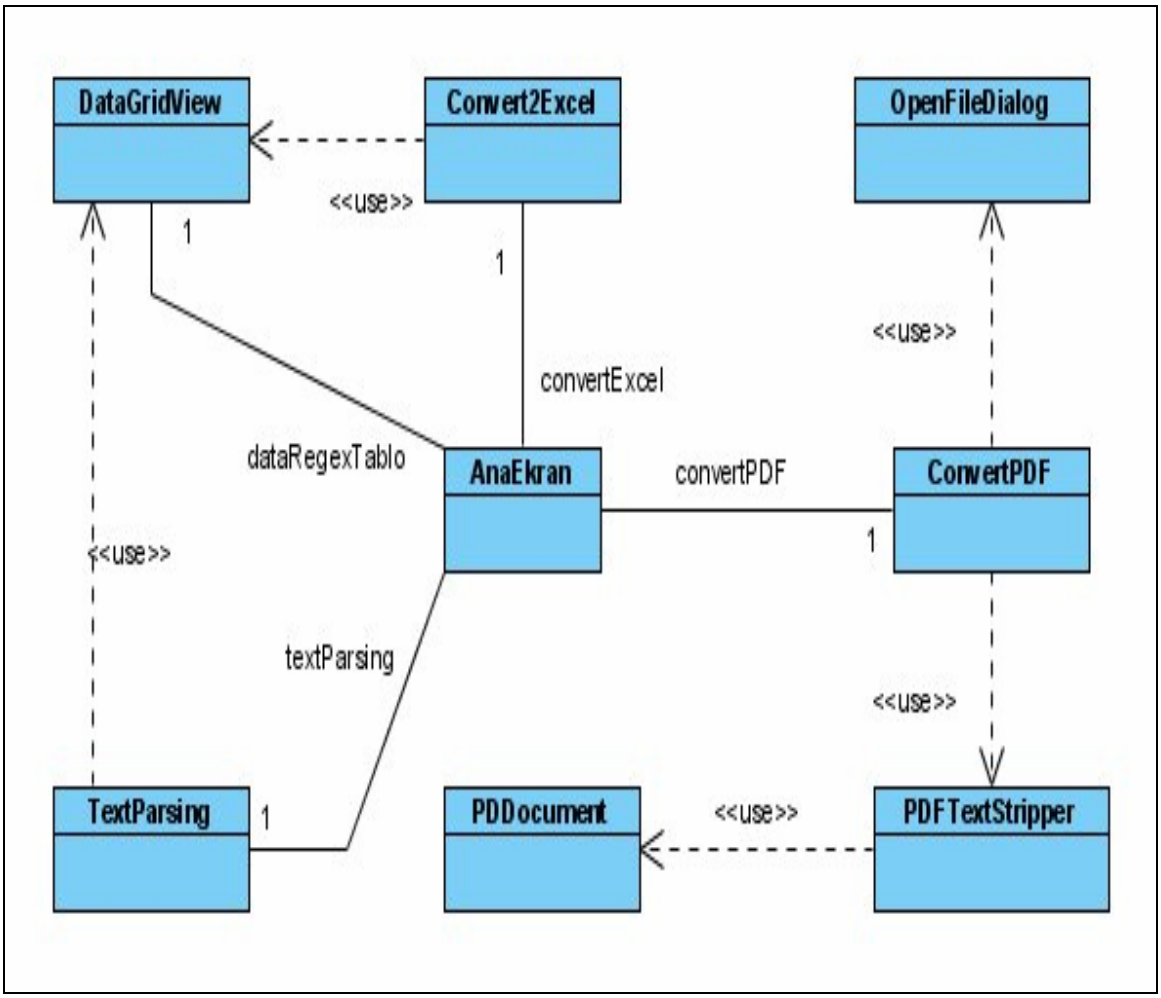
[46] Sippu S., Soisalan E., Soininen I., Parsing Theory Vol.1 Languages & Parsing.

# APPENDIX A

## Entity Relationship Diagrams

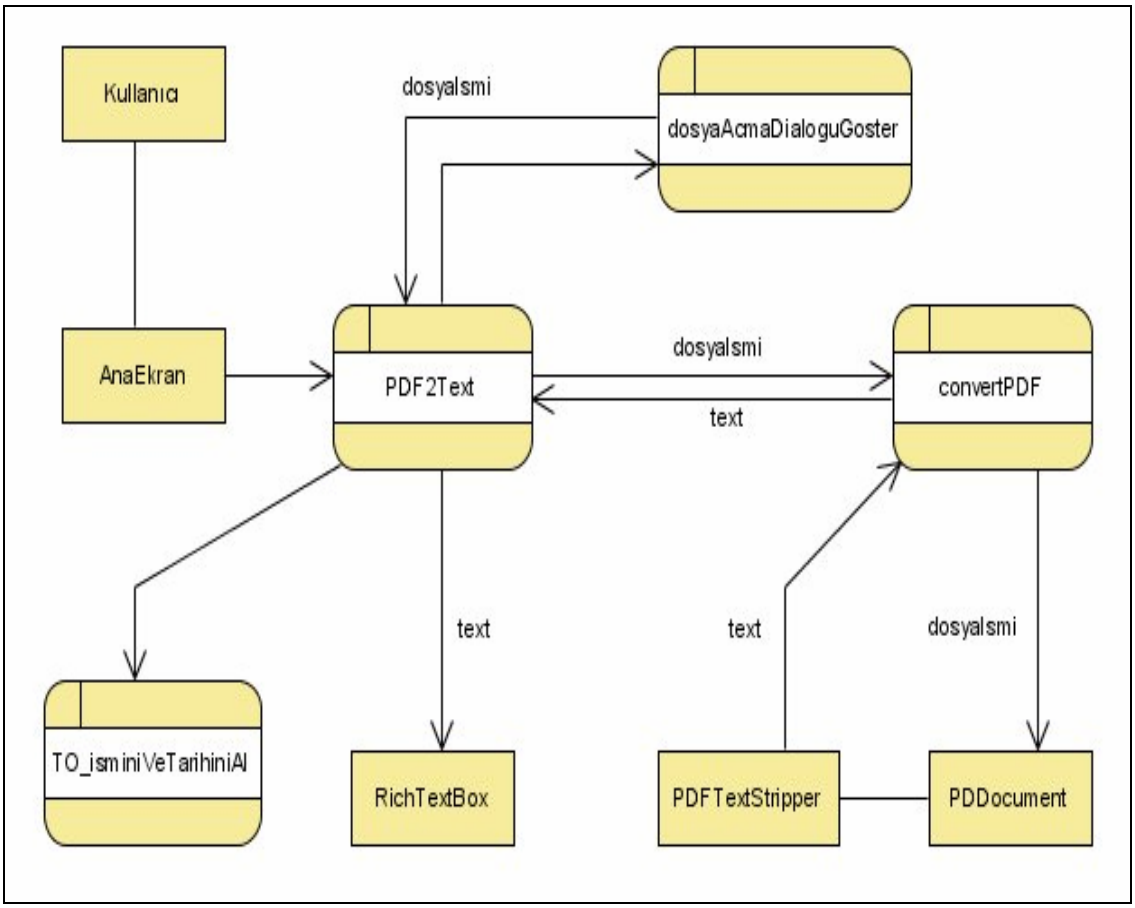



Tablo
-isim
Kayit
-ekranIsmi
alanlar
AlanDegerEsleme
-alanSAPID
-deger
Asama
-asamaNo
-asamaIsmi
alatKayitlar
kolonlar
adimListesi
Adim
-adimNo
-adimIsmi
Kolon
-isim
-sapID
-hiyerarsiNo
-ustHiyerarsiNo
-veriTipi
-StringTipi
altEslemeListesi
Esleme
alan
Alan
-alanIsmi
-sapID
-xmlTag
-tipi
-uzunluk
islemListesi
kolon
ekran
Islem
-islemNo
-islemIsmi
alanListesi
ekranlistesi
KolonYeniDeger
-yeniDeger
eslemeListesi
HiyerarsikEkran
Ekran
-ekranNo
-ekranIsmi
-ekranTipi
-ekranXMLTagi
altEkranListesi
ustEkran

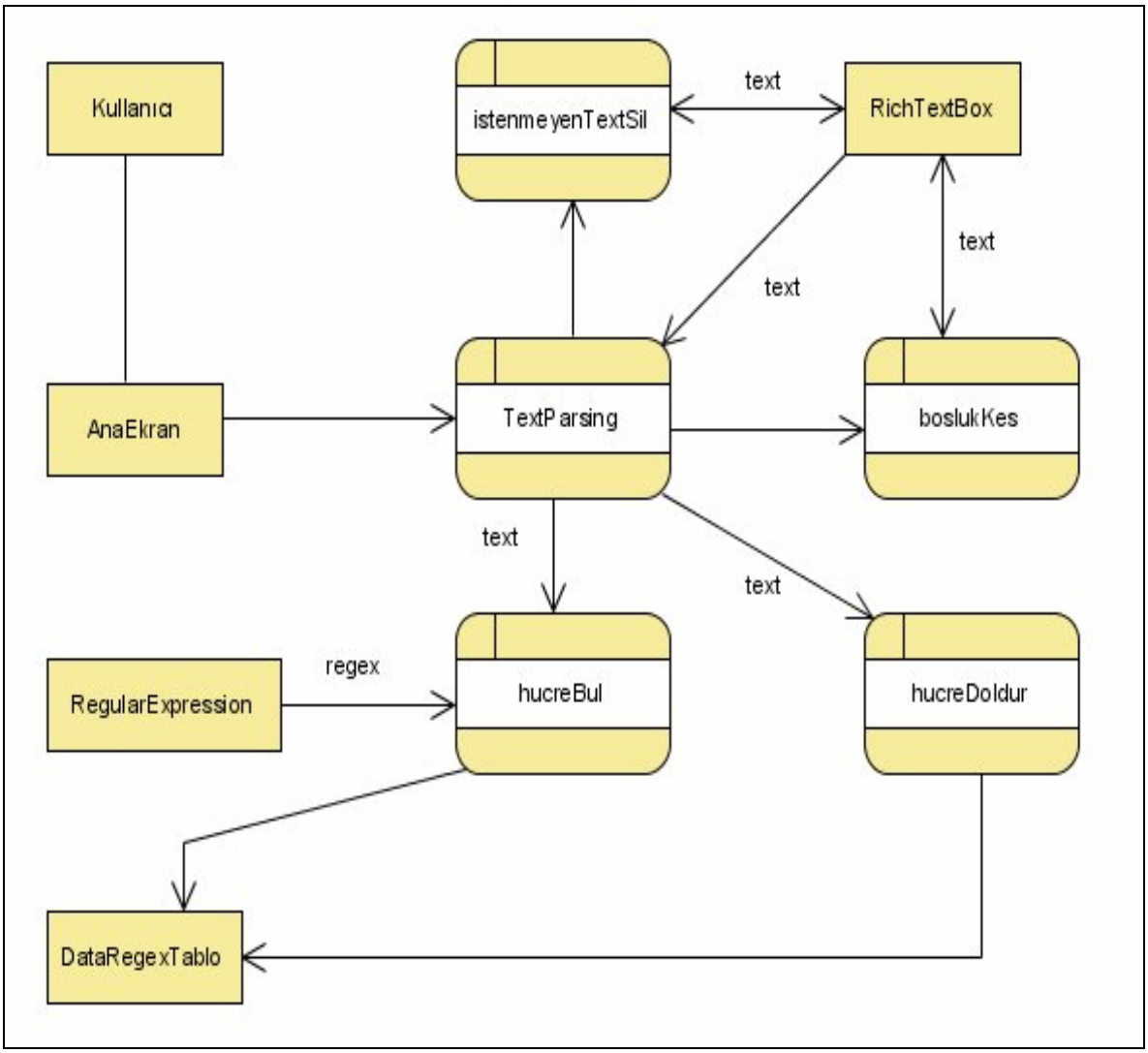

DataGridView
Convert2Excel
OpenFileDialog
<<use>>
1
1
<<use>>
convertExcel
dataRegexTablo
AnaEkran
convertPDF
ConvertPDF
1
<<use>>
textParsing
<<use>>
TextParsing
1
PDDocument
<<use>>
PDFTextStripper

# APPENDIX B

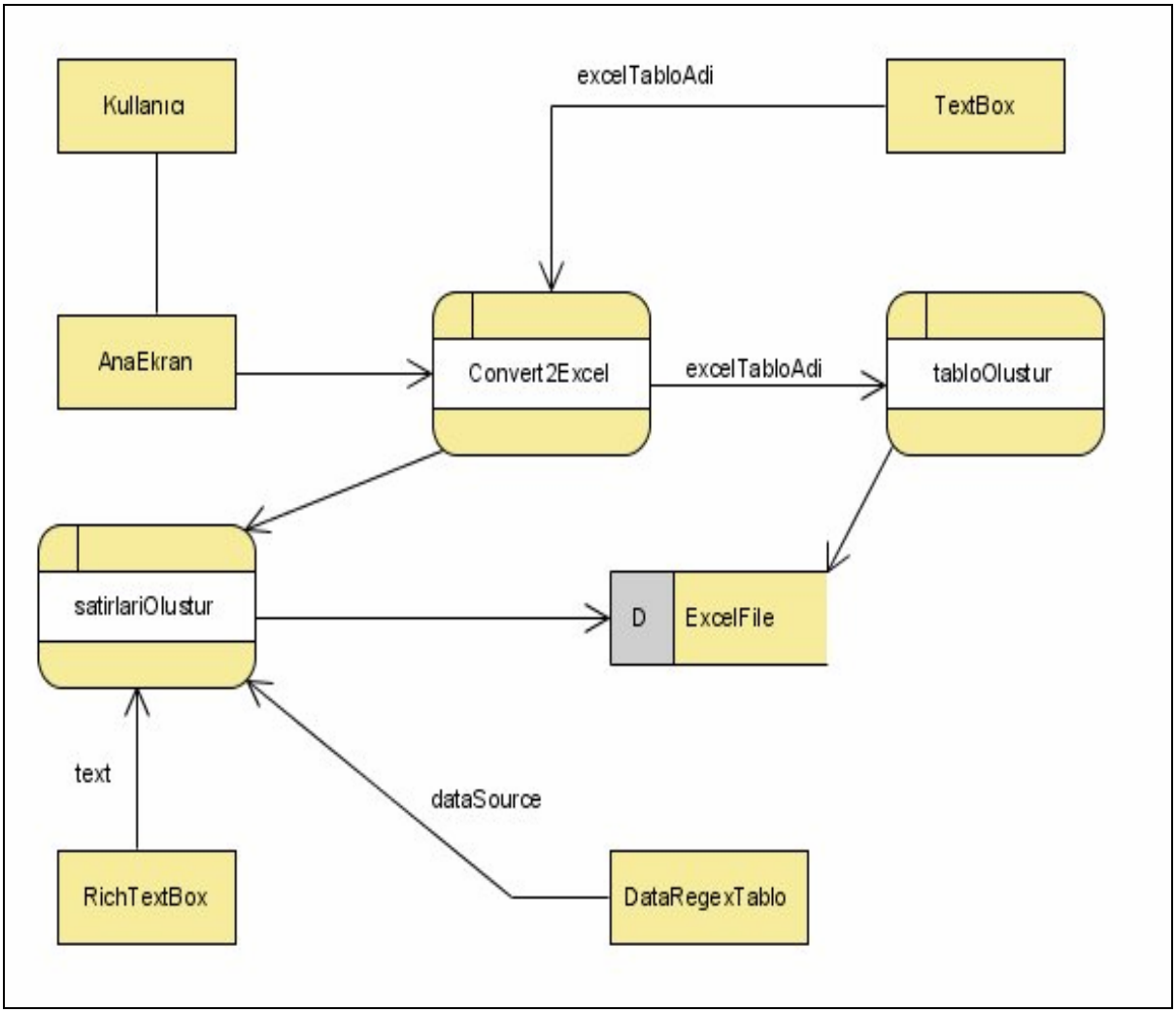
## DataFlow Diagram for "PDF to TEXT" (2nd Level)



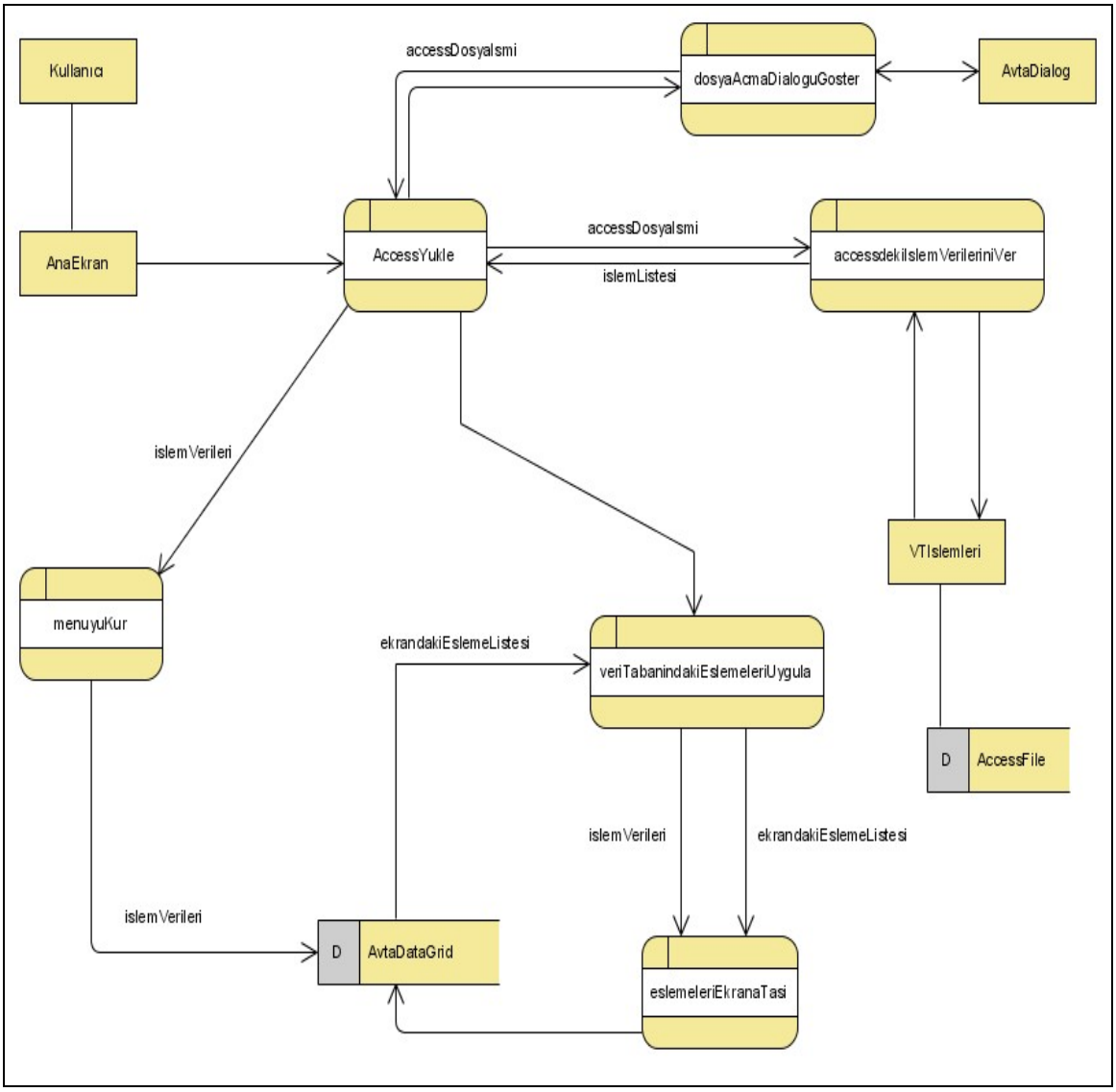
**DataFlow Diagram for "Text Parsing" (2nd Level)**



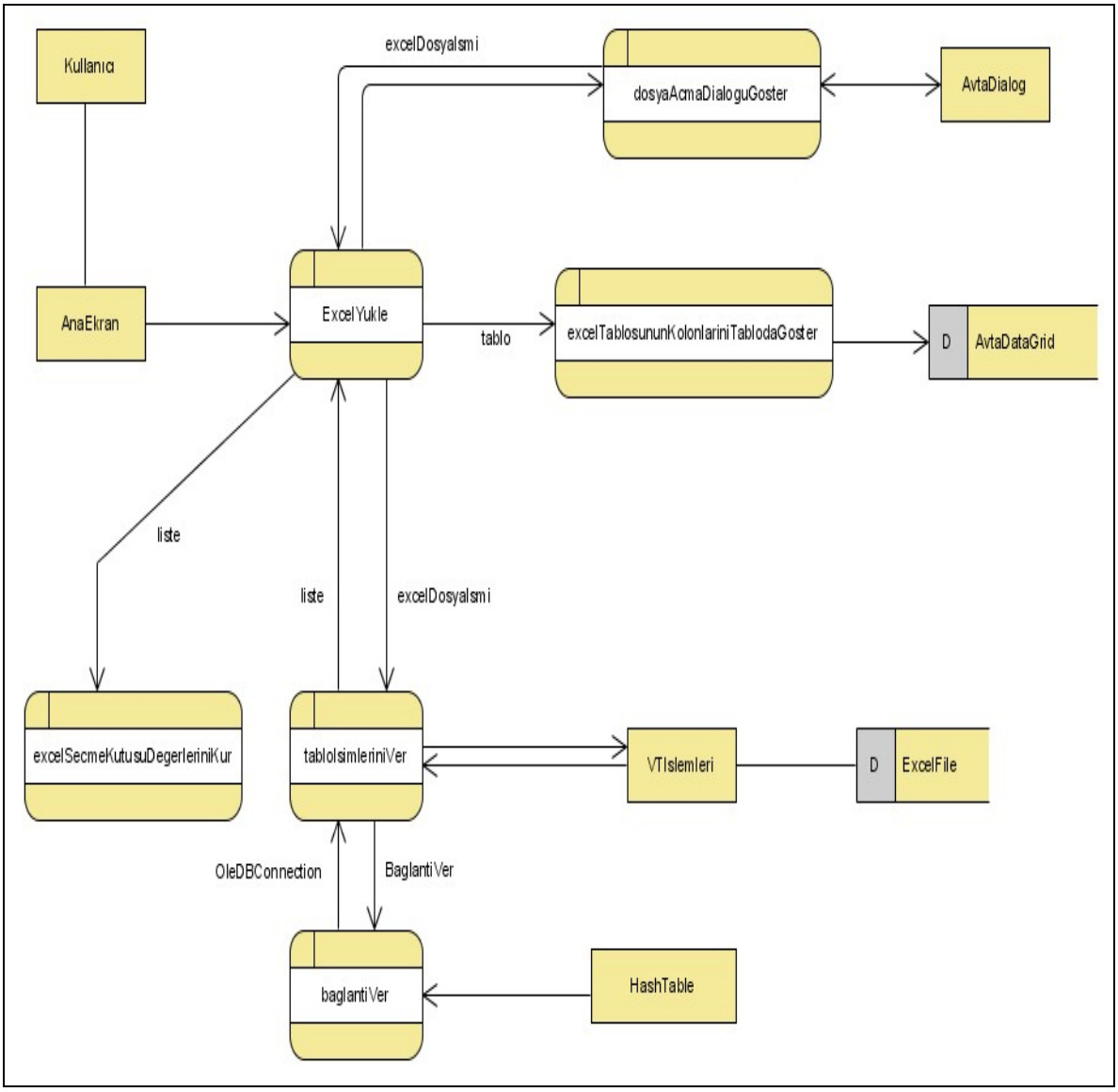
**DataFlow Diagram for "Converting to MS Excel" (2nd Level)**



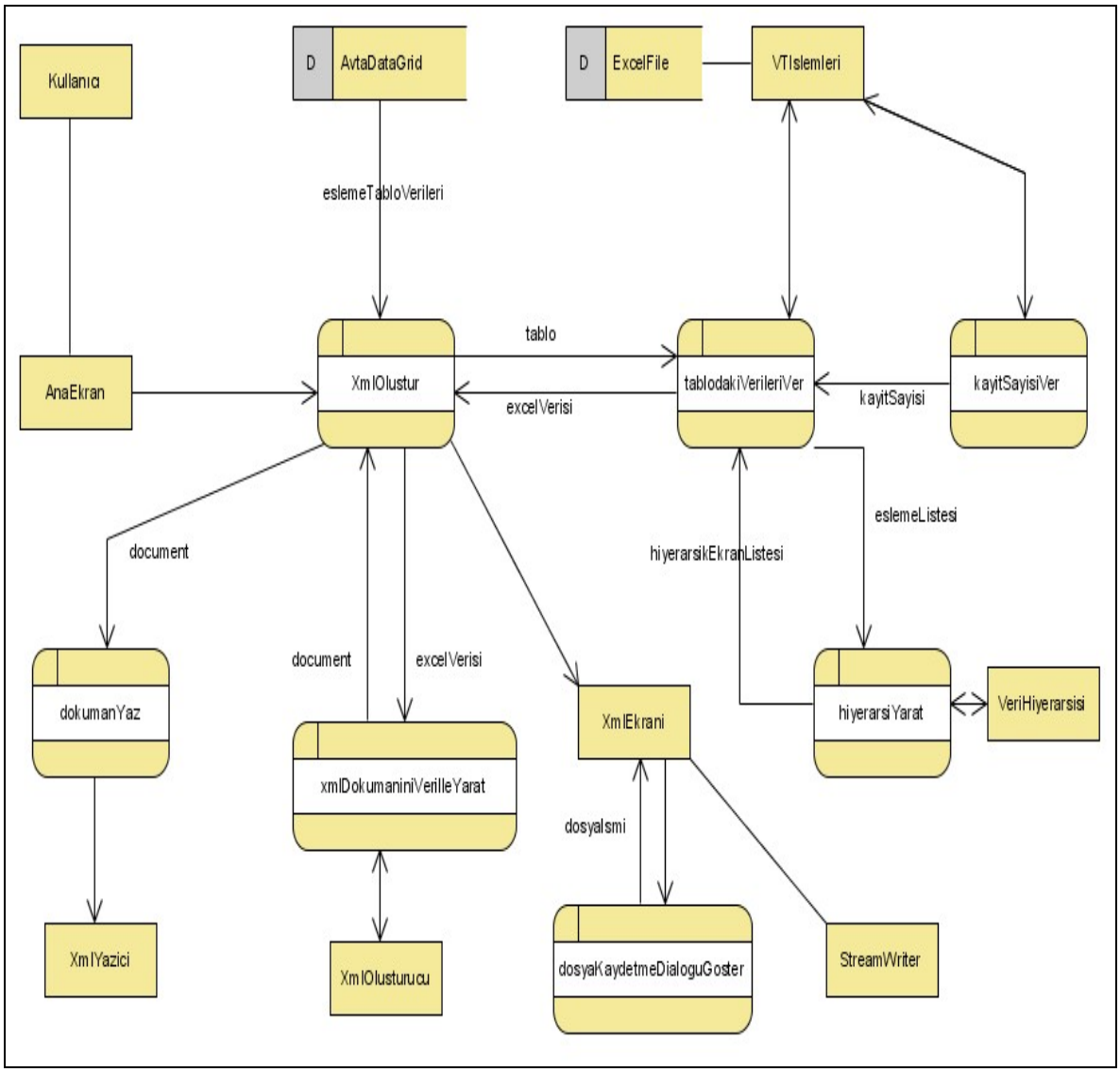
**DataFlow Diagram for "Loading to MS Access" (2nd Level)**



**DataFlow Diagram for "Loading to MS Excel files" (2nd Level)**

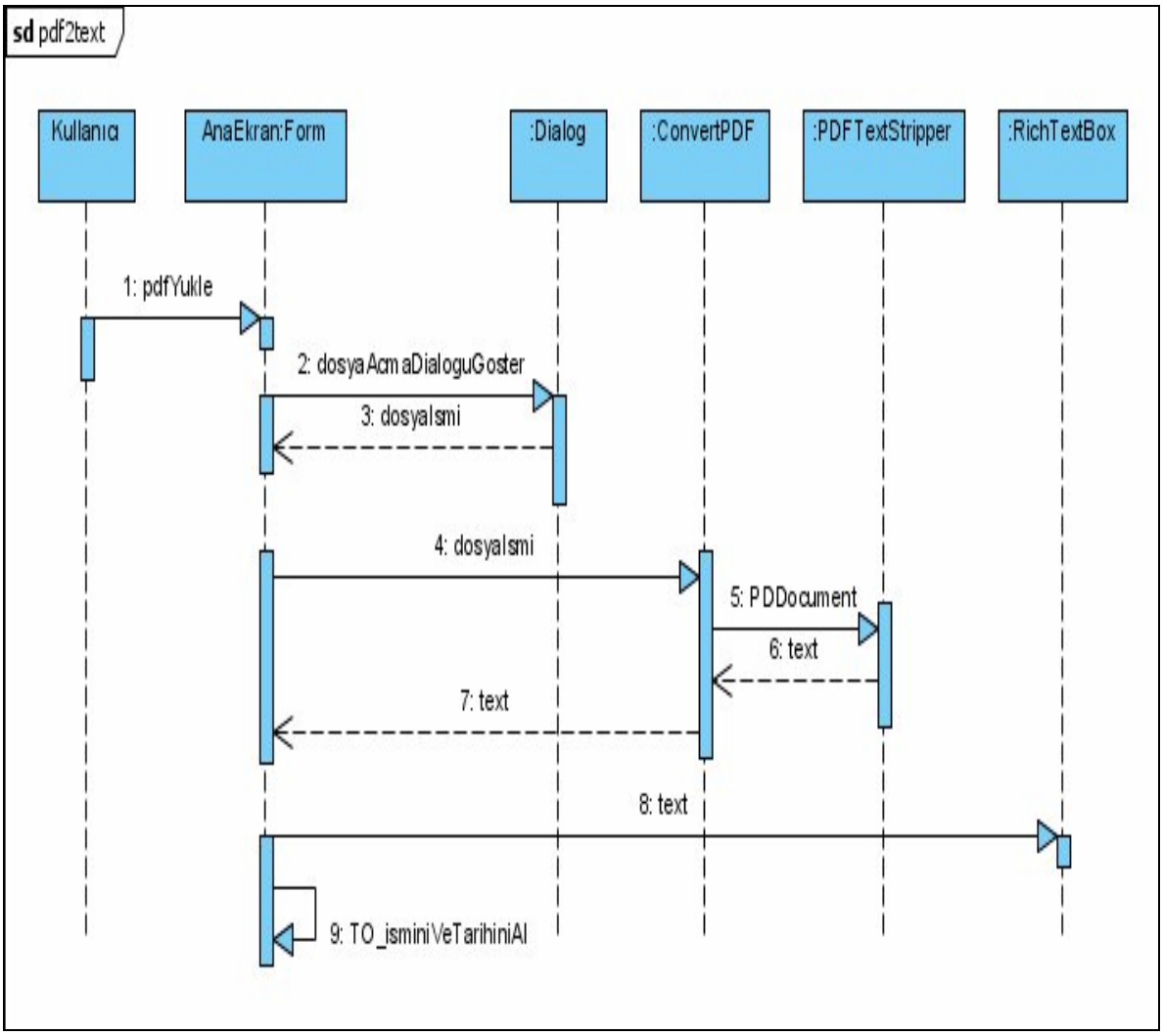


**Dataflow Diagrams for "Forming XML file" (2nd Level)**

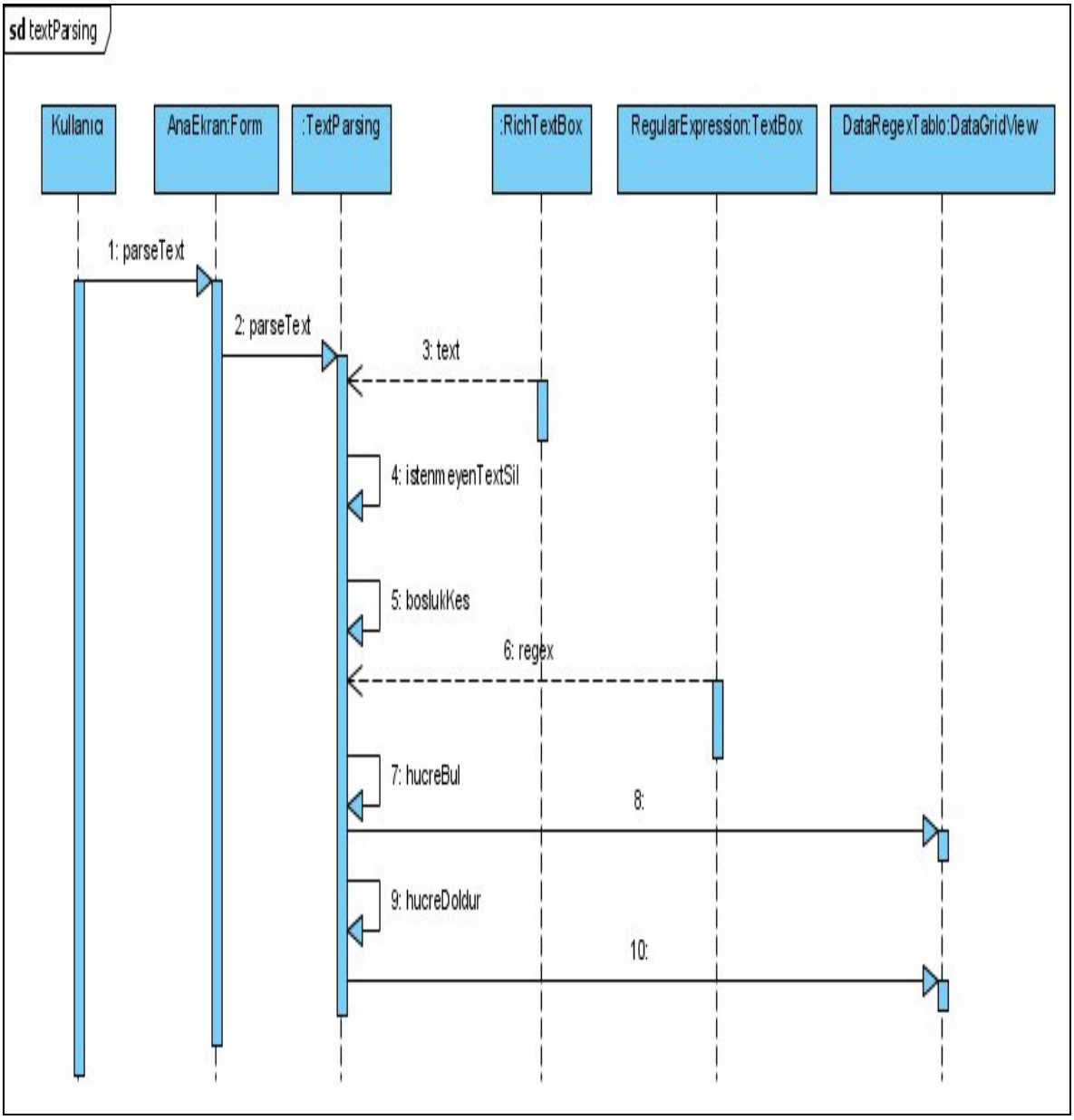


# APPENDIX C

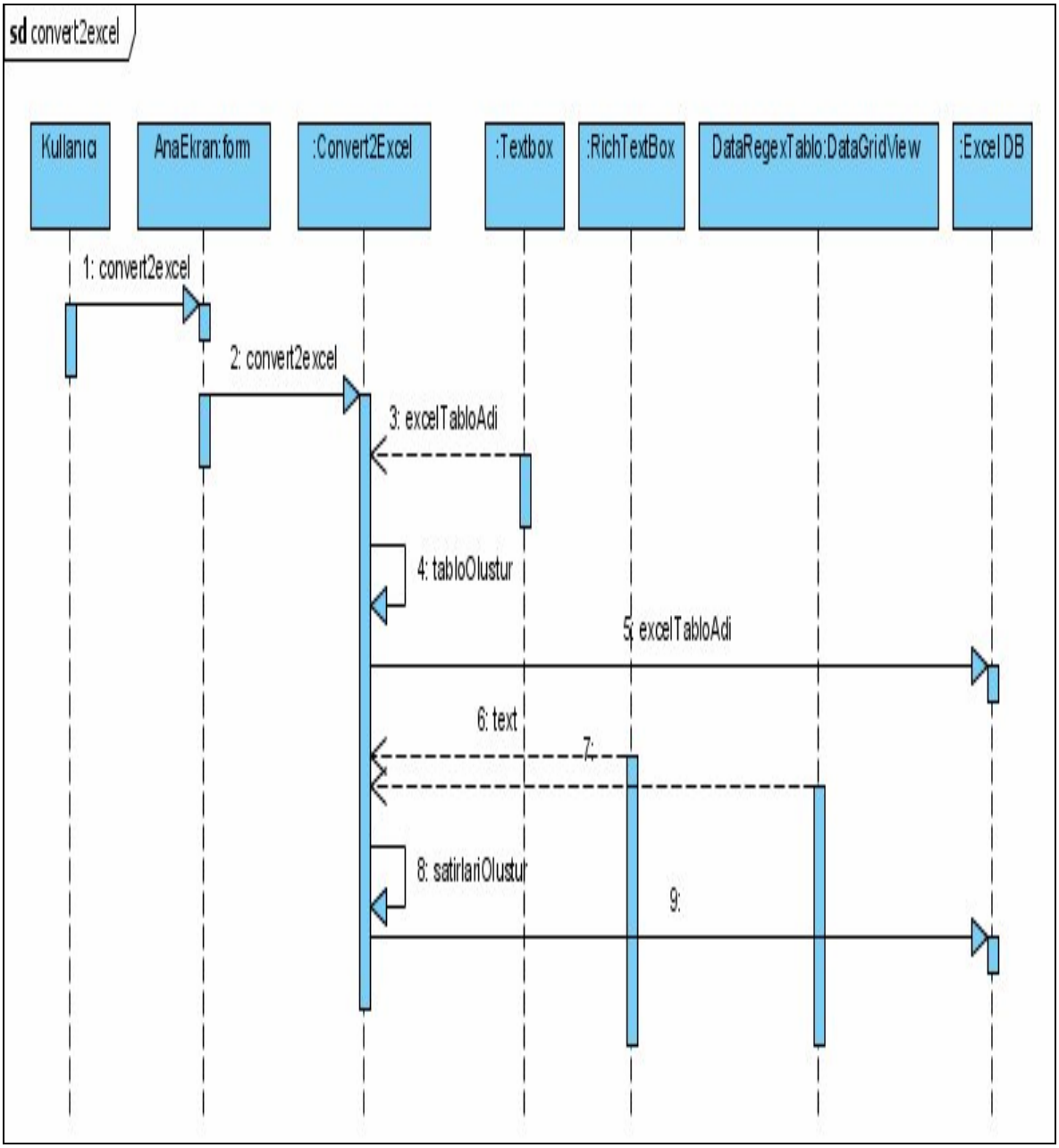
## Sequence Diagram for "PDF to TEXT"



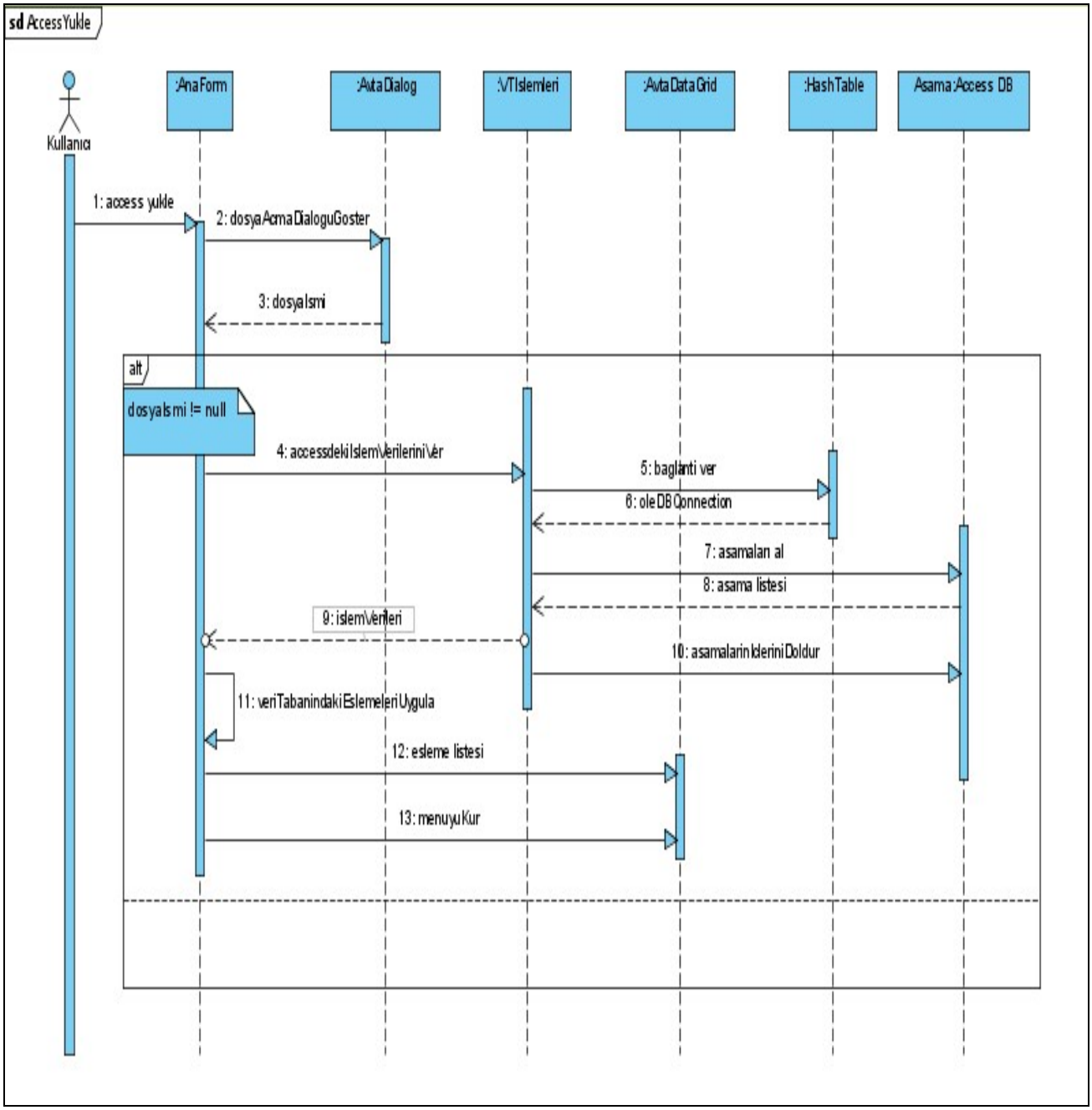
**Sequence Diagram for "Text Parsing"**



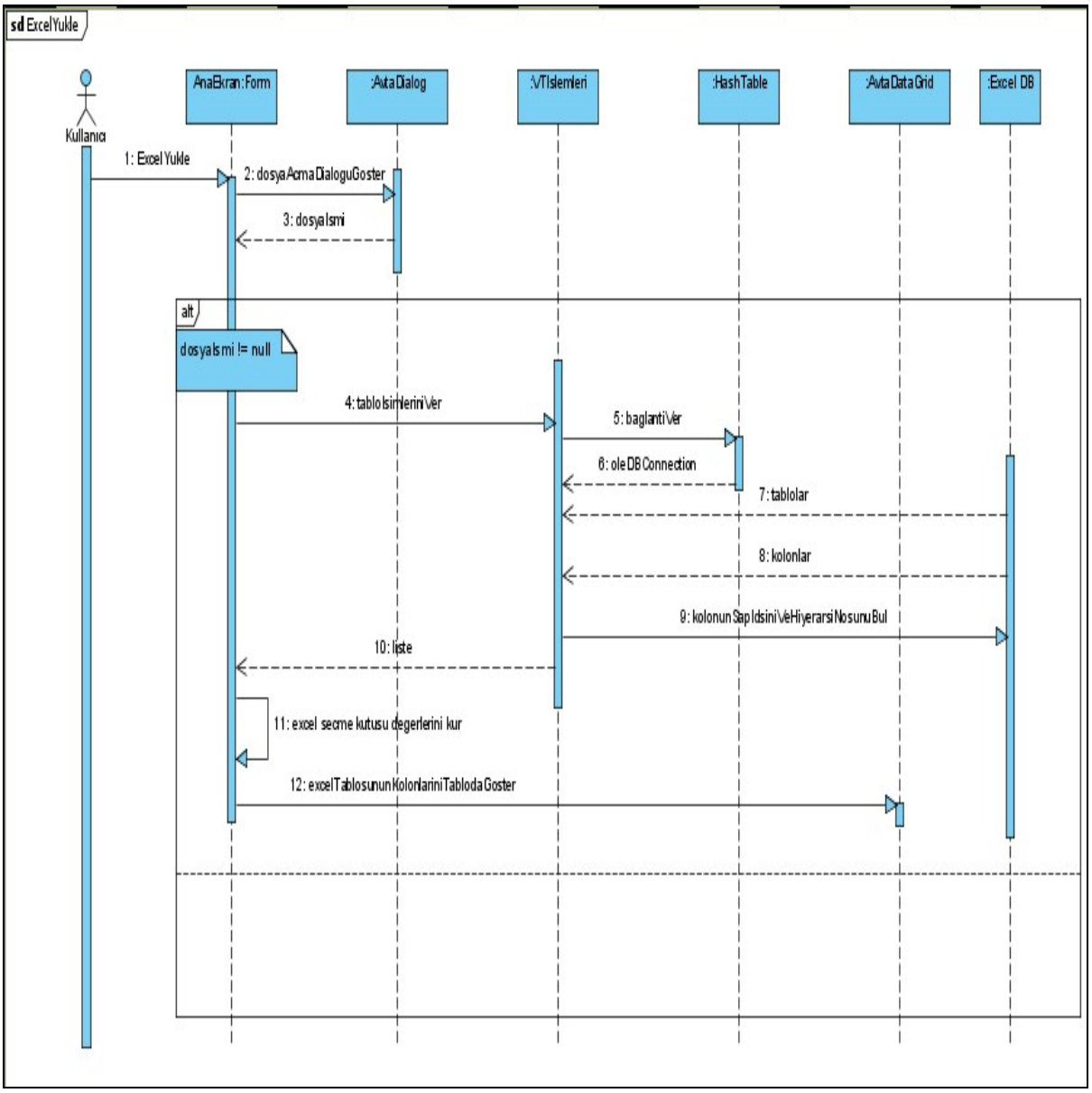
**Sequence Diagram for "Converting to MS Excel"**



**Sequence Diagram for "Loading to MS Access"**

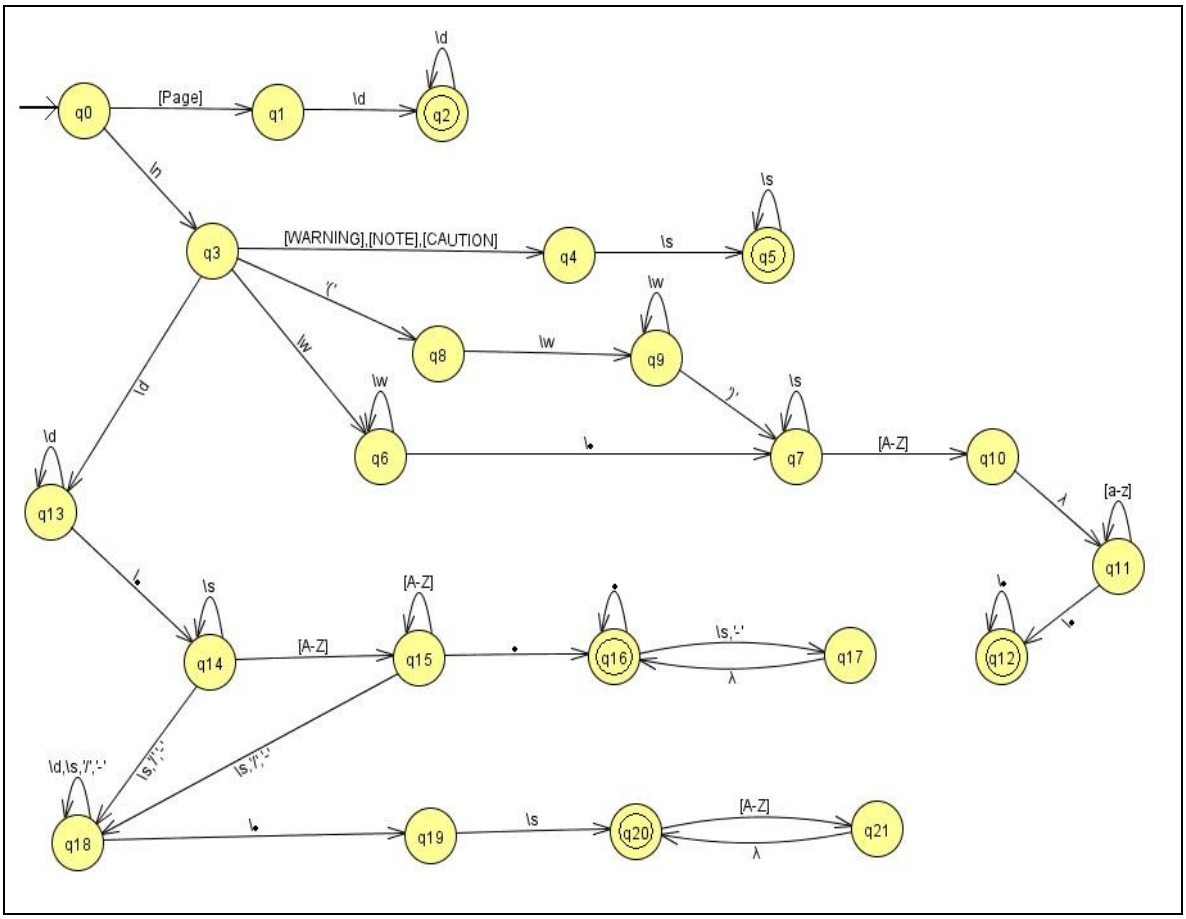


## Sequence Diagram for “Loading to MS Excel”



# APPENDIX D

## Regular Expression




In the Finite State Automata diagram, the used symbols and their meanings are as follows;

‘\d’ indicates a digit [0-9]

‘\s’ indicates a white space char [\t\n\r\f]

‘\w’ indicates a word char [a-zA-Z0-9_]

‘\n’ indicates new line char

‘.’ indicates any character

# APPENDIX E

**Pseudo Code of Finite Sate Automata for the Selected Regular Expression**

```
q0:     if str=”Page” then begin
                store(str); getnext(ch); go to q1
        end;
        if ch='\n' then begin
                getnext(ch); go to q3
        end;
q1:     if ch in digit then
                store(ch); getnext(ch); go to q2
        end else begin
                empty(buffer); go to q0
        end;
q2:     write(buffer);
        while ch in digit then begin
                write(ch); getnext(ch);
        end;
        go to q0
q3:     if str="WARNING" or str="CAUTION" or str="NOTE" then begin
                store(str);getnext(ch); go to q4
        end else begin
        if ch in digit then begin
                store(ch);getnext(ch);go to q13
        end else begin
        if ch='(' then begin
```

```
            store(ch);getnext(ch);go to q8
        end else begin
        if ch is a word char then begin
            store(ch);getnext(ch);go to q6
        end else begin
            getnext(ch);go to q0
        end;
q4:     if ch in white space char then begin
            store(ch);getnext(ch);go to q5
        end else begin
            empty(buffer);getnext(ch);go to q0
        end;
q5:     write(buffer);
        while ch in white space char then begin
            write(ch);getnext(ch);
        end
        go to q0
q6:     while ch is a word char then begin
            store(ch);getnext(ch);
        end;
        if ch is any char then begin
            store(ch);getnext(ch);go to q7
        end else begin
            empty(buffer);getnext(ch);go to q0
        end;
```

```
q7:     while ch in white space char then begin
                store(ch);getnext(ch);
        end;
        if ch is a uppercase letter then begin
                store(ch);getnext(ch);go to q10
        end else begin
                empty(buffer);getnext(ch);go to q0
        end;
q8:     if ch in white space char then begin
                store(ch);getnext(ch);go to q9
        end else begin
                empty(buffer);getnext(ch);go to q0
        end;
q9:     while ch in white space char then begin
                store(ch);getnext(ch);
        end;
        if ch=')' then begin
                store(ch);getnext(ch);go to q7
        end else begin
                empty(buffer);getnext(ch);go to q0
        end;
q10:    go to q11;
q11:    while ch is lowercase letter then begin
                store(ch);getnext(ch);
        end;
```

```
        if ch is any char then begin
                store(ch);getnext(ch);go to q12
        end else begin
                empty(buffer);getnext(ch);go to q0
        end;
q12:    write(buffer);
        while ch is any char then begin
                write(ch);getnext(ch);
        end
        go to q0
q13:    while ch is digit then begin
                store(ch);getnext(ch);
        end;
        if ch is any char then begin
                store(ch);getnext(ch);go to q14
        end else begin
                empty(buffer);getnext(ch);go to q0
        end;
q14:    while ch in white space char then begin
                store(ch);getnext(ch);
        if ch='/' or ch='-' then begin
                store(ch);getnext(ch);go to q18
        end else begin
        if ch is uppercase letter then begin
                store(ch);getnext(ch);go to q15
```

```
        end else begin
                empty(buffer);getnext(ch);go to q0
        end;
q15:    while ch is uppercase letter then begin
                store(ch);getnext(ch);
        if ch in white space char or ch='/' or ch='-' then begin
                store(ch);getnext(ch);go to q18
        end else begin
        if ch='.' then begin
                store(ch);getnext(ch);go to q16
        end else begin
                empty(buffer);getnext(ch);go to q0
        end;
q16:    write(buffer);
        while ch='.' then begin
                write(ch);getnext(ch);
        end;
        if ch in white space char or ch='-' then begin
                store(ch);getnext(ch);go to q17
        end else begin
                empty(buffer);getnext(ch);go to q0
        end;
q17:    go to q16
q18:    while ch is digit or ch in white space char or ch='/' or ch='-' then begin
                store(ch);getnext(ch);
```

```
        end;
        if ch is any char then begin
                store(ch);getnext(ch);go to q19
        end else begin
                empty(buffer);getnext(ch);go to q0
        end;
q19:    if ch in white space char then begin
                store(ch);getnext(ch);go to q20
        end else begin
                empty(buffer);getnext(ch);go to q0
        end;
q20:    write(buffer);
        if ch is uppercase letter then begin
                store(ch);getnext(ch);go to q21
        end else begin
                empty(buffer);go to q0
        end
q21:    go to q20
```