

USING METADATA TO IMPLEMENT EFORMS AND THEIR  
ASSOCIATED DATABASES

A Thesis Submitted to the College of  
Graduate Studies and Research  
In Partial Fulfillment of the Requirements  
For the Master of Science Degree  
In the Department of Computer Science  
University of Saskatchewan  
Saskatoon

By

DAVID EDGAR KIPROP LELEI

© David Edgar Kiprop Lelei, December 14<sup>th</sup>, 2010. All rights reserved.

## PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science

176 Thorvaldson Building

110 Science Place

University of Saskatchewan

Saskatoon, Saskatchewan

Canada

S7N 5C9

## ABSTRACT

Web forms (eForms) and databases are at present widely used for data handling in most web applications. While eForms are used for data gathering and display, databases are used for data storage. To connect and interface an eForm to a database, an eForm processor is used. The eForm processor supports data saving, retrieval, update, and delete. In most web applications, eForms, eForm processors, and databases are designed and implemented separately. This leads to two main challenges: One, complexity in the manipulation of eForms and their associated database; and two, difficulty in the reproduction and reuse of existing eForms.

To address the above-identified challenges, this thesis proposes the use of metadata in the creation and implementation of both eForms and their associated databases. Our approach comprises a two-part solution: One, modeling domain's metadata and two, creating a tool, called Delk eForm Creator. To model domain metadata, Resource Description Framework Schema (RDFS) was used. However, to analyse the tool's requirement, Putting Usability First (PUF) approach was used.

In order to demonstrate the applicability of our solution approach, Delk eForm Creator was used to create a set of Metadata and three specific eForms based on a generic eForm. The created eForms were rendered in different web browsers and used to enter data into the associated databases. It was observed that Delk eForm Creator successfully generated a Generic eForm based on the Domain Metadata. Moreover, three different Specific eForms were successfully generated based on one Generic eForm, thereby leading to a reusable Generic eForm.

We conclude that the metadata-based approach of implementing eForms, as proposed in this thesis, is a viable technique to creating eForms and their associated databases. The approach enables users to easily create, maintain, and reuse eForms and databases.

## ACKNOWLEDGMENTS

At last, I am very glad to write this page. The relief and accomplishment I feel in having come to this stage comes with a deep sense of gratefulness to first God who blessed me with this opportunity and to the help, support, and inspiration of the many people to whom the thesis owes its existence.

First of all, I wish to express my sincere gratitude to Professor Jim Carter, who guided me through this work and helped me whenever I was in need. Dr. Carter's direction and support have been helpful and greatly appreciated.

To the committee members: Professor Dr. Ganesh Vaidyanathan, Professor Dr. Simone Ludwig, Professor Dr. Chanchal Roy, and the chair of committee Professor Dr. Gord McCalla. I am most grateful for the precious time you all devoted to this thesis work. It is my honour and I thank you all for the advice and the constructive criticism that contributed substantially to bringing the original conception to this final stage.

I am thankful to all present and former colleagues in USERLab who have provided me with advice, and encouragement, and sharing their knowledge with me. Especially, I am grateful to my colleagues Lisa Tang and Sunny Pal for their useful discussions, we had on different occasions of which served greatly to influence my thesis work. I am also grateful to my friends Johnson Iyilade and Rita Orji for their contribution.

I am so thankful to Ms. Jan Thompson, Graduate Correspondent at the department of Computer Science, who has been very helpful throughout my study life here at the University of Saskatchewan and very kind.

I am also gratefully to the financial support I received from the University of Saskatchewan scholarship offered under the Graduate Student Scholarships. I am grateful for the financial support I got from the Graduate Student Association offered through the GSA Bursary.

I dedicate this work to my parents, my brother and his family, and my extended family at large.

## CONTENTS

PERMISSION TO USE.....	i
ABSTRACT.....	ii
ACKNOWLEDGMENTS .....	iii
CONTENTS.....	v
LIST OF TABLES .....	vii
LIST OF FIGURES .....	viii
ABBRECIATIONS.....	x
Chapter 1     INTRODUCTION.....	1
1.1     Introduction to the use of eForms and database.....	1
1.2     Summary of the main goals and methodologies used.....	4
1.3     Thesis outline and structural overview .....	6
Chapter 2     BACKGROUND AND PROBLEM DEFINITION .....	8
2.1     Existing eForms systems and the different technologies .....	8
2.1.1     Existing eForm systems .....	8
2.1.2     Different technologies used.....	11
2.2     Challenges with current implementation approaches .....	12
2.2.1     Difficulty in manipulation of eForms and their associated databases.....	12
2.2.2     Difficulty in eForms reuse and reproduction .....	13
2.3     Causes of the challenges .....	13
2.3.1     Use of incompatible tools and technologies.....	13
2.3.2     Different vendor structural approaches .....	15
2.4     Introduction of a solution to the causes of challenges .....	15
Chapter 3     ANALYSIS AND USES OF EFORMS.....	17
3.1     Use cases .....	18
3.1.1     Ecommerce applications use case .....	18
3.1.2     Uses of EForms within the ecommerce domain.....	21
3.2     Objects for the use cases .....	24
3.2.1     Generic eForms, Specific eForms, and eForm elements.....	24
3.3     Users of the use cases and objects .....	25
3.3.1     Administrators .....	26
3.3.2     Service Providers.....	27
3.3.3     Customers.....	27
3.4     Operations/Tasks on the objects .....	28
3.4.1     Maintaining EForm Elements .....	29
3.4.2     Maintaining Generic EForms .....	30
3.4.3     Maintaining Specific EForms.....	32
3.5     Attributes for the objects.....	35
3.5.1     EForm Elements Attributes .....	36
3.5.2     Generic EForm Attributes .....	37
3.5.3     Specific EForm Attributes.....	38
Chapter 4     METADATA MODELLING AND TOOL HIGH LEVEL DESIGN .....	40
4.1     Metadata and Delk eForm Creator.....	40

4.1.1	Advantages of using metadata.....	41
4.1.2	Use of metadata .....	41
4.1.3	Using Delk eForm Creator .....	42
4.2	Metadata for eForms and databases creation .....	43
4.2.1	Use of metadata in eForm and database creation .....	43
4.2.2	Types of metadata used in a domain .....	44
4.2.3	Capturing and representing metadata in a domain .....	45
4.3	Using Delk eForm Creator to create eForms and databases .....	48
4.3.1	System requirements .....	48
4.3.2	System components .....	50
4.3.3	System use cases .....	55
Chapter 5	TOOL DETAILED DESIGN AND IMPLEMENTATION .....	59
5.1	Delk eForm Creator Use cases and their interfaces (presentation tier).....	59
5.1.1	Create Domain Metadata interface.....	62
5.1.2	Create Generic eForm interface .....	66
5.1.3	Create Specific eForm interface .....	70
5.2	Delk eForm Creator Classes and component functionality (application tier) .....	77
5.2.1	System Overview Diagram .....	77
5.2.2	Major Components step by step Functionality.....	78
5.3	Database Backend (data tier) .....	80
5.3.1	Table and their attributes .....	81
Chapter 6	TESTING AND EVALUATION.....	84
6.1	Creating eForm objects .....	84
6.1.1	Create Domain Metadata.....	85
6.1.2	Create Generic eForm Metadata.....	87
6.1.3	Create Specific eForm Metadata .....	89
6.2	Using eForm objects .....	94
6.2.1	EForm and database setup.....	94
6.2.2	Using eForms .....	97
6.2.3	Checking database.....	99
6.3	Evaluation .....	100
6.3.1	Comparison between DeC tools and other tools .....	101
6.3.2	Description of an experimental evaluation.....	104
Chapter 7	CONCLUSION AND FUTURE WORK.....	106
7.1	Conclusion .....	106
7.2	Contributions.....	107
7.3	Future work .....	108
APPENDIX I	.....	112
APPENDIX II	.....	114
APPENDIX III	.....	122
APPENDIX IV	.....	137
APPENDIX V	.....	142
REFERENCES	.....	148

## LIST OF TABLES

Table 6-1: Comparison of the functionalities concerning the eForm and database creation .....	103
Table 6-2: Comparison on advanced functionalities concerning the eForm creation ....	103



## LIST OF FIGURES

Figure 1.1: Data handling process .....	2
Figure 2.1: EForm and database implementation .....	14
Figure 3.1: Ecommerce use case.....	19
Figure 3.2: Uses of eForms .....	22
Figure 3.3: Use cases objects .....	25
Figure 3.4: Users .....	26
Figure 3.5: Users and tasks relationship .....	29
Figure 3.6: Maintaining eForm elements.....	29
Figure 3.7: Maintaining generic eForm .....	30
Figure 3.8: Changing generic eForm .....	32
Figure 3.9: Maintaining specific eForm.....	33
Figure 3.10: Changing specific eForm.....	34
Figure 4.1: EForm and database implementation using metadata .....	42
Figure 4.2: EForm and database creation using Delk eForm Creator.....	43
Figure 4.6: EForm and database metadata creation flow.....	50
Figure 4.7: DeC major components .....	51
Figure 4.8: General use for creating eForms .....	56
Figure 5.1: DeC general User Interface Layout.....	60
Figure 5.2: DeC Default Interface .....	61
Figure 5.3: DeC Log-In Interface .....	62
Figure 5.4: Default DeC Create Domain Metadata Interface .....	63
Figure 5.5: Using DeC Create Domain Metadata Interface.....	64
Figure 5.6: DeC Create Generic eForm Interface .....	66
Figure 5.7: DeC Interface Populated with Domain Metadata attributes.....	67
Figure 5.8: DeC Create Specific eForm Interface .....	71
Figure 5.9: DeC Interface Populated with entity attributes .....	72
Figure 5.10: DeC Classes diagram.....	78
Figure 5.11: DeC backend database E-R diagram .....	81
Figure 6.1: Generic eForm example created by DeC .....	85
Figure 6.2: Specifying Domain Metadata attributes .....	86
Figure 6.3: Domain Metadata represented and formatted in RDFS .....	87
Figure 6.4: Customizing Generic eForm Metadata entities and their attributes .....	88
Figure 6.5: Generic eForm Metadata represented and formatted in XSD .....	89
Figure 6.6: Customizing Specific eForm Metadata entities and their attributes.....	90
Figure 6.7: Specific eForm 1 example created by DeC rendered using Chrome browser	91
Figure 6.8: Specific eForm 2 example created by DeC rendered using Firefox browser.	92
Figure 6.9: Specific eForm 3 example created by DeC rendered using internet explorer browser.....	93
Figure 6.10: A real world eForm example shown in an e-Commerce page .....	93

Figure 6.11: eForm example created by DeC based on a real world eForm .....	94
Figure 6.12: Interface for setting up the created database .....	95
Figure 6.13: Interface showing the database setup results.....	96
Figure 6.14: Interface for database management.....	96
Figure 6.15: eForm example created by delk eForm creator .....	97
Figure 6.16: Specific eForm 1 usage .....	98
Figure 6.17: Specific eForm 2 usage .....	98
Figure 6.18: Specific eForm 3 usage .....	99
Figure 6.19: Database management interface showing number of data rows entered....	100
Figure 6.20: phpAdmin interface showing the actual data entered in the database.....	100

## ABBRECIATIONS

**ACORD:** Association for Cooperative Operations Research and Development.

**APEH:** Hungarian Tax and Financial Control Administration.

**CAPTCHA:** Completely Automated Public Turing test to tell Computers and Humans Apart.

**CRUD:** Create, Retrieve (Read), Update, Delete

**DBMS:** Database Management System.

**DB:** Database.

**DeC:** Delk eForm Creator.

**ER Diagram:** Entity-Relationship Diagram.

**HTML:** HyperText Markup Language.

**IBM:** International Business Machines.

**ID:** Identifier.

**IEC:** International Electrotechnical Commission.

**ISO:** International Organization for Standardization.

**MS:** Micro-Soft.

**MySQL:** Structured Query Language.

**OWL:** Web Ontology Language.

**PEOU:** Perceived ease-of-use.

**PHP:** Hypertext Preprocessor.

**PU:** Perceived usefulness.

**PUF:** Putting Usability First.

**RDF:** Resource Description Framework.

**RDFS:** Resource Description Framework Schema.

**SE:** Software Engineering.

**SQL:** Structured Query Language.

**TAM:** Technology Acceptance Model.

**UE:** Usability Engineering.

**UML:** Unified Modeling Language.

**USERLab:** USability Engineering Research Lab.

**W3C:** World Wide Web Consortium.

**XHTML:** Extensible Hypertext Markup Language.

**XML:** Extensible Markup Language.

**XSD:** XML Schema Definition.

## CHAPTER 1 INTRODUCTION

The main goal of this thesis is to illustrate an approach of using metadata to create eForms and their associated databases. Section 1.1 provides a discussion on the introduction to the use of eForms and databases. In section 1.2, a summary of the main goals and methodologies used in this thesis are presented. Section 1.3 presents the thesis outline and structural overview.

### **1.1 Introduction to the use of eForms and database**

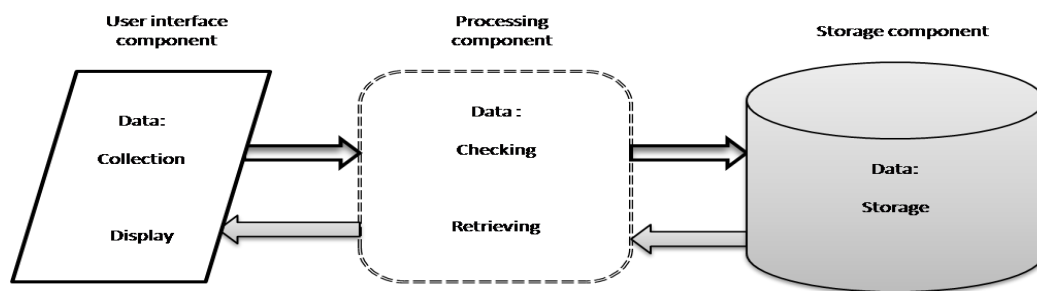
Web applications used in different domains increasingly make use of eForms to perform data handling. A domain is an environment that defines a set of common requirements, terminology, and functionality. An example of a domain is ecommerce. Data handling includes input (i.e. gathering), output (i.e. displaying), processing, and storage of data. In the past, the use of paper forms and databases was important in accomplishing data handling. The paper forms were used for gathering and displaying data. Data gathered using the paper forms would then be entered and stored in the databases.

With increased use of the Internet and the advancement in technology, there is a shift from the use of paper forms to the use of web (digital) forms [15, 45]. In this thesis, web forms are referred to as eForms. EForms serve the same purpose of data handling as paper forms do; however, they appear on the display screen. The display screen can be that of a traditional computer or other devices, such as personal digital assistants or cell phones, devices used to access data.

The various tasks a user can accomplish using eForms include but are not limited to data entry, data searching and data comparing. To enter data, the user fills in the eForm by either selecting options provided on the eForm with a pointing device i.e. a mouse or the user types in

data in form of text using the device keyboard. The data is then sent directly to an eForm processing application, which then transfers the data into a database.

Data collected using eForms are mostly checked against data constraints before being stored in the databases. Once the data is stored, they can be retrieved from the databases and displayed on appropriate eForms using retrieval mechanisms present in the eForm processors [43]. Checking and retrieval of data are part of the eForm processor's task. Data handling can be grouped into three major web application components: the user interface component, the processing component, and the storage component. Figure 1.1 shows these three web application components. The figure also shows how the data flow through the three components.



**Figure 1.1: Data handling process**

EForms help reduce printing costs, save time, and problems most often encountered while using paper-based forms. For example, lack of feedback and data validation, unreadable handwriting, delay in data entry to database system [2]. Furthermore, when used for data handling, eForms offer more flexibility and functionality. It is possible to automate eForms so as to be used for complex business processes. EForm automation reduces the need for manual eForm processing. EForms are used to deal with large amounts of data electronically [46]. The use of eForms helps the eForm user to become more productive, efficient and swift in using data since there are few input errors, thanks to validation checks. The data can also be accessed and reused easily [16].

The advantages of eForms over paper forms have led to a rising need for and adoption of eForms within web applications. This has made eForms an important part of today's information systems [7, 46]. The need and adoption of eForms has created a huge demand for eForms use. There is a need to create eForms to serve this demand. This need has led to emergence of vendors to satisfy this demand for creating eForms and their associated databases.

EForms and their associated databases can be created separately from web applications, but more often than not, eForms are created within web applications and the databases created separately. There are three current approaches (ways) of creating these eForms. These approaches are; 1) an application developer can choose to create eForms and databases by themselves; 2) an application developer can employ a vendor to create both eForms and databases; or 3) an application developer can employ different specialized vendors to create eForms and databases separately. When there is high need to create and maintain large number of eForms to satisfy a large number of users' needs, application developers are pushed to employ vendors to create eForms and databases for them.

These current approaches of creating eForms and databases used within web applications have some challenges. This thesis examines the challenges related to these current eForm creation approaches and provides a possible solution that is implemented and tested. The challenges of the current approaches arise from how the eForms and their associated databases are created and integrated within a web application.

Most of the time when web application developers develop a web application, they develop different range of applications in terms of size. The web application might be huge such as ebay.com and amazon.com. However, sometimes these developers are asked to create small application e.g. registration web application. The application could either be registration for

membership or registration for participation. While these applications are similar they require different functionality. With current approaches developers cannot reuse eForm (code) already developed. Modifying existing code requires more work because changes have to be made to the form, eform processor and the database. Moreover, if the big application supports service providers, it comes a point in time when they need to create and use similar eForms as those created by the application developers. This thesis identifies the following two challenges related to these scenarios:

- Difficulty in the manipulation of eForms and their associated database.
- Difficulty in the reuse and reproduction of existing eForms.

## **1.2 Summary of the main goals and methodologies used**

The main concerns that distinguish the approach of this thesis in illustrating the creation of eForms, eForm processors, and their associated database are:

1. Creation of eForms and their databases using metadata.
2. Creation of easy to manipulate eForms and databases.
3. Creation of reusable eForms and eForms components by maximizing reusability of metadata.
4. When eForms are created, their databases should be automatically generated.

Generally, metadata is defined as data that describes or provides information about other data entity [25]. The metadata makes the described data entity easy to search and reuse. In this thesis, metadata is used to maintain the association between the eForm and the database entities in a semantic manner based on a domain. The metadata is also used to control the creation and manipulation of the eForms, eForm processors, and the databases.



There are diverse domains where eForms and databases could be used, the focus in this thesis is on the ecommerce domain in general and online art ecommerce application in specific.

Keeping in mind the core aims stated above, the methodologies used in this thesis are as follows:

- *Problem definition.* The two main challenges to be addressed in this thesis are discussed. First, difficulty in the manipulation of eForms and their associated databases; second, difficulty in eForms reuse and reproduction. Causes of these challenges are also discussed.
- *Methodology.* A solution of using metadata in the creation process of eForms and databases is presented. This solution is broken down into two parts for better understanding. The parts are, 1) finding a way of modeling domain's metadata geared towards eForm creation and 2) creating a tool that is used to model the Domain Metadata and create eForms and their associated databases based on the Domain Metadata. An XML based technology; Resource Description Framework Schema (RDFS) [37] is used to model Domain Metadata. By doing this, the reusability of the Domain Metadata and the resulting eForms is achieved. Putting Usability First (PUF) methodology [10] is used to help in the analysis of the tools requirements. Based on the requirement analysis and ideas presented in this thesis, a tool has been developed. The tool is called Delk eForm Creator (DeC). This tool currently supports the following operations: (1) the modelling and manipulation (i.e. create, modify, and search) of metadata; (2) the manipulation of simple eForms; (3) automatic creation of databases associated to created eForms; (4) automatic creation of eForm processors that enable interfacing between eForms and databases by providing the CRUD

methods (i.e. Create, Retrieve, Update, and Delete); (5) provides an interface to setup and manage the created databases.

- *Evaluation.* To illustrate the concept of this thesis, the tool (DeC) is used to create a set of Metadata (Domain, Generic eForm, and Specific eForm metadata) and some eForms. A documented demonstration of how to create metadata for a Generic eForm and how the resulting Generic eForm is rendered on a web browser and used by a typical user is provided. Moreover, a demonstration of how to create metadata for 3 Specific eForms based on one Generic eForm and how the resulting Specific eForms are rendered on different web browsers and used by a typical user is provided. In addition a documented demonstration of how to create metadata for an eForm based on some real world ecommerce webpage and then how the eForm is rendered on a web browser and used by a typical user is provided.

### **1.3 Thesis outline and structural overview**

The thesis is organized in seven chapters. The challenges are identified, a solution is identified, an implementation of the solution is shown, and finally a conclusion is provided.

**Chapter 1** (*Introduction*). In this chapter an introduction of the thesis work is presented. The main goal of the thesis is identified and discussed. A brief discussion on the focus of the thesis and what has been done to fulfill the requirements of the thesis is provided.

**Chapter 2** (*Background and Problem Definition*). In this chapter a background study on other approaches and tools used in creating eForms and databases is presented. The challenges of these approaches are identified and discussed. The identified solution is also introduced.

**Chapter 3** (*Analysis and Uses of eForms*). In this chapter analysis of eForms and databases is presented. Example of ecommerce scenarios are used to help in the analysis. The analysis discussion on ecommerce covers use cases, uses of eForms, operations on eForms, and attributes of eForms.

**Chapter 4** (*Metadata Modeling and Tool High Level Design*). This chapter presents and discuss in more details about this thesis approach of using metadata as an approach of creating eForms and their associated database. A way of modeling and structuring the metadata in a manner that is useful in eForm creation is presented. The system requirements and high level design of the tool is presented.

**Chapter 5** (*Tool Detailed Design and Implementation*). In this chapter a detailed design and implementation of the tool components is presented. The functionalities supported by Delk eForm Creator are also discussed.

**Chapter 6** (*Testing and Evaluation*). This chapter presents the test result of using Delk eForm Creator. This chapter also provides evaluation derived from examining Delk eForm Creator's functionality to see if it fulfills the requirements of this thesis and provide a solution to the identified challenges and proves the concept of this thesis.

**Chapter 7** (*Conclusion and Future Work*). This chapter summarizes the main ideas of this thesis, and suggests directions for future work.

## CHAPTER 2 BACKGROUND AND PROBLEM DEFINITION

In this chapter, a background study on other approaches and tools used in creating eForms and their associated databases is presented. This background study is useful in identify the causes of the challenges in the current eForms and databases implementation approaches. Finally this chapter introduces this thesis solution to the identified challenges.

### **2.1 Existing eForms systems and the different technologies**

#### ***2.1.1 Existing eForm systems***

Many vendors have sprung up to exploit the profitable opportunities offered by the great demand for eForms. Most of these vendors provide batch services; because, they do not produce only one or two web application systems for one enterprise, but usually are involved in producing hundreds of different eForms application for different web application developers and enterprises [46].

Some of the vendors that are currently involved in the creation of multi-context of use eForms include; IBM Workplace Forms [46], EZ-Forms [19], and FormDocs [22]. There are also vendors that are involved in creation of specific context of use eForms, they include: APEH [24], ACORD [45], and Admin Tool [33]. While these vendors have their benefits, they also have limitations. This section describes these eForm systems briefly and identifies their benefits and limitations.

IBM Workplace Forms is one of the implementations of XForms standard, which is supported by W3C organization. IBM Workplace Forms helps accomplish eForm related tasks and services. The main service they offer is to convert the original forms (either paper or electronic) format to Workplace Forms format. It then refines the converted results to share some common properties. When application developers change any of these properties, “the change

applies to all eForms, both finished and unfinished” [46]. The creation process involves the application developer providing sample eForms with all requirements they need.

IBM Workplace Forms vendor then creates sample eForms that need to be reviewed before being finalized. The limitations of this approach are: first, any change on a single eForm affects other eForms, second, the database and the eForms are created using different programs hence increasing the possibility of incompatibility, and third, the application developer cannot make changes to the eForms without the help of an IBM Workplace Forms vendor.

EZ-Forms provide eForm and database solution for the desktop and internet based applications for application developers [19]. With EZ-Forms it is possible to create database enabled or stand-alone eForms for use with EZ-Forms Filler. It is also possible to share eForms created by using EZ-Forms viewer. EZ-Forms enable the application developers to simply scan their forms into EZ-Forms ULTRA Designer. EZ-Forms also allow the creation of a database using a wizard. The limitation of this approach is that separate programs are essentially used to create eForms and the databases causing possibilities of incompatibility.

FormDocs is software that enables application developers to create eForms. In its description it seems there is no need for a database to be used with the resulting eForms. “FormDocs is much simpler to use than database or spreadsheet programs so your users don't need special skills to get the job done [22].” The application developers can easily design the eForms exactly the way they want it. This approach has the same limitation as EZ-Forms, i.e. the creation of eForms and databases is done by different programming tools hence the possibility of incompatibility. There is less regard for the use of databases for data storage.

The Hungarian Tax and Financial Control Administration system was developed by APEH to provide government services that include e-payment to the community [24]. The initial

idea was to develop general purpose e-communication software that would be used by citizens and businesses, regardless of the cause of communication. According to [24] there exist more than 160 downloadable eForms that are currently supported on the portal of APEH servers. These eForms are used to serve different purposes. The system is divided into three sections developed to ensure safe communication.

The sections are: the form designer and generator section, the client section for filling, signing and sending eForms, and the server section including a portal and a processing part. The system provides general services including: form filling support, client programs to sign and send eForms, support for data collection, interface for registration, time-stamp, and support for certification control. The designer works on MS Windows panels. The different versions of the form under construction are stored in an Oracle database. To support the standardization of forms, APEH defines libraries of components and supports the use of sub-forms. The form designer generates a code interpretable for the form-filling program from the stored form-plans [24]. One limitation of this approach is that the system works only on one operating system hence would be incompatible with systems that do not run on windows. Another limitation is that changes to the already stored components and sub-forms might be hard to achieve.

ACORD [45] provides a description of how eForms used by a group of insurance companies are created. “ACORD (Association for Cooperative Operations Research and Development) maintains more than 450 standard forms to meet the requirements of the industry and government regulators” [45]. Lately, ACORD are moving away from use of paper forms towards the use of eForms. They use programs managed by IBM that includes files based on Lotus Forms pure XML technology, which allows the XML data captured in the ACORD eForms to be submitted, processed, filed, archived and digitally signed. The current ACORD

Standard forms are usually used as static forms. XML is only used to formalize the form presentation layer. XML is not being used to express data metadata. There is no mapping between ACORD Data schema and ACORD eForms [45]. The limitations with ACORD eForms are that they are not dynamic and non-reusable in reproducing similar eForms.

The Administration CASE Tool created by USERLab [33] is used to create eForms for multiple methodologies used in software engineering (SE), usability engineering (UE) and their related projects. It is designed to be used to “explore the requirements of new UE methodologies, support existing methodologies and support their translation into SE requirements” [33]. The limitation of this approach is that, its use is limited to specific domain of methodologies. A domain is an environment that defines a set of common requirements, terminology, and functionality.

### ***2.1.2 Different technologies used***

The eForm system described in section 2.1.1 uses different programming technologies to create the eforms. The increase in the number of these programming technologies increases the difficulty of creating eForm applications. This difficulty is increased by the complexity involved in the interactions between the older generation of technologies and the newer generation of technologies. There are many technologies used for every aspects of the web application thanks to generational development of these web technologies. These technologies, if not used well might contribute further to the complications presented by incompatibility issues between different technologies.

When dealing with web applications, the first generation web technology used HTML to help render data and eForms as user interface on the web [9, 5]. HTML provided the basic document and eForm formatting and hyperlinks for online browsing [8]. A major challenge of

the first generation web technology (HTML) is the fact that all its eForms are manually hard-coded on the page. This type of user-interface is too inflexible when dealing with content that is stored in existing databases or that is subject to frequent changes and should be displayed on eForms.

The second generation web technology ushered in a more dynamic, interactive web by defining ways to automate eForms to collect or display data to web application users [7]. The dynamic capabilities of second generation web technologies though they solve the challenges posed by manual coding of eForms, they do not offer ways of searching and comparing data easily.

## **2.2 Challenges with current implementation approaches**

The shift towards the use of dynamic web data manipulation has improved the "web's utility and the experience of web users" [9]. This shift has led to new ways of creating and programming eForm based web applications. Such new ways include use of eForm systems to create eForms as discussed in Section 2.1. Each the described systems have one or both of the following challenges:

- Difficulty in the manipulation (i.e. creation, modification) of eForms and their associated database.
- Difficulty in the reproduction and reuse of existing eForms.

### ***2.2.1 Difficulty in manipulation of eForms and their associated databases***

There is certain level of difficulty involved in creating new eForms. There is even more difficulty in making changes to existing eForms to accommodate eForm attributes which were not earlier identified. The difficulty increases where the eForms have associated databases. This requires that changes on the eForm have to be reflected on the associated databases and eForm



processor. This scenario requires some eForm, eForm processor, and database programming expertise.

In some cases, eForms are implemented in such way that it is difficult to make minimal needed changes on the entities of the eForm. A situation where minor changes to the eForm entities would have solved the problem, might require that the existing eForm be gotten rid of and a new one created. A new set of eForm entities are created and existing database structure and entities are changed. This practice take time, resources to accomplish, and might require complex operations to be done on both the eForm entity structure and the already existing database entity structure. These cases often might require the intervention of the vendors initially contracted to create the databases or the eForms. This practice might prove costly and inefficient where a change is needed to address an urgent need.

### ***2.2.2 Difficulty in eForms reuse and reproduction***

It is difficult to reuse code for already created eForm for reproduction of similar eForms with minor. New eForms are created each time there are new needs to either add or delete fields from existing eForms. Lack of reproduction means reuse of existing eForm entities and their structure to produce new eForms is not possible.

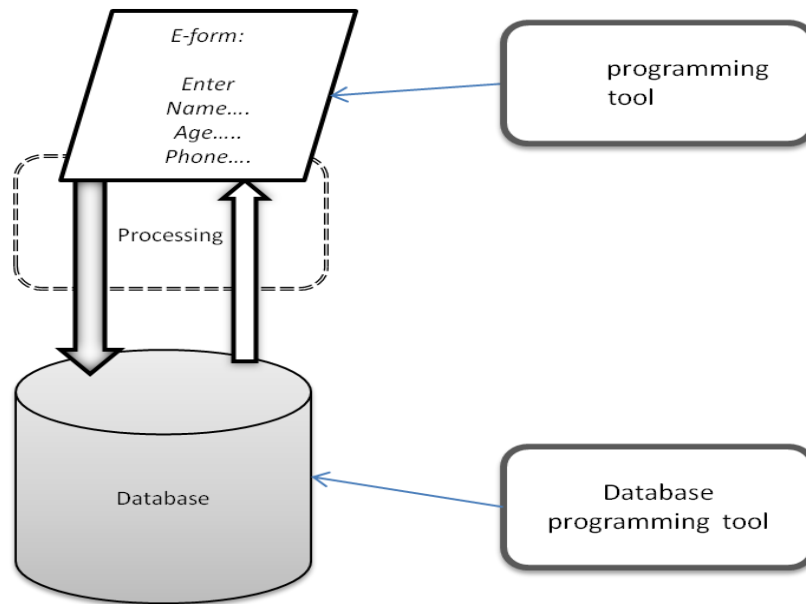
## **2.3 Causes of the challenges**

Some of the factors that contribute to the challenges identified in section 2.2 include use of incompatible programming technologies to create eForms and their associated databases and emergence of multiple vendors who implement eForms in different structural ways.

### ***2.3.1 Use of incompatible tools and technologies***

When application developers want to create their web application they can either develop it themselves or employ external vendors. The development might use different technologies to

create the different parts and components. When the vendors are chosen to create different components (i.e. the eForms or the databases) of the application, there is a high chance that the different vendors will use different technological approaches to build the application component they are assigned. Figure 2.1 show different technologies used to create eForms and databases separately.



**Figure 2.1: EForm and database implementation**

There are numerous programming tools used for creating eForms, eForm processors, and databases separately. Often, eForms, eForms processors, and databases created using different technologies eventually are combined together to create one web application. This approach contributes to the challenge concerning difficulty in eForm manipulation because of the incompatibility issues that might exist between programming tools used. The interoperability and incompatibility issues are addressed during the merging of eForms and databases within the web application being developed. If there are changes to be done in the eForm structure, it often requires a change to be made in the databases and the eForm processor.

### **2.3.2 *Different vendor structural approaches***

More often than not, eForms and their associated databases are developed without encapsulating the domain information. When different vendors are employed to create eForms, it might be difficult for the vendors to encapsulate the domain expertise into the eForm creation process. The reason for lack of domain encapsulation is that, often the web application developer implicitly assumes that the vendors know about the domain's knowledge. The domain knowledge is implicit to the web application developer, yet the knowledge is not passed on explicitly to the vendors who create eForms [42] and the databases.

There is also a likelihood that the different vendors will never directly communicate. This is because they each communicate to the application developer directly. Because of the lack of direct communication between vendors, there is no way of establishing common parameters needed for the components to work together hence their work resulting in incompatible components.

Even if the vendors were provided with the same component requirements, there are possibilities that the vendors would come up with different entity structures to address the requirements presented to them. Hence, the resulting eForms and databases would have different entity structures. This leads to eForms and databases that have interoperability and incompatibility issues within the web application. The limitations of the current eForm implementation approaches calls for change on how these eForms and their associated databases are designed and implemented.

## **2.4 Introduction of a solution to the causes of challenges**

The development of the Web has created demand for machine computable information [40]. To make the Web and web applications more accessible, there is a need to use the metadata

in building components of these web applications. Metadata can be used to implement “intelligent” eForms and databases used within web applications.

To implement the eForms and the databases using metadata as part of web application, a two part solution is presented in this thesis. The first part is the modeling of the metadata and the second part is the use of a manipulation tool called Delk eForm Creator (DeC) in the creation of the metadata, the eForms and the databases.

The use of metadata to control the creation of both the eForms and the database is a possible solution to the challenges concerning difficulty in eForm reuse. The use of DeC is a possible solution to the challenges caused by the use of incompatible programming tools and technologies. The DeC would be used to create the metadata used for both creating eForms and databases, thus harmonizing the difference in tools and technologies used.

### CHAPTER 3 ANALYSIS AND USES OF EFORMS

This chapter presents an analysis of eForms and databases. EForms are used widely in different web application domains. Some examples of domains include but not limited to the software engineering domain, the usability engineering domain, the ecommerce domain, and the education domain. In this thesis, the focus is on the ecommerce domain in general and art ecommerce application specifically. Art ecommerce application is an application used for buying and selling with art products. In order to understand how eForms and databases are created and used in an art ecommerce domain, a requirement analysis is done.

The Putting Usability First (PUF) methodology is used to help in analyzing of the use cases, users of the use cases (actors), objects of the use cases, operations on the objects, and attributes of the objects. PUF is a usability engineering methodology that attempts to balance usability concerns for both users and developers for a given system [10]. It is a methodology that is under development in USERLab at the University of Saskatchewan.

The basic concept behind PUF is that the right consideration is given to usability issues in each activity of application development life cycle [32]. Such issues include (all main scenarios (use cases), user groups, tasks (operations), content (attributes), and tools related to the application systems, which are identified and briefly described. Usability is defined by ISO 9241-11[30] as the “extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use”. ISO 9241-11 [30] defines context of use as the “users, tasks, equipment (hardware, software and materials), and the physical and social environments in which a product is used”.

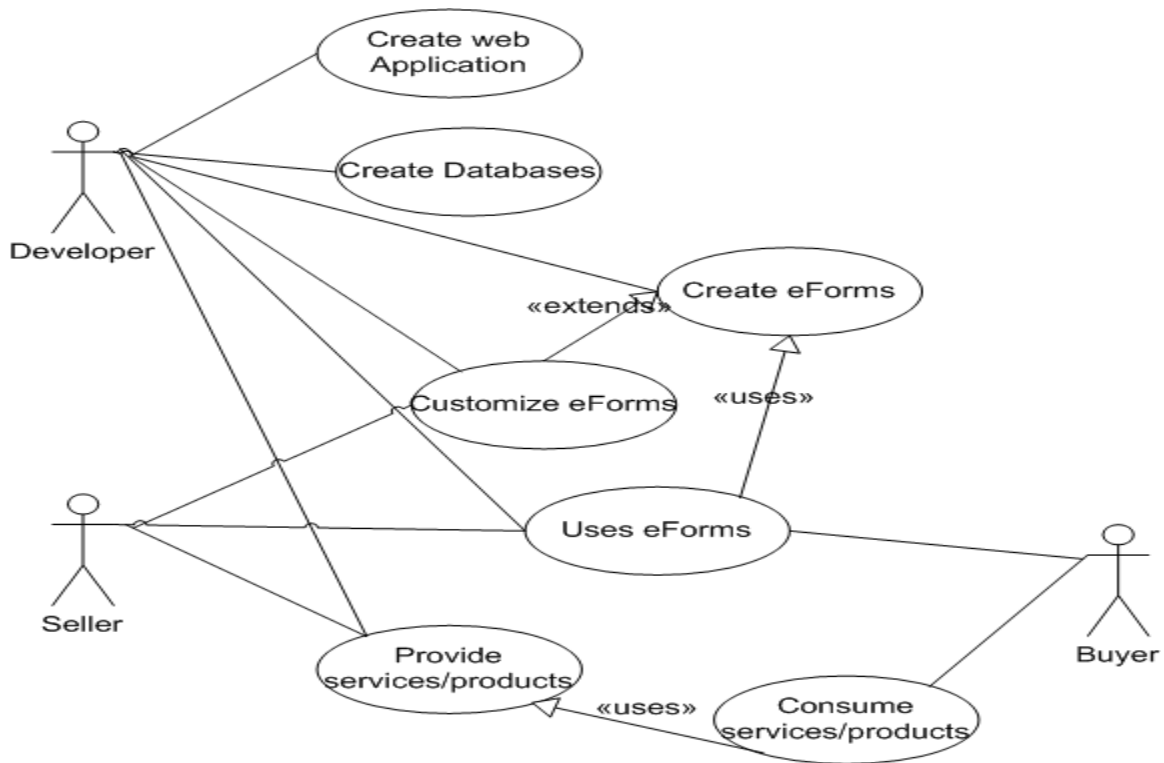
### **3.1 Use cases**

A use case is a description of sets of interactions between users and the web application system. Use cases used in the UML approach are comparable to scenarios used in usability engineering [32]. A scenario is mostly taken as an instance of a use case. Scenarios like use cases might be simple or complex involving multiple different types of users, tasks, content, and tools. In the following sections, a general ecommerce domain use case is described to illustrate the different roles played by different user groups in the creation and use of eForms and databases within ecommerce web applications.

#### ***3.1.1 Ecommerce applications use case***

In general, ecommerce consists of the buying and selling of products or services via electronic systems such as the Internet and other computer networks. At present there are various web applications that support ecommerce. In most ecommerce web applications eForms are used as user interfaces, means of gathering and displaying data and databases are used for the storage of the data.

There are lots of user groups that interact with ecommerce web applications: buyers, sellers, agents, product suppliers, wholesalers, auditors, and customer service representatives. In this general use case the focus is on the three general grouping of users: application developers, sellers, and buyers. Figure 3.1 shows how different users interact with eForms and databases within the ecommerce domain.



**Figure 3.1: Ecommerce use case**

In the following sections, examples of the general use cases where eForms and databases are used are described. These examples include Amazon.com, eBay.com, and art shop use cases.

#### *3.1.1.1 Amazon.com use case*

Amazon.com started out as a bookseller and now serves as a sales platform for almost any product that could be sold. In Amazon.com use case, there are 3 general grouping of users identified in this thesis: the Amazon.com application developers, the partner stores, and the customers. Amazon.com application developers provide eForms and databases as part of their ecommerce web application. The partner stores use the eForms provided by Amazon.com to support their tasks in using the Amazon.com web application. The partner stores could customize the use of eForms by filling or not fill all the fields that are provided on the eForm. However, they cannot customize on the number or the types of field to be on the eForm.

The partner stores use the Amazon.com databases to store data gathered using the customized eForms. The partner stores need customizable eForms that its fields could easily be added or deleted. Such customizable eForms could easily be customized to capture specific data from customers and other partner stores.

It is expected that each customer accesses the web application and perform tasks that help them fulfill their goal of finding a desired product. A customer uses Amazon.com application to purchase different products. The customer searches the ecommerce web application for the product that meets their needs using the eForms. The customer can also search for a partner store that provides the product that the customer desires. After searching and browsing, the customer can request more information about the product before ordering it. The customer can then place orders on an already existing product using the eForms. The customer can use the eForm to communicate their needs, complaints or suggestion to the partner store.

#### *3.1.1.2 EBay.com use case*

EBay.com started as a place for individuals to auction off their collectibles. While the selling model is considerably different between EBay.com and Amazon.com, their use cases are very similar. EBay.com has three general groups of users: eBay.com application developers, sellers, and buyers which have the same meaning as amazon.com developers, partner store and customer respectively.

#### *3.1.1.3 Art Shop use case*

Art Shop unlike eBay.com and Amazon.com is a web application used to sell and buy custom art products (e.g. paintings, sculptures, and photos). While the selling model of Art Shop would be considerably different from that of EBay.com and Amazon.com, the use of eForms and databases is mostly similar except for specific instances (e.g. eForms used to order customized



painting might be used within art shop use case and not on eBay or Amazon.com). Art Shop has three general groups of users: Art Shop application developers, artists, and art customers, which have the same meaning as Amazon.com developers, partner store and customers respectively.

To support their different selling models, eBay.com and Amazon.com provide different sets of eForms to their users (both the individuals who use their sites to sell products and the individuals who use their sites as customers). While both eBay.com and Amazon.com started with a single set of eForms, for all types of products that they help to sell, the wider the variety of products that are sold within their application, the greater the diversity of how those eForms are used and the greater the pressures to provide product specific variants of the eForms. In addition to the product specific eForms, web application developers need to provide specialized eForms to address needs such as that of negotiation that might arise during product customization process in the case of art shop use case.

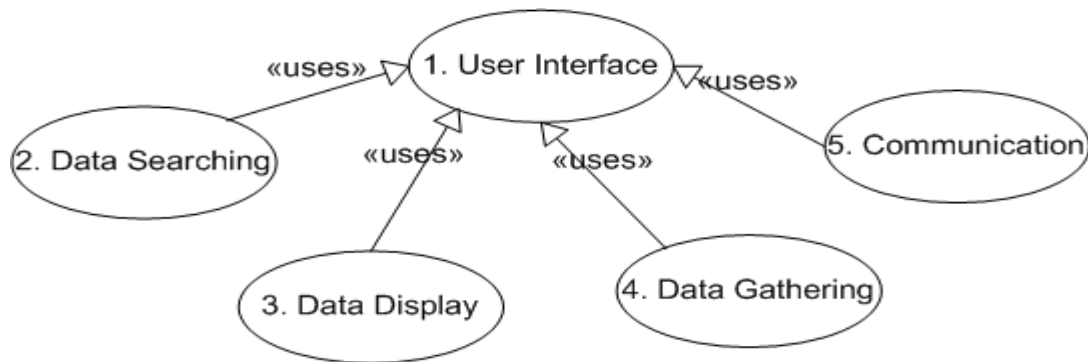
From the three use cases examples provided, there are three different user groups that perform similar roles in each use case. For the purposes of this thesis, the following terms will be used to represent the different groupings.

- Administrators: represent application developers described in the 3 use cases.
- Service providers: represent the partner stores, sellers, and artists in the Amazon.com, eBay, and art shop use cases respectively.
- Customers: represent the customers, buyers, and art customers in the Amazon.com, eBay, and art shop use cases respectively.

### ***3.1.2 Uses of EForms within the ecommerce domain***

Ecommerce domain contexts require customers to interact with service providers in various different ways regarding services or products offered and available data. The web

application eForms offer a different dimension on how the data might be provided or changed to suit its users' desire. There exist different interactive systems that might be used to either retrieve or provide information. This section describes what eForms could be used for within web application systems. Figure 3.2 shows the uses of eForms.



**Figure 3.2: Uses of eForms**

#### *3.1.2.1 User interface*

EForms can serve as part of the user interface for the web application, i.e. what the user sees and interacts with on the computer or other display device screen [1]. At the same time, it is more than just an interface since it is the medium that the customers can use to communicate their needs to other users of the web application system or perform other tasks. The content of the eForm may have some impact on how these tasks are achieved.

#### *3.1.2.2 Data searching tool*

The users of ecommerce web applications should be able to search for product data using the eForms. Electronic interaction between sellers and customers from different parts of the world increases the size of the market and purchasing efficiency. The use of eForms has changed how product information and services can be accessed. Instead of searching through huge volumes of catalogue pages, a simple search on a service provider web application for any product using eForms can be done.

The searching experience could sometimes turn out to be tiresome if the customers do not find what they want [35]. The difficulty in precise searching might depend on the degree of flexibility of product or service description provided [7]. Such inflexible description might result in difficulty while searching or matching products or services. The challenge with current eForm implementation is that, the “intelligence” of these eForms is often limited [41].

#### *3.1.2.3 Data display*

EForms can be used to display most comprehensive and current content in order to keep the users of the application well updated with the latest content. The eForms used by the customers to add products to the shopping cart should automatically display the total cost of the products as the customers adds or removes products from the cart.

#### *3.1.2.4 Data gathering*

EForms are used to collect data from customers using a web application. EForms can be used to collect various kinds of information (e.g. surveys, personal information) from the customers. The eForms are faster in information collection, delivery, and have greater ease-of-use than with paper forms service. The need for and the presence of data validity allows correct entry of information.

#### *3.1.2.5 Communication (Negotiation)*

We live in electronic times where text messaging, emailing, and video conferencing are frequently used ways of communication. For this reason most organizations are adopting these quicker means of exchanging information. Through eForms, ecommerce organizations could change their eForms entity structure and content on the fly. EForms could be seen as tool for communication through which ecommerce web application users could perform different communicative actions between each other [1]. The eForm can be used to facilitate negotiations.

The eForm can be used to make initial enquires on products or services: the same eForms could be used to respond back on the inquiries and report any changes deemed necessary during the process of creating the product.

### **3.2 Objects for the use cases**

In the ecommerce use cases, the administrators are expected to provide eForms for all types of products that are sold by the service providers using the ecommerce web application. The wider the variety of products that are sold via the application, the greater the need to provide eForm elements that support the entire list of products attributes on the eForms.

Different service providers provide different amount of data and information to be used for describing the products or services they offer. Some service providers might provide all the data for all the product attributes on the eForm while others provide data for some attributes. This necessitates the need for customization of the eForm used by individual service providers.

#### ***3.2.1 Generic eForms, Specific eForms, and eForm elements***

Since different service providers using the same web application might provide information on various product attributes to be captured on the eForms using eForm elements, it is clear that two types of eForms (i.e. Generic and the Specific) could be used and therefore need to be created. A Generic eForm is an eForm created by the developers to include all possible attributes for a product or service. A Specific eForm is an eForm customized to meet the needs of a specific service provider. The Specific eForms are based on Generic eForms.

Both Generic and Specific eForms are created based on the attributes of the product that need to be gathered or presented based on a specific ecommerce domain. An eForm element is a special control component (e.g. checkbox, radio button, menu...) and label on the component that helps build the eForm presentation [18]. The eForm elements can represent containers for eForm

data. They specify the layout of the eForm depending on the content and type of the eForm element used. They also allow users to interact with the eForm (i.e. entering text, selecting menu items). Figure 3.3 illustrates how different Specific eForms can be generated from the same generic eForm based on same eForm elements and their labels. The Generic eForms, Specific eForms, and eForm elements are identified as the three objects for the use cases described in Section 3.1.

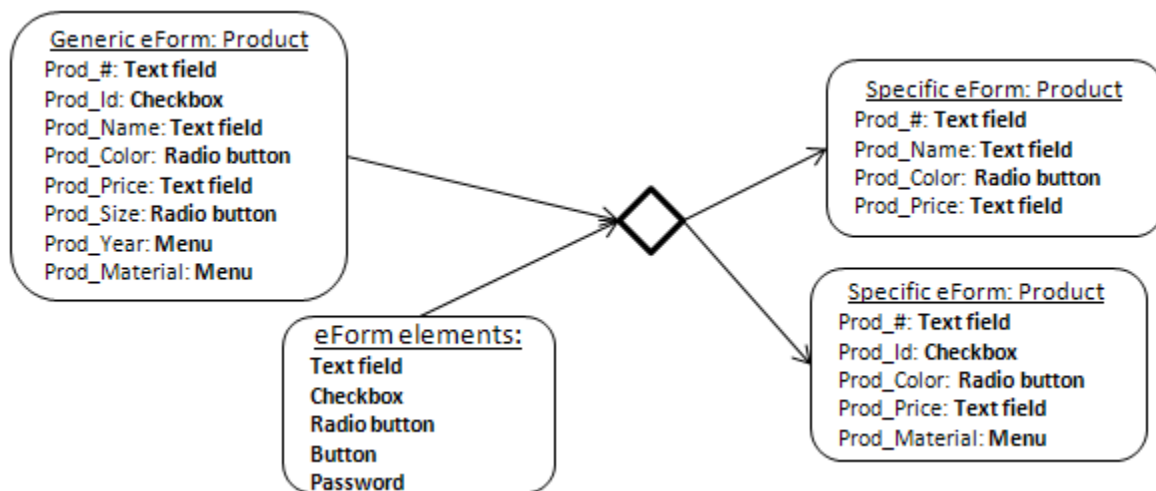


Figure 3.3: Use cases objects

### 3.3 Users of the use cases and objects

Figure 3.4 shows the roles and the interactive relationship between user groups identified in the use cases described earlier in Sections 3.1. It also shows the interaction flow between the different user groups. In discussing the user groups, it is important to describe users according to different criteria. The identified criteria are: roles and interactions.

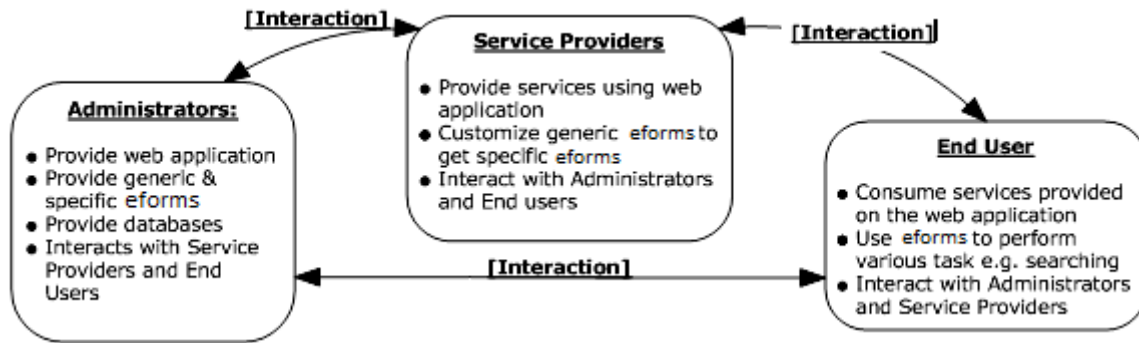


Figure 3.4: Users

### 3.3.1 Administrators

#### 3.3.1.1 Roles:

An administrator is the individual or organization that provides eForms and databases as part of the web application used within the ecommerce domain. They use the eForms to perform some of their tasks (e.g. interacting with other users). They create and manage generic eForms and databases based on the domain description the eForms are created for. They also analyze opportunities to add new types of products and their associated attributes to the existing set of products hence identifying needs for new eForm elements.

#### 3.3.1.2 Interactions with other Groups

There is a constant interaction between the administrators and the service providers. The service providers' suggestions and input are important for the administrators in their tasks of creating new or manipulating existing generic eForms to accommodate the service providers' needs. The administrator might also be in contact with customers both to encourage them to use the ecommerce web application and to identify additional eForm elements that they might be interested in having added to one or more eForms. For example: Amazon.com and ebay.com undertake various initiatives to recruit customers and to keep them satisfied.

### **3.3.2 Service Providers**

#### *3.3.2.1 Roles:*

A service provider is the individual or organization that provides services using the ecommerce web application. They might require customized specific eForms to use in offering their services. They provide product or service related data used to fill the eForm element and facilitate the various eForms tasks. They also use the eForms to perform some of their tasks (e.g. interacting with other users). They use the generic eForm elements to create customized specific eForms. The service providers decide on the functionality to be added on the specific eForms that would help in accomplishing various tasks.

#### *3.3.2.2 Interaction with other Groups*

Service providers are expected to be in constant interaction with both administrators and customers. If a service provider feels they have needs that are not currently addressed by the eForms, they can suggest to the administrators new eForm elements to be added to existing eForms entity structure. They fulfill the needs of the customers by interacting with them via the specialized eForms. They examine customer's complaints about eForms to identify need for structural improvements. They could ask for help from the administrators on any tasks related to eForms.

### **3.3.3 Customers**

#### *3.3.3.1 Roles:*

A customer is the individual or organization that consumes the services provided on the ecommerce web application. The customers use the eForms to perform most of their tasks on the web application (e.g. interacting with other users or place orders by entering text, selecting menu

items). The customers access and use products data provided by the service providers on the eForms. They provide data that is required of them by the service providers.

#### *3.3.3.2 Interaction with other Groups*

The customers interact with administrators and service providers. The customers are expected to complain about eForms and the web application if there are usability problems with the eForm. Such complaints are useful in improving the system.

### **3.4 Operations/Tasks on the objects**

Operations on the objects can also be referred to as tasks for the users. The tasks for users identify the high level operations on the set of objects. To keep a user-centered focus, operations on the objects are referred to as tasks in this section, but recognize that these tasks will later need to be implemented as operations on the objects. Tasks are specific accomplishment of individual users or a group of users using the web application system [32].

Figure 3.5 shows the relationship between tasks and user groups associated with eForms. It also shows the communication flow between different user groups that help determine users' needs on eForms. The administrators are responsible for creating, maintaining, and providing generic eForms and specific eForms.

The service providers create their own customized specific eForms based on the generic eForms or other specific eForms. Maintenance of eForm elements, generic eForms, and specific eForms involves one or more of the following: searching, creating, changing, approving, and deleting. The customers use the generic or specific eForm to perform most of their task on the application i.e. search to identify products or service providers or provide data.



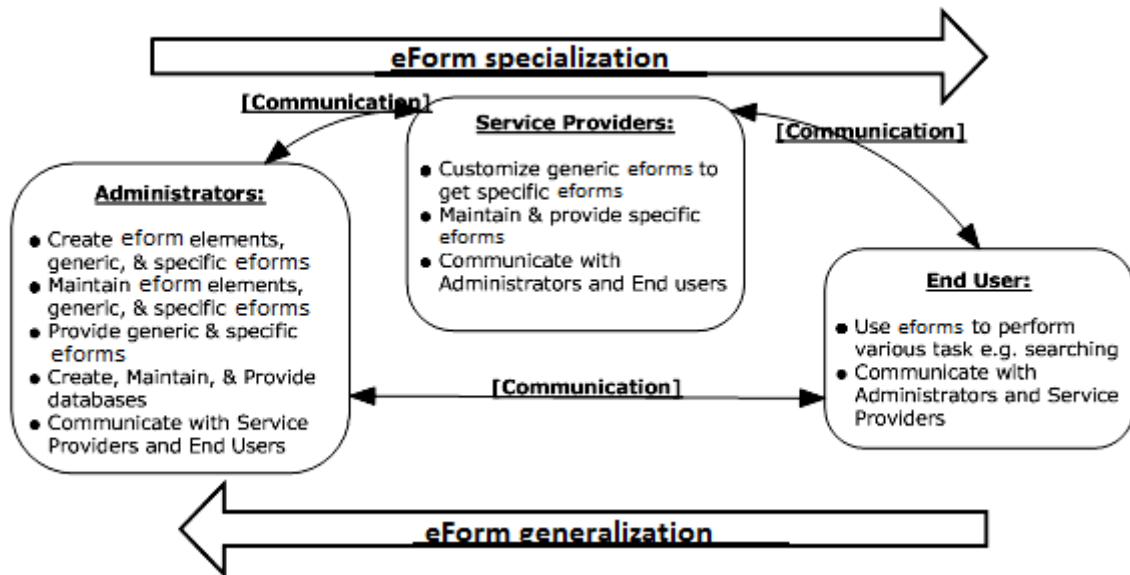


Figure 3.5: Users and tasks relationship

### 3.4.1 Maintaining EForm Elements

Figure 3.6 shows tasks that can be done to maintain eForm elements. It also shows the relationships that exist between the different tasks.

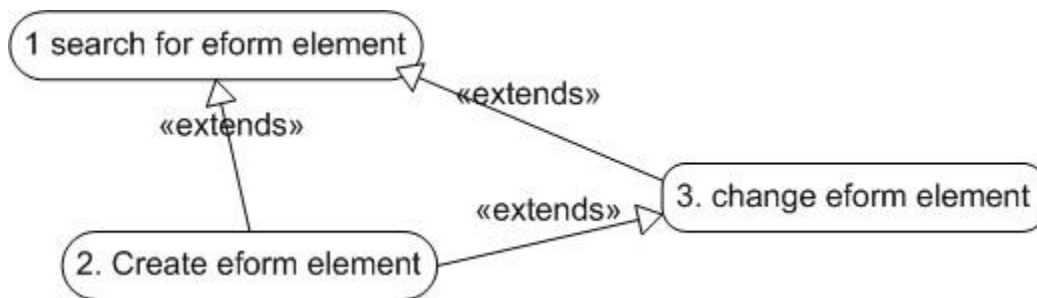


Figure 3.6: Maintaining eForm elements

#### 3.4.1.1 Searching for eForm elements

The administrators should be able to search and find existing eForm elements in the eForm element list. Searching can help determine if there is an existing eForm element or if there is need to create a new one.

### 3.4.1.2 Creating the eForm element

New eForms elements can be created by either creating a new element or cloning an existing eForm element.

#### 3.4.1.2.1 Creating new eForm element

The administrators can create an eForm element by taking an eForm control component (e.g. textbox, checkbox) and assigning a named label. The new eForm element attributes are also defined. Once the eForm element has been created, it is saved within the eForm elements list.

#### 3.4.1.2.2 Cloning existing eForm element

The administrators can also create an eForm element by cloning an existing eForm element and changing their attributes.

### 3.4.1.3 Changing the eForm elements

The administrators can change already existing eForm elements by changing one or more of their attributes.

## 3.4.2 Maintaining Generic EForms

Figure 3.7 shows tasks that can be done to maintain generic eForms. It also shows the relationships that exist between the different tasks.

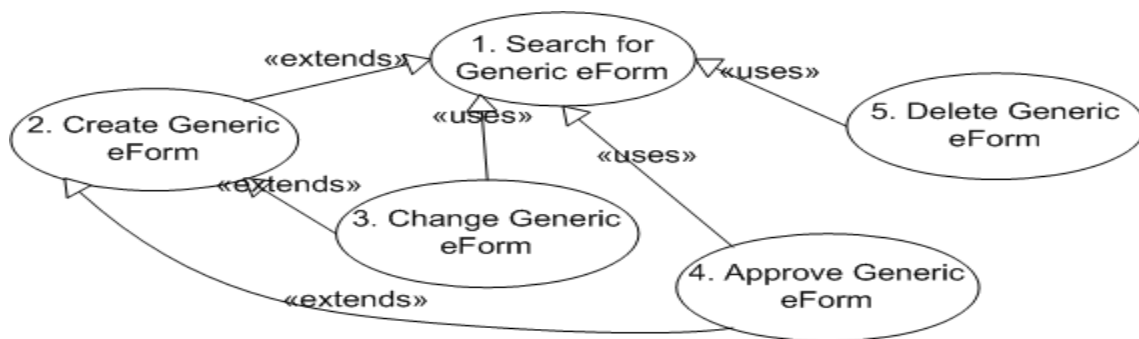


Figure 3.7: Maintaining generic eForm

#### *3.4.2.1 Searching for the generic eForms*

The administrators should be able to search and find existing saved generic eForms in their respective databases. Searching can help determine if there is an existing type of eForm or if there is need to create a new one.

#### *3.4.2.2 Creating the generic eForms*

New generic eForms can be created by either creating a new generic eForm or cloning an existing generic eForm.

##### *3.4.2.2.1 Creating new generic eForms*

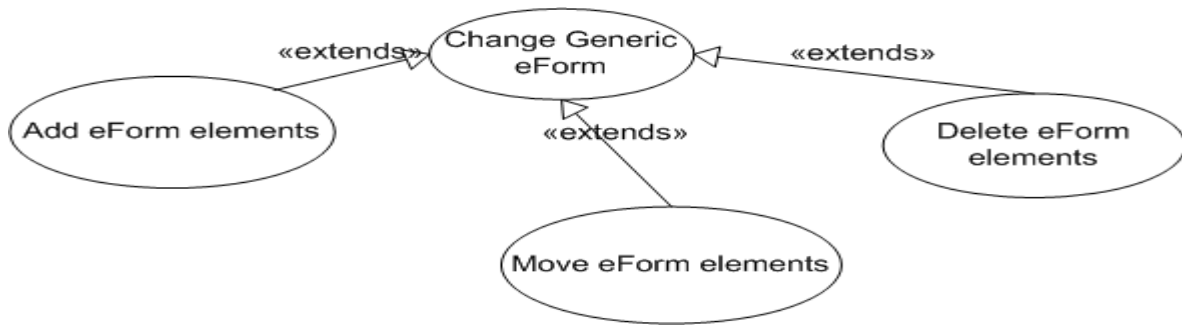
The administrators can create a generic eForm by identifying the product attributes and creating eForm elements for each of the attributes. Once the generic eForm has been created, it is saved and approved so it can be reused.

##### *3.4.2.2.2 Cloning existing generic eForms*

The administrators can also create a generic eForm by cloning an existing generic eForm. They can then rename some of the eForm elements attributes of the copy as the basis of creating a new eForm and provide a new name for the generic eForm.

#### *3.4.2.3 Changing the generic eForms*

The administrators can change an already existing generic eForm by adding new eForm elements, by moving, or deleting eForm elements within the eForm. These changes can be done to both approved and unapproved generic eForms to fulfill the needs of the tasks for which the eForms would be used. Figure 3.8 shows two possible ways of accomplishing the task of changing the generic eForms.



**Figure3.8: Changing generic eForm**

#### *3.4.2.4 Approving the generic eForms*

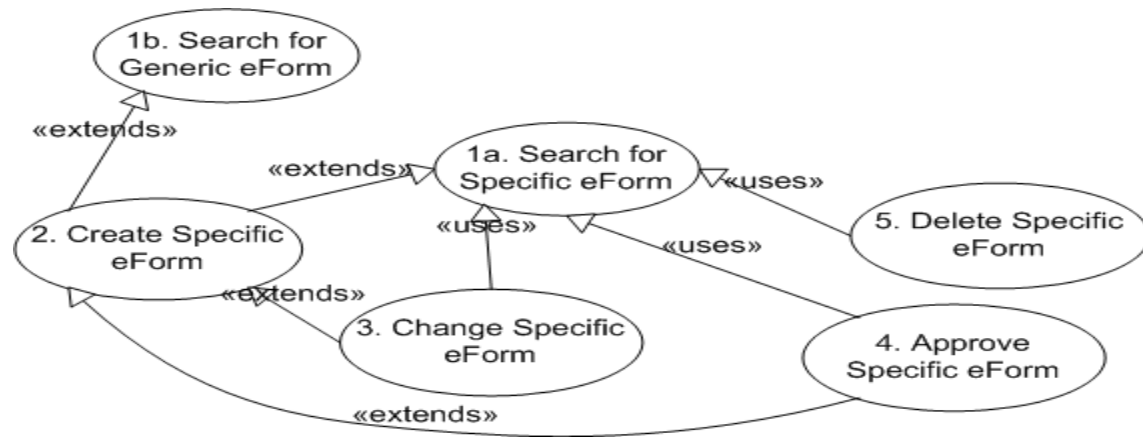
The completed generic eForms and their elements should be accessible to be used to create specific eForms or other generic eForms. Since both completed and uncompleted generic eForm are saved, the completed ones should be approved by administrators hence it can be copied and be used to create specific eForms and generic eForms.

#### *3.4.2.5 Deleting the generic eForms*

The administrators can delete a generic eForm that is no longer needed to support creation of specific eForms. A generic eForm that is considered no longer useful could be transfer into a repository database used for historical storage.

### **3.4.3 Maintaining Specific EForms**

Figure 3.9 shows tasks that can be done to maintain specific eForms. It also shows the relationships that exist between the different tasks.



**Figure 3.8: Maintaining specific eForm**

#### *3.4.3.1 Searching for specific eForms*

The administrators should be able to search and find already existing specific eForms. Searching for specific eForms can help determine if there is an existing a specific eForm or if there is need to create a new one.

#### *3.4.3.2 Creating specific eForms*

New specific eForms can be created by either customizing an existing generic eForm or cloning an existing specific eForm.

##### *3.4.3.2.1 Customizing generic eForms*

Administrators and service providers create the specific eForms by customizing the generic eForm i.e. cloning the existing generic eForm then selecting elements that they want to retain in the new specific eForm while deleting the elements that would not be needed.

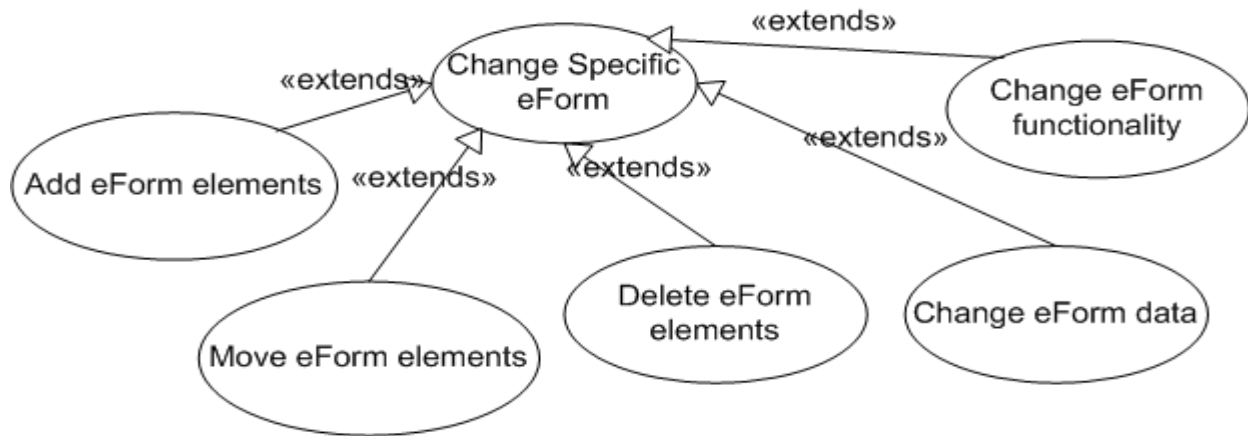
##### *3.4.3.2.2 Cloning existing specific eForms*

The administrators and service providers can also create a specific eForm by cloning an existing specific eForm. They can then delete the elements they don't want in the existing specific eForm, add new elements from the eForm elements list hence creating a new specific

eForm. The service providers can also rename eForm elements, add data, and add functionality to the new eForm.

### 3.4.3.3 Changing the specific eForms

Figure 3.10 shows possible ways of accomplishing the task of changing the specific eForms.



**Figure 3.9: Changing specific eForm**

The changes that apply to generic eForms can also apply to specific eForms: add eForm elements and move eForm elements. Their description is similar to those of generic eForms except here it applies to specific eForms (see section 3.4.2.3). In addition, three other ways that changes can be made to specific eForms are identified: delete eForm elements, change eForm data, and change eForm functionality.

#### 3.4.3.3.1 Delete specific eForm elements

The administrators or the service providers can delete eForm elements from a specific eForm if the elements are deemed not necessary in accomplishing the task the eForm is used for.

#### 3.4.3.3.2 Changing specific eForms' data

After creating the specific eForms, the service providers can add, move, or delete data that would be used by the customers in accomplishing the different tasks that the specific eForms

are used to accomplish. The service provider decides on the information content that is provided and in what way this content is presented. A change in data might include a change on eForm elements attributes.

#### *3.4.3.3.3 Changing specific eForms' functionality*

The administrators and the service providers can add, move, or delete constraints and functionality to the specific eForms. In order to accomplish some tasks, the specific eForms might require some functionality and constraints to be added to them.

#### *3.4.3.4 Approving the specific eForms*

The description of this task is similar to that for generic eForms (see section 3.4.2.4) except that it is directed to specific eForms.

#### *3.4.3.5 Deleting the specific eForms*

The description of this task is similar to that for generic eForms (see section 3.4.2.5) except that it is directed to specific eForms.

### **3.5 Attributes for the objects**

Attributes (properties) of the objects are the content that facilitates the accomplishment of various tasks on the objects. All application systems and their tasks involve the presentation and manipulation of content [32]. Most interactions between the eForms and its users are based on the content presented on eForm. The content should therefore serve the users' needs in accomplishing their tasks using the eForm.

Different chunks of content serve varying user groups in performing different tasks. Often specific user groups use specific content to perform specific tasks [32]. It is also important to note, while some content could be predefined and fixed, most content is subject to change in a variety of ways due to data processing and updating to meet the needs of its users.

### **3.5.1 EForm Elements Attributes**

#### **3.5.1.1 EForm elements identification**

The eForm elements are used to build the layout of generic and specific eForm objects. These eForm elements should have attributes that is used to identify them. The ISO 11179 “Specification and Standardization of Data Elements” standard [31] and the World Wide Web Consortium (W3C) "XML Schema" standard [11] identify the different attributes for describing elements in varying domains.

ISO/IEC 11179 [31] identifies the following basic attributes for a data element.

- *Name*: the primary identifier of an element.
- *Id*: unique identifier assigned to an element.
- *Version*: the attribute that specifies the version of the element.
- *Definition*: statement that clearly describes the element and its use.
- *Data type*: the type of data that can be represented using the element.

#### **3.5.1.2 Types of EForm control components (elements)**

The layout of most eForms could consist of one or more of the following eForm control components and their labels. These controls are based on the commonly used web applications form elements. The eForm control components used to layout the eForm is not limited to the provided list. Service providers can use specialized widget (e.g. date widget) in their eForm layout.

- *Text field*: This control is used to collect a small amount of data from the web application user.
- *Radio buttons*: Radio buttons are used when the application user is expected to choose one of a limited number of choices.



- *Checkboxes*: Checkboxes are used when the application user is expected to choose one or more options of a limited number of choices.
- *Passwords*: This control is used to collect data used for authentication of the user with either registering or logging in to use the application.
- *Buttons*: Simple buttons have no predefined behavior: the buttons can be set to call functions or trigger other behavior when a relevant event occurs.

### **3.5.2 Generic EForm Attributes**

#### *3.5.2.1 Generic eForm elements*

The creation of generic eForm layout requires the use of eForm elements. Different combinations of these eForm elements are used to create different eForm layouts. The use of consistent eForm elements would allow easy control of the eForm layout, can lead to common control elements of different specific eForms can be identified and be combined to form a single generic eForm [23].

#### *3.5.2.2 Generic eForm identification*

Multiple generic eForms are created to meet the needs of a given domain. In addition, there are tasks that need to be done on existing generic eForms i.e. cloning them, changing them, and reusing them. In order to accomplish these tasks, the administrators need to search for the relevant generic eForm to be changed or cloned. Searching is therefore important in accomplishing these tasks on the eForms. It is an essential issue to consider while determining the identification attributes of an eForm. Domain name, eForm name, and eForm id are identified as attributes that a generic eForm layout should have to help make searching easy.

- *Domain name*: This is the identifier of the main purpose and the domain for which the eForm is created for.

- *EForm id*: This would be the unique identifier given to generic eForms created by the administrators.
- *EForm name*: This is the identifier given to different types of generic eForms created by the administrators. The names could be given to eForms based on the format or the layout difference they possess. This would help distinguish eForms that have similar functionality but different layout.

### **3.5.3 *Specific EForm Attributes***

#### **3.5.3.1 *Specific eForm elements***

The specific eForms are created by customizing generic eForms (i.e. choosing needed elements to retain and rename while deleting the unwanted elements). The specific eForms like the generic eForms has eForm elements as part of their content. The control elements would be based on the context of use environment and identify what content will be displayed on the specific eForms.

#### **3.5.3.2 *Specific eForm data constraints***

Constraints and integrity elements are added at the specialization domain. The integrity of the data collected using eForms is important. Integrity includes user authentication and data validation. The service provider needs to collect correct data and from the right users. Validation helps detect errors in data input and check that correct data types are entered in their respective eForm elements.

#### **3.5.3.3 *Specific eForm purpose data***

Specialization domain identifies the information being presented on the eForms without referring to how it is presented. There are specific purpose data that might be presented on a specific eForm as helpful information in accomplishing some tasks in a given domain. Specific

eForms are used to collect data related to the specific purpose of the web application as input. Some information to be collected might require instructions on how to provide the response. Specific eForms tailored for different purpose might require different data. The specific purpose data does not change the structural looks of the eForm but only deals with content. The service providers would follow the domain rules to create data content for the specific eForms to be used by their customers in need of their services.

#### *3.5.3.4 Specific eForm identification*

In the same manner searching is important while dealing with generic eForms, it is important for dealing with specific eForms as well. Domain name, eForm name, eForm id, and service provider name are identified as attributes that a generic eForm layout should have to help make searching easy.

- *Domain name*: This is the identifier of the domain for which the eForm is created for. This name would be based on the generic or specific eForm the new specific eForm is customized from.
- *Specific eForm id*: This would be the unique identifier given to specific eForms created by the administrators.
- *Specific eForm name*: This is the identifier given to different specific eForms.
- *Service provider name*: This is the name of the service provider for which the specific eForm is created for. This service provider name would be useful in searching for specific eForms associated to a given specific service provider.

## CHAPTER 4 METADATA MODELLING AND TOOL HIGH LEVEL DESIGN

This chapter presents and discusses a way of modeling and structuring the metadata in a manner that is useful in eForm creation as a solution to the challenges identified in section 2.2. The system requirements and high level design for DeC as a tool used to manipulate the metadata is presented. The usability and acceptance of eForms on web applications is determined by their design, therefore, the process of eForm design is important.

The main goal of creating eForms and their associated databases using metadata is to provide solutions to the challenges identified in Section 2.2. As earlier mentioned in the introduction, metadata is data used to describe other data entities, making the described data entities easy to relocate and reuse [25]. The resulting eForms should be reusable, easy to manipulate, and have Domain Metadata for better search results. To illustrate this approach as a solution, metadata is used in the eForm creation process. When using metadata to create eForms and databases for a given domain, it is essential to create the metadata based on the domain description (i.e. how terminology is defined and used within the given domain context).

### **4.1 Metadata and Delk eForm Creator**

Metadata and Delk eForm Creator are essential in the design and implementation of eForms and their associated databases. In web domain, metadata is machine understandable information about web resources (data entities) [3]. The data entities being described should be based on a domain where they are used (e.g. sculptures, pictures, paintings are for art ecommerce domain). Metadata is often referred to as “data about data” [36]. Metadata generally has three broad categories [25]: the administrative, which provides information to help manage an entity; the structural, which describes the structure of entity; and the descriptive, which describe an entity for a purpose as used in the domain.

#### ***4.1.1 Advantages of using metadata***

Using metadata to create eForms, helps achieve scalability, reusability of eForms' components, and a high level of eForm consistency. The ability to control the entity structure of an eForm using metadata, allows the web application administrator to create new eForms and reuse existing ones. The metadata-based eForms are different from the custom-coded eForm applications in that, the metadata-based eForms contain declarative metadata, and they can be accessed and processed by metadata driven applications [5].

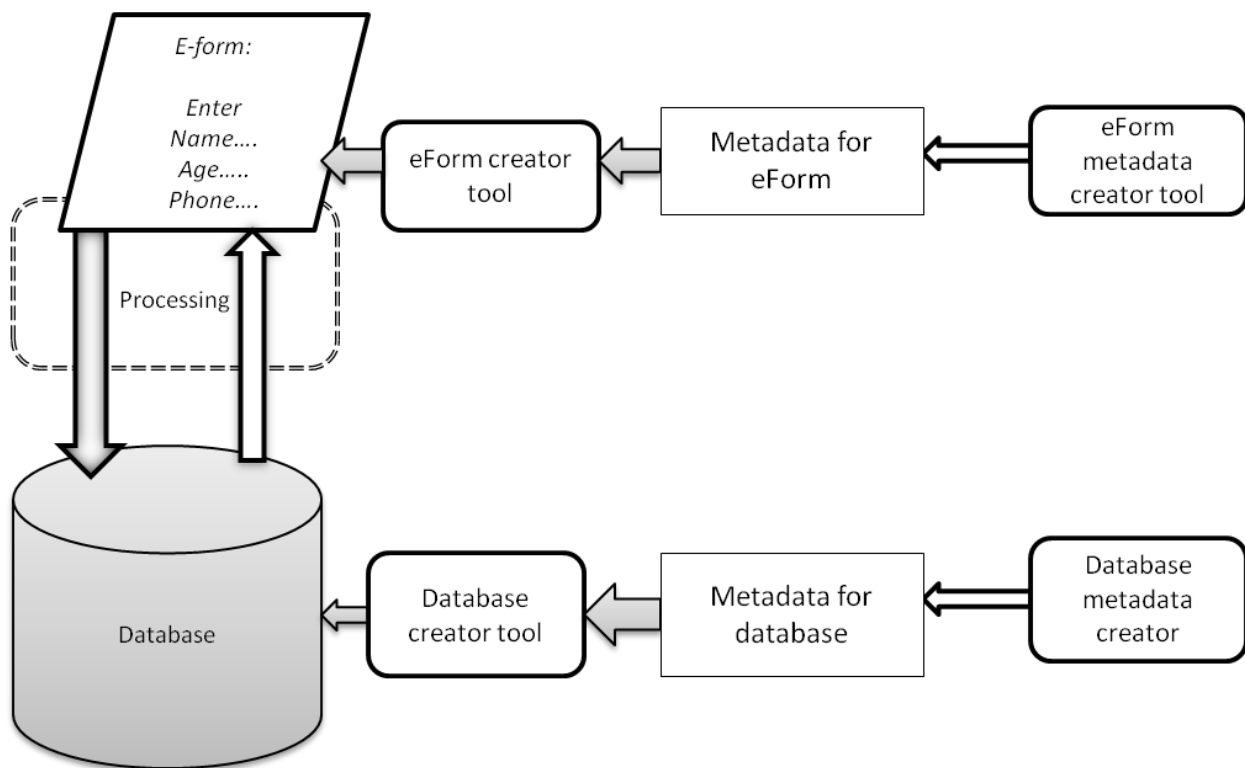
The data provided on the eForms are easily understandable by both the computer and human. This allows and facilitates information exchange between different web based applications. The use of metadata to create eForms make precise searching and comparing of such information's attributes and elements easier across different varieties of eForm entity structures used to present the information [35]. The use of metadata help users find information relatively easy, enable reuse of data, and facilitate better data management [21].

#### ***4.1.2 Use of metadata***

Using metadata for create and maintain the eForms, eForm processors, and their associated databases provides a solution to the challenge concerning vendor using different structural approaches. This necessitates the creation of metadata for domain within which the eForms will be used (e.g. art ecommerce domain). Figure 4.1 is a diagram showing how metadata can be created and used to create the eForms and their associated database.

Figure 4.1 extends Figure 2.1 by replacing programming tools and technologies used to create eForms and database with creator tools that use metadata to create the eForms and their databases. In addition, two tools that are used to create the metadata used by the creator tools are shown. It is important to provide easy to use tools to create metadata and a repository to store

metadata. The metadata used to create eForms and their databases needs to be structured such that if and when there is change on the eForms, the change is performed synchronously and with integrity to both eForms and databases. That is, whenever there is change on the eForm metadata, corresponding changes are reflected in their associated database metadata. However, in Figure 4.1, the eForm and database metadata implementation remain separate, which would place the burden of synchronizing the metadata for eForm and metadata for database on the administrator.



**Figure 4.1: EForm and database implementation using metadata**

### **4.1.3 Using Delk eForm Creator**

Using a single manipulation tool is a solution to the challenge caused by the use of multiple incompatible programming tools and technologies to create eForms and their databases. Figure 4.2 expands on Figure 4.1 by combining the four tools into a single tool (i.e. DeC) with three component tools. The tools used to create eForm and database metadata are combined into

a single component tool (i.e. metadata creator) that maintains synchronicity and integrity between eForm metadata and database metadata. The metadata creator tool would be used by the administrators to create and maintain sets of metadata used to drive the creation of eForms and their associated databases.

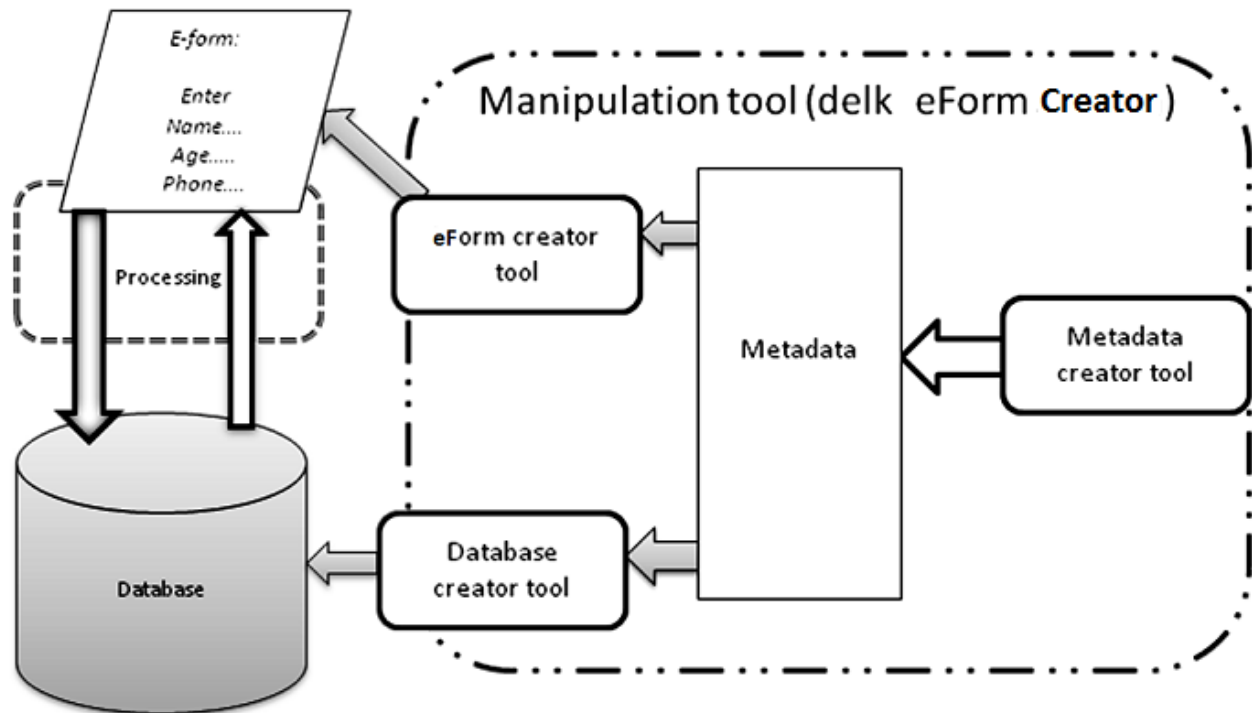


Figure 4.2: EForm and database creation using Delk eForm Creator

## 4.2 Metadata for eForms and databases creation

### 4.2.1 Use of metadata in eForm and database creation

Creating eForms and databases using metadata calls for the creation of an architecture that would be used to format eForms in a way that is both understandable to humans and computers. The process of creating eForms and databases using metadata could be augmented with the help of Semantic Web technologies. Semantic Web technologies mostly “concentrate on the definition, discovery, orchestration and execution of Semantic Web services”. More often

than not, their focus is on “system-to-system communication and less on human-computer interaction” [41].

However, in this thesis, a way to create eForms using metadata (i.e. semantic description) of an art ecommerce domain is presented. The metadata description utilizes Semantic Web technologies to provide a logic description of the domain. Semantic Web is an evolving extension of the World Wide Web in which the semantics of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines using the web content [40].

To create the metadata, it is important to use technologies that are platform independent. Technologies that can be used to create data formats; describe different kinds of data, store data, transport and share data on the web; and reduce the raising complexity of web technologies. XML is a technology with the mentioned qualities. XML can be used as a “standard for exchanging data between different systems, platforms, applications, and organizations” [39]. XML is vendor and platform independent, flexible, and can be used to format data to varying structures because new XML tags can be defined as needed.

#### ***4.2.2 Types of metadata used in a domain***

As earlier mentioned in Section 3.2.1, there are two types of eForm objects: generic and specific are identified. Their associated databases that are used to store the information gathered using the eForms is also identified. To create the generic and specific eForms using metadata, each eForm object type needs its metadata description (i.e. Generic eForm metadata and specific eForm metadata). In the same way the specific eForm are created based on generic eForm, the specific eForm metadata are created based on the Generic eForm Metadata. To create databases using metadata, there is a need for database metadata.



The idea of augmenting eForms and their associated databases creation with metadata necessitates the definition of a general metadata vocabulary to be used within a domain (i.e. Domain Metadata). Using the art ecommerce domain as an example, Domain Metadata define entities (e.g. sculptures, pictures, paintings) and provide their semantic descriptions as used within a domain in a general manner (e.g. a sculpture is three-dimensional artwork created by shaping or combining hard materials).

Based on the Domain Metadata, sets of Generic eForm Metadata, Specific eForm Metadata, and database metadata are created. This is achieved by first transforming the Domain Metadata into the Generic eForm Metadata. The transformation is done by grouping together attributes (e.g. the dimensions, the material of the sculpture) that are related to one entity an eForm is being created for. The grouped attributes are also assigned a data type (e.g. integer, string) each during the transformation process. Generic eForm metadata is therefore a structured set of entities derived from Domain Metadata geared towards concrete eForm creation.

Database metadata is be derived from the Generic eForm Metadata. In this thesis, each Generic eForm and Specific eForm will be mapped to a table within a database. Each entity attribute in the Generic eForm metadata will be transformed to associated database metadata. The varying types of Specific eForm metadata will be derived from the Generic eForm Metadata. The process will involve choosing entities that meet the specific need to create specialized eForm. The semantics and constraints of the entities can then be modified by either adding or deleting from the main eForm.

#### ***4.2.3 Capturing and representing metadata in a domain***

Based on the metadata definition in Section 4.1, metadata used to describe entities consists of a specific entity and description concepts that are based on a specific domain. In the

case of art ecommerce as example of a domain, it is essential to identify the entities and understand how these entities and their attributes are used or described within art ecommerce context before specifying the Domain Metadata. After identification, the entities, their description, and attributes have to be specified and represented in a semantic way.

Metadata for the entities can be represented in semantic way using various formatting technologies. The technologies include but are not limited to RDF [37], Dublin Core Metadata Element Set [13], and Web Ontology Language [14]. The Resource Description Framework (RDF) is a foundation for processing metadata [37], it is a W3C standard and a general-purpose language for representing information in the Web [44, 6]. RDF uses XML to exchange descriptions of Web resources of any type. The RDF data model, defines a simple model for describing interrelationships among resources (i.e. entities) in terms of two properties; attributes-properties of resources which correspond to attribute-value pairs and resource-properties which represent relationships between resources. RDF Schema (RDFS) provides a mechanism for defining the relationships between these properties and the resources. RDF Schema is a set of RDF resources (i.e. classes/entity), their properties, and constraints on their relationships.

The Dublin Core Metadata Element Set is a vocabulary of fifteen properties for use in resource description. Dublin Core elements are “broad and generic, usable for describing a wide range of resources” [13]. Ontology is a "formal, explicit specification of a shared conceptualization"[26]. Ontology is a formal representation of the knowledge using a set of concepts within a domain and the relationships between those concepts. It may be used to describe the domain and the domain properties. OWL Web Ontology Language is created for use by applications that do not only present information to humans but also need to process the

content of information [14]. Additional information vocabulary provided alongside formal semantics, allows OWL to facilitate greater machine interpretability of Web content.

For the purposes of this thesis, the main concepts of RDFS are used to help to model the concepts used with metadata. Entity is used to represent resources, entity properties to represent resources properties, and entity constraints to represent constraints attached to the resources. The entities are domain objects being modeled (e.g. products, users), while properties are description of entities' attributes (e.g. age, gender, name for users).

Three types of entity properties are identified: entity-property, data type-property, and description-property. Entity-property is a property that links different entities (classes, e.g. a customer buys a product). Data type-property is a property that links entities to data values (e.g. product's price = \$10). Description property is a property that provides a brief explanation about the entity (e.g. the product is elegant). RDFS has a similar property called comment. While RDFS uses comment-property to provide a human-readable description of a resource, the description-property will be used to aid searching.

The first step identified in this thesis is to describe the art ecommerce's domain entities, entities' description, and entities' properties (i.e. attribute value, attribute description and attribute constraints (represented as data type) in a semantic way using RDFS. The second step is to find a way of transforming the RDFS semantic description into a simpler semantic language (i.e. XSD format) as the basis for concrete eForm and database creation. This requires the transformation of the Domain Metadata to create Generic eForm Metadata (i.e. transform RDFS to XSD). The transformation process could be achieved by using the XSL technology (i.e. an XML based language used to transform XML files to a desired file type). Creation of new specific eForm metadata from the Domain Metadata should trigger creation of corresponding

database metadata. The creation of database metadata and concrete database is automated; the database elements (e.g. column) are added to the database metadata. Any changes to the existing specific eForm metadata should be reflected in the corresponding database metadata and the actual databases.

The next step is the creation of a concrete generic eForm entity structure that is rendered by the web browsers. This involves transforming the XSD by adding the relevant eForm elements the Generic eForm Metadata. XForms is a W3C recommendation web technology [35] that can be used to create and structure web forms using XML. XForms is based on XML [17]. XForms acts as markup language for eForms and was designed to be the next generation of HTML forms. It is used for specifying eForms in a way that separates eForm use, eForm presentation, and eForm data, clearly separating the content, structure and user input [42, 39]. The separation of content and presentation allows for reuse [4] of the eForm. For a detailed pre-prototype example with code see appendix II.

### **4.3 Using Delk eForm Creator to create eForms and databases**

#### ***4.3.1 System requirements***

In this section, the analysis of system requirements for Delk eForm Creator (DeC) are introduced and discussed. This tool supports the creation of metadata for eForms and metadata belonging to art ecommerce domain. In order to support the projects from the different domains, DeC qualities include and are not limited to the ones identified below:

- It provides a web-based user interface for interaction with domain administrators and service providers.

- It has the ability to support the operations involved in creating and maintaining Domain Metadata, generic and specific metadata, database metadata, and concrete eForms (i.e. generic and specific).
- It provides a storage database used to save the created eForms and databases.
- It has a searching mechanism that enables the administrators and service providers to access the saved metadata and eForms.
- It provides a mechanism that allows the administrators and service providers to integrate the created eForms to a web application.

The creation of eForms and their associated databases using DeC includes one or more of the following operations: creating new, searching, or changing (modifying) of metadata or eForms. The metadata and eForms created are saved in storage databases. The storage database is different from the databases generated by the using metadata. Searching is done against the storage database to check if metadata or eForm for a given issue within a given domain has been created or not. In addition, the tool creates a script that enables communication and data transfer between the created eForms and their databases.

Figure 4.6 expands on by modifying the metadata creator tool component into two components: Metadata Creator and eForm Metadata Creator. Further expansion includes the addition of the Script Creator. This component is deemed necessary because it will be used to create the eForm processors that would enable data processing and communication between the created eForms and their databases. Figure 4.3 to shows an overview of the proposed metadata based eForms and database creation process.

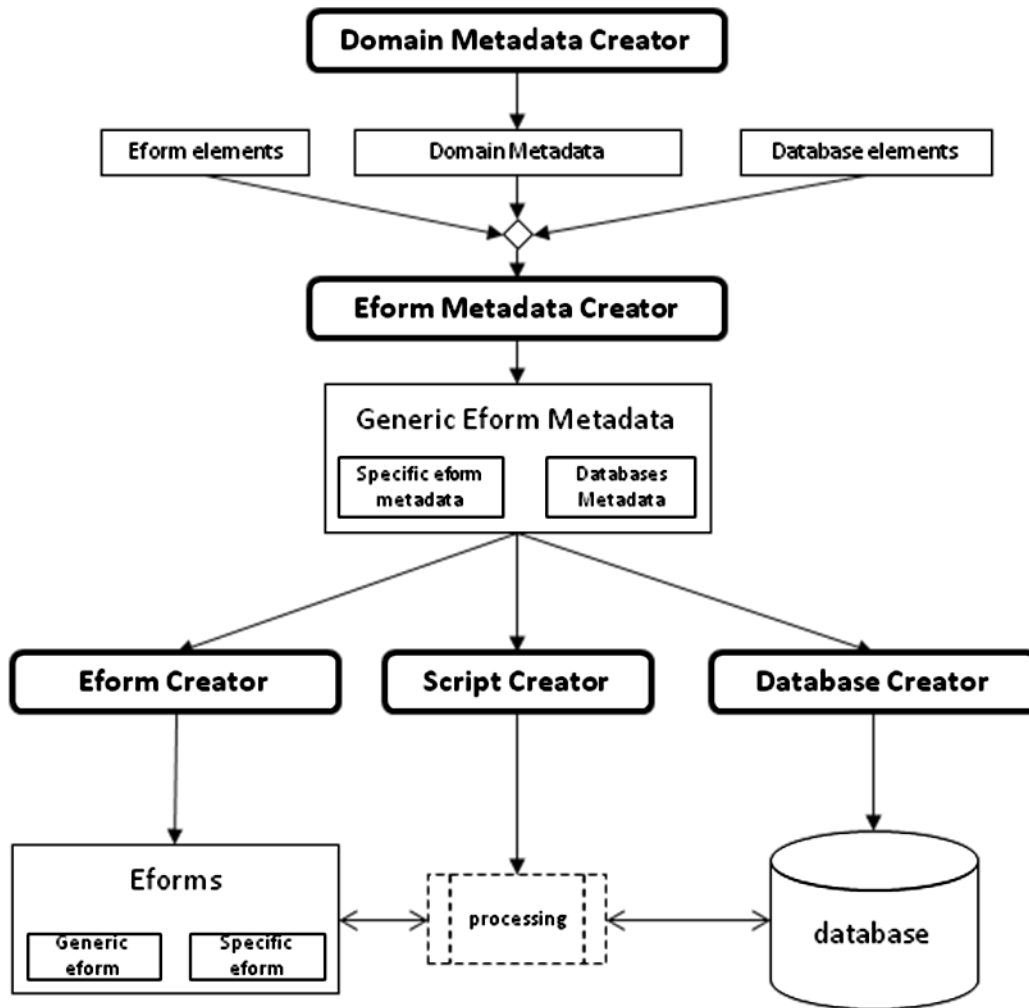
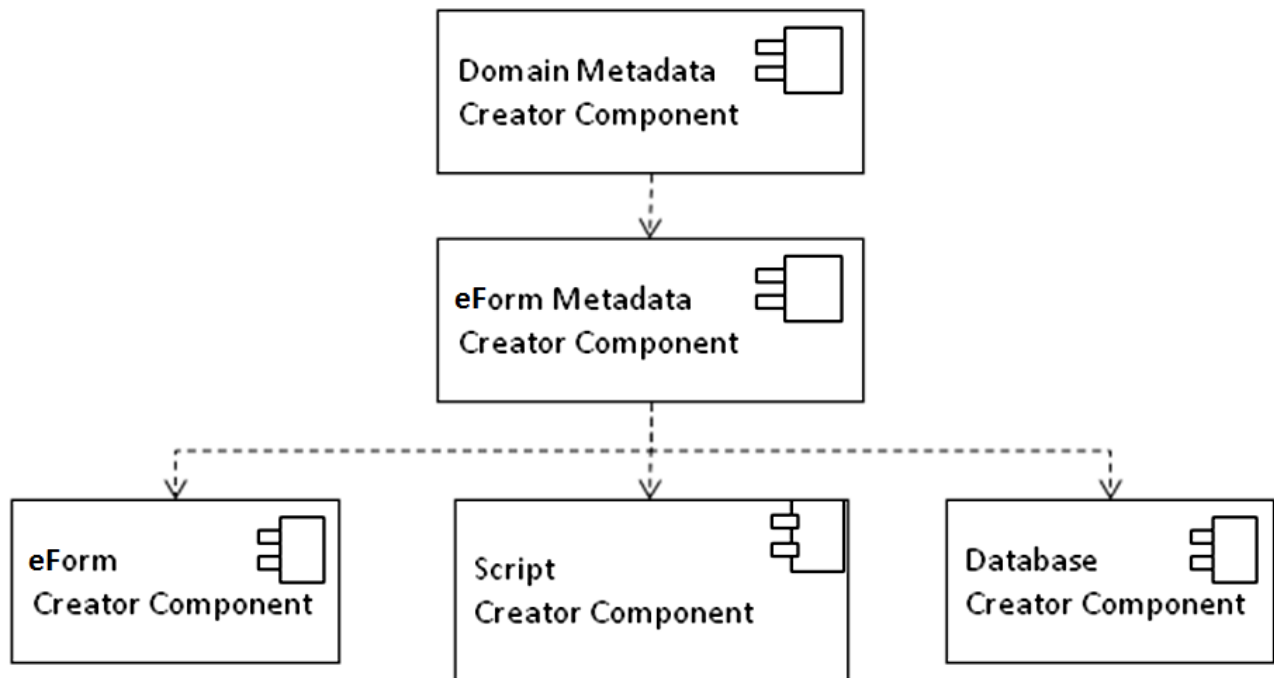


Figure 4.3: EForm and database metadata creation flow

#### 4.3.2 System components

DeC is implemented around five major components as shown in Figure 4.7. To support its functionalities, addition five supportive components are also implemented. These components are modeled based on Figure 4.6 and the qualities of the tool identified in section 4.3.1. The major components are; Domain Metadata Creator, eForm Metadata Creator, eForm Creator, Database Creator, and Script Creator components.



**Figure 4.4: DeC major components**

#### *4.3.2.1 Domain Metadata Creator component*

The Domain Metadata Creator component is only accessible by administrators. As the name suggests, this component is used to create Domain Metadata that is described and formatted using RDFS as described in section 4.2.3. The Domain Metadata is created by identifying and specifying all the attributes of entities in a domain. This name, data type, and domain based description of the attributes is needed to create Domain Metadata. After the Domain Metadata has been created, they are available to the eForm Metadata Creator component. The Domain Metadata Creator component allows the saving of created Domain Metadata, the saved Domain Metadata could be searched, retrieved, and be modified. The input for the Domain Metadata Creator component is the domain description (i.e. entity attribute name, data type, and description) entered by the administrator. The output is a list of domain based attributes (Domain Metadata) represented and formatted using RDFS.

#### *4.3.2.2 EForm Metadata Creator component*

The eForm Metadata Creator component is accessible by administrators and service providers. This component creates and maintains new and existing copies of Generic eForm Metadata (i.e. XSD) and database metadata. The input for the eForm Metadata Creator component is the Domain Metadata from the Domain Metadata Creator. The outputs are the Generic eForm Metadata and database metadata.

After the Generic eForm Metadata have been created, they are available to the eForm Creator and the Database Creator components. The eForm Metadata Creator is also used to create specific eForm metadata based on the Generic eForm Metadata. The eForm Metadata Creator component allows the saving of generic and specific eForm metadata, the saved Domain Metadata could be searched, retrieved, and be modified..

#### *4.3.2.3 EForm Creator component*

The eForm Creator component is accessible by administrators and the service providers. The eForm Creator uses the eForm elements identified in Chapter 3 to create the generic and specific eForms. A Generic eForm is created by combining a product's attributes with the appropriate eForm elements. The attributes are selected from the list defined as Domain Metadata. The eForm Creator can also be used to customize Generic eForms to create Specific eForms by deleting the attributes that are not deemed require in a Specific eForm or adding specific data to the eForm. Eform creator can also be used to enter fixed data on the eForms (e.g. default data). The inputs for the eForm Creator component are the generic and specific eForm metadata from the eForm Metadata Creator and eForm elements. The outputs are the generic eForm and specific eForms. The eForm Creator component would be used to update, edit, change, and add content and structure to generic and specific eForms.



#### *4.3.2.4 Script Creator component*

The script creator component creates the eForm processor (script) that will enable the communication, the processing, and transfer of data between the created Forms and databases. The inputs for the Script Creator component are eForm and database metadata from the eForm Metadata Creator. The output is a script that provides a connection between the eForms and databases created based on the metadata.

The Script Creator produces script files which have Create, Read, Update and Delete (CRUD) methods, i.e. the four basic functions of persistent storage [27, 73]. These four basic provides ways to a set and manipulate data in the databases when using the created eForms and databases. CRUD supports all major functions that need to be implemented in a relational database application to consider it complete. Each CRUD function can be mapped to a standard SQL statement: (Create: INSERT, Read (Retrieve): SELECT, Update: UPDATE, Delete (Destroy): DELETE).

#### *4.3.2.5 Database Creator component*

This component uses the database metadata from the eForm Metadata Creator and creates a database that will hold the data for the eForm. The input for the Database Creator component is the database metadata from the eForm Metadata Creator. It is essential to automate functions of this component so that changes made to the Generic eForm Metadata are reflected in their associated databases. The input has the constraint included in the Domain Metadata. The constraints could be modified while creating Generic eForm Metadata; modifications are automatically reflected on the database metadata. The output is a concrete database. The Database Creator will operate like a Database Management System (DBMS). It will control the creation, management and use of the database storage structure.

#### *4.3.2.6 User Interface component*

The user interface component will provide the graphical user interface that would enable the user (i.e. administrator and service provider) to perform all the operations on the system, access and interact with some DeC components (i.e. Domain and eForm Metadata Creators and Integration components). It will also support an authentication mechanism which will help identify if the administrator or service provider is logged in.

#### *4.3.2.7 Database component*

DeC has databases for storage. The currently identified databases are: Domain Metadata DB, eForm metadata DB, eForm DB, and users DB. The inputs for the Database component are any data that need to be saved (i.e. eForms, metadata, or user information) created by any of the DeC component. The output is save storage of the data. These will be used as follows:

*Domain Metadata DB:* used to store the Domain Metadata created by the Metadata Creator.

*EForms Metadata DB:* used to store Generic eForm Metadata and specific eForm metadata created by the eForm Metadata Creator.

*EForms DB:* be used to store concrete generic eForm and specific eForm created by the eForm Creator.

*Users DB:* used to store the users of the delk eForm creator, i.e. the administrators and the service providers.

#### *4.3.2.8 Database Connection component*

The database connection component allows the communication between the front end (i.e. user interface) and the back end (i.e. database) of the delk eForm creator. The inputs for the Database Connection component are the username, password, and database name to connect to; data provided by the administrator or service provider. The process is automated, so the data can

be fetched and connection be established when the user logs in. The output is an established connection to the database.

#### *4.3.2.9 Search component*

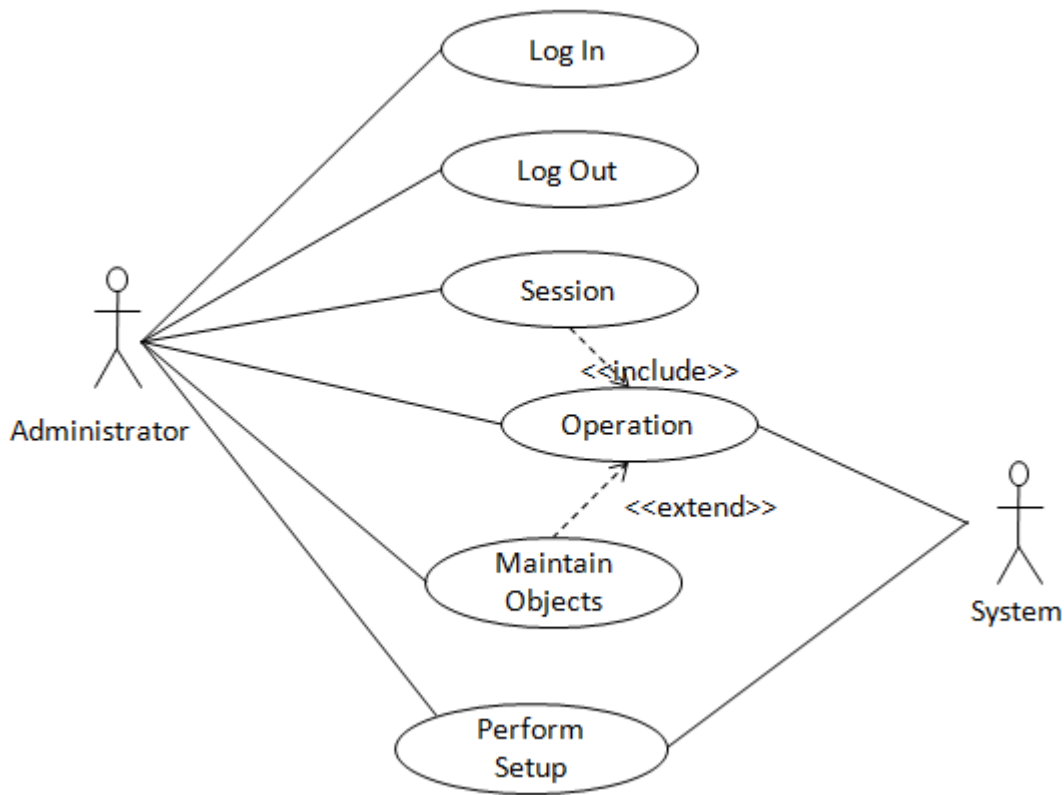
The search component is used to search the database for existing domain, generic or specific eForm metadata. It is used to search saved generic and specific eForms that an administrator or a service provider might be interested to reuse. This component supports the administrators and service providers in searching the following databases: Domain Metadata, eForm metadata, eForm, and users. The input for the Search component is some phrase from the administrator or service provider. The outputs are metadata or eForms associated with the phrase or null result.

#### *4.3.2.10 Application Integrator*

The application integrator component is used to setup and integrate (incorporate) the created eForms and their databases within a web application. The inputs for the Application Integrator component are the eForms, scripts, and their databases from the eForm Creator and server information for the web application hosting. The output is an eForm being integrated within a web application and database being setup.

### **4.3.3 System use cases**

The eForm creation involves the creation and maintenance of Domain Metadata, eForm metadata, eForms, eForms elements and their associated databases. Figure 4.8 is a diagram that shows the general use case for creation of eForms and databases. (UML diagrams use a stick person to represent any interactive entity within the use case, including both humans and the system.) This section describes briefly some general high level use cases. More detailed use cases are presented in Chapter 5.



**Figure 4.5: General use for creating eForms**

#### *4.3.3.1 Log in use case*

At the start of the metadata manipulation operations, the administrator will be required to log in by entering a user name and a password, both of which will be sent to the database for validation. A successful log-in operation starts a session. The administrator will then be able to perform one of more operations using the system.

#### *4.3.3.2 Log out use case*

The session will terminate when the administrator logs off.

#### *4.3.3.3 Perform setup*

This use case will enable the administrator to setup the eForms created with the web applications. The administrator will be required to provide the servers authentication information.

#### *4.3.3.4 Session use case*

If the administrator's log in data is valid, a logged in session for the user will be created. This session will be maintained until the administrator logs off. Else if the administrator's log in data is invalid, a message is displayed requiring the administrator to re-enter the data before a session can be started. While the session is on, the system should allow the administrator to perform the following operations. Maintain Domain Metadata, maintain eForm metadata, maintain eForms, maintain database metadata and maintain databases.

#### *4.3.3.5 Operation use case*

The operation use case is an abstract generalization of the specific operations use cases that the administrator can perform using the system. Each specific operation implements the administrator's tasks in an appropriate way. The operations use case includes the search, create\_New, and modify use cases. The operation use case is associated with the object maintenance use case. An operation use case is started within a session when the administrator chooses a specific object maintenance use case from a menu of options.

#### *4.3.3.6 Object maintenance use case*

The object maintenance use case is an abstract generalization of specific object maintenance use cases that the administrator can perform using the system. A specific object maintenance use case implements one or more of the operation use cases on a specific object (e.g. maintain domain eForm implements operations on domain eForm). The object maintenance use case describes a general common event flow for the different operations. The flow of events for each individual type of use case gives detailed operations that are specific to that use case.

The object maintenance use case is started within a session when the administrator chooses an operation type from a menu of options. The administrator accesses the appropriate

user interface generated depending on the chosen option from a menu (e.g. the user interface to create Domain Metadata if the administrator choose ‘maintain Domain Metadata’ from the menu option). The administrator should be able to abort the use case in progress by pressing the Cancel button.

The object maintenance use case will be considered complete once it has been approved. In the case of creating Domain Metadata, a Boolean value in the table storing Domain Metadata in database is changed and an email is send to the administrator to confirm the operation. If an operation fails for any reason, the system will display a message related to the problem. The administrator is then asked whether they want to do another operation.

## CHAPTER 5 TOOL DETAILED DESIGN AND IMPLEMENTATION

This chapter describes the detailed design for Delk eForm Creator. It expands on use cases described in section 4.3.3 by providing descriptions for user interfaces used to fulfill the needs of the use cases. Description of class implementation and responsibilities that support functionality of the major system components identified section 4.3.2 is also provided. Further description of the database implementation is provided.

Considering all the features and components identified in the analysis and high level design (i.e. chapter 3 and chapter 4 respectively), Delk eForm Creator (DeC) is a complex system. For the purposes of this thesis and proof of concept most but not all of the identified features are implemented. As earlier identified in the introduction section of Chapter 4, the main goal for using metadata is to create eForms that are reusable, easy to manipulate, and encapsulate Domain Metadata that is expected to aid data searching.

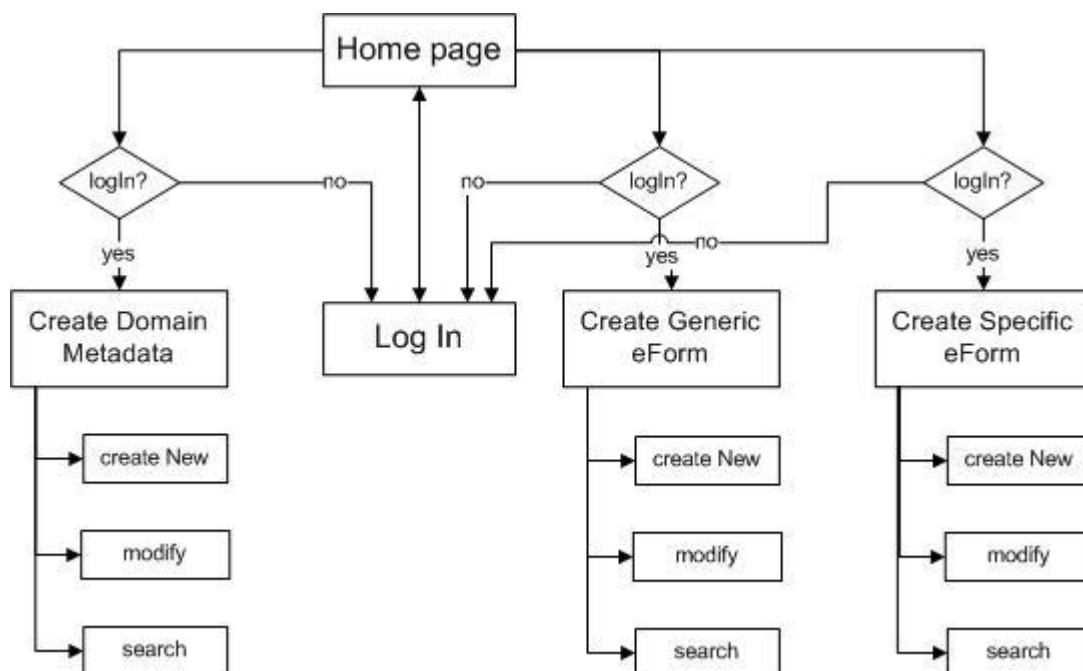
DeC is used to create simple eForms. Its functionality and use illustrates that use of metadata is a possible solution to the issues identified in section 2.2. DeC produces eForms that could be used for user interface, data display, and data gathering. The implementation is based on 3 tier architecture. The presentation (user interface) tier, application (functional) tier, and data (database backend) tier.

### **5.1 Delk eForm Creator Use cases and their interfaces (presentation tier)**

This section describes the presentation tier (user interface). This presentation tier provides a communication interface that enables the user of DeC to manipulate, navigate, and use the functionalities provided by the other tiers of DeC tool as a whole. The following implementations of user interfaces are for the uses cases described earlier in section 4.3.2. Figure 5.1 shows the general user interface layout.

The Home Page is the default interface that is first accessed whenever DeC is started. From the Home Page the user is able to navigate to four interfaces that would enable them to manage the eForm objects identified in Section 3.2 and expanded in Section 4.2.2. The four interfaces are Create Domain Metadata, Create Generic eForm, Create Specific eForm, and Log In. The user is able to access 'Log In' directly from the default interface. In order to access the 'Create Domain Metadata', 'Create Generic eForm', and 'Create Specific eForm', the user will have to be logged-in.

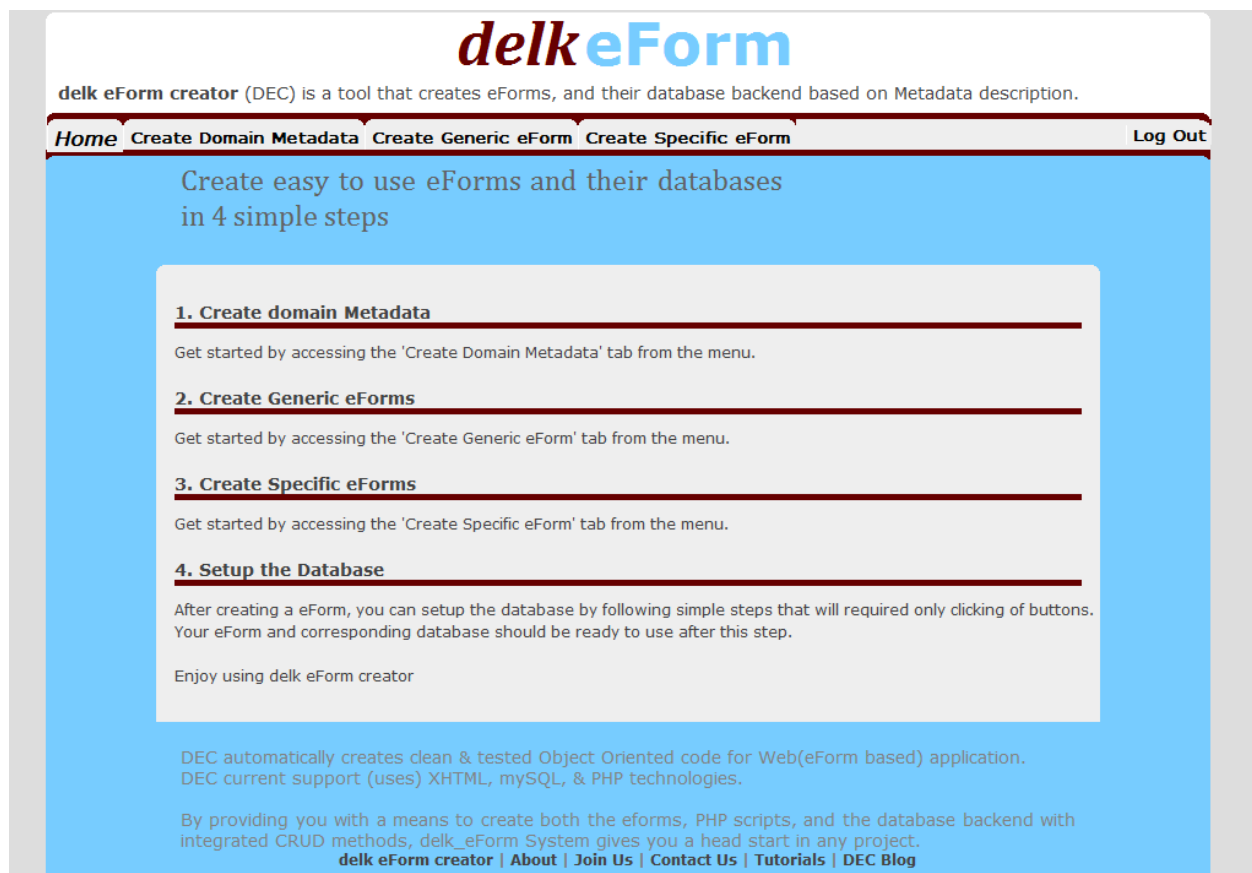
In case the user is not logged-in and they tried to access any of the other interfaces, they will be redirected to the 'Log In' interface. The log-in information is provided by the administrator who is in charge of user registration. From each of the three interfaces (Create Domain Metadata, Create Generic eForm, and Create Specific eForm), the user is able to access the operations they can perform on that specific interface. The operations accessible are: create\_New, modify, and search.



**Figure 5.1: DeC general User Interface Layout**



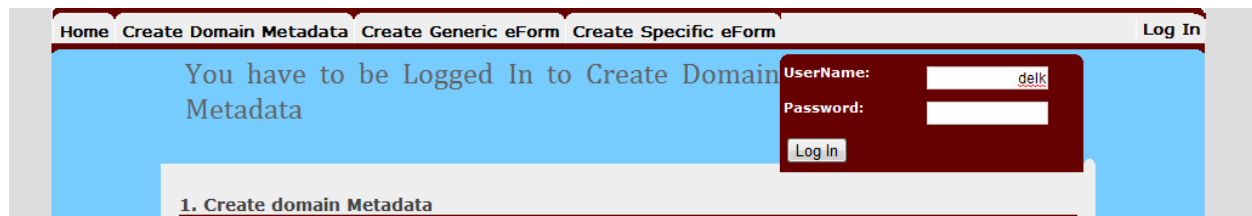
Figure 5.2 shows the default (Home Page) interface. This is the first interface the user encounters when the system starts. The Home Page interface provides a layout that enables the user to choose which eForm object they want to work on. The three eForm objects the user can work on are the Domain Metadata, Generic eForm, and Specific eForm. In Section 3.4, six operations on each eForm object were identified (i.e. search, create\_New, clone, change, approve, and delete). For the purposes of this thesis, three of six operations that are implemented are create\_New, modify, and search. The rest of the identified operations (clone, approve, and delete) will be implemented before the tool is deployed.



**Figure 5.2: DeC Default Interface**

Figure 5.3 shows the log in interface within the default home interface that implements log in use case described in section 4.3.3.1. The Log in interface is activated when the user clicks

on the log in tab on the Home Page or when the user tries to access the other interface and they are not logged-in. Successful log-in starts a session use case and takes the administrator to the Home Page (see Figure 5.2). The user has to be logged-in in order to use the DeC tool, therefore this Log in interface is needed.



**Figure 5.3: DeC Log-In Interface**

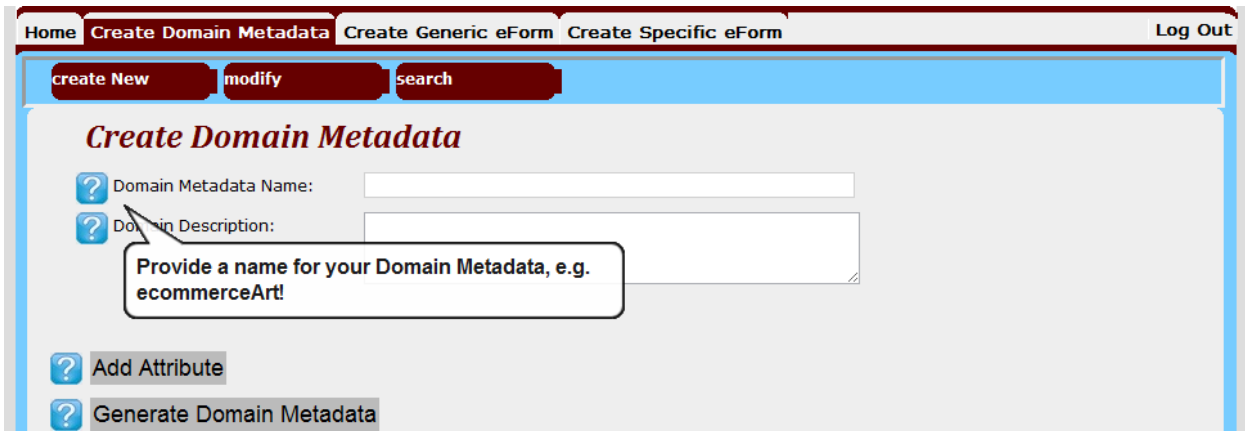
Some common features for the all user interfaces are the operation menu and instance help buttons. Just below the main menu that allows the user to navigate and work between different eForms objects, are buttons that allows the user to navigate between the operations they can do on the eForm object (i.e. create\_New, modify, and search). Instant help that explains the element functional description and shows an example are provided by using a question mark image as shown in Figure 5.4. When a user moves the mouse over the question mark image, a balloon containing the help message is displayed. This is useful when a user needs help on the functionality of the element.

### **5.1.1 Create Domain Metadata interface**

This interface allows the administrator to carry out the following activities supported by Delk eForm Creator prototype: search for, create\_New, or modify Domain Metadata. This section focuses on create\_New activity for Domain Metadata. For more information on search and modify activities see appendix V.

Figure 5.4 shows an interface an administrator is presented with when they access the Create Domain Metadata interface and select create\_New from the drop-down menu. This

interface provides text eForm elements called the Domain Metadata Name that the administrator enters the name of the Domain Metadata (e.g. ecommerceArt). The interface also provides a text area eForm element called Domain Description where the administrator enters a description on the domain the attributes are created for (e.g. Information related to art products sold via internet/online shops).



**Figure 5.4: Default DeC Create Domain Metadata Interface**

Figure 5.5 shows Create Domain Metadata user interface with some attributes added. A Domain Metadata without any entities cannot be created. To add the first entity, the interface provides a mechanism to add new attributes to the identified list by using the Add Attribute button. The interface also provides text eForm elements to add data about the attribute name (e.g. price), attribute's data type (e.g. decimal), and attribute's description (e.g. the price of the item).

The tool provides and populates the dropdown menu with data types supported by MySQL databases. Specifying data type is important for data validation when using the eForm that would be created based on the Domain Metadata. The attribute description provides more information on how the attribute is used within a specific domain which might be different from how the same attribute is used in another domain. To remove or delete any of the added attributes, the interface provides a button represented by the recycle bin image on the left side of

the attribute. To trigger the tool to create the Domain Metadata; the interface provides the Produce Domain Metadata button.

**Figure 5.5: Using DeC Create Domain Metadata Interface**

This section provides a step by step instruction on how to manage Domain Metadata. For a detailed pre-prototype example with code see appendix II.

#### *5.1.1.1 Create New Domain Metadata*

1. The first step is to access DeC Home Page, see Figure 5.2.
2. The next step is to click on the Log In tab on the right side of the page. An interface with a log in interface is presented as a result of this step (2), see Figure 5.3.
3. Next is to enter the username and password into their respective text input boxes on the interface, then press Log In button (the administrator should be registered as a user to have

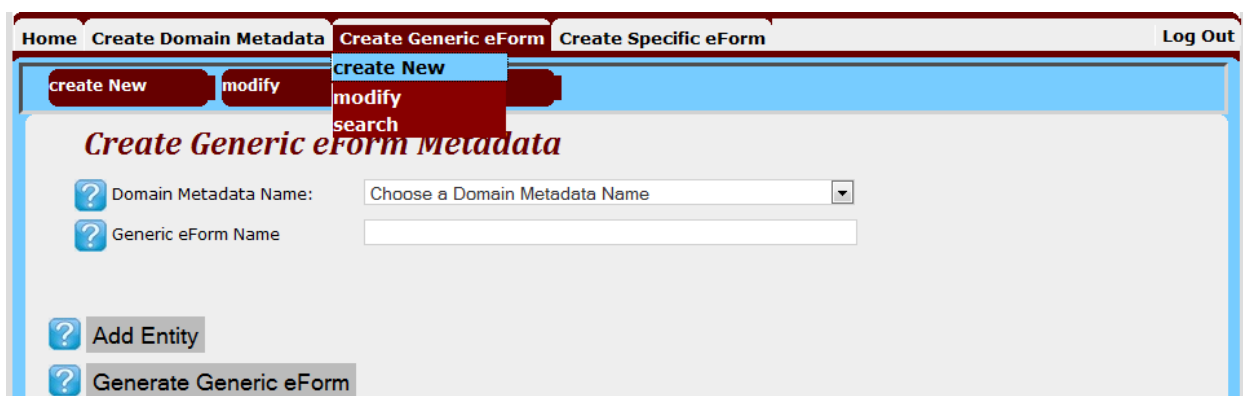
log-in information). The Home Page will re-load if the log in process was successful, if not, the administrator will be asked to re-enter the log-in information.

4. To create a new Domain Metadata, the administrator has to click on the Create Domain Metadata at the top of the interface, and then choose create\_New from the resulting dropdown menu.
5. The resulting interface from step 4 is the interface that will allow the administrator to enter the content of the Domain Metadata. Note from this interface the administrator can access other operations (i.e. modify and search) Domain Metadata.
6. Next step is to specify the name of the domain in the Domain Metadata name section (e.g. ecommerceArt). Provide domain description.
7. A Domain Metadata without any attribute cannot be created. To add the first attribute, the administrator clicks on the Add Attribute button. When this button is pressed, an interface that allows the administrator to the enter attribute name is provided.
8. The attributes would have name (e.g. name, age), data type (characters (i.e. char (255)), integer (i.e. Int) and their descriptions. To choose the appropriate data type for the attribute, the administrator chooses one data type from the drop-down list offered. The tool provides and populates the dropdown menu with data types supported by MySQL databases. Specifying data type is important for data validation when using the eForm that would be created based on the Domain Metadata.
9. It is expected that each Domain Metadata would have more than one entity attribute. To add more entities attributes repeat steps 7- 8. To remove or delete any of the added attributes, the administrator clicks on the button represented by the recycle bin image on the left side of the attribute.

10. When the administrator is done adding entity attributes, they click on the Generate Domain Metadata button. When this button is pressed, the system takes the data entered in the interface and transformed them into Domain Metadata represented and structured as RDFS as specified in the requirements (refer to capturing and representing metadata section 4.2.3).
11. Creating a new domain is a onetime process. Addition and modification of domain's properties (entities) can be done multiple times (see modify Domain Metadata in appendix V).

### 5.1.2 Create Generic eForm interface

This interface allows the user (i.e. administrator or the service provider) to carry out the following activities supported by Delk eForm Creator prototype: search for, create\_New, and modify. This section focuses on create\_New activity for Generic eForm for more information on search and modify activities see appendix V.



**Figure 5.6: DeC Create Generic eForm Interface**

Figure 5.6 shows an interface a user is presented with when they access the Create Generic eForm interface and select create\_New from the 'Create Generic eForm' drop-down menu. This interface provides a drop-down select option called Domain Metadata Name that is populated with the names of Domain Metadata already created. The user is to select one Domain Metadata (e.g. ecommerceArt) as the basis for creating a Generic eForm. This interface also

provides a text eForm element called Generic eForm Name that the user enters a name for the Generic eForm (e.g. artistModel).

When the user chooses a Domain Metadata as the basis for creating Generic eForm and clicks the Add Entity button, the interface is populated with all the attributes defined within chosen Domain Metadata as shown in Figure 5.7. The interface provides a mechanism to customize the entities and their attributes based on the eForms elements identified earlier in section 3.5.2. The interface also provides text eForm elements for entering the entity name (e.g. painting, drawing, and artist) for the Generic eForm creation. The interface provides a mechanism to customize the attributes for each entity added using the delete button represented by the red image with the 'minus' sign.

**Create Generic eForm Metadata**

Domain Metadata Name: ecommerceArt Domain

Generic eForm Name: sellerInfo

Entity Name: Artist

Attribute:	Data_Type:	Label:	Control:	Field_Options:	Description:
Name	Char(255)	Davidson	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No	The identifying property for the customer, seller, or the art work
Material	Varchar(255)	Material specialty	Check Boxes	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No Options: paper <input type="checkbox"/> cloth <input checked="" type="checkbox"/>	The material used to create the art work
Address	Varchar(255)			Required? <input checked="" type="radio"/> Yes <input type="radio"/> No	The physical location of a customer, seller, or the art work
Email	Varchar(255)			Required? <input checked="" type="radio"/> Yes <input type="radio"/> No	Electronic contacts for the customer or a seller
Phone	Varchar(255)		Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No	The voice contacts for the customer or the seller

Figure 5.7: DeC Interface Populated with Domain Metadata attributes

The interface provides a text eForm element called Label for each attribute to add label data (e.g. Name) displayed when rendering the Generic eForm. The interface also provides a mechanism to add the eForm elements (e.g. text box, input, or button) and submission element to the Generic eForm. The eForm elements are provided on the interface in a drop-down select element called Control. The eForm element selected affects how the Generic eForm will be rendered and the Field Options. For example if a check box eForm element is chosen, the Field Options provides an interface that allows the user to enter the different options related to the attribute. The Field Options also allows the user to specify whether the attribute will be required or not. This will affect the functionality of the Generic eForm, that is, if the attribute is assigned required, then the eform cannot be submitted with the specified attribute as empty.

As the elements of the Generic eForms are specified, the Generic eForm Metadata is simultaneously created. The creation of the Generic eForm Metadata from the Domain Metadata triggers an automated creation of corresponding database metadata and script file with CRUD methods described earlier in section 4.3.2.4. Any changes to the Generic eForm Metadata should trigger an automated change to the database metadata.

The database elements (e.g. column) are added to the Generic eForm Metadata and represented as SQL commands (see appendix II Figure 4). The sizes of the different elements are allocated by default (e.g. varchar is defaulted to size 255). Each element in the XSD file is transformed into an element of SQL statement. For example, artist element is transformed into ('artist' CHAR (255) NOT NULL). Each data input eForm element is expected to map to a given column of a table in a database. For example, text boxes for inputting name on the product eForm will be mapped to a column for storing names in a table (product-table) within a database.



This section provides step by step instruction on how to create Generic eForm and their Metadata. For a detailed pre-prototype example with code see appendix II.

#### *5.1.2.1 Create New Generic eForm Metadata*

1. The first step is to access DeC Home Page.
2. If the user is logged in, then the next step is 4. Else the next step is to click on the Log In tab on the right side of the page. An interface with a log in interface is presented as a result of this step (2)
3. Next the user enters the username and password into their respective text input boxes on the interface, then press Log In button (the user should be registered as a user to have log-in information). The Home Page will re-load if the log in process was successful.
4. To create a new Generic eForm, the user clicks on the Create Generic eForm then choose create\_New from the dropdown menu.
5. The resulting interface from step 4 is the interface that will allow the user to choose the Domain Metadata to base the Generic eForm on. The names of the Domain Metadata created are presented using a dropdown menu from which the user can choose. Note that from this interface the user can access other operations (i.e. modify and search) Generic eForm Metadata.
6. The next step is to specify the name of new Generic eForm (e.g. artistModel).
7. A Generic eForm without any entities cannot be created. To add the first entity, the user chooses the Domain Metadata to base Generic eForm creation on. When this is done, the attributes specified for the Domain Metadata chosen populate the user interface.
8. Next is to specify the Entity name (e.g. seller) and customize its attributes by deleting the unwanted attributes.

9. If there is need to delete some of the entity's attributes, the attribute to be deleted is identified and the red button with the "minus" sign is pressed.
10. The user then specifies the label name for each attribute.
11. For presentation purposes, the user has to also specify the eForm element type (e.g. radio button, text box) from the drop-down menu provided. When a user chooses eForm element types e.g. radio button or check boxes an interface to specify data options that is related to the given attribute.
12. To add more entities to the Generic eForm, the user repeats steps 7-11.
13. When the user is done adding entities, the user clicks on the Generate Generic eForm button.  
  
When this button is pressed, the tool generated Generic eForm Metadata represented as XSD based on the requirements (refer to capturing and representing metadata section 4.2.3) by transforming the RDFS into XSD and a Generic eForm represented as a XHTML file.

### ***5.1.3 Create Specific eForm interface***

This interface allows the user to carry out the following activities supported by the Delk eForm Creator prototype: search for, create\_New, or modify. This section focuses on create\_New activity for Specific eForm. For more information on search and modify activities see appendix V. This interface is intended to be a proof of concept hence it currently does not support all the activities that could be done to specify the styling of a Specific eForm. One of the features to be included in the future development is the presentation metadata. The presentation metadata would help in specifying the feel and look of the eForm e.g. the location of the eForm elements. Other features are identified in the future work section.

Figure 5.8 shows an interface a user is presented with when they access the Create Specific eForm interface and select create\_New from the 'Create Specific eForm' drop-down

menu. This interface provides a drop-down select option called Generic eForm Name that is populated with the name of the Generic eForms already created. The user is to select one Generic eForm Name (e.g. artistModel) as a basis for creating Specific eForm. This interface also provides a text eForm element called Specific eForm Name that the user enters a name for the Specific eForm (e.g. artistInfo).

**Figure 5.8: DeC Create Specific eForm Interface**

DeC tool at the moment allows the user to create three simple types of Specific eForms (i.e. Registration eForms, data collection eForms, and order eForms). This interface also provides a user a drop-down menu called Types of eForms to choose the type of eForm to create. The type of the eForm chosen helps DeC in deciding what functionality to implement and support in when creating the Specific eForm and the eForm processor. The user selects what type of eForm to create based on their need for the eForm. This interface also provides a text area eForm element called eForm Title that allows the user to enter the title of the Specific eForm. The interface also provides editorial features that a user uses to format the title of the Specific eForm.

When the user chooses a Generic eForm as the basis for creating Specific eForm, the interface is populated with all entities and their attributes defined within the chosen Generic

eForm as shown in Figure 5.9. The interface provides a mechanism to customize the entities and their attributes based on the eForms elements identified earlier in section 3.5.3. The user can delete unwanted entity attributes from the eForm using the delete button represented by the waste bin image. The interface provides text eForm elements called Label for each attribute to add label data (e.g. Artist Name) displayed when rendering the Specific eForm.

**Home** **Create Domain Metadata** **Create Generic eForm** **Create Specific eForm** **Log Out**

**create New** **modify** **search**

### Create Specific eForm Metadata

? Generic eForm Name: paintingModel Generic eForm: based on ecommerceArt

? Specific eForm Name:

? Type of eForm: Choose a Type of eForm to create

? eForm Title: Font Size... Font Family... Font Format... B I U

Entity Name:	Field Name:	Field Label:	Data_Type:	Entity_Relation:	Control:	Field options:
Painting	paintingArtist	Artist	Char(255)	Choose relation	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No Hint: Person that did the art work Default value:
Painting	paintingName	Title	Char(255)	Choose relation	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No Hint: The identifying property for Default value:
Painting	paintingMedia	Medium	Varchar(255)	Choose relation	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No Hint: The art material that is used in a Options: acrylic, canvas, collage <input checked="" type="checkbox"/>
Checkbox	paintingType	Type	Char(255)	Choose relation	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No Hint: The classification

Figure 5.9: DeC Interface Populated with entity attributes

The interface also provides a mechanism to customize the eForm elements (e.g. text box, input, or button) and submission element to the Specific eForm. The eForm elements are provided on the interface in a drop-down select element called Control. The eForm element selected affects how the Specific eForm will be rendered and the Field Options. For example if a check box eForm element is chosen, the Field Options provides an interface that allows the user to enter the different options related to the attribute. This mechanism also allows the user to customize and make changes to the attributes specified on the Generic eForm.

The interface also provides a mechanism to specify the relationship that exists between entities in different Specific eForms because each Specific eForm is mapped to a table, hence establishing a relationship (cardinality) between tables in the database. On the interface, this mechanism is provided by the drop-down menu called 'Entity Relation' which provides three types of relationships (i.e. Parent, Child, and Sibling). Parent and Child relationship are used for entities that have many to 1 or 1 to many relations. Sibling relation is used for many to many entity relations. For the relationship to work, the relationship has to be specified in both ways i.e. when creating both entities. For example, when creating an eForm for an Artist, attribute `artistPaintings` is set as Child.

When creating an eForm for related Paintings eForm, with multiple attributes e.g. Color, the Paintings is set as Parent and the Color as the Child. The two entities (Paintings and Color) can be related by the definition: A Painting has many Colors or even this color belongs to this Painting entity. This relationship illustrates a 1 to many entity relationships. When defining the Painting entity attributes', the Color entity is simply add as one of Painting's attributes and assign the attribute type 'Child'. This tells DeC that Painting is a parent entity and will be connected to a child entity, the Color entity. Therefore DeC will create two tables associated

with the two entities (Painting and Color) database. The Color table will store the repeating data while the Painting table will store the information for every unique painting entered. When any user is using the Specific eForm for searching, the data from the Color will be retrieved recursively. This mechanism is also useful when dealing with order eForm issues (e.g. a sales slip form that has multiple lines for items ordered).

The Field Options also allows the user to specify whether the attribute will be required or not. This will affect the functionality of the Specific eForm, that is, if the attribute is assigned required, then the eform cannot be submitted with the specified as empty. In addition to the required option, this interface provides two text eForm elements for Hint and default data. Hint provides an opportunity for the eForm creator to provide helpful information that will help the eForm user while providing data. Default element provides an interface to enter data that is expected to shown on a Specific eForm as default data.

The creation of the Specific eForm Metadata from the Generic eForm triggers an automated creation of corresponding database metadata and script file with CRUD methods described earlier in section 4.3.2.4. The database created during Specific eForm creation is related to that of the Generic eForm used as the basis.

The database elements are acquired from the Generic eForm process used as the basis for creating Specific eForm. The sizes of the different elements are allocated by default (e.g. varchar is defaulted to size 255). Each element in the XSD file is transformed into an element of SQL statement. For example, artist element is transformed into ('artist' CHAR (255) NOT NULL). Each data input eForm element is expected to map to a given column of a table in a database. Just as is the case with Generic eForm, each data input in Specific eForm element is expected to map to a given column of a table in a database (refer to Section 5.1.2). For example, a text boxes

for inputting name on the artist eForm, will be mapped to a column for storing names in a table (artist-table) within a database.

This section provides a step by step instruction on how to manage Specific eForm Metadata. For a detailed pre-prototype example with code see appendix II.

#### *5.1.3.1 Create New Specific eForm Metadata*

1. The first step is to access DeC Home Page.
2. If the user is logged in then the next step is step 4. The next step is to click on the Log In tab on the right side of the page. An interface with a log in interface is presented as a result of this step (2)
3. The user then enters the username and password into their respective text input boxes on the interface, then press Log In button (the user should be registered as a user to have log-in information). The Home Page will re-load if the log in process was successful.
4. To create a new Specific eForm Metadata, the user clicks on the Create Specific eForm then choose create\_New from the dropdown menu.
5. The next step is to choose the name of the Generic eForm (e.g. artistModel) to base Specific eForm on.
6. The user then provides the Specific eForm name
7. The user then specifies the type of the eForm (e.g. order, survey, or registration) to create from a list provided as a dropdown menu called 'Type of eForm'.
8. The user can then provide and format the Title of the eForm.
9. Based on the Generic eForm chosen, the interface is populated with the entity attributes from the Generic eForm.

10. The next step is to customize the Specific eForm by choosing which Entities to delete and those to retain as part of the Specific eForm.
11. Next is to specify the Entity relation attribute (e.g. Parent) if the eForm is expected to be related with other eForms. If there is no relationship, it is not expected that this attribute is left empty.
12. Next is to specify what type of eForm element (e.g. radio button, check box) will be required to capture the data for the entity attribute being customized.
13. The next step is to specify the field options. The field specifies whether the field will be required or optional when customer is entering data, and provide a hint that will help the customer to enter the correct data. The user provides a default value that will be displayed when the eForm is first loaded. The field option features are eForm specific because two eForms implementing the same Generic eForm Metadata may be required to have different hints or default values.
14. If there is need to delete some of the eForm element options added, identify the element to delete and press the red image with a 'minus' sign on the right side of the element.
15. The user repeats steps 11-14 as many times as the number of entity attributes specified on the Specific eForm.
16. When the user is done choosing and editing eForm elements, they click on the Generate Specific eForm. When this button is pressed, DeC creates an eForm within XHTML file that will be rendered using web browsers, a MySQL database, and PHP script file with CRUD methods to support the manipulation of data to and from the database.
17. Resulting eForm is rendered in a Firefox web browser



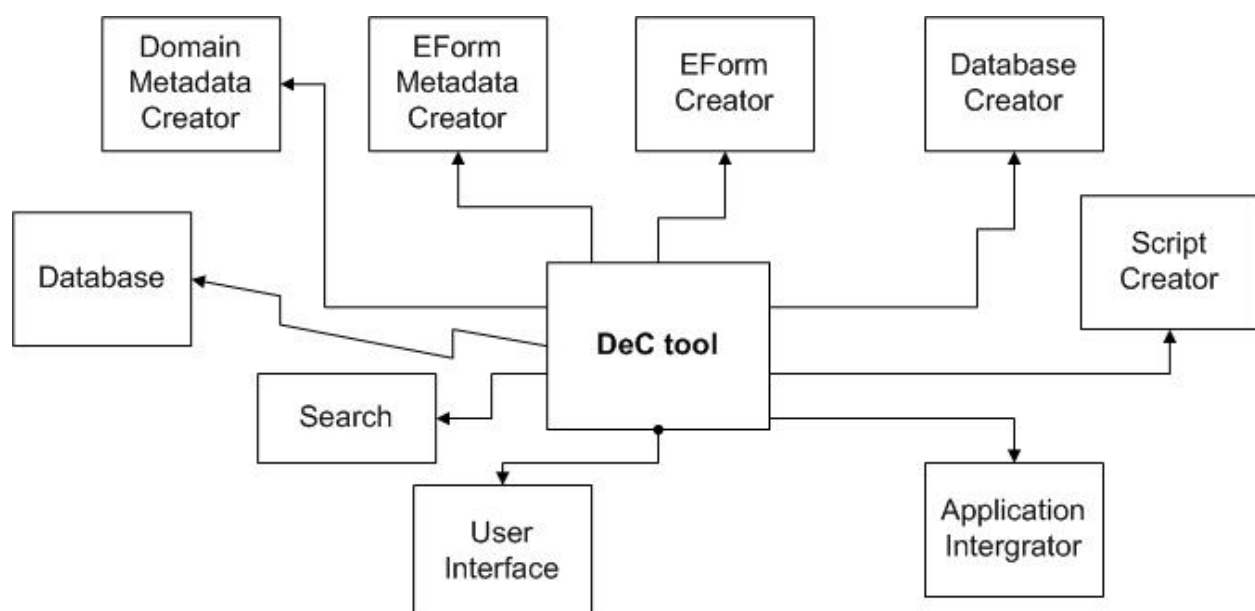
## 5.2 Delk eForm Creator Classes and component functionality (application tier)

This section provides an overview of all the classes and all their responsibilities. This application tier provides and controls the functionality of the tool by performing detailed processing based on defined functions for the application.

### 5.2.1 System Overview Diagram

Based on the components identified in Section 4.3.2, using object oriented design the following classes were identified. A controller class that represents the system itself (i.e. delk eForm creator) is identified. This class manages other object classes that correspond to components within the system. The object classes identified map to system components, use cases, and the output of components of the system.

The classes that represent the components of DeC are: User Interface, Metadata Creator, eForm Metadata Creator, eForm Creator, Database Creator, Script Creator, Search, Application Integrator, and Databases classes. The object classes that corresponding to the output of system components are: Domain Metadata, eForm Metadata, eForm Generic Metadata, eForm Specific Metadata, Database Metadata, eForm, Generic eForm, Specific eForm, Script, and Database.



**Figure 5.10: DeC System Overview diagram**

Figure 5.10 is a diagram showing the system overview diagram for DeC tool. The structure of the diagram arises from the responsibilities description of each chosen class and expected collaboration (communication) between the classes. The classes that corresponding to use cases are: Log In, Log Out, Session, Operation, and Object maintenances. The classes that correspond to operations on object classes are: Maintain Domain Metadata, Maintain eForm Metadata, Maintain Generic eForm Metadata, Maintain Specific eForm Metadata, Maintain eForms, Maintain Generic eForms, Maintain Specific eForms, Maintain Integration, Maintain Script, Maintain Database Metadata and Maintain Database classes. More detailed description of each of the classes is provided in the appendix (see appendix III).

### ***5.2.2 Major Components step by step Functionality***

This section provides a step by step explanation of how each component works. Each of the components communicates with relevant classes to perform their operations.

#### ***5.2.2.1 Domain Metadata Creator Component***

- 1 Receive the data to be used to create Domain Metadata from DeC user interface (Create Domain Metadata).
- 2 Create the RDF file (Domain Metadata) from the data.
- 3 Save data in the database with reference to the Domain Metadata.

#### ***5.2.2.2 EForm Metadata Creator Component***

- 1 Receive the data to be used to create eForm Metadata from DeC user interface (Create Generic eForm).
- 2 Based on the Domain Metadata name chosen, access and get relevant Domain Metadata RDF file.

- 3 Create the XSD file (eForm Metadata) from the resulting combination of data and RDF file.
- 4 Save the resulting data in the relevant database with reference to the eForm Metadata.

#### *5.2.2.3 EForm Creator Component*

- 1 This component is used by more than one user interface.
- 2 Based on the interface interacting with the component, receive the data from the user interface.
- 3 Create the eForm elements based on the entered data.
- 4 Create the XHTML to host the created eForm elements.
- 5 Trigger the Database Creator Component.
- 6 Trigger the Script Creator Component.
- 7 Save the data to the relevant database with reference to the eForm created.

#### *5.2.2.4 Database Creator Component*

- 1 Receive the data from the eForm Creator Component.
- 2 Create the data elements that map to the eForm elements.
- 3 Create the CRUD functionality.
- 4 Create the database file (SQL file).

#### *5.2.2.5 Script Creator Component*

- 1 Receive the data from the eForm Creator Component.
- 2 Based on the type of eForm create the functionalities to be supported.
- 3 Create the elements that map to the eForm elements and database elements.
- 4 Create the script file.

### 5.3 Database Backend (data tier)

This section describes a database that would be used to store all the metadata, eForms, and their related information. The data tier keeps data neutral and independent from application and presentation tiers. The ability to store data free of other tiers, improves scalability and performance of the data storage [20]. This section implements what was discussed briefly in 4.3.2.7 that in turn was based on requirements identified in sections 4.3.1 that the tool should have a storage database.

Figure 5.11 shows an entity relational (ER) diagram of the database that would be used to store Domain Metadata, eForm metadata, and the eForms created. The database consists of 10 tables namely: `domain_metadata`, `domain_class`, `domain_attribute`, `domain_model`, `domain_model_class`, `domain_model_attribute`, `domain_eForm`, `domain_eForm_attribute`, `domain_eForm_option`, and `domain_user`. The tables are named based on the name of the object it will store prefixed with the word 'domain\_'.

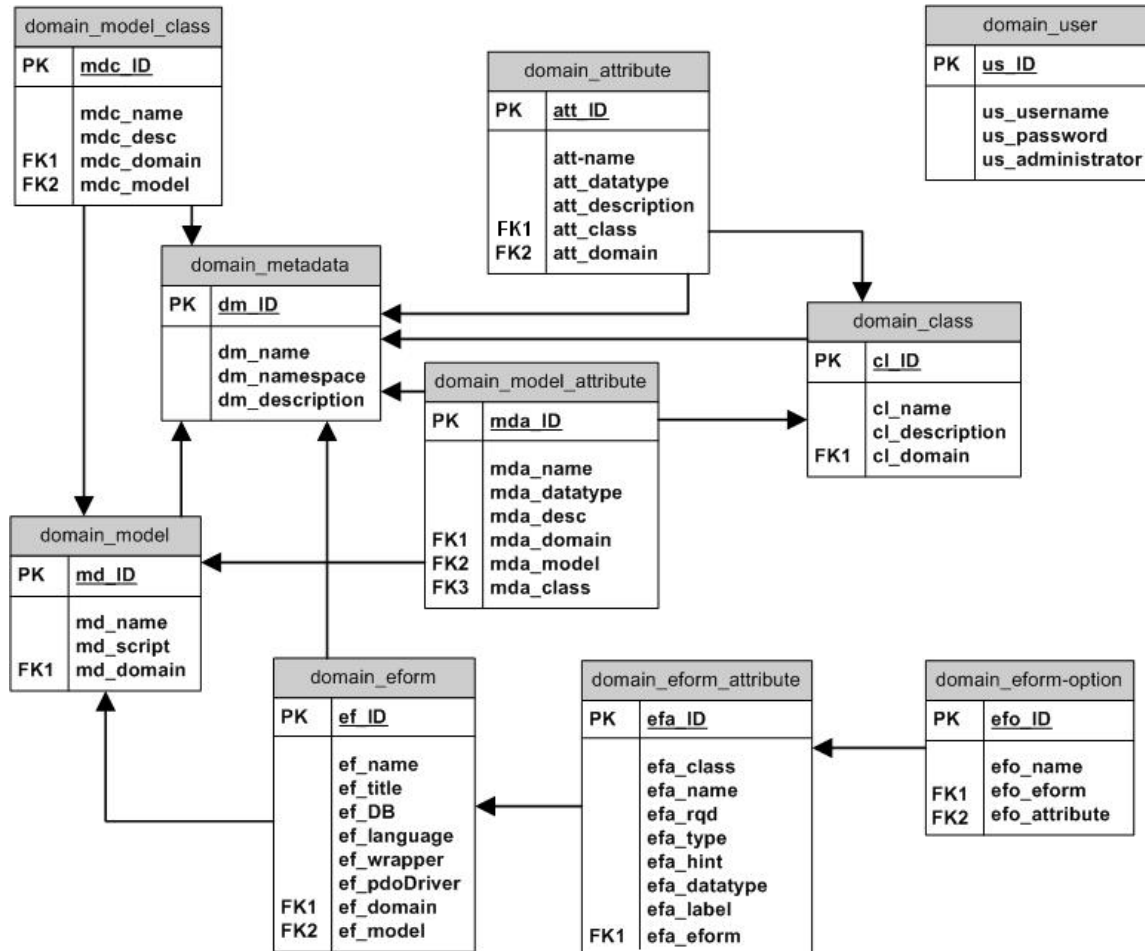


Figure 5.11: DeC backend database E-R diagram

### 5.3.1 Table and their attributes

This section provides descriptions of each table and their attributes. The attributes are named based of the information it will store and are prefixed with the abbreviation of the table name and a hyphen. Detailed description with the column information of each table is provided in the appendix (see appendix IV).

#### 5.3.1.1 domain\_metadata

This table is used for storing Domain Metadata.

#### *5.3.1.2 domain\_class*

This table is used for storing classes for each Domain Metadata. The `cl_domain` acts as a foreign key to the domain table and consist of the Domain Metadata (domain) name combined with domain id.

#### *5.3.1.3 domain\_attribute*

This table is used for storing the attributes of classes for each Domain Metadata. The `att_domain` acts as a foreign key to the Domain Metadata and consist of the domain name combined with domain ID. The `att_class` links the class and their attributes.

#### *5.3.1.4 domain\_model*

This table is used for storing eForm metadata. It has `md_domain` attribute which acts as a foreign key to the domain table and consist of the Domain Metadata (domain) name combined with domain id.

#### *5.3.1.5 domain\_model\_class*

This table is used for storing model classes for each eForm metadata. The `mdc_domain` acts as a foreign key to the domain table and consist of the Domain Metadata (domain) name combined with domain id. The `mdc_model` acts as a foreign key to the model table and consist of the eForm metadata (model) name combined with model id.

#### *5.3.1.6 domain\_model\_attribute*

This table is used for storing the attributes of classes for each eForm metadata. The `mda_domain` acts as a foreign key to the Domain Metadata and consist of the domain name combined with domain ID. The `mda_model` links the eForm metadata (model) and their attributes. The `mda_class` links the class and their attributes.

#### *5.3.1.7 domain\_eForm*

This table is used for storing eForms. The ef\_domain acts as a foreign key to the domain table and consist of the domain name combined with domain id. The ef\_model acts as a foreign key to the model table and consist of the eForm metadata (model) name combined with model id.

#### *5.3.1.8 domain\_eForm\_attribute*

This table is used for storing the attributes (fields) for each eForm. The efa\_eForm links the eForm and their attributes.

#### *5.3.1.9 domain\_eForm\_option*

This table is used for storing the options of attributes (fields) for each eForm that required options, for example the case of check boxes. The efo\_eForm links the eForm and the options. The efo\_attribute links the attribute with the options.

#### *5.3.1.10 domain\_user*

This table is used for storing the user information. The us\_administrator column with be used to state whether the user is an administrator or not.

## CHAPTER 6 TESTING AND EVALUATION

In this chapter, Delk eForm Creator(DeC) tool is examined and evaluated to see if it fulfills the requirements of this thesis to provide solutions to the identified challenges by creating functional eForms using metadata. To prove the concept of this thesis, different sets of simple eForms are created. This chapter provides documented demonstrations of how DeC is used to create the various eForm objects based on the following activities;

- An illustration of how DeC is used to create metadata for a Generic eForm and how the resulting Generic eForm is rendered from the metadata and used by a typical user is provided.
- An illustration of how DeC is used to create metadata for 3 Specific eForms based on the Generic eForm and how the resulting Specific eForms are rendered from the metadata and used by a typical user is provided.
- An illustration of how DeC is used to create metadata for a Specific eForm based on some real world e-Commerce page (e.g. ARTshops - <http://www.artshops.com>) and how the eForm is rendered from the metadata and used by a typical user is provided.

Based on the evaluation requirements, above, the following sections provide a documented demonstration of the creation and usage of the eForm objects. The approach taken in this thesis evaluation is to first show how the eForms are created and then secondly show how they are rendered and used.

### **6.1 Creating eForm objects**

For evaluation purposes, a simple eForm that could be used to specify or collect data about a painting is developed. Figure 6.1 shows what a simple illustration of the targeted eForm



output of the DeC looks like. Figure 6.1 shows, in the chrome browser, an eForm document that could be used for a painting attributes.

The image shows a web browser window with a URL bar displaying 'https://localhost/delk\_eForm1/bank\_eforms/eform.artSelling.genericPainting.xml'. The browser's address bar also shows 'quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...'. The main content area features a form titled 'EForm containing painting's attributes' in a dark red header. The form fields are as follows: 'Painter' (text input), 'Theme' (text input), 'Price' (text input with value '605'), 'Size' (dropdown menu with 'Medium' selected), 'Width' (text input), 'Length' (text input), 'Date' (text input), 'Material' (list box with 'waterPaint', 'oilPaint', and 'sprayPaing'), 'Colors' (checkboxes for 'Red' and 'Blue'), 'Texture' (dropdown menu with 'Smooth' selected), and 'Space amount' (dropdown menu with 'Small' selected). A 'Submit' button is located at the bottom left of the form. At the bottom right, it says 'Created by delk eForm'.

**Figure 6.1: Generic eForm example created by DeC**

### **6.1.1 Create Domain Metadata**

The first step in the process of creating eForms using metadata is to create the Domain Metadata based on requirements in section 4.2.3. This step includes the identification of relevant domain (i.e. ecommerceArt), entities (e.g. painting, sculpture, drawing), and the attributes of the entities. Possible list of attributes include but are not limited to: artist/painter's name, style, theme, medium, size, price, color, and texture. Using the Create Domain Metadata interface, the Domain Metadata is created by providing Domain Metadata Name, Description, and Entity attributes. The attributes involves specification of the attribute name, its data type and its description. Figure 6.2 shows an interface of the Domain Metadata creation process.

[Home](#)
[Create Domain Metadata](#)
[Create Generic eForm](#)
[Create Specific eForm](#)
[Log Out](#)

[create New](#)
[modify](#)
[search](#)

### Create Domain Metadata

Domain Metadata Name:

Domain Description:

Domain dealing with information related to art products sold via internet/online shops

	Attribute Name:	Data_Type:	Description:
	<input type="text" value="Artist"/>	<input type="text" value="CHAR(255)"/> ▼	<input type="text" value="Person that did the art work"/>
	<input type="text" value="Name"/>	<input type="text" value="CHAR(255)"/> ▼	<input type="text" value="The identifying property for the customer, seller, or the art work"/>
	<input type="text" value="Price"/>	<input type="text" value="DOUBLE"/> ▼	<input type="text" value="The cost in terms of money that is need to acquire the art work"/>
	<input type="text" value="Material"/>	<input type="text" value="VARCHAR(25)"/> ▼	<input type="text" value="The material used to create the art work"/>
	<input type="text" value="Media"/>	<input type="text" value="VARCHAR(25)"/> ▼	<input type="text" value="The art material that is used in a work of art such paint"/>
	<input type="text" value="Type"/>	<input type="text" value="CHAR(255)"/> ▼	<input type="text" value="The classification identification of the art work"/>
	<input type="text" value="Style"/>	<input type="text" value="CHAR(255)"/> ▼	<input type="text" value="An &lt;u&gt;artist's&lt;/u&gt; characteristic manner of expression on the work of art"/>
	<input type="text" value="Theme"/>	<input type="text" value="TEXT"/> ▼	<input type="text" value="The graphical appearance of the ark work"/>
	<input type="text" value="Size"/>	<input type="text" value="CHAR(255)"/> ▼	<input type="text" value="The Physical dimensions, proportions, magnitude, or extent of the art work"/>
	<input type="text" value="Color"/>	<input type="text" value="CHAR(255)"/> ▼	<input type="text" value="what is seen when light is reflected of the art work"/>
	<input type="text" value="Texture"/>	<input type="text" value="CHAR(255)"/> ▼	<input type="text" value="An element if art, the surface quality or 'feel' of the art work"/>
	<input type="text" value="Space"/>	<input type="text" value="CHAR(255)"/> ▼	<input type="text" value="An element of art that refers to distance or area between and around the art work"/>
	<input type="text" value="Width"/>	<input type="text" value="INT"/> ▼	<input type="text" value="One of the dimensions of the art work or material used"/>
	<input type="text" value="Length"/>	<input type="text" value="FLOAT"/> ▼	<input type="text" value="The longest dimension of the art work or the material used"/>

Figure 6.2: Specifying Domain Metadata attributes

Once the Domain Metadata is created, it is represented and stored in a RDFS format as shown in Figure 6.3 and described in section 4.2.3. This RDFS documents serves as the Domain Metadata template and framework for creating eForm metadata and eForm objects in the domain.

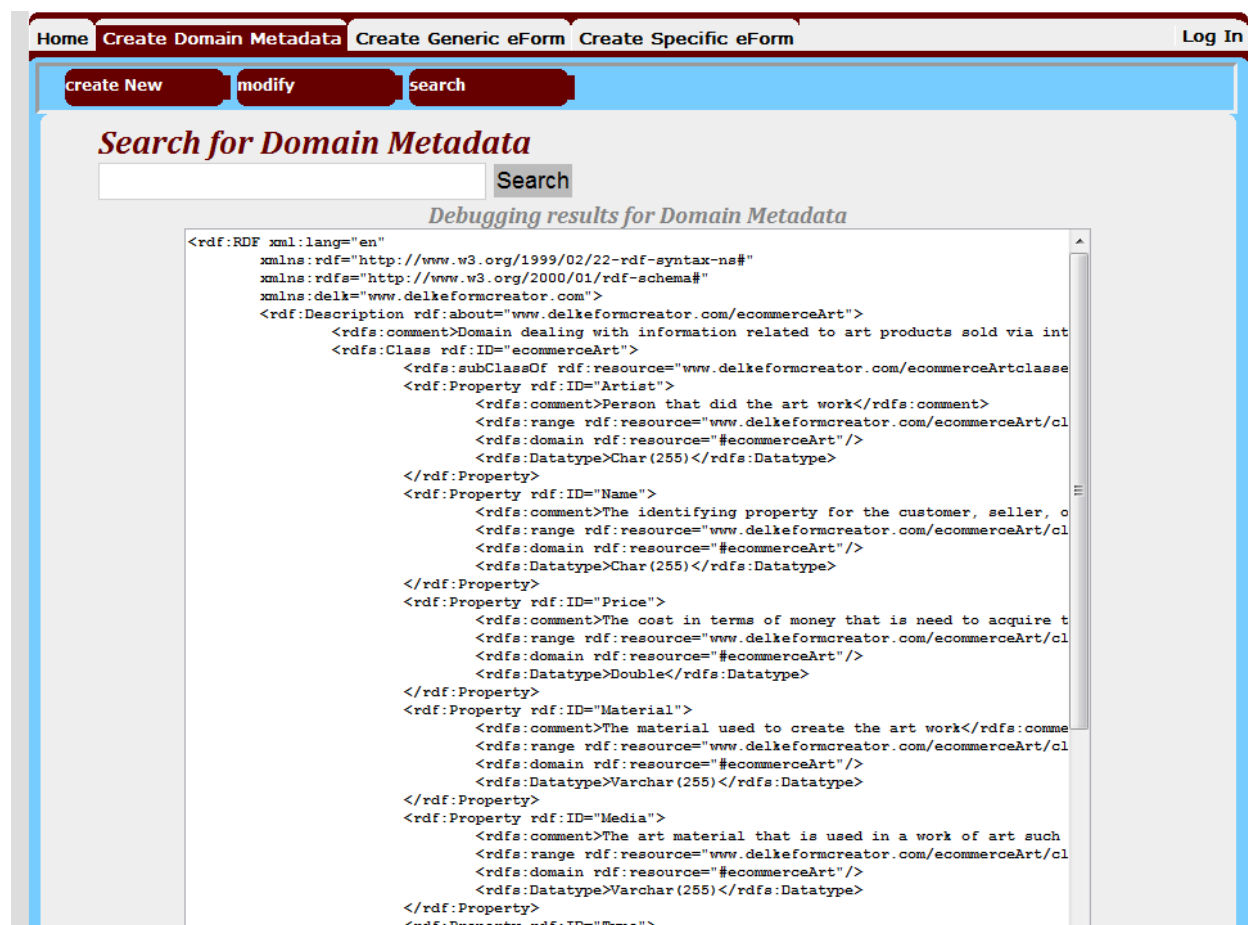


Figure 6.3: Domain Metadata represented and formatted in RDFS

### 6.1.2 Create Generic eForm Metadata

The second step is to create Generic eForm Metadata. Using the Create Generic eForm Metadata interface, the Domain Metadata which is the basis for creating the Generic eForm Metadata is chosen (e.g. ecommerceArt); the name for the Generic eForm Metadata is also specified in this step. The entities to be on the Generic eForm Metadata are identified and the relevant entity's attributes data from the Domain Metadata are identified. Different renderings of eForm elements are specified depending on the data to be collected. Figure 6.4 shows an

interface of the Generic eForm creation process. Generic eForm Metadata itself does not define the file format. Instead, a host language such as XHTML combined with the eForm elements and styling, are used.

**Create Generic eForm Metadata**

Domain Metadata Name: ecommerceArt Domain:

Generic eForm Name: paintingModel

Entity Name:	Entity Attributes:					
Painting	Attribute:	Data_Type:	Label:	Control:	Field_Options:	Description:
	Artist	Char(255)	Artist	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No	Person that did the art work
	Name	Char(255)	Title	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No	The identifying property for the customer, seller, or the art work
	Price	Double	Price	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No	The cost in terms of money that is needed to acquire the art work
	Media	Varchar(255)	Medium	Check Boxes	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No Options: acrylic, canvas, collage	The art material that is used in a work of art such as paint
	Type	Char(255)	Type	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No	The classification identification of the art work
	Theme	Text	Theme	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No	The graphical appearance of the art work
	Size	Char(255)	Size	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No	The Physical dimensions, proportions, magnitude, or extent of the art work
	Color	Char(255)	Colors	Menu (drop-down)	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No Options: Red, Orange, Purple	what is seen when light is reflected of the art work

**Figure 6.4: Customizing Generic eForm Metadata entities and their attributes**

To create Generic eForm Metadata, the Domain Metadata represented in RDFS format are transformed into XSD format as shown in Figure 6.5, a transformation process is described in

section 4.2.3. This XSD documents serves as the template and framework for creating eForms and databases.

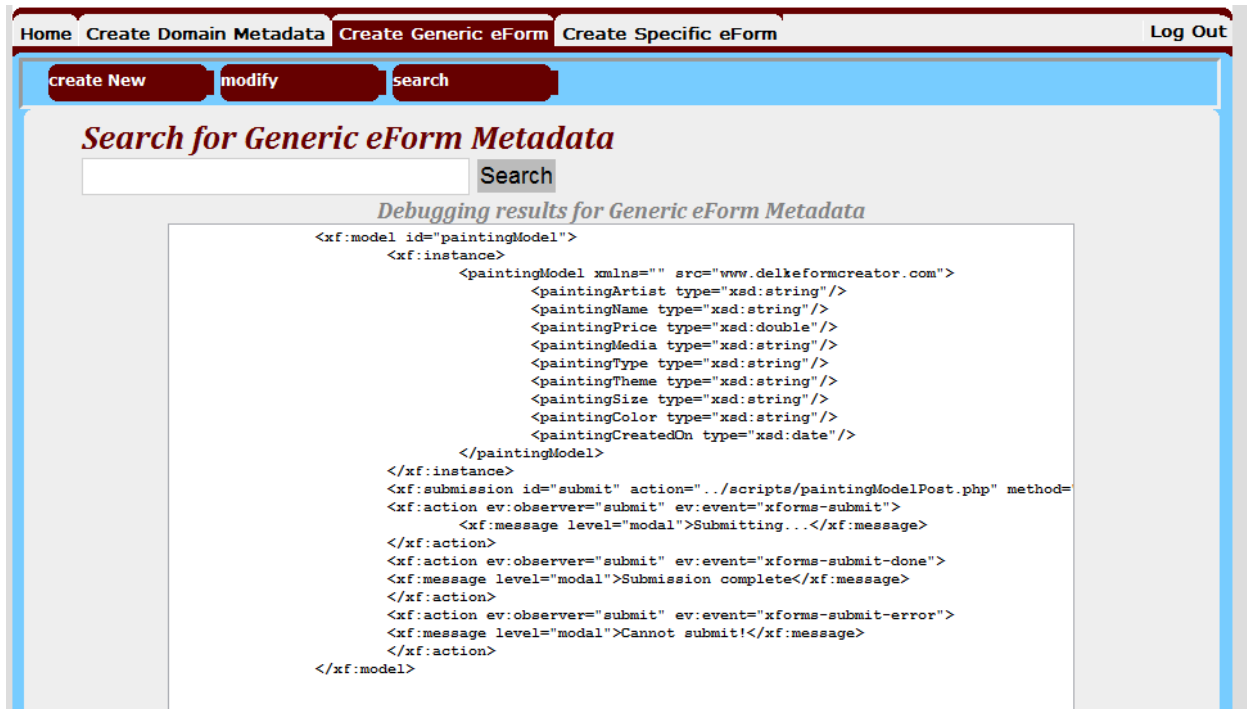


Figure 6.5: Generic eForm Metadata represented and formatted in XSD

### 6.1.3 Create Specific eForm Metadata

The next step is to use the Create Specific eForm Metadata interface to create the Specific eForm file. The different rendering eForm elements are specified depending on the data to be collected. Specific eForm Metadata itself does not define the file format. Instead, a host language such as XHTML combined with the eForm elements and styling, are used. XHTML is used in this thesis with eForm elements inserted the appropriate places. Currently DeC does not support styling except for eForm title; hence the resulting eForms files are rendered using web browsers with less concern for their presentation, look and feel.

Home
Create Domain Metadata
Create Generic eForm
Create Specific eForm
Log Out

create New
modify
search

### Create Specific eForm Metadata

Generic eForm Name:
paintingModel Generic eForm: based on ecommerceArt

Specific eForm Name:

Type of eForm:
Choose a Type of eForm to create

eForm Title:

Font Size...
Font Family...
Font Format
B
I
U

Specific eForm 1

Entity Name:	Field Name:	Field Label:	Data_Type:	Entity_Relati:	Control:	Field options:
Painting	paintingArtist	Artist	Char(255)	Choose relati	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No Hint: Person that did the art work Default value:
ab	Text					
Painting	paintingName	Title	Char(255)	Choose relati	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No Hint: The identifying property for Default value:
ab	Text					
Painting	paintingPrice	Price	Double	Choose relati	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No Hint: The cost in terms of money that is need Default value:
ab	Text					
Painting	paintingMedia	Medium	Varchar(255)	Choose relati	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No Hint: The art material that is used in a Options: acrylic canvas collage Default value:
ab	Text					
Painting	paintingType	Type	Char(255)	Choose relati	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No Hint: The classification identification Default value:
ab	Text					
Painting	paintingTheme	Theme	Text	Choose relati	Text Box	Required? <input checked="" type="radio"/> Yes <input type="radio"/> No Hint: The graphical appearance of the ark work Default value:
ab	Text					

Figure 6.6: Customizing Specific eForm Metadata entities and their attributes

Figure 6.6 shows an interface for creating Specific eForm Metadata. Figure 6.6 shows the different eForm elements required to create the Specific eForm Metadata used to create the Generic eForm shown in Figure 6.1. These Specific eForm Metadata are transformed into XForms formatted using XHTML. This Generic eForm is used as the basis for creating 3 specific eForms.

Figure 6.7 shows the first specific eForm created based on Figure 6.1 Generic eForm rendered using chrome web browser. To create Specific eForm 1, the attributes (i.e. Painter, Theme, Price, and Size) are identified based on the attributes present in the Generic eForm. Specific eForm 1 is rendered in chrome web browser. The red star besides the eForm element indicates that the field is required; the red circle with a cross indicates data needs to be entered in the field, and the blue diamond provides the user of the eForm with useful hint will filling the eForm with data.

The screenshot shows a web browser window with the address bar displaying `https://localhost/delk_eForm/bank_eforms/eform.artSelling.specificPainting1.xml`. The page title is "Specific eForm 1". The form contains the following fields:

- Painter:** A text input field with a red star icon to its right.
- Theme:** A text input field with a red star icon to its right.
- Price:** A text input field containing the value "605", with a blue diamond icon to its right.
- Size:** A dropdown menu with a red star icon to its right.

Each field also has a red circle with a cross icon next to the validation icons. At the bottom of the form is a "Submit" button. The footer of the form says "Created by delk eForm".

**Figure 6.7: Specific eForm 1 example created by DeC rendered using Chrome browser**

Figure 6.8 shows the second specific eForm created based on Figure 6.1 Generic eForm. To create Specific eForm 2, the attributes needed (i.e. Painter, Price, Size, Width, Length, and Material) are identified based on the attributes present in the Generic eForm. Specific eForm 2 is rendered in Firefox web browser.



**Figure 6.8: Specific eForm 2 example created by DeC rendered using Firefox browser**

Figure 6.9 shows the third specific eForm created based on Figure 6.1 Generic eForm. To create Specific eForm 3, the Painting attributes needed (i.e. Painter, Price, Size, Material, Color, and Texture) are identified based on the attributes present in the Generic eForm. Specific eForm 3 is rendered in internet explorer web browser.

Figure 6.10 shows a real world ecommerce eForm from ARTshops ecommerce website. This eForm is rendered in chrome web browser. It contains the following entity attributes: a photo, Artist, Title, Medium, Date, Size, and Price. Attributes that were already defined in the Generic eForm Metadata (Figure 6.4) and the Generic eForm (Figure 6.1).



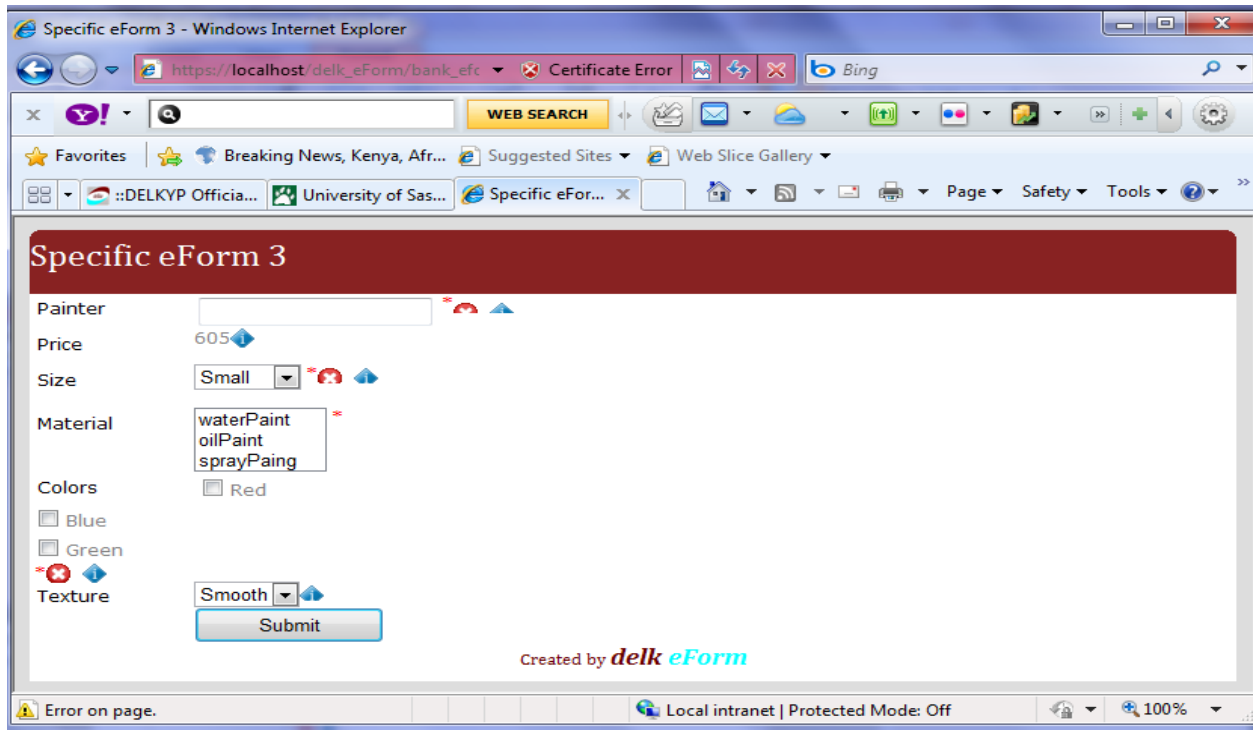


Figure 6.9: Specific eForm 3 example created by DeC rendered using internet explorer browser

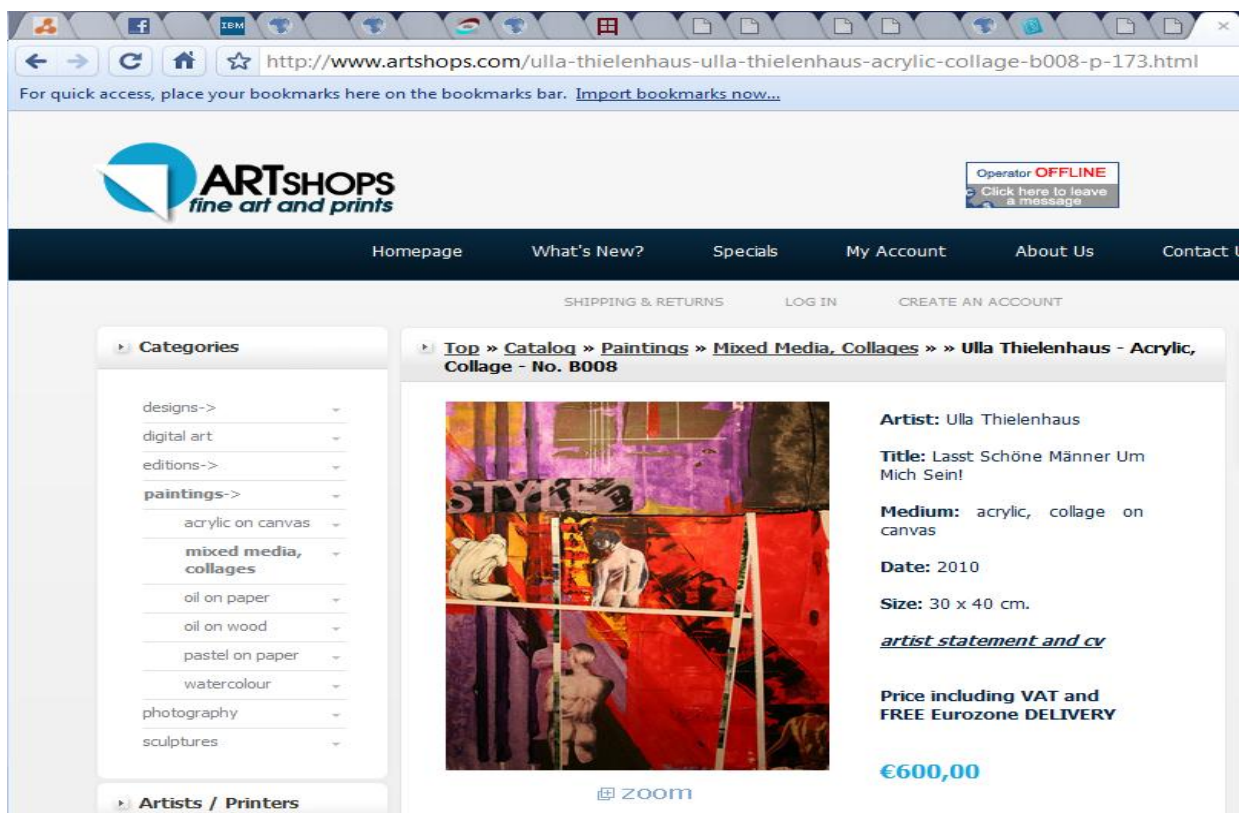


Figure 6.10: A real world eForm example shown in an e-Commerce page

Figure 6.11 shows an eForm created based on a real world ecommerce eForm shown in Figure 6.10. The eForm is rendered using the chrome web browser. It contains the following entity attributes: a photo, Artist, Title, Medium, Date, Size, and Price. The eForm created by DeC does not have the photo because currently DeC functionality does not support creation of uploading mechanism. This mechanism is complex and is not require illustrating the concepts of this thesis.

The screenshot shows a web browser window with the address bar displaying [https://localhost/delk\\_eForm/bank\\_eforms/eform.artSelling.specificARTshop.xml](https://localhost/delk_eForm/bank_eforms/eform.artSelling.specificARTshop.xml). The page title is "EForm based on ARTshops website". The form has a header "upload Photo" and a sub-header "upload photograph of the painting". The form fields are: Artist (text input with value "Ula Thielenhaus"), Title (text input with value "Lasst Schone Manner Um"), Medium (dropdown menu with value "acrylic collage on canvas"), Date (text input with value "2010/09/10"), Size (text input with value "30 x 40 cm"), and Price (text input with value "600.00"). A "Submit" button is located at the bottom left. A tooltip for the Artist field says "The name of the person who painted the painting". The footer says "Created by delk eForm".

**Figure 6.11: eForm example created by DeC based on a real world eForm**

## 6.2 Using eForm objects

Before the created eForms are used by typical users, the eForms and their associated databases need to be integrated within web applications. This step includes the setting up of the database created when the Generic eForm was created. For the evaluation purposes, the eForms created were not integrated to any web application. To use the eForms the database needs to be setup. Currently this step is accomplished by using the interface shown by Figure 6.12.

### 6.2.1 EForm and database setup

Figure 6.12 shows an interface for database setup. The user is not expected to change any of the data when setting up the database within the same server as the web application. This

interface enables the user to specify database name, server, user name, password, and whether there is need to create new table or align with existing. For the purposes of this thesis, the user is not expected to change any information. The user is to press the “proceed” button to setup the database.

**delk eForm creator** , (DEC) is a tool that creates eForms, their databases & their scripts based on Metadata description. DEC automatically creates clean & tested Object Oriented code for web application. DEC uses XHTML, MySQL, & PHP technologies.

**What is DEC Setup?**  
DEC Setup is part of the delk eForm creator. It is meant to help the users setup their databases.

DEC Setup is a 3 step process which:

1. Creates tables for the created DB objects.
2. Performs DB tests on all objects within the 'objects' directory.
3. Provides a light interface to the object tables.

**About Setup:**  
DEC allows users to easily create database objects with integrated CRUD methods.

DEC Setup process enables the mapping of the object's attributes onto columns of a database table without having to write SQL queries by automating **table creation, unit testing** and providing a light **scaffolding** environment.

**Specify the configurations then click on the button below to proceed:**

Database Name:

Database Server:

Database User:

Database Password:

Plugin Path:

TABLES:

TESTS:

Proceed

**Figure 6.12: Interface for setting up the created database**

Figure 6.13 shows an interface for database setup results. The interface provides the information status of the Database setup, Scripting file structure and Essentials showing if all the CRUD (Create, Retrieve, Update and Delete) functions of the Script file are ok. If everything is setup properly, the user then presses the proceed button.

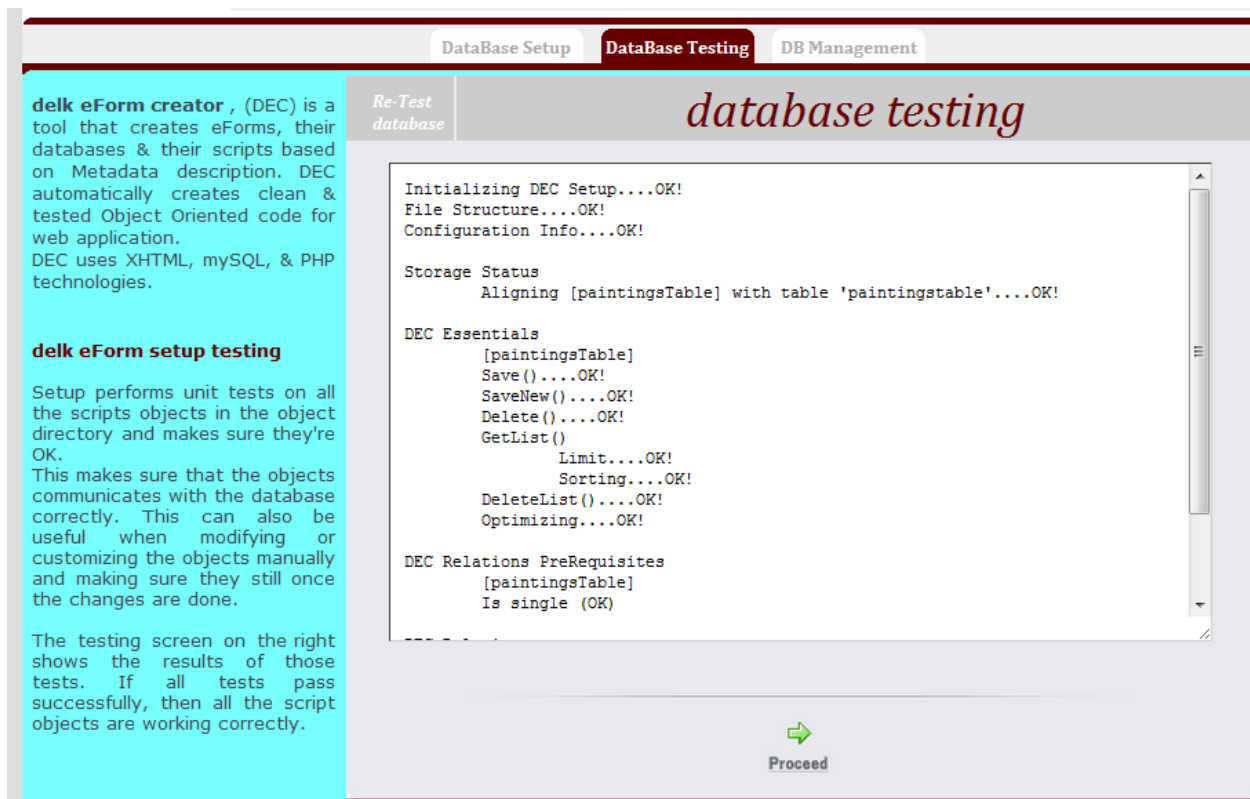


Figure 6.13: Interface showing the database setup results

Figure 6.14 shows an interface for database management. The interface shows the database table name created (paintingsTable) and manipulation options. The interface also shows the number of data entries in the database, currently at 0 because the eForms and the databases have not been used to collect or save any data yet.

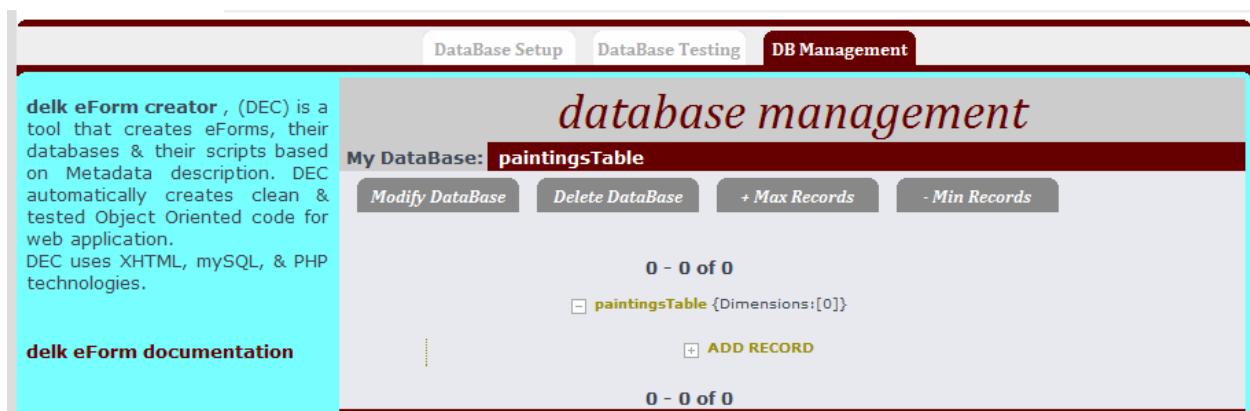
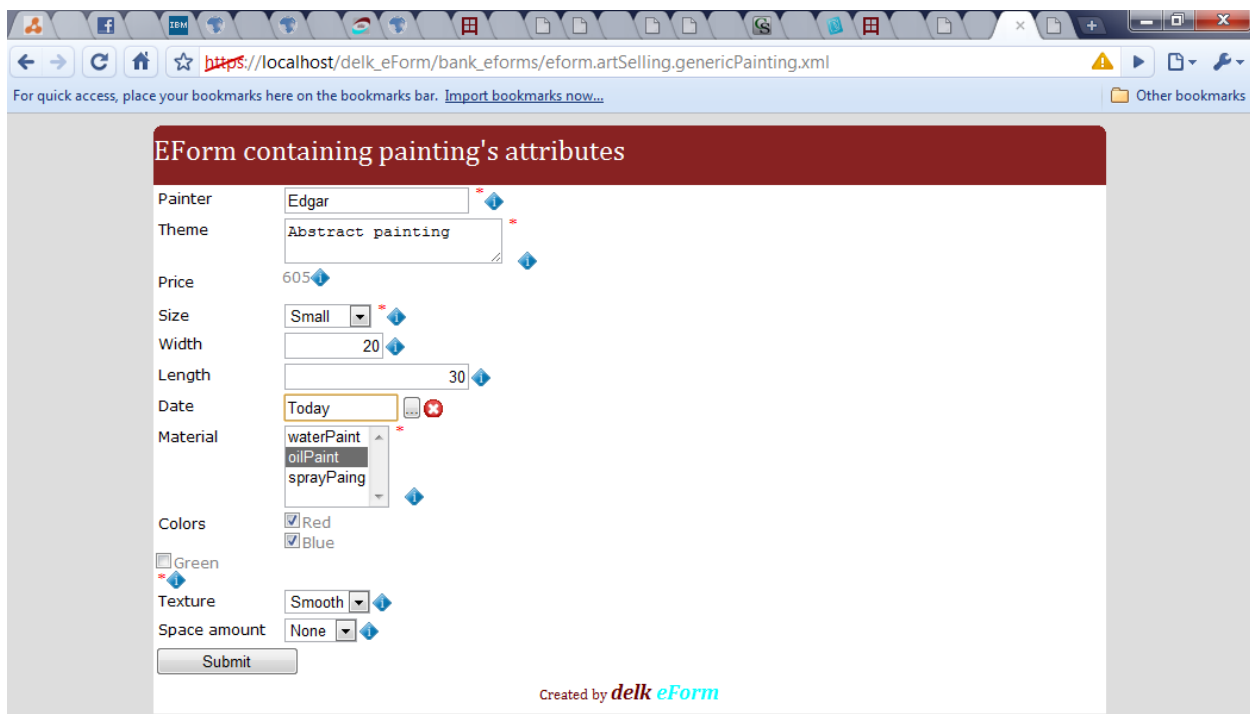


Figure 6.14: Interface for database management

## 6.2.2 Using eForms

Figure 6.15 shows how a Generic eForm is used and rendered using the chrome web browser. A typical user enters information required in the different fields. In the field where the correct data type is entered, the red circle with a cross mark displayed in Figure 6.1 is no longer displayed. If wrong data type is entered in a field the red circle does not disappear and remains as a warning (e.g. the Date field in Figure 6.15). Depending on the type of eForm element used, some data requires the user to enter text while others requires the user to select, such is the case for menu, check boxes and radio buttons elements. After entering the data on the eForm, the user is expected to press the submit button for the data to be entered in the database.



The screenshot shows a web browser window with the address bar displaying [https://localhost/delk\\_eForm/bank\\_eforms/eform.artSelling.genericPainting.xml](https://localhost/delk_eForm/bank_eforms/eform.artSelling.genericPainting.xml). The browser's bookmarks bar is visible with the text "For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...". The main content area displays an eForm titled "EForm containing painting's attributes". The form includes the following fields and controls:

- Painter:** Text input field containing "Edgar".
- Theme:** Text input field containing "Abstract painting".
- Price:** Text input field containing "605".
- Size:** Dropdown menu set to "Small".
- Width:** Text input field containing "20".
- Length:** Text input field containing "30".
- Date:** Calendar icon and text input field containing "Today". This field is highlighted with a red border and a red cross icon, indicating an error.
- Material:** Dropdown menu with options "waterPaint", "oilPaint", and "sprayPaing".
- Colors:** Checkboxes for "Red" (checked), "Green" (unchecked), and "Blue" (checked).
- Texture:** Dropdown menu set to "Smooth".
- Space amount:** Dropdown menu set to "None".
- Submit:** A button at the bottom of the form.

At the bottom right of the form, it says "Created by delk eForm".

**Figure 6.15: eForm example created by delk eForm creator**

Figure 6.16 shows how Specific eForm 1 is used and rendered in chrome web browser. A typical user enters information required in the different fields. In the field where the correct data type is entered, the red circle with a cross mark displayed in Figure 6.7 is no longer displayed.

After entering the data on the eForm, the user is expected to press the submit button for the data to be entered in the database.

The screenshot shows a web browser window with the address bar displaying `https://localhost/delk_eForm/bank_eforms/eform.artSelling.specificPainting1.xml`. The page title is "Specific eForm 1". The form contains the following fields:

- Painter:** A text input field containing "Daudi".
- Theme:** A text input field containing "landscape painting".
- Price:** A text input field containing "605".
- Size:** A dropdown menu with "Medium" selected.

There is a "Submit" button at the bottom left of the form. A red asterisk is visible next to the Painter and Theme fields, indicating required fields. A blue diamond icon is next to the Price and Size fields. The text "Created by delk eForm" is displayed at the bottom right of the form area.

**Figure 6.16: Specific eForm 1 usage**

Figure 6.17 shows how a Specific eForm 2 is used and rendered using the Firefox web browser. A typical user enters information required in the different fields. In the field where the correct data type is entered, the red circle with a cross mark displayed in Figure 6.8 is no longer displayed. If no data type is entered or selected in a field the red circle does not disappear and remains as a warning (e.g. the Material field in Figure 6.17). After entering the data on the eForm, the user is expected to press the submit button for the data to be entered in the database.

The screenshot shows a web browser window with the address bar displaying `https://localhost/delk_eForm/bank_eforms/eform.artSelling.specificPainting2.xml`. The page title is "Specific eForm 2". The form contains the following fields:

- Painter:** A text input field containing "Smith".
- Price:** A text input field containing "605".
- Size:** A dropdown menu with "Large" selected.
- Width:** A text input field containing "100".
- Length:** A text input field containing "120".
- Material:** A dropdown menu with "waterPaint" selected.

There is a "Submit" button at the bottom left of the form. A red asterisk is visible next to the Painter and Material fields, indicating required fields. A blue diamond icon is next to the Price, Size, Width, and Length fields. A red circle with a cross mark is visible next to the Material field, indicating a warning. The text "Created by delk eForm" is displayed at the bottom right of the form area.

**Figure 6.17: Specific eForm 2 usage**

Figure 6.18 shows how a Specific eForm 3 is used and rendered using the chrome web browser. A typical user enters information required in the different fields. In the field where the correct data type is entered, the red circle with a cross mark displayed in Figure 6.1 is no longer displayed. If no data type is selected in a field the red circle does not disappear and remains as a warning (e.g. the Colors field in Figure 6.18). Figure 6.18 also illustrates how the hint is used and displayed to the user. After entering the data on the eForm, the user is expected to press the submit button for the data to be entered in the database.

The screenshot shows a web browser window with the address bar displaying `https://localhost/delk_eForm/bank_eforms/eform.artSelling.specificPainting3.xml`. The page title is "Specific eForm 3". The form contains the following fields:

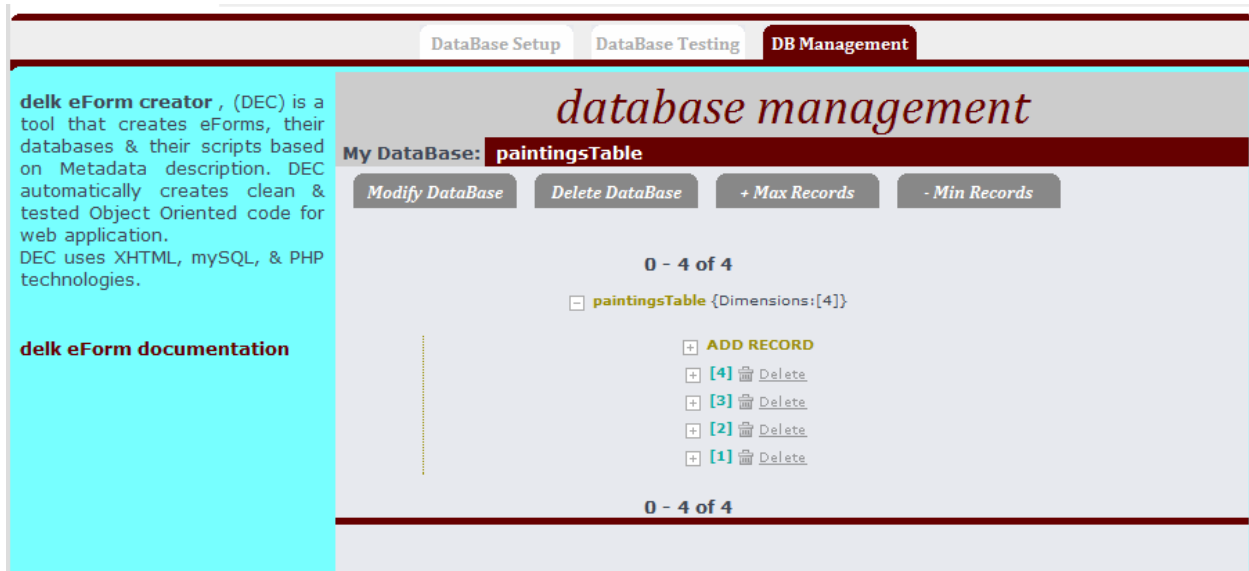
- Painter:** A text input field containing "Smith".
- Price:** A text input field containing "605".
- Size:** A dropdown menu with "Medium" selected.
- Material:** A list box with "waterPaint", "oilPaint", and "sprayPaing" (sic) visible.
- Colors:** A section with checkboxes for "Red", "Green", and "Blue". The "Red" checkbox is checked.
- Texture:** A dropdown menu with "Smooth", "Rough", "Soft", and "Shiny" as options.

Below the form fields is a "Submit" button. To the right of the "Texture" dropdown, there is a text box containing the hint: "Specify the textures to be shown on the painting". At the bottom right of the form, it says "Created by delk eForm".

**Figure 6.18: Specific eForm 3 usage**

### 6.2.3 Checking database

The database management interface introduced in Figure 6.14 is used to check if the data entered by the typical user were entered into the database. Figure 6.19 shows a database management interface showing four records. This illustrates that the data entered in the 4 eForms (Generic eForm, Specific eForm 1, Specific eForm 2, and Specific eForm 3).



**Figure 6.19: Database management interface showing number of data rows entered**

To check the real data that was entered into the database, phpMyAdmin is used. phpMyAdmin is a tool used for managing databases and is not part of delk eForm created. It is just used to evaluate the results of using eForms created by delk eForm creator. Figure 6.20 shows the rows of data that were entered using the eForms.

			paintingstableid	painter	painttheme	paintprice	paintsize	paintwidth	paintlength	paintcreatedon	paintme
<input type="checkbox"/>			1	Edgar	Abstract painting	605	Small	20	30	2010-09-10	Water pa
<input type="checkbox"/>			2	Daudi	A landscape painting	605	Medium	0	0	0000-00-00	
<input type="checkbox"/>			3	Smith		605	Large	100	120	0000-00-00	
<input type="checkbox"/>			4	Smith		605	Medium	0	0	0000-00-00	Water

**Figure 6.19: phpMyAdmin interface showing the actual data entered in the database**

### 6.3 Evaluation

This section describes how documented demonstrations in chapter 6 section 6.1 and 6.2 help illustrate the concept and idea of this thesis. The evaluation requirements established at the beginning of the chapter created a basis for illustrating how the Delk eForm Creator fulfils the requirements identified in chapter 3 and 4.

Section 6.1.1 shows how DeC is used to create Domain Metadata by capturing and representing it in RDFS format. This fulfils the requirement to have a tool for manipulating



Domain Metadata identified in section 4.1.3 and the requirement to represent Domain Metadata in RDFS identified in section 4.2.3.

Section 6.1.2 shows how DeC is used to create Generic eForm Metadata based on Domain Metadata and representing it in XSD format. This fulfils the requirement to have a tool for manipulating Generic eForm Metadata identified in section 4.1.3 and the requirement to represent Generic eForm Metadata in XSD identified in section 4.2.3.

Section 6.1.3 shows how DeC is used to create Specific eForm Metadata based on Generic eForm Metadata and representing it in XForms and XHTML format. This fulfils the requirement to have a tool for manipulating Specific eForm Metadata identified in section 4.1.3 and the requirement to represent Specific eForm Metadata in XForms identified in section 4.2.3.

A documented illustration of how 3 specific eForms based on a Generic eForm is provided in section 6.1.3. Section 6.2.2 illustrates how each of the 3 eForms are used by typical users and rendered in web browsers. Section 6.1.3 also illustrates how DeC is use to create eForm based on real world ecommerce eForms.

### ***6.3.1 Comparison between DeC tools and other tools***

This section provides a comparison between the functionalities supported by DeC tool and those of other tools earlier analyzed in the background study section. The comparisons concentrate more on the tools used to create eForms. There is less concern on the tools used to create the databases. The functionalities for the comparison are;

1. Using metadata to create eForms
2. Create generic and specific eForms
3. Platform independent

4. Reuse existing eForms to create new eForms
5. Scan paper forms
6. Create database tables
7. Create mapping between eForms and database tables
8. Support digital sign
9. Allow downloads
10. Provides eForm Interface formatting
11. Provides what you see is what you get (WYSIWYG)
12. Provide spam security
13. Create different types of eForms
14. Allow printing

Table 6.1 shows the comparison for the first seven functionalities. This Table shows how DeC tool is superior to other tools by supporting functionalities generating and maintaining both the eForms and their associated databases. Most eForm tools concentrate only on creating the eForms, while DeC tool goes beyond these functionalities and adds the creation of databases to its features. DeC tool helps generate and map eForms to their associated databases as described in section 4.2.3 and section 5.1.3.

Another functionality that makes DeC stand out is the ability to have automated changes on the tables where necessary. The DeC interface for creating eForms could also be used to change the eForms and their associated tables. If the user what to add more attributes to the eForm, similar number of attributes would be added to the corresponding table in the databases. This allows for easy maintenance for eForms and their associated databases.

**Table 6-1: Comparison of the functionalities concerning the eForm and database creation**

	<b>eForm creation related functionalities</b>					<b>Database functionalities</b>	
	Use metadata	Generic to specific approach	Platform independent	Reuse existing eForms	Scan paper form to eForm	Create database(DB) tables	eForm-DB tables mapping
DeC tool	Yes	Yes	Yes	Yes	No	Yes	Yes
IBM Workplace forms	No	No	Yes	No	Yes		
FormDocs	No	No	No	No	Yes	No	No
EZ-Forms	No	No	No	No	Yes	No	No

Table 6.2 shows comparison for the other seven functionalities. Generally, the comparison shows that other tools perform more specialized and advanced features for creating eForms. This Table helps demonstrate that DeC tool could be improved further. The features that are currently not supported by DeC are interesting future work opportunities and directions. Chapter 7 provides some suggestions that could be done to improve the functionalities of DeC tool.

**Table 6-2: Comparison on advanced functionalities concerning the eForm creation**

	<b>eForm creation related functionalities</b>						
	Support digital sign	Allow downloads	Formatting user interface	Support WYSIWYG	Provides spam protection	Different type of eForms	Allow printing
DeC tool	No	Yes	No	No	No	no	No
IBM Workplace forms	No	Yes	Yes	Yes	Yes	Yes	Yes
FormDocs	Yes	No	Yes	Yes	No	Yes	Yes
EZ-Forms	No	Yes	Yes	Yes	No	Yes	Yes

### **6.3.2 *Description of an experimental evaluation***

This section tries to go beyond the evaluation on the test and the use of the prototype DeC tool to providing an experimental description for user testing. The one low points of this thesis work is that, user testing and evaluation has not been done yet. There are some statistical data that when collected during user testing would provide a compelling case for or against the DeC tool. This section provides a description of an experiment that would be interesting to address as part of the future works resulting from this thesis work.

One of the challenges in developing and describing an experiment for this tool is identifying the parameters to test. One of the advantages of using metadata in describing data elements is the reusability of the describe data element. Reusability is one aspect that would be in interesting to test in this experiment. Domain metadata is used to capture and describe the overall vocabulary of a given business domain. Domain metadata therefore focuses on the patterns that can encompass generic business model. Domain metadata enables the users (service providers) to understand their business better. These two aspects concerning the metadata form the basis for this experiment.

The main goal of this experiment is to study the effects of using metadata to create eForms on the service providers (mom and pop business). The two questions (effects) that this experiment concentrates on are; first, does using metadata help the service provider reuse existing eForms, and second, does using metadata and drill down (Generic to Specific) description helps the service provider understand their business better.

The requirements for the experiment are; presence of experimenters, two sets of users (representing service providers), the DeC tool, a venue, and a simple ecommerce web application. For the experiment to flow, the DeC tool has to be improved to include some domain

metadata based on the ecommerce application. It would be more interesting if the service providers are randomly selected and placed on the different groups. The service providers will be placed in different groups. They will be introduced to the general purpose of the simple ecommerce web application. The service providers would then be required to create eForms for the different services they would have to support within the web application. Only one group will be given access to the DeC tool. The service providers will be required to create several eForms that would require reuse of their own and other service provider's eForms. Questionnaire could be used to collect data from the service providers based on their experience. A set of 5 to 7 questions concerning the two issues could be arranged.

By contrasting the results from the two user groups, the data collected will go a long way in illustrating the usefulness of the tool, both in affecting how service providers can reuse existing eForms to create new ones and understanding the business model better. The aim of this thesis work is not to produce a tool that is useful by the developers. Instead, the aim is producing a tool usable by the service providers. It is aimed that this experiment illustrates this point as it answers the main two questions of this experiment.

## CHAPTER 7 CONCLUSION AND FUTURE WORK

This chapter concludes this thesis work. A discussion on concluding remarks is provided on section 7.1. Contributions are presented in section 7.2. Suggestion of a list of areas for future research work is presented in section 7.3.

### 7.1 Conclusion

In this thesis two challenges concerning eForms and database implementation approaches are identified: complexity in the manipulation of eForms and their associated database, and difficulty in the reproduction and reuse of existing eForms. Based on these challenges, a solution to use metadata in the implementation of eForms and databases was implemented. To illustrate our solution, we identified a way of modeling the metadata so it can be used in creating eForms and databases. We further created a tool (Delk eForm Creator) to both model the metadata and manipulate eForms, databases, and their associated scripts.

Resource Description Framework Schema (RDFS) was used to model the metadata based on domain context. A three level categorization approach was used to model the metadata. First, *Domain Metadata*: a general vocabulary used with a domain derived from domain description, second, *Generic eForm Metadata*: metadata derived from Domain Metadata geared towards structuring Generic eForms, third, *specific eForm metadata*: metadata derived from generic metadata geared towards creating Specific eForms.

Delk eForm Creator (DeC) was used to model Domain Metadata, Generic eForm Metadata, and specific eForm metadata. Based on the metadata modeled, DeC was used to create Generic eForms and their associated databases. Based on the Generic eForm metadata, DeC was used to create Specific eForms and their associated databases. DeC automatically creates databases. Once the eForms and the databases were implemented, the eForms elements (e.g.

input text field) are mapped to the database elements (i.e. columns of a table) in the database. To fulfill the requirements of this thesis, DeC has been tested by using it to create simple eForms (see section 6.1). The resulting eForms rendered in web browsers are used to enter data into their associated databases. This result shows that the idea of using metadata to implement eForm is a viable solution to the identified challenges.

As stated in the evaluation section, one low point in this thesis is work is the lack of user testing and evaluation. The experiment description in 6.3.3, should however pave way for possibilities for future work. The approach of using drill down approach (i.e. having domain metadata to specific metadata) is good in modeling business patterns at the domain metadata level. One interesting aspect to study from this case is whether the service providers would use the domain and generic metadata as useful information to understand the business model they are involved in.

Furthermore, it would be interesting to compare the functionalities of the DeC tool against how eBay.com and amazon.com create their own eForms. But because of the competition in the market place, eBay.com and amazon.com might not be in a position to reveal how they create their eForms to other competitors.

## **7.2 Contributions**

The work of this thesis contributes an approach of using metadata to implement eForms and their associated databases. The approach includes modeling and reusing metadata models (i.e. Domain, Generic eForm, and Specific eForm Metadata).

The solution presented in this thesis could help service provider reuse existing eForms to create new ones. The metadata could help in maintenance of eForms. For example, when using a tax system and there are changes to be made, changing the domain metadata model can percolate

through the eForms and avoid lots of further working in changing the rest of the eForms. Though other approaches for example, copy & paste may seem faster and useful, it only works for those people who know what copy and paste.

Finally, this work can also be applied to improving XForm eForms and how their data could be stored in a relational database. Most application using XForms use XML databases [7, 9, and 37]. In this thesis, we implemented XForms and MySQL databases.

### **7.3 Future work**

There is no doubt that the work presented in this thesis can be improved upon in numerous ways. We offer the following suggestions as interesting future work and areas of further research:

1. *Using metadata to track changes.* Beyond using metadata to generate eForms and their associated databases, it would be interesting to pursue a research on how the metadata could be used to help in tracking changes and working across eForms. This could help illustrate how metadata could be used in the maintenance and rechecks on eForms. This will involve getting the underlining metadata and semantics right. Though identified as another future direction opportunity, metadata might be used in usability related issues e.g. eForm layout.
2. *Improvement of the query results.* The eForms generated by DeC tool provide a simple querying mechanism, hence, the presence of an opportunity for future work. The eForm querying is somewhat restrictive, i.e., they only support precise set of queries set while generating the eForm. This requires multiple eForms to allow users to ask and perform different types of queries. If a user wishes to perform a query that is not supported, they may leave unsatisfied. The interesting opportunity here is look at mechanism of



improving the query mechanism produced DeC tool to satisfy a diverse group of users that would use the resulting eForm. This could be done by analyzing how the database, its schema, and its content are created. The database schema and content of the database provide a reasonable starting point to estimate these needs. For instance, an Art database is likely to have art pieces as the most important and highest populated entity. It should be no surprise that users accessing such a database will most likely look for information about art pieces and less likely search for information about artist, sellers, or galleries.

3. *Usability test for DeC tool.* A usability test for DeC tool is an interesting future work. An evaluation comparing the automatically-generated eForms with human created eForms side would give a clear illustration of the relative usability of eForms generated by DeC tool. Such an evaluation could potentially be expensive as it will require hiring of an expert who can develop eForms and databases similar to those that DeC tool will generate. Even with the potential cost, it is certainly worth pursuing this usability test in the future to extend the work of this research.
4. *Improve the functionality of the tool to create specialized eForms.* In the analysis Section 3.1.2 various uses of eForms are identified. These include but are not limited to user interface, data searching, data display, data gathering, and communication. EForms can further be used for specialized cases in ecommerce: e.g. order eForms, feedback eForms, and shopping cart eForms. These specialized eForms require advanced mathematical functions in order to perform their tasks. The mathematical metadata can be added to the eForm Creator component. Metadata concerning the types of currencies could be added to the eForm Creator component.

5. *Improve the tool to allow inclusion of presentation metadata.* Presentation is an issue to consider while creating eForms. The ability to render eForms elements on different parts of the eForm page is important for different presentation styles. Currently DeC supports the creation of eForms which are rendered with less consideration to their presentation. The presentation metadata could be added to the eForm Creator Component.
6. *Improving the tool to allow inclusion of security metadata.* Use of security features (e.g. CAPTCHA) is important in countering spamming. These features are important to the functionality and use of the eForms within ecommerce. These features (security metadata) could be added to the eForm Creator Component.
7. *Improve the tool allow manipulation of eForms for other domains.* This thesis focused on creating eForms, and creating Domain Metadata based on an ecommerce domain. The tool could be improved to accommodate the creation of metadata and eForms belonging to different domains.
8. *Investigate the effect metadata has on the ‘visibility’ of the eForms with metadata in the search engines.* A number of approaches towards meta-search have been proposed stating that using metadata improves in searching results [28]. Most of them “focus on deep web query probing and general information retrieval from documents accessible only via form submission” [29]. For example, in ecommerce the customers should be able to find the products or the services they are looking for in the web application using the eForms. It would be interesting research work to identify methodologies to use in testing how the ‘visibility’ of eForms with metadata compare to that without metadata. This research could include developing methodology to prove that using metadata on eForms will make it both easier and faster to search for data stored in eForms with metadata.

9. *Develop a step by step methodology for Domain Metadata modeling.* The Domain Metadata approach that has been presented in this thesis is not yet equipped with step by step instructions on how to create Domain Metadata from a given domain description. Such a step by step methodology is meant to help the administrators in creating and modeling the Domain Metadata by providing a set of guidelines. The methodology should maintain the simplicity of the tool, not make the tool usage complex, and creation of reusable metadata.
10. *Investigating the usability and acceptance rate of the tool among the different user groups (administrators and service providers).* It would be interesting to test the usability and acceptance of the tool in regards to the metadata used to aid the eForm and database creation. The Technology Acceptance Model (TAM) [12] “is an information systems theory that models how users come to accept and use a technology” that could be used in this investigation. According to the model, when users are presented with a new technology, there are factors that influence their decision about how and when they will use it, notably: Perceived usefulness (PU) [12]; defined by Fred Davis as "the degree to which a person believes that using a particular system would enhance his or her job performance". Perceived ease-of-use (PEOU) [12]; also defined by Davis this as "the degree to which a person believes that using a particular system would be free from effort".

## APPENDIX I

**Administrator:** is the individual or organization that provides, maintain, and create the eForms, the databases, and the web application used in a given domain.

**Application developer:** a person, group or organization with an interest in development of a web application.

**Context of use:** is the users, tasks, equipment (hardware, software and materials), and the physical and social environments in which a product is used.

**Database metadata:** is modified Domain Metadata geared towards the database creation.

**Domain:** is an environment that defines a set of common requirements, terminology, and functionality

**Domain metadata:** The Domain Metadata represents common vocabulary of terms used within a given domain. It provides terms (classes) and describes their syntax and semantics as used within a domain in a general manner

**EForm element:** component that helps build the eForm presentation.

**EForm metadata:** is modified Domain Metadata geared towards eForms creation.

**Customer:** is the individual or organization that consumes the services provided on the web application.

**Generic eForm:** an eForm created by the administrators to gather for all possible eForm elements for product or a term within a given domain.

**Metadata:** in general, refer to data used to describe other data entities, making the described data entities easy to reuse and relocate.

**Scenario:** describes the sequences of events that occur during one particular execution of a system

***Schema:*** a way to define structure, content, and semantics of data model within a domain.

***Semantic Web:*** an evolving development of the World Wide Web in which the meaning of information and services on the web are defined

***Semantics:*** meaning or information that provide meaning

***Service provider:*** is the individual or organization that provides services using the web application and the eForms in a given domain.

***Specific eForm:*** an eForm customized to meet the needs of specific service provider within a given domain.

***Usability:*** is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.

***Use case:*** is a description of sets of interactions between users and the web application system.

## Ecommerce domain example

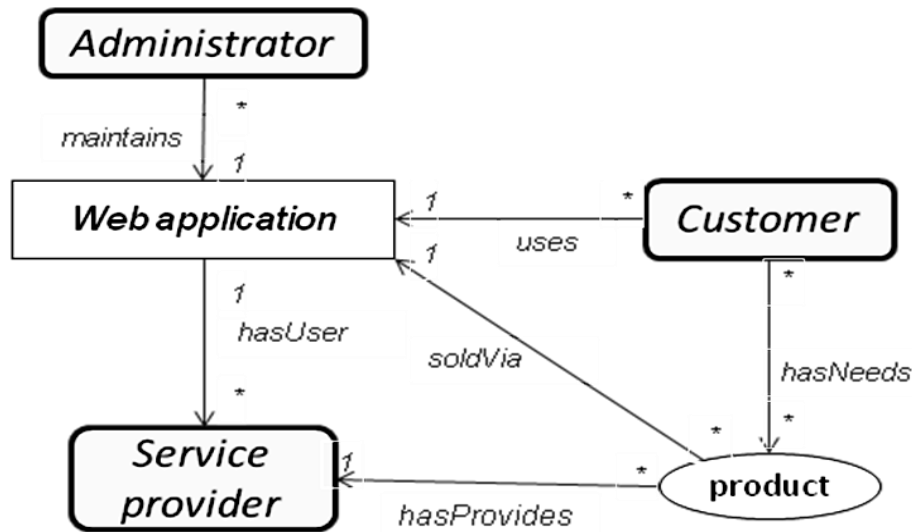


Figure 1: Example of ecommerce scenario

This section provides more details information about the ecommerce example provided in Chapter 4 section 4.2.4. The code and eForm show in this section were used during the analysis and design of delk eForm creator, there are not the results of the tool but prototype examples. This is a description of an example showing how to create Domain Metadata for an ecommerce domain. In this example, a customer uses an ecommerce web application to buy products. Figure 1 is a diagram showing a simple ecommerce scenario. There are five entities in this scenario: administrators, web application, service provider, product, and customer. The object-properties (i.e. relationship between the entities) are shown by the linkages between the entities, e.g. the web application entity has property that it is related to all other entities. The constraints on their relationship are shown, e.g. many administrators can maintain the web application. Though not shown in the diagram, possible data type properties for a product include but are not limited to: name and price.

Simple domain description that could be derived from this scenario is as follows. The administrator maintains the web application. The customer and the service provider use the web application. The service provider provides product. The customer has a need for a product. The product is sold via the web application and is provided by the service provider. The product has name, price, description, and dimension as its data properties. The product is good for human consumption. The customer has property name and address. Hence, the entity product has object-properties web application, service provider, and customer; data type-properties, name, price, type, size, and dimensions; and description-property, good for human consumption.

The first step in creating eForm and their associated databases using metadata is to represent the domain description in a structured format (i.e. Domain Metadata) using RDFS as shown in Figure 2. After the Domain Metadata has been created, it can be maintained (i.e. add to it, change it, or delete from it).

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rdf:RDF >
<rdfs:Class rdf:ID="Product" xsi:type="xsd:string" >
<rdfs:comment>The product is good for human consumption</rdfs:comment>
<rdf:Property rdf:ID="Product">
  <rdfs:Class rdf:ID="Service_Provider" xsi:type="xsd:string" >
    <rdfs:comment>Artist who sell art products</rdfs:comment>
    <rdf:Property rdf:resource="#Product"/>
  </rdfs:Class>
</rdf:Property>
<rdf:Property rdf:ID="Product">
  <rdfs:Class rdf:ID="Customer" xsi:type="xsd:string" >
    <rdfs:comment>customer who needs the product</rdfs:comment>
    <customer xsi:type="xsd:string" ></customer>
  </rdfs:Class>
</rdf:Property>
<rdf:Property rdf:ID="Product">
  <rdfs:comment>Name of product</rdfs:comment>
  <prod_name xsi:type="xsd:string" ></prod_name>
</rdf:Property>
<rdf:Property rdf:ID="Product">
```

```

<rdfs:comment>Price of the product</rdfs:comment>
  <prod_price xsi:type="xsd:float" ></prod_price>
</rdf:Property>
<rdf:Property rdf:ID="Product">
  <rdfs:comment>Type of product</rdfs:comment>
  <prod_type xsi:type="xsd:string" ></prod_type>
</rdf:Property>
<rdf:Property rdf:ID="Product">
  <rdfs:comment>Size of the product</rdfs:comment>
  <prod_size xsi:type="xsd:string" ></prod_size>
</rdf:Property>
<rdf:Property rdf:ID="Product">
  <rdfs:comment>Dimensions of product</rdfs:comment>
  <prod_width xsi:type="xsd:float" ></prod_width>
</rdf:Property>
.
.
.
</rdfs:Class>
</rdf:RDF >

```

**Figure 2: Example of RDFS description of a Domain Metadata**

The next step is to identify the entity (e.g. product) to base the eForm creation. The entity's attributes described within the Domain Metadata are also identified. Possible attributes from the predefined Domain Metadata for the product are listed below. These attributes have been defined in the Domain Metadata even though Figure 2 only shows the definition of some of them. The data type definition helps validate and restrict on the kind of data type that can be entered on eForm field.

- Web application: to show where it is sold
- Service provider: to show its provider
- Customer: to show the buyer of the product
- Product Name
- Product Description
- Product Price



- Product Type
- Product Size
- Product Width
- Product Length

The entity and its attributes description has to then be represented as a Specific eForm Metadata in a simpler semantic language (i.e. XSD format). This requires the transformation of RDFS to XSD. Expected Specific eForm Metadata with data-types derived from the Domain Metadata would resemble the diagram shown in Figure 3.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<product description = "good for human consumption">
    <web_application xsi:type="xsd:string"/>
    <service_provider xsi:type="xsd:string"/>
    <customer xsi:type="xsd:string"/>
    <prod_name xsi:type="xsd:string"/>
    <prod_price xsi:type="xsd:float"/>
    <prod_type xsi:type="xsd:string"/>
    <prod_size xsi:type="xsd:string"/>
    <prod_width xsi:type="xsd:float"/>
    <prod_length xsi:type="xsd:float"/>
</product>
```

**Figure 3: Example of Specific eForm Metadata structure**

The next step is the creation of an instance of the Specific eForm Metadata. This is achieved by adding the eForm elements (e.g. text box, input, or button) and submission element to the Specific eForm Metadata. The creation of the Specific eForm Metadata from the Specific eForm Metadata should trigger an automated creation of corresponding database metadata. Any changes to the specific eForm should trigger an automated change to the database metadata. The database elements (e.g. column) are added to the Specific eForm Metadata and represented as SQL commands as shown in Figure 4. The sizes of the different elements are allocated by default

(e.g. varchar is defaulted to size 255). Each element in the XSD file is transformed into an element of SQL statement. For example, customer element is transformed into ('customer' VARCHAR(255) NOT NULL).

```
CREATE TABLE 'product' (
    'id' int(11) NOT NULL auto_increment,
    'web_application' VARCHAR(255) NOT NULL,
    'service_provider' VARCHAR(255) NOT NULL,
    'customer' VARCHAR(255) NOT NULL,
    'prod_name' VARCHAR(255) NOT NULL,
    'prod_price' FLOAT NOT NULL,
    'prod_type' VARCHAR(255) NOT NULL,
    'prod_size' VARCHAR(255) NOT NULL,
    'prod_width' FLOAT NOT NULL,
    'prod_length' FLOAT NOT NULL,
    PRIMARY KEY ('id'));
```

**Figure 4: Example of database eForm metadata represented in SQL commands**

Each data input (i.e. eForm element) is expected to map to a given column of a table in a database. For example, a text boxes for inputting name on the product eForm, will be mapped to a column for storing names in a table (product-table) within a database.

Figure 5 is shows an example of how the eForm elements and Specific eForm Metadata are used to create Specific eForm Metadata structure. The eFormModel element defines the whole structure of the eForm which includes eForm elements and how the data collected is to be handled. The eFormInstance element is defined within the eFormModel, eFormInstance helps defines an XML template for data to be collected. The eFormInstance element also identifies the various elements that would be present in the eForm. The submission element is also defined within the eFormModel, submission is used to describe how the data will be submitted. This description is based on XForms elements.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<eForm>
    <eFormModel>
```

```

<eFormInstance description = "good for human consumption">
    <formName>Product</formName>
    <web_application xsi:type="xsd:string"/>
    <service_provider xsi:type="xsd:string"/>
    <customer xsi:type="xsd:string"/>
    <prod_name xsi:type="xsd:string"/>
    <prod_price xsi:type="xsd:float"/>
    <prod_type xsi:type="xsd:string"/>
    <prod_size xsi:type="xsd:string"/>
    <prod_dimensions xsi:type="xsd:string"/>
</eFormInstance>
<submission id="product_form" action="product.php" method="get"/>
</eFormModel>
<label ref="formName">Product</label>
<input ref="web_application"><label>web application Name</label></input>
<input ref="service_provider"><label>Provider Name</label></input>
<input ref="customer"><label>Customer Name</label></input>
<input ref="prod_name"><label>Product Name</label></input>
<input ref="prod_price"><label>Product price</label></input>
<input ref="prod_type"><label>Product Type</label></input>
<input ref="prod_size"><label>Product Size</label></input>
<input ref="prod_width"><label>Product Width</label></input>
<input ref="prod_length"><label>Product Length</label></input>
<submit submission="form1"><label>Submit</label></submit>
</eForm>

```

**Figure 5: Example of generic eForm implementation**

The different eForm elements (e.g. text area or input text box) placed on the Specific eForm Metadata could be used to render the individual data elements. The next step is to create Generic and Specific eForms based on the Specific eForm Metadata. The Specific eForm metadata would be used to describe the meaning of the data entered regardless of how the data is rendered by an eForm element on a specific Web page (or browser). The choice of the eForm element to use for specific eForms depends on the service provider because they can change from what was initially used by the administrator to describe the generic eForm structure. Figure

6 shows an example of expected generic eForm that could be generated based on the generic eForm structure described in Figure 5. This is a simple example with less consideration to the layout presentation details.

Product	
web application Name	<input type="text"/>
Provider Name	<input type="text"/>
Customer Name	<input type="text"/>
Product Name	<input type="text"/>
Product Price	<input type="text"/>
Product Type	<input type="text"/>
Product Size	<input type="text"/>
Product Width	<input type="text"/>
Product Length	<input type="text"/>
<input type="button" value="Submit"/>	

**Figure 6: Implementation of generic eForm example**

After the generic eForm has been created, the specific eForm can then be created by choosing needed elements from the generic eForm and modifying them to fulfill the service provider's specific needs. For example, a service provider wants to customize the above generic eForm into a specific eForm showing attributes customer name, product price, product name, and dimensions using a different select eForm element (e.g. changing input box into a dropdown list) as shown in Figure 7. Additional constraints can be added to the one existing on the copied element. The eForms created will be used as the user interface for collecting data that would then need to be stored. While the generic eForms are created using eForm metadata, the databases are created using the database metadata are created. The creation of database would be automated.

The creation of generic eForms is to trigger the creation of databases and tables to work with the created eForms.

Product	
Customer Name	<input type="text"/>
Product Name	<input type="text"/>
Product Price	<input type="text"/>
Product Size	<input type="text"/>
Product Width	<input type="text"/>
Product Length	<input type="text"/>
<input type="submit" value="Submit"/>	

**Figure 7: Implementation of Specific eForm example**

## APPENDIX III

### Classes and their Responsibilities

This section expands on the classes identified and briefly discussed in chapter 5 section 5.2.2. The following tables show the expected responsibilities and methods of the various object classes. The table also shows the classes for which each class collaborates with other classes. The methods for each class are derived from the responsibilities assigned to each object class.

#### *Class Delk\_eFormSystem*

This class represents the delk eForm creator. Table 1 shows its methods and collaborating classes. It is activated when a user starts the system. It starts a new session when a user logs in. It also provides access to all components.

Methods	Collaborators
+Delk_eFormSystem(int: id, systemName: String) +run() +logIn() +logOff() +sessionStart() +getID(): int +getSystemName: String	Class UserInterface Class MetadataCreator Class MetadataManipulator Class eFormCreator Class DatabaseCreator Class ScriptCreator Class Database Class Search Class DatabaseConnection Class ApplicationIntegrator Class LogIn Class LogOut Class Session Class Operation

**Table 1: Class Delk\_eFormSystem's Methods and collaborating classes**

#### *Class UserInterface*

This class represents the User Interface component. Table 2 shows its methods and collaborating classes. It provides the graphical user interface used by users to interact with all the components. It supports the authentication mechanism. It displays messages and response to user interactions.

Methods	Collaborators
+UserInterface() +display(message: String) +readUserName(userName: String) +readPassword(password: String) +readMenuChoice(menu: String[]) +readCancel(cancel: Event)	Class Delk_eFormSystem

**Table 2: Class UserInterface's Methods and collaborating classes**

#### *Class MetadataCreator*

This class represents the Metadata Creator component is only accessible by administrators. Table 3 shows its methods and collaborating classes. It creates and maintains general Domain Metadata. It approves general metadata and allows saving of uncompleted Domain Metadata.

Methods	Collaborators
+maintain(domainMetadata)	Class Delk_eFormSystem Class Database Class Search

**Table 3: Class MetadataCreator's Methods and collaborating classes**

#### *Class eFormMetadataCreator*

This class represents the eForm Metadata Creator component. Table 4 shows its methods and collaborating classes. It is only accessible by administrators. It creates and maintains database, Generic eForm Metadata, and specific eForm metadata.

Methods	Collaborators
+maintain(databaseMetadata) +maintain(eFormMetadata)	Class Delk_eFormSystem Class Database Class Search

**Table 4: Class eFormMetadataCreator's Methods and collaborating classes**

#### *Class eFormCreator*

This class represents eForm Creator component. Table 5 shows its methods and collaborating classes. It creates and maintains instances of generic eForms and instances of specific eForms from specific eForm.

Methods	Collaborators
+maintain(eForm)	Class Delk_eFormSystem Class Database Class Search

**Table 5: Class eFormCreator's Methods and collaborating classes**

#### *Class DatabaseCreator*

This class represents the Database Creator component. Table 6 shows its methods and collaborating classes used to create and maintain database.

Methods	Collaborators
+maintain(database)	Class Delk_eFormSystem Class Search

**Table 6: Class DatabaseCreator's Methods and collaborating classes**

#### *Class StorageDatabase*

This class represents the Database component that provides storage for the output of other components, i.e. to store Domain Metadata and eForm metadata. Table 7 shows its methods and collaborating classes.

Methods	Collaborators
+store(domainMetadata) +store(eFormMetadata) +store(eForms) +store(usersInformation)	Class Delk_eFormSystem Class Search

**Table 7: Class StorageDatabase's Methods and collaborating classes**

#### *Class DatabaseConnection*

This class represents Database Connection component. Table 8 shows its methods and collaborating classes. It initiates connection to the storage database. It sends the log in data for validation, sends data to be saved on the database, and it terminates the connection to database.

Methods	Collaborators
+openConnection() +closeConnection() +sendData()	Class Delk_eFormSystem Class Database

**Table 8: Class DatabaseConnection's Methods and collaborating classes**

#### *Class Search*



This class represents the Search component. Table 9 shows its methods and collaborating classes. It searches the database component for existing domain, generic or specific eForm metadata.

Methods	Collaborators
+search(database)	Class Delk_eFormSystem Class Database

**Table 9: Class Search's Methods and collaborating classes**

### *Class ScriptCreator*

This class represents the Script Creator component. Table 10 shows its methods and collaborating classes. It creates the script.

Methods	Collaborators
+maintain(script)	Class Delk_eFormSystem Class Database

**Table 10: Class ScriptCreator's Methods and collaborating classes**

### *Class ApplicationIntegrator*

This class represents the Application Integrator component. Table 11 shows its methods and collaborating classes. It integrates the created eForms to a web application.

Methods	Collaborators
+performIntegration()	Class Delk_eFormSystem

**Table 11: Class ApplicationIntegrator's Methods and collaborating classes**

### *Class LogIn*

This class represents the Log In use case. Table 12 shows its methods and collaborating classes. It receives the log in data entered by the user and it starts the session.

Methods	Collaborators
+performLogIn() +readLogInData() +startSession()	Class Delk_eFormSystem Class Session

**Table 12: Class LogIn's Methods and collaborating classes**

### *Class LogOff*

This class represents the Log Off use case. Table 13 shows its methods and collaborating classes. It terminates the session.

Methods	Collaborators
+performLogOff() +terminateSession()	Class Delk_eFormSystem Class Session

**Table 13: Class LogOff's Methods and collaborating classes**

#### *Class Session*

This class represents the Session use case. Table 14 shows its methods and collaborating classes. It sends message to user interface if the log in data is invalid.

Methods	Collaborators
+performSession() +sendErrorMessage()	Class Delk_eFormSystem Class LogIn Class LogOut Class Session Class Operation

**Table 14: Class Session's Methods and collaborating classes**

#### *Abstract Class Operation*

This class represents the Abstract Operation use case. Table 15 shows its methods and collaborating classes. It allows the user to choose a type of operation, sends email if operation is approved, and sends message to user interface if problem occurs.

Methods	Collaborators
+performOperation() +sendEmail() +sendErrorMessage()	Class Delk_eFormSystem Class Session Class Operation

**Table 15: Abstract Class Operation's Methods and collaborating classes**

#### *Abstract Class MaintainObjects*

This class represents the Abstract Maintain Objects use case. Table 16 shows its methods and collaborating classes. It allows the user to choose a type of object to maintain.

Methods	Collaborators
	Class MaintainDomainMetadata Class MaintaineFormMetadata Class MaintainDatabaseMetadata Class MaintaineForm Class MaintainDatabase Class MaintainScript

**Table 16: Abstracts Class MaintainObject's Methods and collaborating classes**

*Class MaintainDomainMetadata*

This class represents the Maintain Domain Metadata use case. Table 17 shows its methods and collaborating classes. It checks (i.e. search) for existence of Domain Metadata on storage database. It creates Domain Metadata, clones, modifies, approves, and deletes Domain Metadata. It also loads the Domain Metadata to user interface.

Methods	Collaborators
+search(domainMetadata) +create(domainMetadata) +load(domainMetadata) +clone(domainMetadata) +modify(domainMetadata) +approve(domainMetadata) +save(domainMetadata) +delete(domainMetadata)	Class MaintainObjects

**Table 17: Class MaintainDomainMetadata's Methods and collaborating classes**

*Super Class MaintaineFormMetadata*

This class represents the Maintain eForm Metadata use case. Table 18 shows its methods and collaborating classes. It allows administrator to maintain the eForm metadata.

Methods	Collaborators
+search(genericFormMetadata) +search(domainMetadata) +create(genericFormMetadata) +load(genericFormMetadata) +clone(genericFormMetadata) +modify(genericFormMetadata) +approve(genericFormMetadata) +save(genericFormMetadata)	Class MaintainObjects Class MaintainGenericFormMetadata Class MaintainSpecificeFormMetadata

+delete(genericFormMetadata) +maintain(databaseMetadata)	
---	--

**Table 18: Class MaintainFormMetadata's Methods and collaborating classes**

*Class MaintainGenericFormMetadata*

This class represents the Maintain Generic eForm Metadata use case. Table 19 shows its methods and collaborating classes. It checks (i.e. search) for existence of Generic eForm Metadata. It checks if Domain Metadata exists, creates new, clones, modifies, approves, and delete Generic eForm Metadata. It also loads the Generic eForm Metadata to user interface.

<b>Methods</b>	<b>Collaborators</b>
+search(specificeFormMetadata) +search(genericFormMetadata) +create(specificeFormMetadata) +load(specificeFormMetadata) +clone(specificeFormMetadata) +modify(specificeFormMetadata) +approve(specificeFormMetadata) +save(specificeFormMetadata) +delete(specificeFormMetadata)	Class MaintainObjects Class MaintainFormMetadata

**Table 19: Class MaintainGenericFormMetadata's Methods and collaborating classes**

*Class MaintainSpecificeFormMetadata*

This class represents the Maintain Specific eForm Metadata use case. Table 20 shows its methods and collaborating classes. It checks (i.e. search) for existence of specific eForm metadata. It checks if Generic eForm Metadata exist, create\_New, clones, modifies, approves, and delete Generic eForm Metadata. It also loads the specific eForm metadata to user interface.

<b>Methods</b>	<b>Collaborators</b>
+create(databaseMetadata() +modify(databaseMetadata()	Class MaintainObjects Class MaintainFormMetadata

**Table 20: Class MaintainSpecificeFormMetadata's Methods and collaborating classes**

*Class MaintainDatabaseMetadata*

This class represents the Maintain Database Metadata use case. Table 21 shows its methods and collaborating classes. It creates the database metadata and modifies the database metadata.

Methods	Collaborators
+search(eForm) +maintain(eForm) +load(eForm)	Class MaintainObjects

**Table 21: Class MaintainDatabaseMetadata's Methods and collaborating classes**

#### *Super Class MaintaineForm*

This class represents the Maintain eForm use case. Table 22 shows its methods and collaborating classes. It allows administrator to allow its subclasses.

Methods	Collaborators
	Class MaintainObjects Class MaintainGenericeForm Class MaintainSpecificeForm

**Table 22: Super Class MaintaineForm's collaborating classes**

#### *Class MaintainGenericeForm*

This class represents the Maintain Generic eForm use case. Table 23 shows its methods and collaborating classes. It checks (i.e. search) for existence of generic eForm. It checks if Generic eForm Metadata exist, creates new, clones, modifies, approves, and delete generic eForm. It also loads the generic eForm to user interface.

Methods	Collaborators
+search(genericeForm) +search(gomainMetadata) +create(genericeForm) +load(genericeForm) +clone(genericeForm) +modify(genericeForm) +approve(genericeForm) +save(genericeForm) +delete(genericeForm) +maintain(database)	Class MaintaineForm

**Table 23: Class MaintainGenericForm's Methods and collaborating classes**

*Class MaintainSpecificeForm*

This class represents the Maintain eForm use case. Table 24 shows its methods and collaborating classes. It checks (i.e. search) the existence of specific eForm and generic eForm. It checks if specific eForm metadata exist. It creates new, clones, modifies, approves, and deletes specific eForm. It also loads the specific eForm to user interface.

Methods	Collaborators
+search(specificeForm) +search(genericForm) +create(specificeForm) +load(specificeForm) +clone(specificeForm) +modify(specificeForm) +approve(specificeForm) +save(specificeForm) +delete(specificeForm)	Class MaintaineForm

**Table 24: Class MaintainSpecificeForm's Methods and collaborating classes**

*Class MaintainDatabase*

This class represents the Maintain Database use case. Table 25 shows its methods and collaborating classes. It creates and modifies the database.

Methods	Collaborators
+create(database) +modify(database)	Class MaintainObjects

**Table 25: Class MaintainDatabase's Methods and collaborating classes**

*Class MaintainScript*

This class represents the Maintain Script use case. Table 26 shows its methods and collaborating classes. It creates and modifies scripts.

Methods	Collaborators
+create(script) +modify(script)	Class MaintainObjects

**Table 26: Class MaintainScript's Methods and collaborating classes**

*Class MaintainIntegration*

This class represents the Maintain Integration use case. Table 27 shows its methods and collaborating classes. It integrates the eForms to the web application.

Methods	Collaborators
+integrate(eForm, webapplication, script, database)	

**Table 27: Class MaintainIntegration's Methods and collaborating classes**

*Class DomainMetadata*

This class represents output of metadata creator i.e. Domain Metadata. Table 27 shows its methods and collaborating classes.

Methods	Collaborators
+setDomainId() +setDomainName() +setDomainMetadataId() +setDomainMetadataName() +addTerm() +addTermDescription() +setTermDatatype() +getDomainId() +getDomainName() +getDomainMetadataId() +getDomainMetadataName() +getTerm() +getTermDescription() +getTermDatatype() +editTermDescription() +changeTermDatatype()	Class MetadataCreator Class MetadataManipulator

**Table 27: Class DomainMetadata's Methods and collaborating classes**

*Super Class eFormMetadata*

This class represents the output of eForm Metadata Creator i.e. eForm Metadata. Table 28 shows its methods and collaborating classes.

Methods	Collaborators
	Class MetadataManipulator Class eFormCreator

	Class MaintainGenericFormMetadata Class MaintainSpecificeFormMetadata
--	--

**Table 28: Class eFormMetadata's Methods and collaborating classes**

### *Class GenericFormMetadata*

This class represents the output of eForm Metadata Creator i.e. Generic eForm Metadata.

Table 29 shows its methods and collaborating classes.

Methods	Collaborators
+setDomainId() +setDomainMetadataId() +setScriptId() +setGenericFormMetadataId() +setGenericFormMetadataName() +addGenericFormMetadataTerms() +addGenericFormMetadataTermDescription() +setGenericFormMetadataTermDatatype() +getDomainId() +getDomainMetadataId() +getScriptId() +getGenericFormMetadataId() +getGenericFormMetadataName() +getGenericFormMetadataTerms() +getGenericFormMetadataTermDescription() +getGenericFormMetadataTermDatatype() +editGenericFormMetadataTermDescription() +changeGenericFormMetadataTermDatatype()	Class MetadataManipulator Class eFormCreator Class MaintainGenericFormMetadata Class MaintainSpecificeFormMetadata

**Table 29: Class GenericFormMetadata's Methods and collaborating classes**

### *Class SpecificeFormMetadata*

This class represents the output of eForm Metadata Creator i.e. Specific eForm Metadata.

Table 30 shows its methods and collaborating classes.

Methods	Collaborators
+setDomainId() +setDomainMetadataId() +setScriptId() +setGenericFormMetadataId() +setSpecificeFormMetadataId() +setSpecificeFormMetadataName() +addSpecificeFormMetadataTerms() +addSpecificeFormMetadataTermDescription() +setSpecificeFormMetadataTermDatatype()	Class MetadataManipulator Class eFormCreator Class MaintainGenericFormMetadata Class MaintainSpecificeFormMetadata



+getDomainId() +getDomainMetadataId() +getScriptId() +getSpecificeFormMetadataId() +getSpecificeFormMetadataName() +getSpecificeFormMetadataTerms() +getSpecificeFormMetadataTermDescription() +getSpecificeFormMetadataTermDatatype() +editSpecificeFormMetadataTermDescription() +changeSpecificeFormMetadataTermDatatype()	
--	--

**Table 30: Class SpecificeFormMetadata's Methods and collaborating classes**

### *Class DatabaseMetadata*

This class represents the output of eForm Metadata Creator i.e. Database Metadata. Table 31 shows its methods and collaborating classes.

<b>Methods</b>	<b>Collaborators</b>
+setDomainId() +setDomainMetadataId() +setScriptId() +setDatabaseMetadataId() +setDatabaseMetadataName() +setDatabaseMetadataTerms() +setDatabaseTermDescription() +setDatabaseTermDatatype() +getDomainId() +getDomainMetadataId() +getScriptId() +getDatabaseMetadataId() +getDatabaseMetadataName() +getDatabaseMetadataTerms() +getDatabaseMetadataTermDescription() +getDatabaseMetadataTermDatatype() +editDatabaseMetadataTermDescription() +changeDatabaseMetadataTermDatatype()	Class MetadataManipulator Class DatabaseCreator

**Table 31: Class DatabaseMetadata's Methods and collaborating classes**

### *Super Class eForm*

This class represents Abstract eForm. Table 32 shows its collaborating classes.

<b>Methods</b>	<b>Collaborators</b>
	Class eFormCreator Class MaintainGenericeForm Class MaintainSpecificeForm

**Table 32: Class eForm's Methods and collaborating classes***Class GenericForm*

This class represents the output of eForm creator i.e. Generic eForm. Table 33 shows its methods and collaborating classes.

Methods	Collaborators
+setDomainId() +setDomainMetadataId() +setGenericFormMetadataId() +setScriptId() +setGenericFormId() +setGenericFormName() +addGenericFormTerms() +addGenericFormTermDescription() +setGenericFormTermDatatype() +setGenericFormTermeFormElement() +getDomainId() +getDomainMetadataId() +getScriptId() +getGenericFormId() +getGenericFormName() +getGenericFormTerms() +getGenericFormTermDescription() +getGenericFormTermDatatype() +getGenericFormTermeFormElement() +editGenericFormTermDescription() +changeGenericFormTermDatatype() +changeGenericFormTermeFormElement()	Class eFormCreator Class MaintainGenericForm Class MaintainSpecificeForm

**Table 33: Class GenericForm's Methods and collaborating classes***Class SpecificeForm*

This class represents the output of eForm creator i.e. Specific eForm. Table 34 shows its methods and collaborating classes.

Methods	Collaborators
+setDomainId() +setDomainMetadataId() +setGenericFormMetadataId() +setSpecificeFormMetadataId() +setScriptId() +setGenericFormId() +setSpecificeFormId()	Class eFormCreator Class MaintainGenericForm Class MaintainSpecificeForm

+setSpecificeFormName() +addSpecificeFormTerms() +addSpecificeFormTermDescription() +setSpecificeFormTermDatatype() +setSpecificeFormTermeFormElement() +getDomainId() +getDomainMetadataId() +getGenericeFormMetadataId() +setSpecificeFormMetadataId() +getScriptId() +getSpecificeFormId() +getSpecificeFormName() +getSpecificeFormTerms() +getSpecificeFormTermDescription() +getSpecificeFormTermeFormElement() +getSpecificeFormTermDatatype() +editSpecificeFormTermDescription() +changeSpecificeFormTermDatatype() +changeSpecificeFormTermeFormElement()	
---	--

**Table 34: Class SpecificeForm's Methods and collaborating classes**

### *Class Database*

This class represents the output of database creator i.e. Database. Table 35 shows its methods and collaborating classes.

<b>Methods</b>	<b>Collaborators</b>
+setDomainId() +setDomainMetadataId() +setDatabaseMetadataId() +setScriptId() +setDatabaseId() +setDatabaseName() +setDatabaseTerms() +setDatabaseTermDescription() +setDatabaseTermDatatype() +getDomainId() +getDomainMetadataId() +getDatabaseMetadataId() +getScriptId() +getDatabaseMetadataId() +getDatabaseMetadataName() +getDatabaseMetadataTerms() +getDatabaseMetadataTermDescription() +getDatabaseMetadataTermDatatype() +editDatabaseMetadataTermDescription()	Class DatabaseCreator

+changeDatabaseMetadataTermDatatype()	
---------------------------------------	--

**Table 35: Class Database's Methods and collaborating classes**

### *Class Script*

This class represents the output of script creator i.e. Script. Table 36 shows its methods and collaborating classes.

<b>Methods</b>	<b>Collaborators</b>
+setGenericFormId() +setDatabaseId() +setScriptId() +getGenericFormId() +getDatabaseId() +getScriptId()	Class MetadataManipulator Class ScriptCreator

**Table 36: Class Script's Methods and collaborating classes**

### *Class WebApplication*

This class represents the web application for which the eForms are integrated with. Table 37 shows its methods and collaborating classes.

<b>Methods</b>	<b>Collaborators</b>
+getURL()	Class eForm Class Database Class ApplicationIntegrator

**Table 37: Class WebApplication's Methods and collaborating classes**

## APPENDIX IV

### Delk eForm creator database backend (data tier)

This section describes a database that would be used to store all the metadata, eForms, and their related information. This section provides more detailed information on the E-R diagram shown in Chapter 5 section 5.3.

#### *Table and their attributes*

This section provides description of each table and their attributes identified in Chapter 5 Section 5.3.1. The attributes are named based of the information it will store and are prefixed with the abbreviation of the table name and a hyphen.

#### *domain\_metadata*

This table is used for storing Domain Metadata. Table 38 shows its fields (columns):

Attribute name	Constraints, datatype	Primary or Foreign Key
dm_ID	Not null, int	PK
dm_name	Not null, varchar(50)	
dm_namespace	Not null, varchar(100)	
dm_description	Not null, text	

Table 38: domain\_metadata's Fields (columns)

#### *domain\_class*

This table is used for storing classes for each Domain Metadata. The cl\_domain acts as a foreign key to the domain table and consist of the Domain Metadata (domain) name combined with domain id. Table 39 shows its fields (columns):

Attribute name	Constraints, datatype	Primary or Foreign Key
cl_ID	Not null, int	PK
cl_name	Not null, varchar(50)	
cl_description	Not null, text	
cl_domain	Not null, varchar(60)	FK

Table 39: domain\_class's Fields (columns)

#### *domain\_attribute*

This table is used for storing the attributes of classes for each Domain Metadata. The att\_domain acts as a foreign key to the Domain Metadata and consist of the domain name combined with domain ID. The att\_class links the class and their attributes. Table 40 shows its fields (columns):

Attribute name	Constraints, datatype	Primary or Foreign Key
att_ID	Not null, int	PK
att_name	Not null, varchar(50)	
att_datatype	Not null, varchar(15)	
att_description	Not null, text	
att_class	Not null, varchar(60)	FK
att_domain	Not null, varchar(60)	FK

**Table 40: domain\_attribute's Fields (columns)**

#### *domain\_model*

This table is used for storing eForm metadata. The md\_domain acts as a foreign key to the domain table and consist of the Domain Metadata (domain) name combined with domain id. Table 41 shows its fields (columns):

Attribute name	Constraints, datatype	Primary or Foreign Key
md_ID	Not null, int	PK
md_name	Not null, varchar(50)	
md_script	Not null, varchar(100)	
att_domain	Not null, varchar(60)	FK

**Table 41: domain\_model's Fields (columns)**

#### *domain\_model\_class*

This table is used for storing model classes for each eForm metadata. The mdc\_domain acts as a foreign key to the domain table and consist of the Domain Metadata (domain) name combined with domain ID. The mdc\_model acts as a foreign key to the model table and consist of the eForm metadata (model) name combined with model id. Table 42 shows its fields (columns):

Attribute name	Constraints, datatype	Primary or Foreign Key
mdc_ID	Not null, int	PK

mdc_name	Not null, varchar(50)	
mdc_desc	Not null, text	
mdc_domain	Not null, varchar(60)	FK
mdc_model	Not null, varchar(60)	FK

**Table 42: domain\_model\_class's Fields (columns)**

#### *domain\_model\_attribute*

This table is used for storing the attributes of classes for each eForm metadata. The mda\_domain acts as a foreign key to the Domain Metadata and consist of the domain name combined with domain ID. The mda\_model links the eForm metadata (model) and their attributes. The mda\_class links the class and their attributes. Table 43 shows its fields (columns):

<b>Attribute name</b>	<b>Constraints, datatype</b>	<b>Primary or Foreign Key</b>
mda_ID	Not null, int	PK
mda_name	Not null, varchar(50)	
mda_datatype	Not null, varchar(15)	
mda_desc	Not null, text	
mda_domain	Not null, varchar(60)	FK
mda_model	Not null, varchar(60)	FK
mda_class	Not null, varchar(60)	FK

**Table 43: domain\_model\_attribute's Fields (columns)**

#### *domain\_eForm*

This table is used for storing eForms. The ef\_domain acts as a foreign key to the domain table and consist of the domain name combined with domain id. The ef\_model acts as a foreign key to the model table and consist of the eForm metadata (model) name combined with model id. Table 44 shows its fields (columns):

<b>Attribute name</b>	<b>Constraints, datatype</b>	<b>Primary or Foreign Key</b>
ef_ID	Not null, int	PK
ef_name	Not null, varchar(50)	
ef_title	Not null, varchar(60)	
ef_DB	Not null, varchar(60)	
ef_language	Not null, varchar(60)	
ef_wrapper	Not null, varchar(60)	
ef_pdoDriver	Not null, text	
ef_domain	Not null, varchar(60)	FK
ef_model	Not null, varchar(60)	FK

**Table 44: domain\_eForm's Fields (columns)**

*domain\_eForm\_attribute*

This table is used for storing the attributes (fields) for each eForm. The efa\_eForm links the eForm and their attributes. Table 45 shows its fields (columns):

Attribute name	Constraints, datatype	Primary or Foreign Key
efa_ID	Not null, int	PK
efa_name	Not null, varchar(50)	
efa_class	Not null, varchar(60)	
efa_rqd	Not null, char(3)	
efa_type	Not null, varchar(60)	
efa_hint	Not null, text	
efa_datatype	Not null, varchar(15)	
efa_label	Not null, varchar(60)	
efa_eForm	Not null, varchar(60)	FK

**Table 45: domain\_eForm\_attribute's Fields (columns)**

*domain\_eForm\_option*

This table is used for storing the options of attributes (fields) for each eForm that required options, for example the case of check boxes. The efo\_eForm links the eForm and the options. The efo\_attribute links the attribute with the options. Table 46 shows its fields (columns):

Attribute name	Constraints, datatype	Primary or Foreign Key
efo_ID	Not null, int	PK
efo_name	Not null, varchar(50)	
efo_eForm	Not null, varchar(100)	FK
efo_class	Not null, varchar(60)	FK

**Table 46: domain\_eForm\_option's Fields (columns)**

*domain\_user*

This table is used for storing the user information. The us\_administrator column with be used to state whether the user is an administrator or not. Table 47 shows its fields (columns):

Attribute name	Constraints, datatype	Primary or Foreign Key
us_ID	Not null, int	PK
us_name	Not null, varchar(20)	
us_datatype	Not null, varchar(20)	



us_administrator	Not null, char(3)	
------------------	-------------------	--

**Table 47: domain\_user's Fields (columns)**

## APPENDIX V

### *Modify Domain Metadata*

1. The first step is to access DeC Home Page.
2. If you are logged in move to step 4. If you are not logged in, the next step is to click on the Log In tab on the right side of the page. A interface with a log in interface is presented as a result of this step (2)
3. Enter the username and password their respective text input boxes on the interface, then press Log In button (you should be registered as a user to have log-in information). The Home Page will load if your log in was successful.
4. To modify Domain Metadata, click on the Create Domain Metadata then choose modify from the dropdown menu.
5. The resulting interface from step 4 is the interface that will allow you to choose the name of the Domain Metadata you want to modify.
6. After selecting the name of the domain to be modified, click the modify button. All the information (entities, their description and their attributes) are loaded to the interface.
7. To add the first entity click on the Add Entity button.
8. Specify the further entity name (e.g. seller), its description, and its attributes
9. The attributes would have name (e.g. name, age), data type (characters (i.e. char (255)), integer (i.e. Int)) and their descriptions. To choose the appropriate data type for the attribute from the drop-down of the different data types.
10. After specifying the attribute name, data type and its description, click on the button with the tick sign to add the attribute.

11. To add Entity attributes to existing Entities, identify the Entity for which add the attributes and repeat steps 9 and 10.
12. To remove or delete an entity from the Domain Metadata, click of the waste bin image on the right side of Entity you want to delete.
13. To delete the Entity attribute, identify the attribute you want to delete and press the image with the minus sign beside the attribute.

#### *Search for Domain Metadata*

1. The first step is to access DeC Home Page.
2. To search for Domain Metadata, click on the Create Domain Metadata then choose search from the dropdown menu.
3. The resulting interface from step 2 is the interface that will allow you to type in the name of the Domain Metadata you want to search.
4. You will be provided with the results present in the database based on the text you typed into the search box.
5. Clicking on the modify button, redirects you to an interface that will enable you to modify the Domain Metadata (i.e. step 6 of section 6.2.1.3)
6. Clicking on the Debug button, results in an interface that will enable you to see that code implementation of the Domain Metadata.

#### *Modify Generic eForm Metadata*

(The interfaces are the same as those of modifying Domain Metadata; the difference would be the content.)

1. The first step is to access DeC Home Page.

2. If you are logged in move to step 4. If you are not logged in, the next step is to click on the Log In tab on the right side of the page. A interface with a log in interface is presented as a result of this step (2)
3. Enter the username and password their respective text input boxes on the interface, then press Log In button (you should be registered as a user to have log-in information). The Home Page will load if your log in was successful.
4. To modify Generic eForm Metadata, click on the Manage Generic eForm Metadata then choose modify from the dropdown menu.
5. The resulting interface from step 4 is the interface that will allow you to choose the name of the Generic eForm Metadata you want to modify.
6. After selecting the name of the generic eForm to be modified, click the modify button. All the information (entities, their description and their attributes) of the chosen Generic eForm Metadata are loaded onto the interface.
7. To add an entity click on the Add Entity button.
8. Specify the Entity name (e.g. seller) from the drop-down menu showing the entities related to the Domain Metadata.
9. If there is need to delete some of the entity's attributes, identify the attribute to delete and press the image with a minus sign beside that attribute.
10. To add more entities repeat steps 7- 9.
11. When you are done adding entities, click on the Modify Generic eForm Metadata button to implement changes on the Generic eForm Metadata.

*Search for Generic eForm Metadata*

(The interfaces are the same as those of modifying Domain Metadata; the difference would be the content.)

1. The first step is to access DeC Home Page.
2. To search for Generic eForm Metadata, click on the Manage Generic eForm Metadata then choose search from the dropdown menu.
3. The resulting interface from step 2 is the interface that will allow you to type in the name of the Generic eForm Metadata you want to search.
4. You will be provided with the results present in the database based on the text you typed into the search box.
5. Clicking on the modify button, redirects you to an interface that will enable you to modify the Generic eForm Metadata (i.e. step 6 of section 6.2.2.3).
6. Clicking on the Debug button, results in an interface that will enable you to see that code implementation of the Generic eForm Metadata.

#### *Modify Specific eForm Metadata*

(The interfaces are the same as those of modifying Domain Metadata; the difference would be the content.)

1. The first step is to access DeC Home Page.
2. If you are logged in move to step 4. If you are not logged in, the next step is to click on the Log In tab on the right side of the page. A interface with a log in interface is presented as a result of this step (2)
3. Enter the username and password their respective text input boxes on the interface, then press Log In button (you should be registered as a user to have log-in information). The Home Page will load if your log in was successful.

4. To modify Specific eForm Metadata, click on the Manage Specific eForm Metadata then choose modify from the dropdown menu.
5. The resulting interface from step 4 is the interface that will allow you to choose the name of the Specific eForm Metadata you want to modify.
6. After selecting the name of the Specific eForm to be modified, click the modify button. All the information (entity names, field names, field label, data type, and field options) of the chosen Specific eForm Metadata are loaded onto the interface.
7. To add the first eForm element, choose the type of eForm element you want to add from the drop-down menu of eForm elements and click on the Add Element button.
8. Choose the Entity name (e.g. seller) from the drop-down menu showing the entities related to the Domain Metadata chosen in step 5.
9. Choose the Entity attribute (e.g. age) from the drop-down menu under the Field name label.
10. Specify the label name.
11. Specify the field options; specify whether the field will be required or optional when customer is entering data, and provide a hint will help the customer to enter the correct data. Note other options like read only are yet to be implemented.
12. If there is need to delete some of the eForm element added, identify the element to delete and press the waste bin image on the left side of the element.
13. To add more elements repeat steps 7 - 11.
14. If there is need to delete some of the eForm element's options (i.e. checkbox, radio, or menus), identify the options to delete and press the image with a minus sign beside that attribute.

### *Search for Specific eForm Metadata*

(The interfaces are the same as those of modifying Domain Metadata; the difference would be the content.)

1. The first step is to access DeC Home Page.
2. To search for Specific eForm Metadata, click on the Manage Specific eForm Metadata then choose search from the dropdown menu.
3. The resulting interface from step 2 is the interface that will allow you to type in the name of the Specific eForm Metadata you want to search.
4. You will be provided with the results present in the database based on the text you typed into the search box.
5. Clicking on the modify button, redirects you to an interface that will enable you to modify the Specific eForm Metadata (i.e. step 6 of section 6.2.2.3).
6. Clicking on the View button to view the created Specific eForm.

## REFERENCES

- [1] Axelsson, K., & Ventura, S. (2007). Reaching Communication Quality in Public E-Forms—A Communicative Perspective on E-Form Design. *Electronic Government*, 342–353. Springer. Retrieved from <http://www.springerlink.com/index/7h769078327151n6.pdf>.
- [2] BellHawk Systems Corporation, B. (2010). Problems with Paper Forms. Retrieved from <http://www.bellhawk.com/Solutions/Replace PaperForms.php>.
- [3] Berners-Lee, T. (1997). Axioms of Web Architecture: Metadata Architecture.
- [4] Boyer, J. M., Landwehr, D., Merrick, R., Raman, T. V., Dubinco, M., & Klotz, L. L. (2009). XForms 1.1 W3C Recommendation. Retrieved from <http://www.w3.org/TR/xforms/>.
- [5] Brand, N., & XML Web Services Working Group, W. (2003). E-Forms Applications Use of XML Standards.
- [6] Brickley, D., Guha. (2000). Resource Description Framework (RDF) Schema Specification 1.0. Retrieved from <http://www.w3.org/TR/2000/CR-rdf-schema-20000327>.
- [7] Calvanese, D. (1999). Representing and reasoning on XForms document. *Journal of Logic and Computation*, 9(3), 295-318. doi: 10.1093/logcom/9.3.295.
- [8] Calvanese, D, Giacomo, G. D., Lenzerini, M., Nardi, D., Rosati, R., La, R., et al. (1998). Information Integration : Conceptual Modeling and Reasoning Support. *Components*.
- [9] Cardone, R., Soroker, D., & Tiwari, A. (2005). Using XForms to simplify Web programming. *Proceedings of the 14th international conference on World Wide Web - WWW '05*, 215. New York, New York, USA: ACM Press. doi: 10.1145/1060745.1060780.
- [10] Carter, J., Liu, J., Schneider, K., & Fourney, D. (2005). Transforming Usability Requirements into Software Specifications. *Human-Centered Software Engineering – Integrating usability in the Development Process*, 145-167.
- [11] David, F., & Priscilla, W. (2004). XML Schema. Retrieved from <http://www.w3.org/TR/xmlschema-0/>.
- [12] Davis, F. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly* 13(3), 319-340.
- [13] DCMI, D. C. M. I. (1995). Dublin Core Metadata Element Set. *Online Computer Library Center*. Retrieved from <http://dublincore.org/documents/dces/>.
- [14] Deborah, M., & Frank, V. H. (2004). OWL Web Ontology Language. Retrieved from <http://www.w3.org/TR/owl-features/>.
- [15] Dekeyser, S., Hidders, J., Watson, R., & Addie, R. (2006). Peer-to-Peer Form Based Web Information Systems. *Reproduction*, 49.



- [16] Dewitt, M. (2004). PureEdge 's XML E-Forms Streamline Workflow. *Business*.
- [17] Dubinko, M., Klotz, L., Merrick, R. And Raman, T. (2003). XForms 1.0, World-Wide Web Consortium, (Recommendation). Retrieved from <http://www.w3.org/MarkUp/Forms> .
- [18] Dubinko, M. (2003). *O'Reilly XForms Essentials*.
- [19] Eckerson, W. (1995). Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications. In *Open Information Systems 10 (January 1995)*.
- [20] EZ-Forms. (2008). EZ-Forms. Retrieved from [http://www.filebuzz.com/fileinfo/43290/EZ\\_Forms\\_ULTRA\\_Filler\\_5\\_50\\_ec\\_220.html](http://www.filebuzz.com/fileinfo/43290/EZ_Forms_ULTRA_Filler_5_50_ec_220.html) .
- [21] Fitzwater, L. (2006). Introduction to MDR - ISO/IEC 11179 Metadata Registries Edition 3. *9th Open Forum on Metadata Registries Harmonization of Terminology, Ontology and Metadata*.
- [22] FormDocs. (2008). FormDocs. Retrieved from <http://www.formdocs.com>.
- [23] Futo, I. (2006). A general purpose front-office system for eGovernment communications. *28th International Conference on Information Technology Interfaces, 2006.*, 333-338. Ieee. doi: 10.1109/ITI.2006.1708502.
- [24] Gouscos, D., Rouvas, Stathis, Vassilakis, Costas, & Georgiadis, P. (2002). An Object-Oriented Approach for Designing Administrative E-forms and Transactional E-services, (i), 19-30.
- [25] Greenberg, J. (2005). Understanding Metadata and Metadata Schemes. *Cataloging & Classification Quarterly*, 40(3), 17-36. doi: 10.1300/J104v40n03\_02.
- [26] Gruber, T. R., & Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Creation Diffusion Utilization*, (April).
- [27] Heller, M. (2007). REST and CRUD: the Impedance Mismatch. Retrieved from <http://www.infoworld.com/d/developer-world/rest-and-crud-impedance-mismatch-927>.
- [28] Hepp, M. (2008). GoodRelations : An Ontology for Describing Products and Services Offers on the Web. *Business*, 329-346.
- [29] Holzinger, W., Krüpl, B., & Baumgartner, R. (2008). Exploiting Semantic Web Technologies to Model Web Form Interactions, 1145-1146.
- [30] International Organization For Standardization, I. (a). INTERNATIONAL Ergonomic requirements for office work with visual display terminals ( VDTs ) - Part 11 : Guidance on usability. *International Organization*, 1998.
- [31] International Organization For Standardization, I. (b). INTERNATIONAL STANDARD ISO / IEC and standardization of data elements —, 1999.
- [32] James, C. (2008). *Putting Usability First*.

- [33] Jon, T. (2005). *Building an administration CASE Tool for CASE Tools*.
- [34] Kilov, H. (1990). From semantic to object-oriented data modeling. *Proceeding ISCI '90 Proceedings of the first international conference on systems integration on Systems integration '90*, 385-393.
- [35] Martin, H. (2009). Semantic Web-based E-Commerce: Webmasters Get Ready! Retrieved from [http://semanticweb.com/semantic-web-based-e-commerce-webmasters-get-ready\\_b11088](http://semanticweb.com/semantic-web-based-e-commerce-webmasters-get-ready_b11088).
- [36] Mathes, A. (2004). Folksonomies - Cooperative Classification and Communication Through Shared Metadata. Retrieved from <http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html>.
- [37] McBride, B., with Brickley, D., Guha. (2004). RDF Vocabulary Description Language 1.0: RDF Schema. Retrieved from [http://www.w3.org/TR/rdf-schema/#ch\\_sumclasses](http://www.w3.org/TR/rdf-schema/#ch_sumclasses).
- [38] Mikko, H., Oskari, K., & Markku, L. (2007). Connecting XForms to Databases An Extension to the XForms Markup Language.
- [39] Nicola, M., Avenue, B., & Jose, S. (2005). Native XML Support in DB2 Universal Database. *Database*, 1164-1174.
- [40] Shadbolt, N., Berners-Lee, T., & Hall, W. (2006). The Semantic Web Revisited. *IEEE Intelligent Systems*, 21(3), 96-101. doi: 10.1109/MIS.2006.62.
- [41] Stadlhofer, B., & Salhofer, P. (2007). Automatic generation of e-government forms from semantic descriptions. *Proceedings of the 1st international conference on Theory and practice of electronic governance - ICEGOV '07*, 12. New York, New York, USA: ACM Press. doi: 10.1145/1328057.1328064.
- [42] Tambouris, E., Boukis, G., Vassilakis, C., Lepouras, G., Rouvas, S., Cañadas, R., et al. (2002). SMARTGOV - A Governmental Knowledge-based Platform for Public Sector Online Services, 128-129.
- [43] Tornqvist, A. (1999). XML and Objects - the future for e-Forms on the Web. *Structure*.
- [44] W3Schools. (2010). Introduction to RDF. Retrieved from [http://www.w3schools.com/rdf/rdf\\_intro.asp](http://www.w3schools.com/rdf/rdf_intro.asp).
- [45] Wang, Q., Guo, Y. C., Li, M., & Zhao, X. F. (2008). ACORD Standards Based SOA Solution for Insurance Industry - Combine ACORD eForms with Business Services through XForms Standard. *2008 IEEE International Conference on e-Business Engineering*, 247-254. Ieee. doi: 10.1109/ICEBE.2008.76.
- [46] Zhoulin, D., Yi, G., Jun, L., & Jie, X. Y. (2007). Experiences in Accurately Estimating Electronic Forms Conversion Services with a Spiral Estimate Process. *31st Annual International Computer Software and Applications Conference - Vol. 2 - (COMPSAC 2007)*, (Compsac), 497-500. Ieee. doi: 10.1109/COMPSAC.2007.113.