A Web-based System for
Publishing Publications to Both Växjö University Library and RICS.

Israr ul haq
Acknowledgment
This thesis is submitted for fulfillment of the degree Master in Software Technology at Växjö University of Sweden.

Thanks to my supervisor Prof. Dr. Welf Löwe for his kind assistance. His help was always available throughout my project.

Big thanks to Växjö University for being such great place of learning. I have really enjoyed the academic excellence and the fun of meeting with people from all over the world.

And to all those who indirectly contributed to this work- Lots of thanks to Jonas Lundberg for providing kind guidance during the entire program and all the faculty members of Computer Science Department.

And to the one upstairs, who made all this possible. Thanks God.
Abstract
The Department of Computer Science of Växjö University has a web based system where thesis finished by the students as well as available thesis can be seen. Research work made by the faculty members of the Department is also available here.

The Växjö University Library also has a web- based System called DiVA. DiVA stands for Digitala Vetenskapliga Arkivet (Academic Archive Online). It enables students and researchers to publish their research work to University Library. Students and Researchers can also search, research made by other researchers.

Each year researcher/student publishes their result in conference papers, journal articles, thesis report, books etc. These publications should be register both at the Växjö University Library and RICS (Research in Computer Science) web site in a systematic way so to avoid any kind of redundancy and errors. To develop a system which follow the principle” publish once and view everywhere” is the objective of this project.

The System makes it possible to extract the already published publication at DiVA and put that publication’s information into RICS web site.

When a user requests for registering a publication, the system should verify whether he/she is eligible to register a publication. If he/she qualifies, that is if he/she is a registered user, then the system will register the publication along with the required information like title, author, date of publication, kind of publication etc. on different Web sites.

The web-based publication information system was implemented in C#.NET.

The project was successfully completed but the deliver system requires more live testing.

Keywords: Publication, RICS, DiVA(Digitala Vetenskapliga Arkivet), Publish, Regex.
# Table of Contents

1  INTRODUCTION ................................................................................................................................. 1  
1.1  PROBLEM DESCRIPTION ...................................................................................................................... 1  
1.2  DESIGNED SOLUTION ........................................................................................................................... 2  
1.3  GOAL OF THE PROJECT ........................................................................................................................ 3  
1.4  MOTIVATION ......................................................................................................................................... 3  
1.5  REPORT STRUCTURE ............................................................................................................................. 4  

2  BACKGROUND ........................................................................................................................................ 5  
2.1  SCREEN SCRAPING ...................................................................................................................................... 5  
2.1.1  Tools used for Screen Scraping ........................................................................................................... 5  
2.1.2  Regular Expression .............................................................................................................................. 5  
2.2  C# REGULAR EXPRESSION ..................................................................................................................... 6  
2.2.1  The C# System.Text.RegularExpressions Class ....................................................................................... 6  
2.2.2  RegexOptions ...................................................................................................................................... 7  
2.2.3  Regular Expression Patterns ............................................................................................................... 7  

3  WEB APPLICATION TECHNOLOGIES ..................................................................................................... 8  
3.1  LINUX, APACHE, MYSQL AND PHP (LAMP) ...................................................................................... 8  
3.1.1  Linux .................................................................................................................................................. 8  
3.1.2  Apache .............................................................................................................................................. 8  
3.1.3  MySQL ............................................................................................................................................... 9  
3.1.4  PHP ................................................................................................................................................... 9  
3.2  JAVA/J2EE ............................................................................................................................................. 9  
3.3  .NET ARCHITECTURE ............................................................................................................................ 10  
3.3.1  Common Language Runtime ............................................................................................................. 10  
3.3.2  Common Type System ....................................................................................................................... 11  
3.3.3  The .NET Framework class library .................................................................................................. 13  
3.3.4  Assemblies ....................................................................................................................................... 16  
3.3.5  ADO.NET ....................................................................................................................................... 17  
3.4  HTML AND CSS .................................................................................................................................... 17  
3.5  CHOSEN ARCHITECTURE ..................................................................................................................... 18  

4  INFRASTRUCTURE AT DIVA AND RICS ............................................................................................. 19  
4.1  INFRASTRUCTURE OF DIVA ................................................................................................................. 19  
4.1.1  What is JSF? ....................................................................................................................................... 19  
4.1.2  JSF Component ................................................................................................................................. 20  
4.1.3  JSF Request Processing Lifecycle .................................................................................................... 23  
4.2  INFRASTRUCTURE OF RICS ............................................................................................................... 24  
4.2.1  PHP .................................................................................................................................................. 24  
4.2.2  MySQL ............................................................................................................................................. 25  

5  SYSTEM DESIGN ..................................................................................................................................... 26  
5.1  A DIFFERENT APPROACH FOR DESIGNING WEB APPLICATIONS ...................................................... 26  
5.2  WEB ENGINEERING ............................................................................................................................. 26  
5.2.1  The waterfall model ........................................................................................................................... 26  
5.2.2  Evolutionary development ................................................................................................................. 27  
5.2.3  Component-based software engineering .......................................................................................... 27  
5.2.4  A WebE Spiral Model ........................................................................................................................ 28  
5.2.5  Prototyping Model ............................................................................................................................. 29  
5.3  CHOSEN SOFTWARE DEVELOPMENT MODEL FOR THIS PROJECT ................................................. 31  

6  REQUIREMENT ANALYSIS ..................................................................................................................... 32
6.1 SYSTEM BOUNDARY ..................................................................................................................32
6.2 STAKE HOLDERS OF THE SYSTEM ...........................................................................................32
6.3 USE CASE LIST ............................................................................................................................32
   6.3.1 Registration with System ........................................................................................................33
   6.3.2 Log in to the System ................................................................................................................33
   6.3.3 Publishing Specific Publications ............................................................................................34
   6.3.4 Publishing Any Publications ..................................................................................................34
   6.3.5 Search for Publications ...........................................................................................................34
6.4 FUNCTIONAL AND DATA REQUIREMENTS ...............................................................................34
   6.4.1 Functional Requirements .......................................................................................................34
   6.4.2 Data Requirements ................................................................................................................34
   6.4.3 Look, Feel and Use Requirements ..........................................................................................35
6.5 PERFORMANCE REQUIREMENTS ............................................................................................35
   6.5.1 Speed Requirements ...............................................................................................................35
6.6 SAFETY CRITICAL REQUIREMENTS .........................................................................................35
6.7 RELIABILITY AND AVAILABILITY REQUIREMENTS .............................................................36
7 IMPLEMENTATION AND TESTING ..............................................................................................37
   7.1 COMMUNICATION WITH DIVA .................................................................................................37
      7.1.1 Request.cs ............................................................................................................................37
      7.1.2 Response.cs ..........................................................................................................................38
      7.1.3 CookieManager.cs ................................................................................................................38
   7.2 EXTRACTING INFORMATION FROM DIVA AND PUSHING IT INTO RICS ................................38
      7.2.1 Publication.cs .......................................................................................................................39
      7.2.2 MyRegularExpression.cs .......................................................................................................41
      7.2.3 Connection.cs ........................................................................................................................42
      7.2.4 Database.cs ............................................................................................................................42
8 CONCLUSION AND FUTURE WORK ...........................................................................................43
   8.1 FUNCTIONAL REQUIREMENTS .................................................................................................43
      8.1.1 Simple and Nice Main Screen of Software ............................................................................43
      8.1.2 Saving Published Publication at DiVA into RICS ..................................................................44
      8.1.3 Registration of user ...............................................................................................................46
      8.1.4 Screen after Login ...............................................................................................................47
      8.1.5 The Researcher can view and edit his Profile .......................................................................47
   8.2 NON FUNCTIONAL REQUIREMENTS ..........................................................................................47
      8.2.1 Usability .................................................................................................................................47
      8.2.2 Reliability ...............................................................................................................................48
      8.2.3 Scalability ...............................................................................................................................48
      8.2.4 Security ..................................................................................................................................48
      8.2.5 Look and feel requirement .....................................................................................................48
      8.2.6 Availability .............................................................................................................................48
   8.3 FUTURE WORK ..........................................................................................................................48
      8.3.1 Webbrowser Control ............................................................................................................48
      8.3.2 IFrame ....................................................................................................................................49
9 REFERENCES ....................................................................................................................................50
MATEMATISKA OCH SYSTEMTEKNISKA INSTITUTIONEN ..........................................................54
1 Introduction
Each year researchers/students (referred to as users) publish their results in conference papers, journal articles thesis reports, books etc. (referred to as publications). These publications should be registered both at the Växjö University and the RICS Web sites, in a systematic way, so to avoid any kind of redundancy and errors. A publishing system (referred to as the system) following the principle "publish once - view everywhere" should help here.

When a user requests for registering a publication, the system should verify whether he/she is eligible to register a publication. If he/she qualifies, that is if he/she is a registered user, then the system will register the publication along with the required information like title, author, date of publication, kind of publication etc. on different Web sites.

1.1 Problem Description
At Växjö University there are already two systems exits. One is RICS (Research in Computer Science) and the other one is DiVA.

The RICS has all the information about the available thesis and also the thesis which has already been completed and published here.

The DiVA system also enables researchers/students to publish their research work here. It also enables its users to search different research made by the other researchers.

It is boring, error prone and time consuming to visit and fill forms at each system and publish research work. There is a need for a system by which researcher give information about his research work to only one system and the same information is available to the other system automatically, without filling any forms. That is if the research work is published at DiVA, that work will be available to the RICS database. Direct access to the RICS database is available but to the DiVA system database is not available.

The figure 1.1 depicts the problem. The figure 1.1 shows that researchers publish their publications to DiVA , but those publications are not available to RICS (Research in Computer Science).
1.2  **Designed Solution**  
The solution which has been designed for the problem mentioned in section 1.1 can be explained from the figure 1.2.

According to figure 1.2, a new system called Publication Information System should be designed which makes an HTTP request to DiVA and received HTTP response from DiVA. The Publication Information System then parsed that response to extract information about publications from DiVA. Then that information can be put into RICS database.
1.3 Goal of the Project
The goal of the project is to develop such a publication system, which
a. Helps the user to publish their research work to both Växjö University Library and RICS (Research in Computer Science) in a simple and easy way. Collect information about the publication from users once and publish this information to different already existing systems.
b. Helps to extract information about different published publications at DiVA and push that information into RICS database.
c. Helps the user to search different publications at DiVA.

The full details of the requirements of the system are given in the requirement analysis, section 6 of this report.

1.4 Motivation
There are many scenarios in which certain web applications have been designed for very specific purposes. For example, almost every organization has its own website and its own database. Such organization’s web application not only have information about there organization and employees but it also have information for the available vacant positions, for which the organization is looking for to hire people. Similarly, many organization, use to advertised there vacant positions to certain news papers. It will be very friendly and convenient, if we can designed an application, which has the ability to fetch information from these different sources to a single platforms. Through such an application, the user will be able to see information from different resources with out visiting to these resources individually and thus can help in tremendous way to not only save time but also reduce labor for capturing specific information from different resources.

DiVA also has been designed for a specific purpose. That is for saving and searching publications from researchers and students working at different research groups and different departments of the university. There are many research groups at Växjö University. These are
- CAFO - Centre for Labour Market Policy Research
- CeLeKT - Centre for Learning and Knowledge Technologies
- FPF - Forum for Profession Research (in Swedish)
- Forum for Gender Research (in Swedish)
- Forum for Intermedial Studies
- HERE - Higher Education Research and Evaluation
- ICMM - International Centre for Mathematical Modelling
- INID - International Network for Industrial Diagnostics
- LOIS Science Centre - LOFAR Outrigger in Scandinavia
- MiV – Citizen Participation and Social Inclusion
- RICS - Research in Computer Science
- WDAT - Wood Design and Technology
- WSCC - Web Services Competence Centre

The publications from these research groups must be saved at DiVA as well as to their own web sites. It is very boring, time consuming and error prone to give same information to twice. That is once for publishing at DiVA and once for publishing at the
research groups own websites. There must be a system by which once the publication is published at DiVA, it is available to different research groups.

The above idea can only be turn into realization if we could answer the following questions:

- Is it possible to share information between different independent systems without having direct access to database of these systems?
- Is it possible to share information between different, independent tested system without compromising at the integrity of data of these systems?
- If it is possible to share information between different independent tested systems, why develop new systems?
- Why not use already working and tested system, to develop new system?

The search for answers to these questions were the main motivation factor to pursue this project and thus to design a system by which two independent and already existing, tested system can share information between each other without compromising the integrity of data of these systems and without making a single change to these systems.

Apart from this, this research can further help in the field of web re-engineering research area.

1.5 Report Structure

Chapter 2 introduces concepts and tools which have been used to develop this project. Chapter 3 discusses different web technologies and their pros and cons. Chapter 4 discusses infrastructure of the existing systems about which this project is concerned. Chapter 5 discussed the different approaches for software development and the approach which has been chosen for this project. Chapter 6 discusses the requirement analysis phase of the project and defines functional and non-functional requirements for this project. Chapter 7 discusses the implementation phase of the project. It discusses the various classes which have been written for the implementation of this project. Chapter 8 concludes the outcome of the project and gives suggestion how can further be improved the application.
2 Background
There were many challenges faced in this project. The greatest challenge was the DiVA security. The DiVA is a secure system and it is very difficult to push and retrieve data from this system without direct access to its database.

This chapter provides review of these challenges and gives different alternative by which data can be extract and push into systems without direct access to the DiVA database.

2.1 Screen Scraping
Screen scraping, in terms of programmer mean fetching data from a website into your application. To talk more technically, it is actually a way by which application extracts data from the output of some other application. This technique is basically making the request and parsing the response. To extract information from DiVA we, use screen scraping.

2.1.1 Tools used for Screen Scraping
To perform screen scraping, we need help of different software. Among these are Firefox with firebug add INS. Through fire bug we can inspect the whole document that we receive as HttpResponse and then we can design different regular expressions to extract information of our interest.

The other software that is used extensively in this project is fiddler. Through this software we can monitor the traffic between the browser and the server, to which we want send or extract information. This tool helps to monitor different cookies send and received by the browser and thus we designed our code which can manage cookies for communication with the secure server.

2.1.2 Regular Expression
Regular expression (or regex, for short) is a tool, and like all tools, regular expression is designed to solve a very specific problem. The best way to understand regular expressions and what they do is to understand the problem they solve.

Consider the following scenario:

- We are searching for a file containing the text car (regardless of case) but do not want to also locate car in the middle of a word (for example, scar, carry and incarcerate).
- We are generating a Web page dynamically (using an application server) and need to display text retrieved from a database. Text may contain URLs and we want those URLs to be clickable in the generated page (so that instead of generating just text, we generate a valid HTML <a href> </a>).</a>.
- We create a web page containing a web form. The form prompts for user information including an email address. We need to verify that specified addresses are formatted correctly. That is to verify the information given by the user is syntactically correct.
We are editing a source code and need to replace all occurrences of size with isize, but only size and not size as part of another word.

We are displaying a list of all files in you computer file system and want to filter so that we locate files containing the text Application.

We need to search a file for some specific text, but only at a specific location (perhaps at the start of a line or at the end of sentence).

All these scenarios present unique programming challenges. And all of them can be solved in any language that supports conditional processing and string manipulation. But how complex will the solution of such task be? We will need to loop through words or characters once at a time, perform all sort of if statement tests, track lots of flags so as to know what we found and what we had not, check for white space and special character and more. And we will need to do it all manually.

But each of the preceding challenges can be solved using regular expression in a simple and easy way. We just define a pattern for which we are interested to capture it from a bunch of text and the rest of the work is done for us by the regular expressions.

In fact, all regular expressions either match text (performing a search) or match and replace text (performing replace).

2.2 C# Regular Expression
Here is a brief overview of the main classes and methods in the System.Text.RegularExpressions namespace. It does not cover Regex pattern in any real depth at all, but gives an introduction to the power of regular expression using C#. [9]

2.2.1 The C# System.Text.RegularExpressions Class
The Regex class contains the regular expression pattern and has a number of methods. The most commonly used of these are:

- IsMatch(string) - Returns True or False to indicate whether the pattern is matched in the string passed as an argument
- Match(string) - Returns 0 or 1 Match object, depending on whether the string contains a match.
- Matches(string) - Returns a Match Collection object containing zero or more Match objects, which contain all matches or none in the string that is passed as an argument
- Replace(pattern, string) - Replaces all instances of the regular expression pattern with the string
- Split(string, pattern) - Takes the pattern as a delimiter and returns an array of strings.

Most of these methods can be statically overloaded, so a new Regex object doesn't necessarily need to be created each time.

Other classes in System.Text.RegularExpressions
In addition to Regex, Match and Match Collection there are 6 other classes:
• Capture - Represents the text captured by a single set of parentheses (....) surrounding a sub expression
• CaptureCollection - Represents a collection of Capture objects
• Group - Represents the result of a single capturing group of paired parentheses
• GroupCollection - Represents a collection of Group objects
• RegexCompilationInfo - Provides information for the compiler to use to compile the regular expression to an assembly.
• RegexOptions - Specifies which if the available options are or are not set.

2.2.2 RegexOptions
These are the most common options
• None - specifies that no options are set
• IgnoreCase - matching is case-insensitive (default is case-sensitive)
• Multiline - Treats each line as a separate line for matching purposes. Consequently ^ matches the beginning of each line position, and $ matches the end of each line position
• Compiled - specifies whether the pattern is compiled to an assembly. Slower start-up, but faster for repeated use.
• IgnorePatternWhiteSpace - ignores un escaped white space with reference to the pattern, and allows comments preceded by #
• ExplicitCapture - Changes the capturing behavior of parentheses
• SingleLine - Forces the period character to match every character. Default behavior is that it does not match \n.

2.2.3 Regular Expression Patterns
Patterns are the key to successful use of regular expressions. Depending on the complexity of the problem that regex is used to solve, these can vary from a straightforward human readable string pattern, to a complex combination of string characters, meta characters, grouping and capturing characters. The real skill is in creating a pattern that does the job, so a thorough understanding of the usage of meta characters and in particular, the quantifiers is essential to efficient use of regular expressions. There is no real shortcut to learning this aspect of the discipline, although I found that unpicking regular expression patterns that I find employed successfully elsewhere a good start. One place where a lot of these can be found is regexlib.com, a repository for patterns donated by people bitten by the Regex bug.
3  Web Application Technologies
A web application is an application that runs on web server and is accessed by users over the Internet or a local intranet. Web application usually consist of static resources files (e.g. Images), web components, helper classes and libraries. A web browser is commonly used as thin client hence all the processing is done on the server. Web applications are usually organized in three-tier architecture – a user interface level, a functional process logic level, and data storage level. A web browser is the user interface level and dynamic web content technology such as CGI, ASP,ASP.NET or Java Servlets, is used in at functional (business logic) level. Data Storage is handled by a database.

Web applications are an extension of a web server. Web applications are either service oriented or presentation oriented. A presentation oriented web application produces interactive web pages containing mark up languages like (XML and HTML) and dynamic content in response to requests. Many of these open sources LAMP (Linux, Apache, MySQL and PHP). A service oriented web application then implements the endpoint of the web service.

3.1  Linux, Apache, MySQL and PHP (LAMP)
Linux, Apache, MySQL, and PHP/Perl/Python (LAMP) and Window, Apache, MySQL and PHP/Perl (WAMP) are a set of software increasingly being used to run dynamic web sites. Their popularity arises from the fact that they are basically free. These open source software can be easily downloaded from the net, or come bundled with Linux distributions. The LAMP server can be downloaded from [40].

3.1.1  Linux
The Linux operating system is an open-source operating system popular among web developers. Advantages of Linux as a server are its stability, reliability, low cost of implementation and security. Linux distributions usually include the Linux kernel, GNU libraries and tools, application software, and command line shells. Popular commercial distributions include Red Hat Enterprise Linux (run on DCS machines) and SUSE Linux. Non-commercial distributions include Debian GNU Linux and Fedora Core [4] [8].

3.1.2  Apache
Apache is very popular open source HTTP Web Server developed by the Apache Software Foundation. Apache is characterized by its highly configurable error messages, DBMS-based authentication databases and content negotiation and support for various GUI’s. Apache is also distributed in other proprietary packages like the Oracle Database and the IBM Web sphere application server. Apache includes a large set of modules such as mod Perl, a web proxy module, a URL rewriter and an authentication module. Also very useful, is the fact that Apache logs can be analyzed using a web browser and some freely available scripts. The latest currently available release is the Apache 2.X [5].
3.1.3 MySQL
MySQL is a multithreaded, multi-user, SQL relational database server. Programming
languages that can access a MySQL database include C, C++, Java, PHP, and Perl. The
MyODBC interface allows other programming languages which support the ODBC
interface to communicate with MySQL. MySQL runs on many different operating
systems including Linux and Windows. MySQL 4.1.10 offers a lot of improvement over
previous versions including transactions (with save points), SSL support, nested
SELECTS, ACID (atomicity, consistency, isolation, durability) compliance and Query
Caching. It is very important to note that it does not support Triggers or Cursors. It also
does not support Stored Procedures and Views. These have been left for future releases.
[21]

3.1.4 PHP
PHP stands for Hypertext Pre-processor. It is mainly used as a general purpose scripting
language used to develop dynamic web content and can be embedded in HTML. PHP can
be used as an alternative to Macromedia Cold Fusion, ASP.NET/C#/VB.NET and the
JSP/Java System. PHP is easy to use and is very similar to structured programming
languages like Perl. PHP is more than just a scripting language. It is a full programming
language and can be used from a command line and also be used to develop Graphical
User Interface Applications. PHP runs on many of the major operating systems, including
Linux and Windows and also supports many database systems, including MySQL. One
feature that leads to the popularity of PHP is that it is dynamically typed. Variables do
not have to be declared and they can hold any type of object. The arrays in PHP can hold
objects of different types, including other arrays. PHP includes many open-source
libraries and includes modules built in for accessing FTP and database servers. [7]

3.2 Java/J2EE
The Java 2 Platform Enterprise Edition (J2EE) provides developers with the tools and
Application Programming Interface's (API's) they need to create and deploy interoperable
web services and clients. According to Sun, “The J2EE platform simplifies enterprise
applications by basing them on standardized, modular components, by providing a
complete set of services to those components, and by handling many details of
application behavior automatically, without complex programming. [36]”. The Java 2
Platform, Enterprise Edition has full support for Enterprise JavaBeans components, Java
Servlets API and JavaServer Pages and XML Technology. Web components like
JavaServer Pages (JSP) and Java Servlets provide dynamic extension capabilities for web
servers. A client sends a HTTP request to a web server which implements the Java
Servlet and JavaServer Pages technology. The web server converts the request into an
HTTPServletRequest object which is delivered to a web component. The web
component interacts with JavaBeans or a database to generate dynamic content. A web
component produces a HTTPServletResponse object which is converted by the web
server into a HTTP response that is sent to the client. These web components can run on
the Tomcat Web container supplied in the Java Web Component Software Development
Package (JWSDP). Tomcat provides services such as life cycle management,
concurrency, security and requests as well as providing accesses for components to API's for transactions, email.

3.3 .NET Architecture
The .NET Framework is Microsoft’s platform for building different type of applications. It offers language independence and language-interoperability which is the most important and powerful aspect of the .NET platform. [44].

The .NET framework is composed of three main components. Common Language Runtime, Common Type System and a set of class libraries.

3.3.1 Common Language Runtime
The Common Runtime Language is responsible for loading and running .NET applications. The Common Language Runtime makes it possible to write an application partially in VB.net and partially in another .NET language like C#.

All the .NET language source code is translated to Microsoft Intermediate Code or IL (Intermediate Language) with CLR. The MIDL code needs to be interpreted and translated into a native executable. The CLR analogues to Java Runtime Environment (JRE), translates the MIDL code to target machine code with the help of JIT compiler.

This process, the compilation and execution of .NET language source code is shown by the figure 3.2.
Apart from language independence and language interoperability, it offers many features such as garbage collection, exception handling, cross language inheritance, debugging and side by side execution of different versions of the same .NET component.

### 3.3.2 Common Type System

The CTS defines all of the basic data types as well as the operation that can perform on these data types. It defines how classes, interfaces and primitive data type are represented. It is due to CTS that a class written in any .NET language can be extended by a class written in any other .NET language.

The Common Type System offers two basic classes of types. These are values types and reference type.

The value type is also called build in type that is supported mostly in all programming languages. Table 3.1 lists the value type supported by CTS and their corresponding keywords in VB.net and C#.net.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>VB.NET</th>
<th>C#.NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>A Boolean value (true or false)</td>
<td>Boolean</td>
<td>bool</td>
</tr>
<tr>
<td>Byte</td>
<td>An 8-bit unsigned integer</td>
<td>Byte</td>
<td>byte</td>
</tr>
<tr>
<td>Char</td>
<td>A Unicode (16-bit) character</td>
<td>Char</td>
<td>char</td>
</tr>
<tr>
<td>Decimal</td>
<td>A 96-bit decimal value</td>
<td>Decimal</td>
<td>decimal</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td>VB.NET</td>
<td>C#.NET</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Classes</td>
<td>A class is a data structure that may contain data members (constants, variables, and events), function members (methods, properties, indexers, operators,</td>
<td>Class</td>
<td>class</td>
</tr>
</tbody>
</table>

The reference type store a reference to the value’s memory address and are allocated on heap. The reference type supported by the CTS and their corresponding keywords in VB.net and C#.net are give in table 3.2.
Delegates
A delegate is a reference type that refers to a shared method of a type or to an instance method of an object.
Delegate
delegate

Arrays
An array is a data structure that contains a number of variables (elements of the array).
Dim MyArray(5) As Integer
myArray = new int[]

Interfaces
Interfaces are implemented by other types to guarantee that they support certain operations. An interface defines a contract. A class or structure that implements an interface must adhere to that contract.
Interface
interface

Pointers
Pointers reference blocks of memory. There are three kinds of pointers supported by the runtime: managed pointers, unmanaged pointers, and unmanaged function pointers.
N/A
int*

Table 3.2. CTS reference type.

3.3.3 The .NET Framework class library
The .NET Framework class library consists of a number of programming classes that help to achieve different programming tasks. These related classes have been grouped and are referred to as Namespace. The .NET Framework uses dot notation syntax scheme to describe hierarchical organization of classes. For example System.Data is the root level of ADO.Net classes. Two classes that are grouped at next level of System.Data class are System.Data.SqlClient and System.Data.OleDbClient. Each of these set of classes are independent. No function in the System.Data.SqlClient requires the System.Data.OleDbClient.

Microsoft provides a number of namespace as part of .NET framework. The classes of these namespaces provide big support to achieve different programming task in a simple and easy way.

Table 3.3 gives lists some of the .net Framework supported namespace and gives a brief description of these namespaces.
<table>
<thead>
<tr>
<th>Namespace</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>The most fundamental of all the .NET Framework Namespaces; it must be used by all applications, and it contains the classes that represent the basic data types.</td>
</tr>
<tr>
<td>System.CodeDom</td>
<td>Contains classes that are used to represent a source code document.</td>
</tr>
<tr>
<td>System.Collections</td>
<td>Contains classes used to manage collections of objects.</td>
</tr>
<tr>
<td>System.ComponentModel</td>
<td>Provides classes that control the design-time and runtime behavior of components and controls.</td>
</tr>
<tr>
<td>System.Configuration</td>
<td>Contains classes that enable your application to access the .NET Framework configuration settings.</td>
</tr>
<tr>
<td>System.Data</td>
<td>Provides support for ADO.NET and its database access classes and data types. The basic ADO.NET data management classes contained in the System.Data Namespace are the DataSet and DataTable classes that enable disconnected data access for Windows and Web-based applications.</td>
</tr>
<tr>
<td>System.Diagnostics</td>
<td>Contains classes that enable application to manage system processes as well as read the system event logs and performance monitor counters.</td>
</tr>
<tr>
<td>System.DirectoryServices</td>
<td>Contains classes that enable your application to access the Active Directory.</td>
</tr>
<tr>
<td>System.Drawing</td>
<td>Contains classes that access the system's GDI+ functions, which provide graphics support.</td>
</tr>
<tr>
<td>System.EnterpriseServices</td>
<td>Contains classes that provide access to COM+</td>
</tr>
<tr>
<td>Namespace</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>System.Globalization</td>
<td>Contains classes that enable National language Support (NLS) as well as support for Calendar objects.</td>
</tr>
<tr>
<td>System.IO</td>
<td>Contains classes that enable your applications to read and write to data streams either synchronously or asynchronously.</td>
</tr>
<tr>
<td>System.Management</td>
<td>Contains classes that provide access to the WMI (Windows Management Interface) infrastructure to provide systems monitoring and management support.</td>
</tr>
<tr>
<td>System.Messaging</td>
<td>Contains classes that enable application to read and write messages for the Microsoft Messaging Queuing technology.</td>
</tr>
<tr>
<td>System.Net</td>
<td>Contains classes that enable your application to conduct network communications using HTTP, as well as TCP and UDP sockets.</td>
</tr>
<tr>
<td>System.Reflection</td>
<td>Contains classes that enable the application to read the metadata of a loaded assembly.</td>
</tr>
<tr>
<td>System.Resources</td>
<td>Contains classes that enable your application to store and load regional-specific resources.</td>
</tr>
<tr>
<td>System.Runtime.ComplierServices</td>
<td>Contains classes that allow compiler developers to control aspects of the runtime behavior of the CLR.</td>
</tr>
<tr>
<td>System.Runtime.InteropServices</td>
<td>Contains classes that enable your .NET applications to interface with COM (Component Object Model) objects and native Win32 APIs (Application Program Interfaces).</td>
</tr>
<tr>
<td>System.Runtime.Remoting</td>
<td>Contains classes that enable application to manage remote objects required for developing distributed applications.</td>
</tr>
<tr>
<td>System.Runtime.Serialization</td>
<td>Contains classes that allow applications to store and load objects by converting them into a</td>
</tr>
<tr>
<td>Namespace</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>System.Security</td>
<td>Contains classes that enable your application to control .NET Framework security features. These classes can manage security features such as permissions, policies, and cryptography.</td>
</tr>
<tr>
<td>System.ServiceProcess</td>
<td>Contains classes that allow your applications to create, install, and manage Windows services.</td>
</tr>
<tr>
<td>System.Text</td>
<td>Contains classes that represent ASCII, Unicode, UTF-7, and UTF-8 character sets.</td>
</tr>
<tr>
<td>System.Threading</td>
<td>Contains classes that enable us to develop multithreaded applications.</td>
</tr>
<tr>
<td>System.Timers</td>
<td>Contains classes that permit application to raise an event following a specified interval of time.</td>
</tr>
<tr>
<td>System.Web</td>
<td>Contains a set of classes that define ASP.NET. These classes essentially provide support for Web browser to Web server interaction. For example, different classes contained in this Namespace support Web hosting, mail, security, and user interface components.</td>
</tr>
<tr>
<td>System.Windows.Forms</td>
<td>Contains the classes that enable the development of Windows-based applications.</td>
</tr>
<tr>
<td>System.Xml</td>
<td>Contains a set of classes that enable your application to work with XML documents.</td>
</tr>
</tbody>
</table>

Table 3.3 .NET Framework namespaces.

### 3.3.4 Assemblies

To make programming tasks easier, .NET introduces a new concept called ‘Assembly’. Assemblies are the smallest unit that can be individually deployed. An application can contain one or more assemblies.

Assembly consists of the following four elements:

1. Code, compiled into MSIL. This code could either an .EXE or .DLL.
2. Assembly manifest. It is a collection of metadata that contains information about assembly name, culture setting, and list of all files in the assembly, security.
identity, versioning requirements and references to the resources. It can be stored with the intermediated code or in a separate file.

3. Type metadata. Type metadata contains the CTS type that is used in assembly.

4. Resources. It contains the bitmaps, icons and other binary resources that are used by the application.

The basic idea behind assemblies is to solve the problems of versioning, DLL conflicts and simplifying the process of deployment.

### 3.3.5 ADO.NET

One of the most popular features of .net framework is ADO.NET. It supports disconnected environment for accessing database. ADO.net contains classes that help to store and retrieve data from the database as well as from an XML file.

In ADO.NET database access can be connection-oriented or connectionless. In the first case, a permanent connection to a data source is established. That is for doing some operations on a number of tables of the database, the application remains connected to the database. Thus when the number of users increases the performance of the application degrades. In the second case, a part of the database is fetched into DataSet object and then it is processed locally. The application only connects to the database for saving the transactions to the database. This significantly improved applications performance [44].

### 3.4 HTML and CSS

Hypertext Markup Language (HTML) is based on the Standard Generalized Language (SGML) [35]. HTML is a language for describing the structure of a document, not its presentation. HTML defines a set of common styles for web pages: headings, paragraphs, lists and tables. HTML provides a means by which a documents main content can be annotated with various kinds of meta-data and rendering hints. The rendering hints include specifying scripts, image maps and form definitions for web browsers. Macromedia Dreamweaver and Microsoft FrontPage are the leading software tools for editing HTML. Content and presentation can be combined using server side scripting languages like PHP and ASP to make the final HTML.

The four main kinds of markup elements in HTML are structural, presentational, hypertext and widget element markups. Structural markups describe the purpose of the text, an example being the heading markup `<h1> </h1>`. Presentational markups describe the appearance of text, though this has been superseded by cascading style sheets (CSS). Hypertext markup links a document to other documents. Widget elements markups are used to create objects such as buttons and lists. HTML documents normally start with a Document Type Definition Declaration (DTD – a set of declarations in a particular syntax that describe a type of XML documents in terms of the constraints on the structure of those documents).

An example DTD is

```xml
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

This means a document conforms to the strict DTD of HTML 4.01 which is purely structural, leaving formatting to CSS.
XHTML is the current version of HTML. It was produced as a result of the need for a stricter version of HTML. The changes from HTML to XHTML are minor and help increase the conformance with XML. The important changes are that the document must be well formed and all tags must be closed and semantically rich. In XHTML all tags must be written in lowercase. CSS is used by both authors and readers of web pages to define colors and fonts and other aspects of document presentation. Style rules combine HTML tags with properties to format the HTML tag. The HTML tag is used as a selector. The property and value of a selector is combined into a declaration. Style rules define the layout of a tag, as well as other typographic and design properties.

An example of a style rule is:

```

h1 { colour: red }
p { font-family: Arial, colour: blue }
```

This style rules specifies that the first major heading on the page (h1) be in blue and the paragraph be in Arial font and will be black.

### 3.5 Chosen Architecture

The Microsoft.NET architecture was chosen for the implementation. The IIS was configured and asp.NET with C#.NET was used.

The reason for this was:

- Asp.NET is rapid application development environment and provides a bundle of tools for accomplishing complicated task very rapidly.
- C#.NET is also famous for its build library, to request pages through HttpRequest and HttpResponse.
- MySQL is a fast database management system and is also easier to use than many other database system.
4 Infrastructure at DiVA and RICS
This chapter gives an overview of the infrastructure of already existing and working system at Växjö University. There are two main systems DiVA and RICS about which this project is concerned. The architecture of DiVA and RICS will be discussed here. The web technology in which these systems have been implemented will also be discussed in detail.

4.1 Infrastructure of DiVA
The DiVA system has been hosting on Apache Web Server and it has been implemented in JSF (Java Server Faces) technology with the support of jquery. The description of this technology has given below.

4.1.1 What is JSF?
JSF is User Interface (UI) Component framework for J2EE applications developed through Java Community Process (JCP). JSF is part of the J2EE standard specification and it is a top priority for every major J2EE tools vendor in the market (including Oracle, IBM, Borland and Sun) to support it. This guarantee a wide adoption and good tools support [46].

The JSF architecture is based on Model View Controller (MVC) design pattern. This model consists three elements- the view, the navigation model, and the application logic as show in figure 4.1.

![Figure 4.1 MVC with JSF architecture [46]]
JSF eases Java based web development. Here are some of benefits of using JSF:

- JSF framework offers a number of reusable components for creating user interfaces for the web applications.
- JSF has rich library tag for accessing and manipulating the components.
- JSF is a specification and vendors can develop the implementation for JSF.
- It offers a lot of GUIs which makes Java based web development simple and easy.
- It frees programmer to write event handler and thus reduces programmer’s burden.

4.1.2 JSF Component
The JSF component consists of five building blocks:

- **UIComponent**
The true power of JavaServer Faces lies in its user-interface component model. The UIComponent is responsible for the behavior and for accessing the data model. The UI comes in many forms and can be as simple as an output Label which simply displays text or as complex as dataTable which can represent a tabular data from collections of data such as from a database table.

- **Renderer**
It is responsible for the mark up rendered to the client for a specific component family.

- **RenderKit**
This is a library of Renderers with a common rendering technology (for example, HTML).

- **Render-specific component subclass**
The renderer-specific component subclass is convenience class and represents renderer-specific facets and attributes.

- **JSP Tag**
The default page description language is JSP, so JSF needs to follow the contract of JSP and provide JSP tags representing each JSF component.

- **Additional User – Interface Component Libraries**
Because of the richness and flexibility of the JSF API, many Java developers have started to create new JSF Component libraries and implementation.

    Oracle’s ADF Faces is fully compliant JSF component library which offers a broad set of enhanced UI Component for JSF application development. These include renders
per client type, advance tables, color and date pickers along with a host of general
components such as menus, command buttons, shuffle chooser and progress meters.

MyFaces is another popular and rich UI component library for JSF.

Figure 4.2: Oracle's ADF Faces JSF UI Components. [24]
Similarly ICEFaces is another another user interface libraries. DiVA system uses this component library.

ICEFaces provides a rich web presentation environment for JavaServer Faces (JSF) applications that enhances the standard JSF framework and lifecycle with Ajax-based interactive features. ICEFaces replaces the standard HTML-based JSF renders with Direct-to-DOM (D2D) renderers, and introduces a lightweight Ajax bridge to deliver presentation changes to the client browser and to communicate user interaction events back to the server-resident application. ICEFaces provides an extensive Ajax-enabled component suit that facilitates rapid development of rich interactive web-based applications.

The rich web presentation environment enables with ICEFaces provides the following features: [25]

- Smooth, incremental page updates that do not require a full page refresh to achieve presentation changes in the application. Only elements of the presentation that have changed are updated during the render phase.
- User context presentation during page updates, including scroll position and inputs focus. Presentation updates do not interfere with the user’s ongoing interaction with the application.

These enhance presentation features are completely transparent from the application development. Any JSF application that is ICEFaces enabled can benefit from it.
4.1.3 JSF Request Processing Lifecycle

A JSF components constructed page goes through a well define request processing lifecycle. This lifecycle consists of six phases. Restore view, Apply Request Values, Process Validations, Update Model Values, Invoke Application and Render Response. These are shown in Figure 4.4.

Here are the six phases:

1. **Restore View**: This phase is responsible for restoring the component hierarchy from the previous request and attaching it to the FacesContext. If no saved state is available, then the Restore View phase is responsible for creating a new UIViewRoot, which is the root node in the component hierarchy, and storing it on the FacesContext.

2. **Apply Request Values**: In this phase, each component has the opportunity to update its current state with information included in the current request.

3. **Process Validations**: This phase is in charge of processing any validation controls or converters attached to components in the component hierarchy.

4. **Update Model Values**: During this phase, all suitable model data objects will have their values updated to match the local value of the matching component, and the component local values will be cleaned.
5. **Invoke Application**: At this phase, any remaining events broadcast to the application need to be performed.

6. **Render Response**: This phase is responsible for rendering the response to the client and storing the new state for processing of any subsequent request.

### 4.2 Infrastructure of RICS

The RICS (Research in Computer Science) web system has been hosted on open source apache and it has been developed in PHP and MySQL. PhpAdmin has been used for managing database remotely. Below is the description of the technology that has been used in RICS web system.

#### 4.2.1 PHP

PHP stands for PHP Hypertext Processor. Hypertext refers to files linked together using hyperlinks, such as HTML (Hyper Text Markup Language) files. Preprocessing is executing instructions that modify the output. PHP is Open Source server side scripting language which has become incredible popular within the web development community. PHP’s early growth can be largely attributed to its open-source root as a free language that is highly compatible with other (also hugely popular) Open Source technology including MySQL, Apache and Linux.

PHP has a number of advantages over other web scripting languages including [38] [42]:

1. PHP can handle heavily trafficked website and perform common programming task.
2. The beauty of PHP is its ease of use, which greatly enhance productivity of development teams.
3. The most important is that PHP is free. It is open source software so every one can use it and can also make changes to its source code under open source license.
4. PHP is evolving since 1994 and hence thousands of developers have contributed to make PHP best server side scripting language.
5. PHP does not use a lot of system resources so it does not slows down server and run fast.
6. PHP has different security level which can be adjusted in the .ini file. These security levels prevent malicious attack on server.
7. PHP has tons of server interfaces database interfaces which enable it to load into Apache, IIS and other servers. Database interfaces are available for MySQL, MS SQL, Informix, Oracle and Plenty of others.

Millions of web sites across the world use PHP. Those include Cisco, Vodafone, Motorola, Siemens, Ericsson, CBS, Unilever, Philips, Air Canada, and Lufthansa.

The language being adapted by such large organization shows that PHP can be used in the enterprise level projects.
4.2.2 MySQL

MySQL is the most popular Open Source database management system, is developed, distribute, and supported by Sun Microsystems, Inc. [20]. It is Structured Query Language server designed for heavy loads and processing of complex queries. As a relational database system MySQL allows many different tables to be joined together for maximum efficiency and speed.

Some of its features are described below:

- Multiple CPUs usable kernel threads
- Multiple platform operation
- Numerous column types covering every type of data.
- Groups functions for mathematical calculations and sorting.
- Command that allow information about the databases to be easily and succinctly shown to administrator.
- Up to 32 indexes per table permitted.
- International error reporting and thus usable in different countries.

MySQL is the perfect choice for providing data via the Internet because of its ability to handle heavy loads and its advanced security measures.
5 System Design
This chapter presents the various design techniques and process available for building web applications. It explains the design technique chosen, showing its advantages and disadvantages. The tradeoffs between different designs are also pointed out.

5.1 A different approach for designing web applications
Traditionally, software has been broadly classified into different categories. Some of these categories include real-time software, personal computer software, artificial intelligence software and business software. Web-based systems and applications (such as web sites and information processing applications that reside on the Internet or an intranet) require a somewhat different method of development than these other categories of computer software. This is because web based systems involve a mixture of print publishing, software development, marketing, computing, internal communication, external relations, are and technology. Web applications are network intensive, content driven, continuously evolving applications. They usually have a short development time, need strongly security measure, and have to be aesthetically pleasing. In addition, the population of users is usually diverse. These factors all make special demands on requirements elicitation and modeling.

5.2 Web Engineering
Web engineering is the process used to create high quality Web-base systems and applications (WebApps). Web engineering (WebE) exhibits the fundamental concepts and principles of software engineering by following a disciplined approach to the development of computer-based systems, emphasizing the same technical and management activities.

The design and production of a software product (such as a web application) involves a set of activates or a software process. A software process model is an abstract representation of a software process. Three generic process models usually adopted in projects are:

5.2.1 The waterfall model
This model has distinct project phases which can be easily monitored. These phases are requirements specification, software design, implementation and testing.
5.2.2 Evolutionary development
An initial system is developed quickly from abstract specifications. This is later refined with the input of the user to produce a system that meets the user needs. It is an iterative model. Two refinements of this approach are the incremental and spiral models. The incremental model of evolutionary development delivers software in small but usable “increments”, where each increment build on those that have already been delivered. The spiral model couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.

5.2.3 Component-based software engineering
This is based on the existence of a large number of reusable components and is best suited in an object oriented environment.

Component based software engineering is in many ways similar to the object-oriented software engineering. A software team establishes requirements for the system to be built using conventional requirements elicitation techniques. An architectural design is established. Before going into detailed design task, the team examines the requirements to determine what subset is directly amenable to composition, rather than construction.
For each requirement, the team will ask [37]:

- Are commercial off-the-shelf (COTS) components available to implement the requirements?
- Are there any already developed reusable components available that can meet the requirements?
- Are the interfaces for available components compatible within the architecture of the system to be built?

The team will try to modify or remove those system requirements that cannot be implemented with COTS or in-house components. This can reduce the overall system cost and improves the time to market of the software system. It can often be useful to prioritise the requirements, or else developers may find themselves coding components that are no longer necessary as they have been eliminated from the requirements already.

The Component based software engineering process identifies candidate components as well as qualifies each component’s interface, adapts components to remove architectural mismatches, assembles components into selected architectural style, and updates components as requirements for the system change.

5.2.4 A WebE Spiral Model

The spiral model is a software development process combining elements of both design and prototyping in stages, in an effort to combine advantages of top-down and bottom-up concepts.

The spiral model was defined by Barry Boehm in his article ‘A Spiral Model of Software Development and Enhancement’ [32]. This model was not the first model to discuss iterative model, but it was the first model to explain why the iteration matters. As originally envisioned, the iterations were typically 6 months to 2 years long.

The process consists of 6 main stages, outlined below:

1. **Formulation**: This is an activity in which the goal and objectives of the web application are identified and the scope for the first increment in the process is established.

2. **Planning**: This stage estimates overall project cost, evaluates, risks associated with the development effort, prepares a detailed development schedule for the initial web application increment and defines a more coarsely granulated schedule for subsequent increments.

3. **Analysis**: This stage is the requirement analysis stage for the web application. Technical requirements and content items to be used are identified. Graphics design requirements are also identified.
4 **Engineering:** Two parallel sets of tasks make up the engineering activity. One set involves content design and production which is non-technical work. This involves gathering text, graphics, and other content to be integrated into the web application. At the same time, a set of technical tasks (Architecture design, Navigation design, and Interface Design) are carried out.

![WebE spiral model](figure5.2.jpg)

**Figure 5.2 WebE spiral model.** [43]

5. **Page generations:** This is the construction activity that makes use of automated tools for web application creation and the content is joined with the architecture, navigation and interface design to produce executable Web Pages in HTML.

6. **Customer Evaluation:** During this stage, each increment of the WebE process is reviewed.

5.2.5 **Prototyping Model**

A prototype is a working model that is functionally equivalent to a component of the product. [32]

In prototyping model an incomplete version of the required software is implemented. That incomplete version is called prototype. Prototype is functionally equivalent to the product or a part of the product. That is prototype either implements a subset of the final product or the implements the whole product. But the prototype implementation may be completely different from the final product.

There are many scenarios when the client is not sure what exactly he/she expects from the product. That is the client has a general view about that product but don’t know what exactly should he/she expect from the product. In such scenarios where no details are available from the client about the input to the system, the processing needs and the output requirements, the prototyping model may be the best choice.
This model tries to increase flexibility in the software development process by allowing client to interact and experiment with the working model of the product. This approach helps to define the real requirement of the product before starting to implement the final product.

The prototyping model evaluates client’s requirements by allowing clients to experiments with the prototype rather then to evaluate the user requirements based on descriptions. It also helps to determine the accuracy of the initial project estimates and can also decide if the proposed deadlines and milestones will be achieved or not.

Prototyping model involves the following steps:

1. **Identify basic requirements:**
   Basic input and output requirements are determined without going into detail.

2. **Develop initial Prototype**
   The initial prototype which contains only user interfaces is designed.

3. **Review**
   The client examines the prototype and gives feedback what to add or remove from the prototype.

4. **Revise and Enhance Prototype**
   The feedback receive on the prototype are studied and a new version of the prototype is built. If changes are introduced steps#3 and step#4 can be repeated.
   The use of prototyping can give the following benefits in software development.

   **Reduced time and cost**
   Prototyping can help in requirements elicitation and helps to define real requirements and specification of the software. This results into a happy client.

   Since it cost more to implement changes detected in the later stages of software development than to implement changes in the early stages of the software development, prototyping can significantly reduced cost and time for the software development.

   **Improved and increased user involvement**
   Since prototyping requires more user involvement, there are more chances to satisfy user’s desire for look, feel and performance. The user knows better the problem domain than anyone on the development teams does. Thus more interaction of user with software development results better quality of final product.
**Disadvantages of Prototyping**
The requirements elicited from the prototype may not be sufficient and thus it could result into waste in time as well as waste of money instead of increase in productivity and quality of software.

User can become stuck in debates over details of prototype, holding the development team and this can result a delay in the final product.

**Tools:**
The effective use of prototyping requires proper tools. The 4th generation languages like Visual Basic are frequently used since they are cheap, well known and relatively easy and fast to use.

Microsoft Visio is also commonly used software for prototyping development.

**5.3 Chosen Software Development Model for this Project**
The prototyping model was chosen to complete the project. There were several reasons for this choice. An important reason was that it illustrates the input data formats, messages, reports and the interactive dialogues to the customer. Prototyping provides the best mechanism for gaining better understanding of the customer’s needs. The first prototype for this project was not accepted, but that prototype revealed a number of facts which helped to develop the final software.

Prototype is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements. It is also an effective method to demonstrate the feasibility of a certain approach.
6 Requirement Analysis

Requirement Analysis is the process of understanding the customer needs and expectations from a proposed system or application and is a well-defined stage in the Software Development Life Cycle model.

Requirement is a description of how a system should behave or a description of system properties or attributes. It can alternatively be a statement of ‘what’ and application is expected to do.

The Software Requirements Analysis Process covers the complex task of eliciting and documenting the requirements of different users, modeling and analyzing these requirements and documenting them as a basis for system design. Software Requirement Analysis and Documentation Process are critical to software project success. Requirement Engineering is an emerging field which deals with the systematic handling of requirements.

This chapter gives overview of the requirement analysis of the Publication Information System.

6.1 System Boundary

Boundary statements help to separate the thing that is applicable to project from that area that is out of scope. So here are the project boundaries. The RICS web site and Växjö University Library will work as independent component. This project wills only allow the researcher/students in computer science to publish their research work to RICS and Växjö University. It will also provide search facilities for the publication which has been published on RICS and Växjö University.

6.2 Stake holders of the System

Who are the user or customer of an application are broadly referred to as ‘stake holders’. These indicate the group or group of people who will be directly or indirectly impacted by the new application.

By defining in concrete terms, who is the intended user, the Requirements Analyst knows in advance where he has to look for answers. The Requirements Elicitation Process should focus on the wish-list of this defined group to arrive at a valid requirement list.

The users of this Publication Information System are researchers and students of Computer Science Department.

6.3 Use Case List

The most common reason for IT project to be delivered late, over budget, or not at all is poorly defined requirements. Use cases allow specifying a more complete and accurate set of requirements in less time and in a way that is easily understood.

Actor Name

Researcher
Description
Researcher is an individual who is doing some research work at the university. It may be the faculty member of Computer Science department or student of Computer Science department.

Goals:
- Be able to register his/herself with the System
- Be able to register his/her work with System and with VXU Library and with RICS.
- Be able to search different research work already published with the RICS and VXU Library.

Actor Name
SystemAdmin

Description
SystemAdmin is an individual who is can save any kind of research work from DiVA to RICS.

Goals
- Be able to log in to the System
- Be able to register any kind of work with System and with VXU Library and with RICS.
- Be able to search different research work already published with the RICS and VXU Library.

6.3.1 Registration with System
Flow of Events
1. The use case begins when the Researcher request to be register with the system in order to use the system.
2. The system displays a form to the researcher.
3. The Researcher fills the forms and click register button.
4. The system makes sure that the necessary fields have been entered to the system and the user has given a valid vxu email.
5. The system then registers him/her.

6.3.2 Log in to the System
Flow of Events
1. The Researcher enters his VXU email address as id and the corresponding password.
2. The System verifies the user and grant access to the system if it is a valid user of the system.
6.3.3 Publishing Specific Publications
Flow of Events
1. The Researcher enters his/her id and password to the system.
2. The system verifies the Researcher as valid user and login him/her to the system.
3. The Researcher request for registering his/her works with the system.
4. The system displays him/her a wizard like form.
5. Researcher goes through the wizard form and fills the forms.
6. A message is shown to the Researcher that his/her research work has been published to both VXU Library and RICS web site.

6.3.4 Publishing Any Publications
Flow of Events
1. The SystemAdmin enters his/her id and password to the system.
2. The System verifies the actor as SystemAdmin and grants him access with privileges of SystemAdmin.
3. The SystemAdmin search DiVA for publications.
4. A list of publications is displayed to the SystemAdmin.
5. SystemAdmin can save any publication by clicking save button.

6.3.5 Search for Publications
Flow of Events
1. The Researcher has log in to the system.
2. The Researcher click search button to search for publication
3. A search form is displayed to him.
4. The researcher specifies the search criteria.
5. The system performs the search and display him list of publications.
6. The Researcher can further narrow this search by applying further search criteria.

6.4 Functional and Data Requirements
6.4.1 Functional Requirements
Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system is required to perform. Therefore this system has to provide:
• Registration of Researcher
• Registration of publications to RICS and VXU Library.
• Comprehensive search facility for searching different publication.
• Authentication of User

6.4.2 Data Requirements
The data requirement document provides detailed descriptions of the data model that the system must use to fulfill its functional requirements.
The Växjö University email and the password is the basic data, which the system requires from the user to fulfill functional requirements.

6.4.3 Look, Feel and Use Requirements
The system should have simple but an appealing look. It should use simple and light colors.

The system should always give feedback about the action that the user performed in a reasonable time. The feedback to the user should be in the form of a message which should be meaningful and understandable.

When the system interacts with the user, it should avoid any terms which are not familiar to the user. It should use real world conservations to give information to user about any of his/her action and this information should appear in a correct logical order. For example when the user fills a form and forget to enter his name and then enter email address in wrong format, the system should respond in the same order. That is the system must inform the user that he/she should enter his/her name and then he/she should enter his/her email address in a valid format.

The system should help the user so that the user does not have to remember information from going to one page to another.

The system should equally serve the experience as well as non experience users. The dialogues used in the system should be providing relevant information. Extra information wane related information and thus should be avoided.

6.5 Performance Requirements
6.5.1 Speed Requirements
To identify speed requirements, first it is needed to identify the most critical business transaction operations. These transaction operations should be identified in terms of
- Frequently used transaction
- Performance intensive transaction.
- Business critical transactions.

Applying the above thinking to Publications Information System, it turns out that the most frequent transactions are “search”, Registration” and “log in”.

The most performance –intensive transactions are “Search” and “view profile”. The most important business-critical transaction is “Save Publication to RiCS” which gets published publications from DiVA and save it to RICS database.

In the context of “search” operation the system should not take more than 20 seconds. Similar the login operation should not take more than 10 seconds. The most critical business operation is retrieving published publication from DiVA VXU Library to RICS database. This operation should not take more than 10 seconds.

6.6 Safety Critical Requirements
Requirements that define how the business intends to benefit from the system are called business requirements. In other words requirements that define how the risks to the
business from the system and other sources may be minimized are called business requirements. Thus the safety requirements should therefore be thought as BUSINESS REQUIREMENTS rather than system requirements.

Safety requirements are Business requirements that identify risks to the system, undesirable conditions to be avoided or fundamental safety features that must be included

a. At the same level as other business requirement for a system so should be expressed in an abstract way.

b. Specific functional requirement that define how risks are to be reduced and undesirable conditions avoided

c. Equivalent to system requirements – more detailed and specific

d. The more detailed safety requirements are generated from the abstract safety requirements.

Accidents or critical incidents rarely have a single cause. Generally, they arise because several events occur simultaneously e.g.

- Loss of data in critical system.
- User mistype command and instruct data to be deleted.
- System does not check or ask for confirmation of destructive action.
- No back up of data available.

Keeping in view the above, the publication Information system should provide for the following safety requirements:

- When the user adding or removing any information about his/herself or about his publication, the system should notify the user about what he/she is going to do and thus gets confirmation from the user to finalize that operation.
- In case of System failure, the system should rollback all transaction.
- If the transaction to one component succeeded but to other component could not succeeded, it should make a second attempt without involving the user to make the transaction successful to the component to which the transaction could not succeed.
- A failed or abort transaction should not create problem for the integrity of the data.

6.7 Reliability and Availability Requirements

The reliability refers to the ability of a computer-related hardware or software component to consistently perform according to its specifications.

Availability is the ratio of time a system or component is functional to total it is required or expected to function. This can be expressed as a direct proportion (for example, 9/10 or 0.9) or as a percentage (for example, 90%).

Therefore the publication information system should have property of availability and reliability.

The publication information system should provide reasonable amount of reliability and availability.
7 Implementation and Testing
A brief overview of the main algorithm of the system is given in this chapter. Some scripts that contain the core functionality of the system are explained concisely. Security related input validation, session management and cookies are also explained.

7.1 Communication with DIVA
For communication with DIVA two main classes were designed. These classes use httpwebrequest and httpwebresponse objects.

The classes are namely Request and Response. The object of this class can send a request to a web resource with either GET or POST method. A request with POST method is used for sending information either some credentials or sending a form’s data. A request with GET method is used for receiving some information from the external Web Server. The best feature of this class is that, when the external web server redirects the request, it follows automatically that redirect until it did not get the final resource. The getResponse() method is used for getting the final resource, given as answer to the request by the external Web Server. The Request and Response class is given below.

7.1.1 Request.cs
This class is used for communication with DiVA. It uses httprequest to send request to a web server for a web resource. When the request is made to the DiVA, that request is redirected to another location, that is why C#.NET httpwebrequest class object can’t be used directly. The class is design in such a way that automatically it traces the redirect location until it did not catch the final page.

```csharp
public Response getResponse()
{
    buildRequest();

    HttpWebResponse response = (HttpWebResponse)request.GetResponse();
    String statu = response.StatusCode.ToString();
    ccManager.StoreCookies(response);
    if (response.StatusCode == HttpStatusCode.Found)
    {
        url = response.Headers["Location"];
        this.method = HTTP_GET_METHOD;
        response = getRedirectUrl();
        ccManager.StoreCookies(response);
    }

    return new Response(response);
}
```
7.1.2  Response.cs
This class is designed to receive response from the DiVA server. This class has method which can read the source of html page into string which can be used for further processing. This class has method which can save the cookies and session of response into an object of cookieManager class object.

7.1.3  CookieManager.cs
DiVA is secure system, in order to communicate with DiVA the cookies and session must be managed. For this purpose CookieManager class is designed. It can store the cookies and session receives in response object. The following method of the CookieManager class is use to store cookies from response.

```csharp
public void StoreCookies(HttpWebResponse webResponse)
{
    for (int x = 0; x < webResponse.Headers.Count; x++)
    {
        if (webResponse.Headers.Keys[x].ToLower().Equals("set-cookie"))
        {
            this.AddRawCookie(webResponse.Headers[x]);
        }
    }
    webResponse = null;
}
```

The CookieManager class can also use to set coolies for the new request.

```csharp
public void PublishCookies(HttpWebRequest webRequest)
{
    StringBuilder sb = new StringBuilder();
    sb.Append("Cookie:");
    foreach (String key in this.CookieValues.Keys)
    {
        sb.Append(key);
        sb.Append("=");
        sb.Append(this.cookieValues[key]);
        sb.Append(", ");
        break;
    }
    webRequest.Headers.Add(sb.ToString());
    sb = null;
    webRequest = null;
}
```

7.2  Extracting Information from DiVA and Pushing it into RICS
The main challenging task in this project was how to extract information about different publications published at DiVA and push that information to DIVA web site. This task was achieved by creating two classes. One is Publication class and the other MyRegularExpression class.
This class is used for temporarily holding information about the publications which has been currently retrieved from DiVA. The class has methods for setting and getting information about the publications.

The code of this class is given below.

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace PubInfoSys
{
    public class Publication
    {
        private string authorofPub;
        private string tittleofPub;
        private string departmentforPub;
        private string pubType;
        private string yearOfPub;
        private string abstractofPub;
        private string link;

        public Publication()
        {
            authorofPub = null;
            tittleofPub = null;
            departmentforPub = null;
            pubType = null;
            yearOfPub = null;
            abstractofPub = null;
        }

        public String Author
        {
            set
            {
                this.authorofPub = value;
            }
            get
            {
                return this.authorofPub;
            }
        }

        public string Tittle
        {
        }
```
public string TitleOfPub
{
    set
    {
       this.tittleOfPub = value;
    }
    get
    {
        return this.tittleofPub;
    }
}

public string Department
{
    set
    {
        this.departmentforPub = value;
    }
    get
    {
        return departmentforPub;
    }
}

public string PubType
{
    set
    {
        this.pubType = value;
    }
    get
    {
        return this.pubType;
    }
}

public string YearOfPub
{
     set
     {
         this.yearOfPub = value;
     }
     get
     {
         return this.yearOfPub;
     }
}

public string AbstractOfPub
7.2.2 MyRegularExpression.cs

Microsoft .NET has rich regular expression facility. This class is designed to take full advantage of Microsoft .NET regular expression classes. In this class there are methods which contain regular expression correspond to the information needed to extract from the web page. So this class creates objects of Publication class, extract information by regular expression, and populate that information into the object of Publication class and save it in an ArrayList of publications. The ArrayList then can be retrieved and used to do processing on this information.

The MyRegularExpression class is given below:

```csharp
public void getInformationFromPage(String page)
{
    Publication pubObject = new Publication();
    pubObject.Author = this.getAuthor(page);
    pubObject.Title = this.getTitle(page);
    pubObject.Department = this.getDepartment(page);
    pubObject.AbstractOfPub = this.getAbstractOfPub(page);
    pubObject.Link = this.getLink(page);
}
```
pubObject.PubType = this.getPubType(page);
pubObject.YearofPub = this.getYearofPub(page);
pubObject.AbstractofPub = this.getAbstract(page);
pubObject.GetLink = this.urlLink(page);
this.alForPublication.Add(pubObject);
}

7.2.3 Connection.cs
This class is designed to provide database connectivity to our program for saving information into the database. The code for the class is given below:

```csharp
public class Connection
{
    string conStr = null;

    public Connection()
    {
        conStr = "server=localhost;DSN=test;UID=root;PWD=;DATABASE=RICS;pooling=false;
providerName=MySql.Data.MySqlClient";
    }

    public String getConnection()
    {
        return this.conStr;
    }
}
```

7.2.4 Database.cs
This class has different method which performs transaction to the RICS database. The class uses the Connection class for getting connection to the RICS data base and, different method of this class can be called to perform transaction to the database. The publication objects from the arraylist are sending to the insertPublication method of this class and then the method performs all the transaction to the database.
8 Conclusion and Future Work
This chapter contains brief description of what we have done so far and what can be added in the future. The overall description of the application is discussed here. This chapter also gives some suggestions on how could further be enhanced the features of the application or how could be solved the problem for which this software has been designed and implemented by other possible approaches.

We have achieved our goal by meeting the following functional and non functional requirements.

8.1 Functional requirements
Functional requirement define what the system need to do. Since the functional requirement is the major criteria to measure the success of any project; most of the time was spent on it. The following functional requirements were achieved successfully.

8.1.1 Simple and Nice Main Screen of Software
The application has a simple, but nice looking main page. This page gives some information about the system and instructs the user how to use the system.
8.1.2 Saving Published Publication at DiVA into RICS

The main challenging task was to design a solution by which published publications at DiVA is available at RICS web system. This task was achieved with a very simple, easy and friendly interface. Through this page all type of published publication can be retrieved and the user can view it. Upon clicking the save button, the retrieve publication from the DiVA is saved to RICS web system.
Figure 8.2 Saving Publication from DiVA
8.1.3 Registration of user

Researcher can make his/her account with a simple, easy and friendly form. Registration is compulsory so that only legitimate user can use the system. The following form screen shoot, show this.

![Registration Form]

**Publication Information System**

<table>
<thead>
<tr>
<th>Registration</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>First Name</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Last Name</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Email</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Repeat Password</strong></td>
<td></td>
</tr>
<tr>
<td><strong>School</strong></td>
<td>School of Mathematics and System Engineering</td>
</tr>
<tr>
<td><strong>Department</strong></td>
<td>Department of Computer Science</td>
</tr>
<tr>
<td><strong>Sex</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Academic Title</strong></td>
<td>Dipl.Inform.</td>
</tr>
<tr>
<td><strong>User Type</strong></td>
<td>Researchers</td>
</tr>
<tr>
<td><strong>Research Area</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Address</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Phone</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 8.3 Registration of user**
8.1.4 Screen after Login
When the user login to the system, users see the following screen.

![Screen after Login](image)

**Figure 8.4 screen when user login to the system**

8.1.5 The Researcher can view and edit his Profile
The researcher can view and edit his/her personal information.

8.2 Non Functional requirements
The non functional requirement describes how the functional requirements are achieved. Following are the non functional requirements which have been set for this project.

8.2.1 Usability
The system should not too be complex. It should be simple, easy and friendly. This non functional requirement has been achieve by providing simple and easy instruction to the user. When the user give invalid input the system, show an error message. For example if the user enter another email address then Växjö University email, the system give an
error message that email should be name@vxu.se or name@student.vxu.se. This makes the system more usable and friendly.

8.2.2 Reliability
Software Reliability is the probability of failure free software operation for a specified period of time in a specified environment. Software Reliability is also an important factor affecting system reliability. It differs from hardware reliability in that it reflects the design perfection rather than manufacturing perfection.

The Publication Information System can work effectively as far as DiVA system is working. As it made request to DiVA through http and bring information to the RICS.

8.2.3 Scalability
Since the Publication Information System is developed with full fledged object oriented approach, it is scalable. For example, the same classes can be used with minor changes to fetch information to the RICS from other system.

8.2.4 Security
The system has been made fully secure. Only the authorized, registered user can access the system.

8.2.5 Look and feel requirement
The system has a very simple, but nice and appealing look.

8.2.6 Availability
Since the system, interact with DiVA, there is no burden on the system. Publication information System will work as long as the DiVA system is running. The system is available as long as DIVA system works.

8.3 Future Work
Currently the system, retrieve the published information from DiVA and saved it into the RICS database. This is shown by the figure 1.2. In the figure it has been shown that the researcher publishes research work to DiVA and then the new system retrieves that information with HTTP request and response and saves it to the RICS database.

The system can be further improved if it is designed in such a way that it can publish directly publication to DiVA.

It could be done in the following two ways.

8.3.1 Webbrowser Control
The c#.NET has a window forms control called webbrowser control by which a web url can be navigated. This browser control can be used to automate filling a web form and
submission of it. The main challenging task in it is how to host web browser control on asp.NET application as web browser control is a window forms control.

8.3.2 IFrame
The second option is used a hidden IFrame on asp.NET page, load the DiVA web page in this IFrame, then use JavaScript to process the web page in the IFrame. The main challenging task in this is JavaScript can’t manipulate document loaded in IFrame from different domain. To achieve this task some research on cross domain java scripting must be made.
9 References

17th April, 2009

18th April, 2009

19th April, 2009

22nd April, 2009

23rd April, 2009.

24th April, 2009.

25th April, 2009.

26th April, 2009.

28th April, 2009

29th April, 2009.

30th April, 2009

1st May, 2009.

2nd May, 2009


7th May 2009

8th May, 2009.

10th May 2009.

11th May 2009.

12th May 2009.


17th May, 2009


23rd May 2009.

20th May 2009.

27th May 2009.

28th May 2009.

30th May 2009.

2nd June 2009.

3rd June 2009.
5th June 2009.
7th June 2009.
9th June 2009.
[34] http://www.freetutes.com/systemanalysis/sa2-prototyping-model.html#atuid-4ac9b2f91a8c06ad
12th June, 2009.
15th June, 2009.
16th June, 2009.
18th June, 2009
19th June, 2009
20th June, 2009.
10th July, 2009.
2nd August, 2009.
15th October 17, 2009.
