

Final Thesis

**Developing A Secure Web Service for License  
Management in StruSoft**

by

**Dave Alfanso Russell**

LITH-IDA-EX-05/019-SE

2005-02-28

Linköpings universitet  
Department of Computer and Information Science

Final Thesis

# **Developing A Secure Web Service for License Management in StruSoft**

by

**Dave Alfanso Russell**

LITH-IDA-EX-05/019-SE

Supervisor: Yuxiao Zhao  
Department of Computer and Information Science  
Linköpings universitet

Mats Ola Rasmusson  
Structural Design Software in Europe AB  
Box 30069, 200 61 Limhamn Malmö Sweden

Examiner: Kristian Sandahl  
Department of Computer and Information Science  
Linköpings universitet

## **Abstract**

As software increases in complexity and relies more on Internet and Web technology, the challenge of enabling interaction and communication between loosely coupled applications becomes increasingly vital.

Distributed computing presents challenges to loosely coupled applications that require means with which to interact and communicate. There exist technologies that are aimed at solving these problems; Web service is one such technology. Web service is a relatively new and rapidly maturing technology in the area of distributed computing; it offers a standards-based way to exchange information in an interoperable manner.

This thesis is done in partnership with StruSoft and attempts to provide a solution to their problem of distributed computing, by using Web service technology. The paper looks at distributed systems and various solutions to the problems associated with distributed computing. A comprehensive insight into Web service technology is provided, along with rationale as to why it is chosen for the project. In addition, there are guidelines as to how the necessary components of Web service are installed.

Development of License Management Software is also a part of this thesis. The software offers a means with which to store and maintain data about customers and their licenses.

Security is a major focus of this paper and thus extensively mentioned throughout. A detailed explanation of computer security is presented, along with the necessary configurations that are needed to make the Web service and the License Management Software more secure.

**Keywords:** Web service, XML, SOAP, WSDL, Computer Security, Distributed Systems, Open Source, Tomcat, Axis and Software Development Life Cycle.

## Acknowledgements

“No man is an island, no man stands alone.” In times when one embarks on an important project or drafts an important document such as a thesis paper this famous quote becomes startlingly true.

It is with much pleasure that I conduct my thesis work as part of the VIP Energiberäkningssystem project at the Structural Design Software in Europe AB (StruSoft). Special thanks to Mats Ola who made it possible for me to make such a contribution and also for all the information made available. Thanks to Kristian for taking on the responsibility as an examiner and for all the valuable contribution during this period. Very special thanks to Yuxiao my supervisor, for invaluable help and guidance throughout. Finally, to family and friends that offered support, thank you all.

Dave Alfanso Russell  
Linköping  
February 2005

# Table of Contents

1	Introduction.....	1
1.1	StruSoft.....	1
1.2	VIP Energiberäkningssystem.....	1
2	Background.....	3
2.1	Distributed Systems.....	3
2.1.1	CORBA.....	4
2.1.2	JAVA RMI.....	5
2.1.3	DCOM.....	5
2.2	Web Service.....	6
2.2.1	HTTP.....	7
2.2.2	XML.....	7
2.2.3	Enter the Phase of Web Service.....	9
2.2.4	SOAP.....	9
2.2.5	WSDL.....	10
2.2.6	UDDI.....	13
2.2.7	Web Service Stack.....	13
2.2.8	Web Service Tools.....	14
2.3	Security.....	15
2.3.1	Secure System/Communication.....	15
2.3.1	Realizing Security.....	16
3	Problem Formulation.....	18
3.1	StruSoft Mandates.....	18
3.1.1	Non-Functional Requirements.....	19
3.2	Design Decisions.....	19
4	Development Environment.....	22
4.1	Java configurations.....	22
4.1.1	Download.....	22
4.1.2	Installation and Set-up.....	22
4.1.3	Test.....	23
4.2	MySQL Configuration.....	23
4.2.1	Download.....	23
4.2.2	Installation and Set-up.....	23
4.2.3	Test.....	24
4.3	Tomcat Configuration.....	24
4.3.1	Download.....	24
4.3.2	Installation and Set-up.....	25
4.3.3	Test.....	25
4.4	Axis Configuration.....	25
4.4.1	Download.....	25
4.4.2	Installation and Set-up.....	26
4.4.3	Test.....	26
4.5	Open Source.....	27

	4.5.1	Historical View.....	27
	4.5.2	Open Source Fundamentals.....	29
	4.5.3	Pros and Cons of Open Source.....	31
4.6		Development Tools.....	33
	4.6.1	The Apache Project.....	33
	4.6.2	MySQL.....	34
	4.6.3	Eclipse.....	34
5		Discussions.....	35
	5.1	Software Design.....	35
	5.1.1	Waterfall Model.....	35
	5.1.2	Project's Software Processes.....	37
	5.2	License Server Architecture.....	40
	5.3	Web Service Architecture.....	41
	5.3.1	Why Service Oriented Approach.....	41
	5.3.2	Service Oriented Approach Defined.....	41
	5.3.3	Service Oriented Approach Operations.....	42
	5.4	Overall System Architecture.....	43
6		Security Issues.....	45
	6.1	Product Security.....	45
	6.1.1	License Server Security.....	45
	6.1.2	Web Service Security.....	49
	6.2	Open Source Security.....	53
	6.2.1	Expert Opinions.....	53
	6.2.2	Further Open Source Precautions.....	57
7		Testing.....	58
	7.1	Scenario Description.....	58
	7.2	Standalone Testing.....	59
	7.2.1	Evaluation.....	60
	7.3	Network Testing.....	61
	7.3.1	Evaluation.....	62
8		Delivery.....	64
	8.1	StruSoft.....	64
	8.2	Customer.....	65
9		Conclusions.....	67
	9.1	Future work.....	67
10		References.....	69
11		Glossary.....	71
		Appendix A License Server Screenshots.....	74
	A.1	Adding Customer to Database.....	74
	A.2	Search Database for Customer.....	74

A.3	Delete Customer from Database.....	75
A.4	Web Service.....	76
A.5	Email.....	76
Appendix B Web Service WSDL.....		78
Appendix C Related Documents.....		79
C.1	System Overview.....	79
C.2	Project Overview.....	79
C.3	Final Project Proposal.....	80
C.4	Project Time Table.....	80

# 1 Introduction

The age of the Internet and the Web have meant that networks have become larger, very distributed and extremely decentralized [1]. The Internet and Web revolution that have taken place have resulted in the emergence of new programming or language environment, script language and new technologies all allowing for more complex applications. An application need not run entirely on a single user system. Through the use of Internet and Web technology a system can be developed in a loosely coupled manner, where the pieces can be independently distributed.

As the revolution continues and the complexity of applications increases, the way information is exchanged between individuals, companies, and governments have and will continue to change. In spite of the changes that occur, users expect the information or service they require will be available in a reliable and secure manner. Complex applications in particular loosely coupled applications require that the pieces be able to communicate regardless of the underlying operation system or programming language environment. Simply put: there is a need for interoperability between connecting applications.

This thesis is part of an ongoing system development project called VIP Energiberäkningssystem at the Structural Design Software in Europe AB (StruSoft). The intended goal is to allow StruSoft the freedom to build a system that is loosely coupled, interoperable and secure. The resulting solution should provide a scalable and extendable system over the Internet which lends itself to access from a wide variety of communication devices.

## 1.1 StruSoft

The first of July 2002, saw the birth of StruSoft as a company following their spin-off move from Skanska, which they have been apart of for close to twenty years [2].

The niche market of StruSoft is the building trade sector and they develop applications that target that audience. The applications developed comes in two flavours: applications that are exclusively owned and held in trust by StruSoft and others that are developed as consulting agreements [2].

Their main products are the WIN-Statik programmes, the FEM-Design series and IMPACT [3]. Along with the range of products the company also offers various services. The services are Coach, FEM-Design courses and program support and upgrades [4].

## 1.2 VIP Energiberäkningssystem

VIP Energiberäkningssystem translates to VIP Energy Calculating System. VIP is a series of programs that are mutually compatible and integrated for the control and analysis of energy consumption. The entire series of programs utilize the same calculation module and climatic data to guarantee identical results regardless of where the calculation is performed [5].



## 1 • Introduction

The programs that make up the series are [5]:

1. VIP+ - A complete standalone application for analysing energy consumption of detached houses.
2. VIP WEB - A system that is highly adaptable to accept minimum input data for analysing and calculating results for a number of houses.
3. VIP\* - A project between StruSoft, Skanska, Cements and LTH financed by SBUF and FORMAS/BICs for management of energy calculation and inner temperature of a property during the whole planning, building and management phase.

## 2 Background

Connecting applications in a scalable, interoperable and secure manner is the main focus of this thesis, but prior to going into details, it is necessary to provide some background information on the important concepts of distributed systems and computer security.

### 2.1 Distributed Systems

The Internet and Web revolution, in addition the changes that followed brought to the surface the important concept of distributed system. So what is a distributed system?

One humorous, simple but somewhat accurate description is “A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable [6].” A more serious definition describes distributed systems as a collection of computers that communicate via a network and present themselves to the users as a single coherent system [7].

Four of the major solutions to the challenge of application-to-application communication in a distributed system (here in referred to as distributed computing) are CORBA, Java RMI, DCOM and Web Service.

The remainder of the section gives a brief description of the first three. A detailed look at Web Service is presented in the next section entitled Web Service.

#### Classification of Distributed System

Before explaining the four major contributors to distributed computing, a brief look at how these systems could be classified is necessary. Computers use two kinds of data files when storing, transferring and/or accessing data: binary files and text files. One way in which distributed system can be classified is according to the type of file that is passed on the wire: binary versus text.

**Binary Files** – At its simplest, binary files are streams of bits (1’s and 0’s). Applications uses proprietary format to represent information in binary form and this therefore means that these binary files can only be understood and created by certain programs. This introduces the one problem associated with binary files; their proprietary nature means they’re not easily understood by other programs.

**Text Files** – This format also has streams of bits. The differing factor is that text files are grouped together in a structured way so that at any given time they represent numbers. These numbers are then further mapped to characters. This means that this format can be read by numerous applications and humans alike (with the aid of a text editor); therefore the sharing of information is easier.

CORBA, Java RMI and DCOM all convert data to binary format before it is sent, making it crucial for an application or system that knows this binary format to be used on the receiving side. Web Service on the other hand uses text format and send text files on the wire. This makes Web service a very comfortable option when communicating between applications, as

it uses a text format that is a standard and easily understood. It must be also said that binary files can also be sent when using Web Service [8].

### 2.1.1 CORBA

The acronym CORBA stands for Common Object Request Broker Architecture. It is described as an open, vendor-independent framework that permits distributed computer applications to interoperate over networks. Fundamental to CORBA is the Internet Inter-ORB Protocol (IIOP). With the use of this protocol CORBA-based program can interoperate with another CORBA-based program, allowing for distributed, heterogeneous collection of objects to interoperate [9].

The Object Management Group (OMG) is responsible for defining the CORBA standard.

#### CORBA Architecture

Central to CORBA is the use of objects; it defines an architecture of distributed objects. The basic CORBA model is that of a request for service(s) of a distributed object [10].

All services offered are exposed by way of an interface and clients must invoke this Interface Definition Language (IDL) in order to use the service.

With CORBA each object instance has a unique object reference. Clients use this reference to send their invocations to the Object Request Broker (ORB). The ORB is the service that implements the request to the remote object. It acts as a bus that finds the remote object, communicates the request, waits for the result(s) and then sends back the result(s) to the client. The ORB is independent of the programming language from the client and therefore the two can be different.

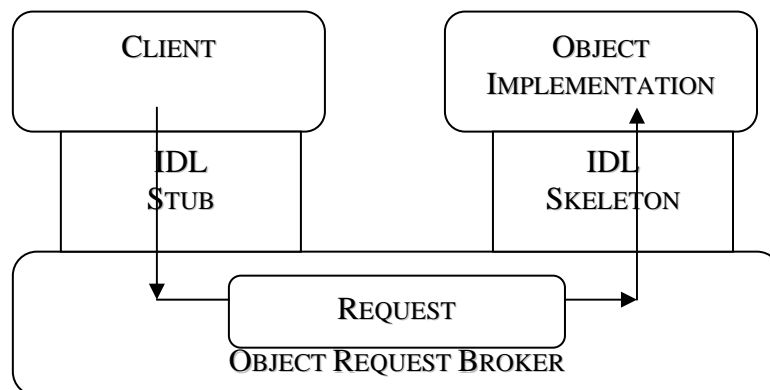


Figure 1: CORBA Architecture.

Figure 1 graphically depicts a request. A client holds an object reference to a distributed object. The object reference is typed by an interface and sent via IDL. The Object Request Broker, or ORB, delivers the request to the object and returns any results to the client using IIOP.

For a more detail look at CORBA, check the OMG home page [11].

### 2.1.2 Java RMI

The acronym RMI stands for Remote Method Invocation (RMI). Java RMI enables software developers to create distributed applications by facilitating object communication between distributed Java-based applications in which calls are made between Java Virtual Machines (JVMs). The protocol on which Java RMI relies is the Java Remote Method Protocol (JRMP). It is with this protocol that the Java RMI delivers connectivity and interoperability.

Sun Microsystem is the company behind Java RMI.

#### Java RMI Architecture

Java is an object-oriented language and therefore objects form the base for Java RMI. The architecture is similar to that of CORBA in that it uses distributed objects. It is based on the important principle that the definition of behavior and the implementation of behavior be separate. This allows for a model that focuses on providing service [12].

The key to offering service in Java RMI is that services must be coded as Java interfaces.

With objects playing such a central role in Java RMI there is a need for marshalling. The Java Object Serialization handles this functionality, which marshal and unmarshal parameters and return values.

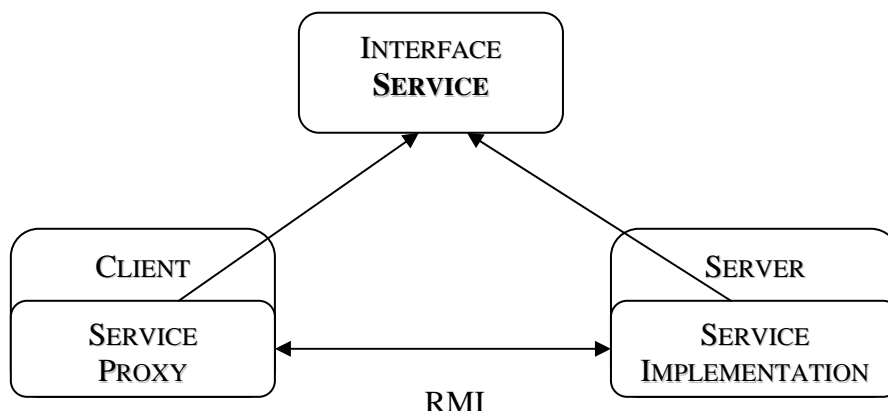


Figure 2: Java RMI Architecture.

Figure 2 above illustrate RMI communication. A server implements a service and announces it through an interface. A client holds an object reference to a distributed object. The object reference is typed by an interface and sent to the JVM on the server. The servers JVM then delivers the request to the object and returns any results to the client using JRMP.

If you wish to read more about Java RMI go to Sun's Java RMI web site [12].

### 2.1.3 DCOM

The acronym DCOM stands for Distributed Component Object Model. The DCOM solution enables applications to interact directly over a network [13]. It supports remote objects through the use of a protocol called Object Remote Procedure Call (ORPC). Through the use of this protocol applications can expose objects to computers across networks and others can communicate with these objects across network boundary.

DCOM is Microsoft's contribution to solving the problems of distributed computing.

#### DCOM Architecture

The concept of objects lends itself well to distributed computing and it is no surprise that fundamental to DCOM is the use of objects. It is comparable to CORBA in terms of offering a set of distributed services [14].

As with the previous two approaches DCOM also uses interfaces. A server object publishes the behavior that it offers through an interface. The client invokes the IDL when a behavior is called.

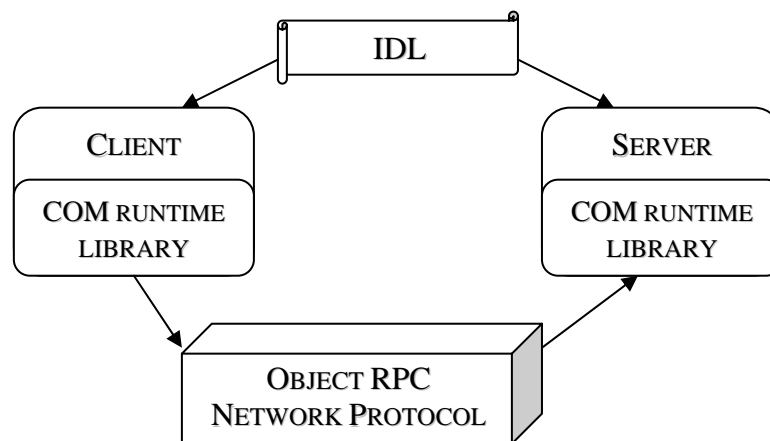


Figure 3: DCOM Architecture.

Figure 3 gives a simplified overview of the DCOM architecture. A server offers a service by defining an interface. The client calls the method of the server by acquiring a pointer to the interface. The client then sends a call to the COM runtime, which communicates with the servers COM runtime. This communications take place over the ORPC.

For further reading on DCOM visit the Microsoft page for DCOM [15].

## 2.2 Web Service

Web Service is the latest contender in the distributed computing arena aimed at providing application-to-application communication across network boundaries. It should not go unsaid that the actual definition of Web service is a hot topic of discussion among industry experts, development communities and various computing organizations. However regardless of the arguments put forward the basic concept remains: Web service refers to the enabling of interaction between software in a platform and programming language neutral manner.

Before delving into more detail about Web Service, it is important to take a brief look at two important building blocks: HTTP and XML.

### 2.2.1 HTTP

The acronym HTTP stands for Hypertext Transfer Text Protocol. Web Service is not specific to HTTP but the protocol is often the chosen transport protocol and it is referred to in many RFCs on Web Service.

The birth of the HTTP protocol was synonymous with that of the Web and date to the early 1990s. HTTP is used to control the delivery of hypertext data. It is a simple stateless response-request text-based protocol intended for sending and receiving text-based information. HTTP has become and still is the protocol of choice for the Web. The format of the HTTP packet includes a header followed by a body, which holds the data. HTTP 1.0 was the first version of the protocol and version 1.1 is the current version that extends the features of the previous by offering among other things stateful connection [16].

### 2.2.2 XML

The acronym XML stands for eXtensible Markup Language. XML can be loosely thought of as a “watered-down” version of Standard Generalised Mark-Up Language (SGML); a simplified version if you wish. It is essentially a meta-language, allowing designers the ability to represent data in a self-describing manner. XML has a hierarchical structure and uses customised tags, allowing for definition transmission, validation and interpretation of data between applications and organisation [17].

XML is one of the fundamental building blocks on which Web Service is built, thus a clear understanding of XML is essential. Below is the exploration of four important aspects of XML: XML Syntax, XML Infoset, XML Namespace and XML Schema.

#### XML Syntax

The rules that govern XML are very simple but very strict. Central to XML is the concept of entities and tags. An entity is enclosed within a tag. All tags must be closed; tags are case sensitive and may contain attributes [18].

Code 1 shows a simple xml file. The first line of the document declares the files to be a XML file, this is called the XML declaration – it defines the version the document is using. The next line describes the root element – all XML document must have a root – “letter” – this is the dominant element in the document and there can only be one root. The syntax for comment is given on line 3. Lines 4 - 6 are referred to as child elements. Code 1 clearly shows the hierarchical nature of XML and also shows tags has a closing tag.

---

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<letter>
  <!-- xml comment -->
  <to>Alice</to>
  <from>Bob</from>
  <body>Just saying hi.</body>
</letter>
```

---

Code 1: XML syntax

## XML InfoSet

The acronym InfoSet stands for Information Set. The XML InfoSet is used for describing the common set of items that are relevant across many XML specifications. It defines a consistent and rigorous set of terms that other specifications can use to refer to the information in a well-formed (but not necessarily valid) XML document [19].

In other words, it is a collection of information items that comprise a description of a particular XML document. An InfoSet is an abstract representation of chunks of an XML document that is characterized by certain properties. Any document that is well-formed and meets XML namespace constraints described in the Namespace specification has an InfoSet. A valid document may also have an InfoSet.

Four of the InfoSet described by the World Wide Web Consortium (W3C) in the recommendation are [19]:

- Document Information Item – Always exactly one, which is the root.
- Element Information Item – There is one element information item for each element in the document, this includes a unique document element.
- Processing Instruction Information Item – There is one for each processing instruction.
- Comment Information Item – There is one for each comment in the instance document, but not for comments in the Data Type Definition (DTD).

## XML Namespace

XML namespaces provide a method to avoid the conflicts in naming of elements/attributes by different people. XML has an attribute call “xmlns” which can be used to specify a namespace. If this element is omitted, then the elements in the document belong to the default namespace. Format – xmlns: namespace-prefix=“qualifiedname”.

Code 2 shows an example of using namespace. Code 2 uses code 1 but there are conflicts with the name of the child elements. In order to solve the conflicts, code 1 is given the namespace “code1” and when one wishes to use an element of code 1, the namespace to the element must be added – example “<code1:body>”.

---

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<brev xmlns:code1="http://code1">
  <to>Bob</to>
  <from>Alice</from>
  <code1:body>Just saying hi.</code1:body>
</brev>
```

---

Code 2: XML Namespace

## XML Schema

XML schema (XSD) is used to define the type and structure of an XML document. Like DTD, XSD restricts well-formed XML documents. Compared to DTD however, XSD is a

formal XML document and keeps the type inheritance in object-oriented way. An application can validate an XML document according to the XSD by using a XSD-supported parser.

Code 3 gives an example of a XML schema that could be used to govern code 1.

---

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<schema xmlns="http://www.w3c.org/2001/XMLSchema">
  <complexType name="letter">
    <element name="to" type="string"/>
    <element name="from" type="string"/>
    <element name="heading" type="string"/>
    <element name="body" type="string"/>
  </complexType>
</schema>
```

---

Code 3: XML Schema

### 2.2.3 Enter the Phase of Web Service

As said earlier in the section, Web service infrastructure is about enabling distributed computing in heterogeneous environments.

A Web service is a modular, self-describing application that supports interoperable application-to-application communication over a network. It offers an interface that is described in text-based format - XML (namely WSDL). Other systems interact with the Web service via the WSDL regardless of operating system, programming language or environment over a protocol of choice, mainly HTTP.

Web service should not be thought of as a specific technology; rather it is a collection of existing established and emerging communication protocol. The cornerstones of Web service are HTTP and XML, both of which are established protocol along with newly emerging protocols such as SOAP, WSDL and UDDI. In the remaining sections these three emerging protocols will be detailed.

### 2.2.4 SOAP

The acronym SOAP stands for Simple Object Access Protocol. SOAP is a XML based protocol that is simple and lightweight. It is used to facilitate information exchange in a decentralized distributed environment [20]. The protocol specification describes an extensible framework for passing parameters and commands (XML messages) between clients and servers.

The World Wide Web Consortium (W3C) is responsible for the SOAP specification with the SOAP 1.1 being the current version (originally introduced in 2000). Version 1.2 has been proposed and is now recommended by W3C. SOAP sits at the Application layer in the protocol stack and offers an operating system, programming language and environment neutral XML based communication.

It is important to note that SOAP does not define a transport protocol to use for sending SOAP messages, but provides some binding mechanisms. This is the task of the application developer. A developer can therefore choose from any of the many transport protocol including HTTP, SMTP and FTP.



## SOAP Messages

SOAP provides an extensible framing mechanism for XML messages. All SOAP messages are XML documents which conform to a schema set forth by W3C [21]. They contain proper namespaces on all elements and attributes. SOAP messages consist of three sections: SOAP Envelope, SOAP Header and SOAP Body. Code 4 shows the syntax of a SOAP message.

---

```

<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2003/05/soap-
  encoding">
  <soap:Header>
    <!-- extensible headers -->
  </soap:Header>
  <soap:Body>
    <!-- payload -->
    <soap:Fault>
      <!-- fault body -->
    </soap:Fault>
  </soap:Body>
</soap:Envelope>

```

---

Code 4: SOAP syntax

**SOAP Envelope.** This is the top or root element of the XML document representing the message. The element must be given the name “envelope” and is a required element in a SOAP message. Inside this element the document may declare namespaces. The element is the container for the SOAP header and SOAP Body.

**SOAP Header.** This element is optional but if present it must be the first immediate child element in the SOAP message. The element name must be “header”. The intended purpose of the header is to encapsulate extension. Any additional functionality to be included in the protocol will be added to the header, and therefore will not affect the specification. This includes extensions that handle security and routing intermediaries. Use of extra functionality of the header has not been specified in the RFC, so to use extra functionality must be some agreement between the sender and receiver.

**SOAP Body.** This element must be present in a SOAP message. As with other elements the name “body” must be used. The body is the location for application specific data. It contains the payload of the message. The payload of the message is serialized according to the chosen convention and encoding. A payload can specify any encoding it desires. If the header element is missing then the body is the first child.

**SOAP Fault.** Carries error and/or status information and must be a child of the body if present.

### 2.2.5 WSDL

The acronym WSDL stands for Web Service Definition Language. With the prospect of application-to-application communication, there are unanswered questions. Having created a service that has been published, how does a potential user know what method to call, the parameter(s) to send if any, what return type to expect, the transport protocol to use and so forth? WSDL is the answer!

WSDL is an arrangement for relating the Web service interface in XML. WSDL is to Web service what IDL is the CORBA. Web service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in application communication. The document produced as a result of the recipe is a platform and language neutral XML document. Simply put, a WSDL document says: This is service “A”. You can find me at endpoint “B”. To use me you need to do “C” and expect “D” in return. And so forth.

### WSDL Document

An agreement of some sort must exist between a server and a client for the service(s) to be consumed. WSDL is where one would find most of these agreements.

The document contains abstract elements that define components in a platform and language neutral manner [22]. The abstract elements of a WSDL document include:

- **Type** – This element holds the data type definitions or schemas that are relevant to the exchanged message. WSDL prefers the use of XML schema but there is no restriction
- **Message** – Consists of one or more logical parts. Its represents an abstract definition of the message to be transmitted.
- **Operation** – Provides an abstract definition of each action support by the service.
- **Port Types** – This is a collection of abstract operation. Basically, it is a grouping of all the methods that can be called. It gives the input (parameter) and the output (return value).

Concrete elements binds abstract elements to concrete protocol, data format specification and endpoint.

- **Binding** – Defines the message format and protocol for the operations and messages defined by a Port Type. Here it can only specify one protocol.
- **Port** – It defines individual endpoints by specifying a single address for a binding. This is where the service is located.
- **Service** – This is a grouping of related ports. It is important that none of the ports communicate with each other.

Code 5 shows the basic structure of a WSDL document.

---

```

<wsdl:definitions name="nmtoken" targetNamespace="uri">
  <import namespace="uri" location="uri"/>
  <wsdl:documentation .... />

  <wsdl:types>
    ...
  </wsdl:types>

  <wsdl:message name="nmtoken">
    ...
  </wsdl:message>

  <wsdl:portType name="nmtoken">
    <wsdl:operation name="nmtoken">
      <wsdl:input name="nmtoken" message="qname">
        ...
      </wsdl:input>
      <wsdl:output name="nmtoken" message="qname">
        ...
      </wsdl:output>
      <wsdl:fault name="nmtoken" message="qname">
        ...
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="nmtoken" type="qname">
    ...
  </wsdl:binding>

  <wsdl:service name="nmtoken">
    <wsdl:port name="nmtoken" binding="qname">
      ...
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>

```

---

Code 5: WSDL syntax

## Web Service Messaging

Four different interaction scenarios between a client and a server of Web Service have been defined [22]. The four are as follows:

1. **One-Way** – The endpoint receives a message. Client sends message.
2. **Response-Request** – The endpoint receives a message and returns a response. This is the most common type where a client sends a request and the server replies.
3. **Solicit-Response** – The style is the same as Response-Request where the endpoint sends a message and receives a response. Here it is the server that sends the message and awaits a response.
4. **Notification** – The endpoint sends a message. Server sends message.

### 2.2.6 UDDI

The acronym UDDI stands for Universal Description Discovery and Integration. Up to this point, some important aspects of Web service has been discussed. Web service use WSDL documents to describe the service(s) on offer. A service requestor is then able to utilize the WSDL to establish communication with the service. It has also being stated that Web service communication builds on the SOAP protocol, which is transport protocol independent. This leads to the question: How does one find the Web service(s) being offered? To this question there exists no single answer.

One possible way to find the location of a service (or endpoint), is to explicitly communicate the endpoint address (URL) to the application or the application's developer(s). This could be done by: sending an email, writing applications that know the intended endpoint address, publish the information on a Web site and so forth.

Another approach is to use UDDI. UDDI is a platform-independent framework for describing services, discovering businesses and integrating business service by using the Internet [23].

A good analogy of UDDI is that of a telephone directory. If you want to find a service simply check the telephone directory, it provides a listings of registered services. Like a telephone directory the UDDI has a concept of yellow pages; where a search can be performed by categories: location, industry or product. There is a white pages section where more detailed information on a service provider such as name, email address, location and so forth is given. There is also a green page that contains more technical information about the service.

UDDI is intended to be used by providers, who wish to make their service(s) public in a standardized manner. If a service provider wants the service to remain private (possibly for security reasons), then it would be best suited to take the first approach of explicitly communicating the endpoint address.

This thesis did not implement a UDDI as the service will only be offered to a limited customer-base. For this reason a more detailed explanation of UDDI will not be given but for further reading on UDDI see UDDI.org [23].

### 2.2.7 Web Service Stack

With Web service being a relatively new technology, it is recommended that developers become familiar with the many layered and interrelated technologies that make up the Web Service stack (architecture). The Web service stack provides a visual representation of these layers and technologies. The technology can be created and used in many ways, thus there is no single concise way to represent the stack. The Web Service stack shown is figure 4 is taken from the W3C [24].

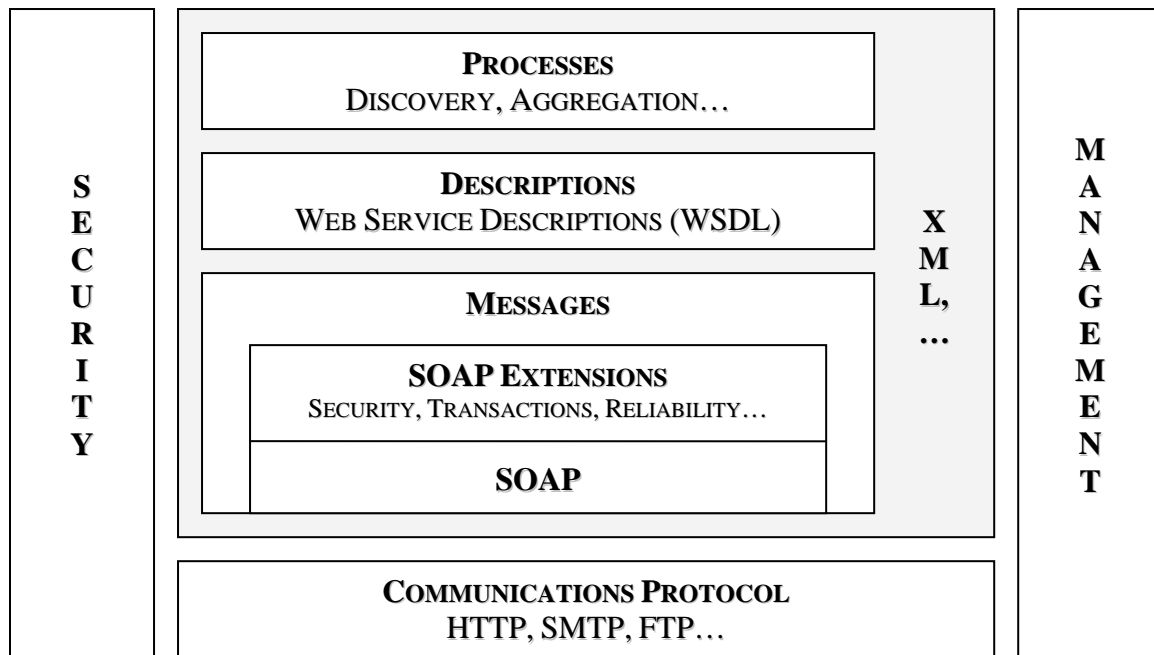


Figure 4: Web Service Architecture Stack

One cornerstone on which Web service is built is XML and this should be evident from viewing figure 4. At the lowest level of the stack there lies a myriad of Transport Protocols over which Web service messages are sent. All Web service messages are sent using SOAP, which runs over the transport protocol of choice. WSDL is used to describe the service(s) on offer. The stack also involves processes like discovery, which is handled by UDDI. All the components used within the Web service arena are built on XML.

Much work and research have already taken place on Web service but much more is on going and some yet to come. The technology is still in its infant stage and is waiting for various proposals to be approved by the Internet governing bodies. There is a lot of focus around Web Service Security (WSS), XML Signature and XML Encryption among others. Security does not stop with Web Service and its related components; security in the various communication protocols is also a concern.

The point is again made that, Web service is not a single technology but rather a collection of existing and emerging technologies to deliver services that are interoperable.

### 2.2.8 Web Service Tools

Although some of the protocols related to Web Service have only being around for a short time, there are already a number of implementations on the market.

In the Java community, there is the Apache Axis project (originally donated by IBM) and the chosen tool for this thesis. IBM Web Service Toolkit is yet another implementation on offer. HP product e-Speak is a Java implementation of Web service. Sun Microsystems has available also a Web Service Development Pack.

Microsoft's latest platform .NET is their implementation of Web Service. One can also find toolkit in such language as C++ and Perl.

Despite the fact that Web service is a relative newcomer to the market, it has without a doubt managed to catch the attention of the industry, which is evident with the support been gathered from small, medium to big industry players.

## 2.3 Security

An important requirement around which this thesis is centred is that of security. The security requirements arise from the fact that the application-to-application communication takes place across the Internet and the Web when using Web Service and this opens up possibilities of security breaches. This section serves to present terms and concepts that are important in the area of computer security.

### 2.3.1 Secure System/Communication

When communication between applications takes place across the boundaries of a trusted network or when using an insecure system, there exist no guarantee that the communication or system will behave as intended. The reality is that messages can be stolen as well as be partially or fully modified. There exists no certainty that communication takes place between intended parties. Therefore, security measures must be added to systems and/or communication when tasks are sensitive or safety crucial.

#### **Principles of Computer Security**

Any attempt to capture the notion of security must first cover Confidentiality, Integrity and Availability commonly referred to as the CIA of computer security [25].

1. **Confidentiality** – This is about ensuring that only authorized persons interpret protected information. This means that no more than the sender and intended receiver(s) should be able to understand the transmitted message. It does not mean that others are unaware of the message but rather that they cannot understand it if they were to access it.
2. **Integrity** – Is aimed at protecting against modification of transmitted messages. Integrity guarantee that the content of the communicated message is not altered, either by accident or deliberately.
3. **Availability** – As the name suggests is concerned with ensuring that information or resource is present when required.

#### **Principles of User Validation**

Making a system or communication secure does not end with CIA, it also involves tracking identity of users who requests service(s) [25]. The concepts fundamental to user validation includes:

- **Identification** – This is about tying a user to a unique identity. The most common means is through username.

- **Authentication** – Guarantees the identity of a party in a system or communication. Here both sender and receiver should be able to confirm the identity of the other party involved in the communication. Simply put are you who you say you are.

### 2.3.2 Realizing Security

There are numerous well established methods and systems developed that serves to fulfil the principles outlined in the earlier section. The remainder of this section will highlight only those that form a central part to the thesis.

#### **Cryptology**

The need for the security within computer systems and the communication between them has led the computer industry to the age-old solution of cryptology. Cryptology has today become a mainstream word whenever security is a need in systems and communication.

It is a technique that allows a sender to mask data so that unauthorized user(s) can gain no information from viewing unauthorized data. The original message is distorted in such a way that it makes no sense to those unauthorized to view it. The act of making the data distorted is called encryption whilst the act of restoring data to its origin is called decryption.

An important element in encryption and decryption is a key. An analogy typically used to explain this element is that of a physical lock and key. To make your home more secure you place a lock on the door to which there exists a key. Once you lock the door with this key only a person possessing the key can open the lock and enter your home. In cryptology a key is used for protection of data. A message is encrypted with a key and only those who have the key can decrypt the message.

When encrypting and decrypting there is a choice of two types of algorithm: Symmetric and Asymmetric. With symmetric key cryptology the same key that is used to encrypt the data must be used to decrypt. This makes symmetric key cryptology the faster of the two as it requires less computation. It is also suited for encrypting large blocks of data. However, it presents a problem of key distribution; the receiver of the encrypted data needs the key that was used for encryption but when parties are located across system, how does the sender give the key to the receiver. Triple-DES is a popular symmetric algorithm.

To solve the problem of key distribution one can use asymmetric key cryptology. This approach uses a pair of keys: a public key and a private key. The public key is known to everyone and the private key is secretly known only to the user. A sender uses the receiver's public key to encrypt messages intended for the receiver and the receiver in turn uses the private key to decrypt. Because the receiver is the only one that knows the private key only the receiver can decrypt the message to its origin. RSA is a popular asymmetric algorithm.

#### **Message Digest**

Both symmetric and asymmetric algorithms offer Confidentiality but what of Integrity? For integrity a message digest must be introduced. This is in some case referred to as a "fingerprint."

A message digest is a function that ensures integrity of a message. It takes a message as input and generates a fix-sized block of bits, usually several hundred bits long that represent the fingerprint of the original message. The message digest is a one-way function. This means that it can generate a fingerprint from a message but given a fingerprint it is extremely difficult to generate the original message.

Message digest algorithm available includes MD5 and SHA-512 [25].

### **Certificate**

With keys playing such a pivotal role in cryptology, it is important that trust be established around the authenticity of the key. How does one party know that the key being presented by another party is indeed from that party? This need for trust has led to the use of certificates.

A certificate is used to identify the owner of a key. It gives information about the holder of the key such as: name, address, email address and country. It also carries information about the certificates expiry date. A certificate authority (CA) is used to issue and sign the certificate. When a certificate is signed by a CA it says to those viewing the party's certificate that the party and the certificate is trustworthy.

### **HTTP Basic Authentication**

Widely referred to as HTTP BASIC-AUTH, it is aimed at introducing a measure of security into the otherwise insecure HTTP protocol. The authentication scheme is based on the model of a client identifying and authenticating itself by means of a username and password. The server then performs a validation check of the credentials to see if it is allowed to enter the protected space.

HTTP BASIC-AUTH scheme though adding security to the HTTP request is a non-secure method of filtering unauthorized access to protected areas on the HTTP server. It is based on the assumption that there is a trusted secure link between the parties [26]. It is insecure because the username and password submitted are sent in the clear.

### **Transport Layer Security**

The Transport Layer Security (TLS) is a protocol that provides authentication and communication privacy between communicating applications over the internet. The protocol allows client/server applications to communicate in a manner designed to prevent tampering, eavesdropping and message forgery.

TLS consists of two layers: TLS Handshake Protocol and the TLS Record Protocol. The TLS handshake Protocol allows parties to authenticate each other and negotiate encryption algorithm and keys. The TLS Record Protocol provides security with the chosen encryption method and provides mechanisms for preventing a message from being modified.

TLS is often the protocol used when security needs to be added to existing protocols. For example to secure HTTP one runs HTTP over TLS, this result is commonly referred to as HTTPS.



## 3 Problem Formulation

Significant parts of the work done on this project fall into the realm of distributed computing. Aside from enabling distributed computing this thesis also implements a small prototype application. This chapter presents requirements set forth by StruSoft that forms the basis for the work done and provides a rationale for choices made.

### 3.1 StruSoft Mandates

In an effort to have tight control over the VIP Energiberäkningssystem and those permitted to use it, StruSoft has decided to use a License Server to aid them in their efforts [Appendix C.3].

The License Server will maintain a list of customers authorized by StruSoft to run the system. To this end, there must be communication between the License Server and the VIP Energiberäkningssystem. This leads to the first clear requirement of the thesis: allow for interaction between the VIP Energiberäkningssystem and a License Server.

The License Server will at all time be the property of StruSoft and therefore will be held in their position and run on their premises. The VIP+ system on the other hand will run at a client's location [5]. Regardless of this physical decoupling of the system the requirement of interaction between the VIP Energiberäkningssystem and the License Server must always be present. With this requirement enters the need for a distributed computing solution that will allow for interaction between the VIP Energiberäkningssystem and the License Server.

When a customer chooses to run the VIP Energiberäkningssystem the application will first communicate with the License Server. Based on the validity of the customer's license the server would decide whether or not to allow this customer access to the VIP Energiberäkningssystem kernel. If a valid license is found then rights are granted to execute the request whereas if an invalid license is found, the access is denied.

When a customer's license is near the expiry date then this customer must be notified. This notification should be automatically send by the License Server.

Asides from enabling the communication between the VIP Energiberäkningssystem and the License Server, a prototype of the License Server should also be developed. This implementation can take the form of a License Server with a traditional hardware lock or a database. If the database solution is chosen, one should be able to perform such actions as: adding a new customer to the server; deleting a customer and searching for customer(s).

The system though primarily used on personal computers and notebook should lend itself easily to other communicating devices including Personal Data Assistant (PDA) and smart phones.

### 3.1.1 Non-Functional Requirements

The mandates given by StruSoft are clear and concise but within these mandates there are unwritten non-functional requirements. This section looks at these non-functional requirements.

- **Security** – The requirement for security stems in part from StruSoft desire to have tight control over the system. Only those users or customers approved by StruSoft should be granted access to the system.  
The nature of distributed computing is such that applications communicate across the Internet and the Web but there exist no guarantee that the communication link established can be trusted. The fact that trust cannot be guaranteed has also prompted the need for establishing secure communication.
- **Expandability** – At this moment StruSoft will only sell the VIP Energiberäkningssystem to a selected number of customers. One future goal for the system is that it be sold to all who wish to buy throughout Europe and eventually worldwide. With a vision to the future, the solution implemented must be flexible and lend itself easily to expansion and changes.
- **Interoperability** – The reality that faces this and other commercial product is that a vendor has no control over their customer's choice of operating system platform and programming language or environment. Therefore it is important that the solution present in this thesis be high interoperable.
- **Performance** – The fact that the system extends beyond the bounds of StruSoft network means that the time taken to complete the interaction between the VIP Energiberäkningssystem and the License Server cannot be guaranteed. Though this is so, the interaction should be as short as possible as the most important thing is to allow the customer rights to run the VIP Energiberäkningssystem.

## 3.2 Design Decisions

Having received clear guidelines for the part this thesis plays within the overall project, decisions now need to be taken with regards to the way forward. This section gives an account of the choices made and why.

The development of software – be it a simple standalone application or a complex distributed application – begins with the choice of a programming language. For this project, Java is the language for development. Java is chosen primarily due to the following reasons:

- Java provides a programming language that is platform independent; write once run anywhere. It holds true only for Java, as language like C++ require the different executables for different platforms. With Java the byte code which is platform neutral can be transported to any platform.
- The Java programming language and environment is freely available for download unlike language like C# which is bundled with Microsoft .NET that requires purchase.

- The Java programming language is built with security in mind and it offers an array of security features [27].

Another decision that shapes the cornerstone of project is the choice of distributed computing solution. Web service comes out as the winner because:

- **Internet Standards** - Web service is built upon open Internet Standards which includes: UDDI, WSDL, XML, SOAP, HTTPS, and SMTP. These standard protocols are within the public domain under the control of the organization like W3C. With XML at the heart of the protocols employed, Web Service offers a text-based data representation which provides readability.
- **Platform Neutral** – Exactly because of the use of open Internet standards Web service is about giving a platform-neutral solution to distributed computing unlike CORBA, DCOM and Java RMI.
- **Interoperability** – This result from the fact that Web service is built with Internet standard protocols.
- **Expandability/Scalability** – Web service allows for handling of increased load, while managing investment in providing service.

The two decisions taken previously would enable one to develop (if necessary) a prototype License Server and allow for the communication to the License Server and the VIP+ application, but there is a need to make one further decision: that of security.

The project takes a twofold approach to meeting the security requirements: securing the communication link and securing the License Server. In order to secure the communication link between the VIP Energiberäkningssystem and the License Server, HTTPS and HTTP BASIC-AUTH is used to provide the user authentication and confidentiality. The second implementation of security was centred on the actual data exchanged between the client and the License Server. Before sending data on the wire it is first encrypted with the symmetric key algorithm and a fingerprint created. This accounts for Integrity.

The actual implementation of the License Server must be decided before work could progress on the project. Implementation could be a traditional hardware lock or through a database. The decision is taken to implement the License Server as a database due to the following very important reasons:

- **Cost** – A primary reason for choosing a database approach. The cost to StruSoft to obtain hardware lock of all the customers that use the system will increase as the customer base increase. And further more, some of the cost of this hardware must be passed on to the customer and thus would increase the final cost of the system.
- **Scalability** – By using a database approach, StruSoft could easily manage a varying number of customers, be it tens or thousands. The primary requirement would be hard disk space. As the customer base increase and they expand into wider market it would be easier to realise this growth with a database rather than hardware lock – which would require the physical distribution of these locks introducing extra logistic problems.

- **Maintenance** – The issue of maintenance is an important aspect of any application. Maintenance of hardware lock would require StruSoft to go to their customers' locations which is a costly venture especially when a customer is in another country. With a database approach maintenance is a simpler task.

To fully simulate communication there is a need for a Web server. This will allow for the implementation of a Web service and provide an endpoint address that the client's application will use to communicate with the License Server. For this Apache Tomcat is used.

The development tools and environment required for a database application, Web service implementation and a Web application is provided by the Apache Software Foundation and MySQL. The Apache Software Foundation provided the Web application or server – Tomcat and the Web service engine – Axis. The database application – MySQL was provided by the MySQL AB. (For more on Apache and MySQL see Chapter 4.6 Development Tools.) The reasons for selecting these solutions are:

- **Cost** – First and foremost the Apache Software Foundation offers solutions that are free.
- **Open Source** – With access to the source code one can modify the application to fit their needs.
- **Platform** – All the solutions chosen requires a Java platform. The Java programming language and platform had already been chosen as the platform and language of choice.
- **Support** – There is fabulous support available to users of Apache solutions not only from the Apache Software Foundation themselves but also from all from all the users. A benefit of using the open source solution from Apache is the support and contribution put forward by both users and developers.

One could write all the code in a simple text editor, but as project size increases it is recommended that a development environment be used. For this project the Eclipse development environment is preferred. (For more on Eclipse see Chapter 4.6 Development Tools.)

## 4 Development Environment

This chapter takes a close look at the development environment and gives a detailed explanation as to how the various components are installed and configured. It also chews over the open source solutions used, and the choices of development tools. One important note to be added is that all configurations that follow in this chapter were done for Microsoft Windows XP operating system.

### 4.1 Java Configuration

Central to this project is Java programming language and platform that all other components to be installed require. Therefore, it is the first piece that should be installed and configured.

#### 4.1.1 Download

In order to have the Java platform, one must first get the Java 2 SDK Standard Edition (there is also an Enterprise Edition but for this project the Standard Edition is enough). Java 2 SDK Standard Edition can be downloaded from the Sun Microsystems Web site [28]. Navigate the pages and download the latest version of the Java 2 SDK that is on offer - the version downloaded for this thesis is version 1.4.2\_05.

#### 4.1.2 Installation and Set-up

To install Java, run the executable file downloaded. This requires no changes, unless one wishes to change the installation directory. If the installation directory is changed, then a note of such should be made of this new location. If no change is made then the Java 2 SDK would be installed in the root directory.

In order to make Java 2 SDK known and readily accessible to all application on the system from any folder, certain Java files need to be added to the path setting. On the Windows taskbar click Start, Settings then Control Panel. Once the control panel opens; double click the System icon then select the Advance tab and click the Environment Variables button [29].

In the lower section named System Variables do the following:

- Click new and create a variable called Java\_HOME. The value field should be the location of the Java 2 SDK directory, for example C:\j2sdk1.4.2\_05.
- Create a second variable called CLASSPATH with the following values:  
.;%Java\_HOME%\bin; %Java\_HOME%\tools.jar; %Java\_HOME%\lib
- Select the variable called Path and add the following value to the end:  
;%Java\_HOME%\bin

After these tasks are complete click OK and exit control panel. This system should automatically update the path but for best result restart your system.

### 4.1.3 Test

To test if the setup and configuration is successful, try executing a simple Java program that prints to the console. Use any text editor and create the following file then save it as `Congrats.Java`. Code 6 shows the content of the file, simply copy and paste.

---

```
public class Congrats{
    public static void main ( String[] args ){
        system.out.println ( "Congratulations You Did It!" );
    }
}
```

---

Code 6: Simple Java program

Open a command terminal and go to the directory where the `Congrats.Java` file is stored, then type the following:

1. > `Javac Congrats.Java`
2. > `Java Congrats`

On the console you should see - *Congratulations You Did It!*

## 4.2 MySQL Configuration

The prototype License Server that will be created for the project requires the uses of a database and MySQL is the chosen database.

### 4.2.1 Download

To download the database visit the MySQL Website [30], navigate the pages and download the latest version on offer. At the time of writing, the latest version on offer is version 4.0. For easiest installation download the zip format of the installer version for Windows OS.

### 4.2.2 Installation and Set-up

Unzip the installer version downloaded to a temporary directory. Form this location run the `SETUP.EXE` to begin the installation process. The installer will choose to put the MySQL installation in the root directory but one could choose to change the location.

(Optional) To make MySQL known and readily accessible to all application on the system it needs to be added to the path setting. On the Windows start bar click Start, Setting then Control Panel. Once the control panel opens double click the System icon then select the Advance tab and click the Environment Variables button.

In the lower section named System Variables do the following:

- Click new and create a variable called `MYSQL_HOME`. The value field should be the location of the MySQL directory, for example `C:\mysql`.

In order to use the MySQL database with Java, download the MySQL connector/J from the MySQL Website [31]. The latest version on offer at the time of writing is mysql-connector-Java-3.0.15-ga. Unzip the file and do the following:

- Copy the .jar file to %Java\_HOME%\jre\lib\ext. This allows Java applications to connect to MySQL database.
- An optional step is to add the jar file to the CLASSPATH of System Variables.

MySQL can also be installed as a service that allows it to run each to Windows start-up. For this and further configurations consult the manual.

### 4.2.3 Test

To test if the installation was successful, open the command window and enter:

- > cd %MYSQL\_HOME%\bin
- > mysql -u root -p
- mysql> show databases

At this point the console should show tables similar to view 1. It is assumed that the MySQLd had started either manually or as a service.

---

+	-----+	+
!	Database	!
+	-----+	+
!	mysql	!
!	test	!
+	-----+	+

---

View 1: MySQL Databases.

Further look at MySQL security will be considered in Chapter 6.2 – Open Source Security.

## 4.3 Tomcat Configuration

Tomcat provides the project with a Web application or server that will allow for the implementing of Web Service engines.

### 4.3.1 Download

To get the Tomcat Web application, visit the Jakarta Tomcat Website [32]. Navigate the pages for the latest available binary version. At the time of writing this is version 5.0.28. Download the binary Windows Installer executable.

### 4.3.2 Installation and Set-up

Run the executable binary file downloaded. During installation such things as: administration password and ports can be set. Also the user can choose the change installation directory. With the version of Tomcat installed, the program provides a convenient shortcut in the Start menu.

Tomcat requires that variables be added to System Variables. So click Start, Settings then Control Panel. Once the control panel opens double click the System icon then select the Advance tab and click the Environment Variables button.

In the lower section named System Variables do the following:

- Create a new variable named CATALINA\_HOME with value being the location of the Tomcat application example `C:\Program\Tomcat 5.0`.
- Add Servlet to your CLASSPATH variable. Append to the end of the list - `%CATALINA_HOME%\common\lib\servlet.jar`.  
If you are not using servlet this is not necessary.

### 4.3.3 Test

To perform the test you must first start the Tomcat Server. There are two ways to do this: through the console or the start menu. The use of the console is recommended as it gives useful error information if any such occurs. Open a console window and go to `%CATALINA_HOME%\bin` then type “startup”. Open your favourite Web browser and type the following URL: `http://localhost:8080/`. At this point you should see the Tomcat home page which indicates that you have a connection to the Web server. To stop the server, type the word “shutdown” in the console window.

With the server up and running and working properly one can choose to change the home page by creating a new page.

Further look at Tomcat and how to add more security will be considered in Chapter 6.2 entitled Open Source Security.

## 4.4 Axis Configuration

Axis provides the project with the Web service engine required and therefore the configuration of this component is vital to the success of the Web Service. Axis is an implementation of SOAP and provides developers with a framework on which to develop Web service it also provides a few tools that are useful example WSDL2Java and TCPmon.

### 4.4.1 Download

A copy of Axis can be downloaded from the Apache site on the Axis home page [33]. Locate the latest binary release on offer; at the time of writing this is Axis 1.1. Download the file in zip format.



## 4.4.2 Installation and Set-up

Axis does not contain an executable file; unzip it to a folder of choice. To install Axis do the following:

- Extract the content of the zip file to a local folder. Example create a directory called `axis-1_1` in the root directory
- Copy the entire `axis-1_1\Webapps\axis` directory to the directory `%CATALINA_HOME%\Webapps`.

Having copied all necessary files, Axis needs to be added to the System Variables. Click Start, Settings then Control Panel. Once the control panel opens double click the System icon then select the Advance tab and click the Environment Variables button.

In the lower section named System Variables do the following [34]:

- Create a variable named `AXIS_HOME` with the value pointing to the Axis directory, example `C:\ axis-1_1`.
- Create another variable named `AXIS_LIB` with the value pointing to `%AXIS_HOME%\lib`.
- Again create a variable named `AXISCLASSPATH` and add the following values:  
`%AXIS_LIB%\axis.jar; %AXIS_LIB%\commons-discovery.jar;`  
`%AXIS_LIB%\commons-logging.jar; %AXIS_LIB%\jaxrpc.jar;`  
`%AXIS_LIB%\saa.jar; %AXIS_LIB%\log4j-1.2.8.jar.`

Save changes. For best result restart Windows.

## 4.4.3 Test

To test the functionality of Axis, start the Tomcat as before and open your favourite Web browser (at least one that works with Axis) and enter the URL `http://localhost:8080/axis`. If all went well you should see the Axis home page. Axis provides a link on the home page called “validate”. This is just a page used to check if all the components needed to run Axis is present. Some components are optional and these have not been provided in this configuration other components are required; all of which are provided.

The happy axis page should now say that all required components are present. This means you are ready to unleash the power of Web service using Axis – “yu up to the time.”

The Axis team recommends the use of Xerces as the XML parser but Java 2 SDK 1.4.2\_05 already has a XML parser. It is up to the individual to download Xerces.

## 4.5 Open Source

Open source solutions forms a central part of the framework of this thesis and therefore it is import that a detail examination is given on this arena.

### 4.5.1 Historical View

Much of the development tools and environment used in this thesis are open source solutions and the origin of open source is of particular personal interest. This sub-section explores this area of interest; therefore readers with no such interest could proceed to the next sub-section.

To trace the origin of Open Source there is a need to revisit the early days of computers. One could “go out on a limb” and claim that in the beginning everything was free: software that is. In the 1960s when vendors like International Business Machines (IBM) delivered their products, it came with software that could be freely distributed to others; it came with the source code and the right to modify the code. It was strongly believed that the value of the computer was in the hardware which was at the time extremely expensive and little thought was given to the cost of software.

The operating systems and applications thou necessary ingredients for the operation of a computer were deemed less valuable than the hardware. Users were few and computer manufacturers actively encouraged users to share improvements to software as it was thought that this would help reduce support cost [35].

By the late 1960s there was a shift for the proprietary software. This meant that users were no longer allowed to redistribute, source code were no longer available and programs could not be modified [35]. During this shift in paradigm two groups were formed which today are seen as the roots of the current Open Source movement. They were UNIX and GNU project. In the early 1990s entered Linux.

### UNIX

The year was 1965 when the Bell Labs, Massachusetts Institute of Technology (MIT) and General Electric (GE) decided to join forces on the Multiplexed Information and Computing Service (MULTICS) project. MULTICS was supposed to be third generation computer equipment with a new interactive, multi-user operating system [36, 37, 38].

In 1969 support was withdrawn from the project and thus marked the end of MULTICS. Though the support was withdrawn from the project, two developers Ken Thompson and Dennis Ritchie continued to develop a file system and write some programs for the GE645 computer. Success came when a game called “Space Travel” original developed from the MULTICS project was ported to the DEC PDP-7 (Programmed Data Processor 4K memory for user programs). Thompson implemented the file system that was designed earlier and continued improving other requirements with the aim of achieving a working operating system. With the promise of developing a text-processing tool for the system, they received a new DEC PDP-11 computer which was shared with others. This was the birth of UNIX.

The typesetting facilities of the PDP-11 gained momentum and soon it was being used outside the research group, this triggered the need for documenting the operating system. The result was the first edition UNIX Programmer’s Manual by Thompson and Ritchie, dated November 3, 1971 [37].

By the third edition in 1973 UNIX was now installed on sixteen sites (all within AT&T/Western Electric) but knowledge of the system was not extended beyond AT&T. October 1973 was the public unveiling of UNIX at a conference. There was an explosion in demand for the operating system but AT&T could not capitalize on this surging demand. This was because of an antitrust ruling in 1956 restricting them from pursuing software as a business. Therefore UNIX was distributed with no support. Without support the community of UNIX users had to help themselves by helping each other. The community started sharing ideas, programs, bug fixes and other information.

One could say this is a significant point in Open Source: the creation of UNIX at the Bell Labs. UNIX was distributed in source code form. Users all created from it and in turn improved on others improvement with the only cost being the cost of distribution (and licensing). AT&T patent and licensing became inferior as the system simply became bigger and more widely used [36].

Over the years, many UNIX derivatives followed. Some of them were based on original AT&T versions, others on the Berkeley Software Distribution (BSD) line.

### **GNU Project**

In 1971, Richard Stallman began working at the MIT Artificial Intelligence lab and became involved in a software sharing community. As an AI lab staff, his job was to improve a timesharing operating system used at the lab called Incompatible Timesharing System (ITS). This software and others designed would gladly be made available to those interested, one would simply ask for the source code and they could in turn read it, edit or copy parts to make a new program [39].

This software sharing community was hit hard in the early 1980s with a series of events. First, the computer that was used for ITS - PDP-10 series – was discontinued, making most of the programs designed for the ITS obsolete. Second, there was a mass exodus of AI staff from MIT. And third, with the 1980s came also proprietary software. The days of sharing and helping your neighbours were ending; owners of software forbid and obstruct sharing among users [39].

As the software sharing community disbanded, Richard Stallman decided that to the first re-igniting the community there must be a free operating system. He decided to build an operating system compatible with UNIX and choose the name GNU. GNU was an acronym used in the hacker community that stands for “GNU’s Not Unix.” In developing the GNU system a decision was taken not to reinvent the wheel and simply adapt and use existing pieces of free software where possible. Stallman achieved the initial goal of a free operating system.

In January 1984 Richard Stallman quit his job at MIT and began writing GNU software. Work on GNU Emacs began September 1984, and in early 1985 it began to be usable, enabling text editing on UNIX systems [39].

With the passing years the GNU project has grown from strength to strength. The GNU project has given: the Copyleft Law, General Public License (GPL) and The Free Software Foundation.

## **Linux**

In 1991 Linus Torvalds a student at a Finnish university started to develop a UNIX compatible kernel [40]. Linus was dissatisfied with the fact that he couldn't run UNIX on his PC with Intel 386 processor and therefore decided to rewrite UNIX to make it compatible.

The design of Linux used a monolithic architecture. Linux was provided under the Copyleft law with the GPL license and others were invited to assist in the developing and improving of the kernel. With a large community of willing developers the Linux community grew rapidly. The Linux kernel was later combined with the GNU system (which was in need of a working kernel) to provide a complete operating system.

With subsequent releases the Linux kernel has improved and extended and today Linux operating system is the fastest growing operating system [41].

## **Up to the Time**

Starting in the late 1990s interests in open source rocketed, with various large industry players embracing the concept and taking an active role in the community.

1997 saw the announcement of Netscape's Mozilla project. Google Inc presented their search engine in 1998 running the first large scale installation of Linux operating system. In the same year The Open Group offered Linux certification to carry the UNIX name. Oracle and Informix vowed support for Linux. Existing open source project Apache received support from IBM. In August 2002 the UK government issued a policy document to promote the use of open source software in “central government departments and their agencies, local government, the evolved administrations as voluntary partners, and the wider public sector, e.g. non-departmental public bodies (NDPBs) and the National Health Service [42].”

Major computer manufacturers started shipping computers with variants of the Linux operating system (for example IBM) and many companies turned to open source products from Internet Service Providers (ISP) - using SendMail – to publishers - using Open Office. These and other significant changes and trends have positioned open source community as a permanent stay and a major contender.

### **4.5.2 Open Source Fundamentals**

So far a conscious effort has been made to give readers the origins of open source and where it is heading, but what of the definition and its core principles.

Open source refers to those computer programs or operating systems for which the original source code is publicly available. Thus, users can examine the source code to see how things work and even modify it if they see fit.

Inherent in the open source philosophy is freedom [39, 43].

1. The freedom to execute the program, for any purpose.
2. The freedom to study how the program works, and modify it to your needs.
3. The freedom to redistribute copies so you can help your neighbour.

4. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits.

There is more to Open Source than access to source code; there are distributions terms that must be complied with. The following are criteria for open source software [44]:

1. **Free Redistribution** – The license shall permit any party to sell or give away the software as a component of an aggregate software distribution containing programs from several different sources. There shall be no royalty or other fee for sale of license.  
Rationale: This eliminates the appeal gain sales dollars in the short run and toss the long term gain.
2. **Source Code** – Source code must be included in the program and the distribution should be in compiled form and source code. Where source code is not provided, there should be a means for downloading this with no more than reasonable reproduction cost – example downloading via Internet without cost. Intermediate forms of code such as those output by pre-processor are not allowed. Intentionally obfuscated source code is not allowed.  
Rationale: Allow for program evolution. Making evolution of program easy requires that modification be made easy.
3. **Derived Works** – Modification must be allowed along with derived works, and the distribution of these works is under the same terms as the license of the original software.  
Rationale: People should be able to experiment with and redistribute modifications, this will allow for evolution.
4. **Integrity of The Author's Source Code** – The license permits the restriction of source code from being modified. This is only permitted if the distribution of the source allows “patch files” with the source code for modifying the program at build time. Distribution of software built from modified source code must be explicitly permitted. Derived works may be required to carry a different name or version number from the original.  
Rationale: All have the right to know who is responsible for the software they are using.
5. **No discriminating Against Person or Groups** – The discrimination against any person or group of persons is not permitted by the license.  
Rationale: This prevents any open source license from barring anybody out of the process.
6. **No discriminating Against Fields of Endeavours** – There should be no field of endeavour that is restricted from making use of the program.  
Rationale: This clause is intended to prohibit license traps that prohibit open source from being used commercially.
7. **Distribution of License** – All rights attached to the program must apply to those whom the program is redistributed without the need for additional license.  
Rationale: This clause keeps the software as open source always.

8. **License Must Not Be Specified To A Product** – The rights of a program must not depend on the program's being part of a particular software distribution. The program is removed from the original distribution and used or redistributed; all parties to whom the program is redistributed should have the same right as with the original software.  
Rationale: This clause forecloses yet another class of license traps.
9. **License Must Not Restrict Other Software** – No restrictions should be placed on other software distributed along with the license software.  
Rationale: Allows distributors of open source software the right to make choices about their own software.
10. **The License Must Be Technology-Neutral\*** – License must not be restricted to an individual technology or style of interface.  
Rationale: This provision is aimed specifically at licenses which require an explicit gesture of assent in order to establish a contract between licensor and licensee.

#### 4.5.3 Pros and Cons of Open Source

Open source has now reach “prime time” and has tremendous support for industry and governments alike. The question remains, why do individuals, organizations and government alike choose or at least consider open source? The answer to this question can be seen in the benefits that are offered. But, not everyone will choose open source product because they too come with shortcomings. This section looks at open source and examines its pluses and minuses.

Some of the major gains from open source product are as follows:

- **Lower Total Cost of Ownership (TCO)** – Total cost of ownership refers to the cost involved in acquiring the product (transport or distribution), actual product price and management of the product. With open source, one can acquire certain products for no more than distribution cost; for example the cost of downloading the product from the Internet. Management cost will be incurred regardless of whether a product is open source or not as someone needs to be responsible for the maintaining and updating of the product. There is also extensive support for open product.
- **Customisation** – Every individual, organization and government have distinctive needs. Software on the other hand tend to be sold on a as is basis, with the hope that all can find what they are looking for in the product. If one is unhappy with software then he or she can adjust setting made available by the software, beyond this there is usually no recourse.  
Open source allows users (those with knowledge and understanding of the source code) to adjust the product to suite the users. With this ability one can go beyond the settings offered by the product in the cases where they are deemed insufficient. This ability of customisation leads to evolution of the product.
- **Encourages Software Re-Use** – The criteria of open source rely heavily on and encourage software re-use and freedom to re-use source code (see previous subsection for more details). Open source development permits programmers to cooperate freely with minimum legal obstacles.

Developers are given freedom to re-use existing code. The approach of not reinventing the wheel was taken by Richard Stallman [39] when he was developing the GNU system.

- **Decreases Vendor Lock-In** – Individuals, organizations and governments no longer have to be tied to a vendor. Most open source products are built on industry standard and uses industry standard and formats, therefore one is not tied to proprietary standards and format and this makes it easier to switch product without major efforts. There is no need to wait on source-vendor to provide update to source code, as one is free to adjust and adapt source code.  
Scenario: Ability to freely change vendor with databases. MySQL database is a popular open source database the uses Standard Query Language (SQL), if a project has MySQL as it database and wishes to change, it becomes a matter of finding and replacement MySQL with another database that used SQL; say for instance Oracle DB.

Having discussed some of the reason why one would go down the open source road it is now time to give a few arguments for why one could or should be vary of this path. The follow are arguments as why not to use open source products.

- **Learning Curve** – When using new software there will be a learning curve; time spent learning how to use the product and this is no different from open source software. This is no different for open source products than for commercial software products.
- **Language** – Communication is a major building block for open source project. More often than not the parties involved in the project are based in different regions, even countries. This importance of this communication also brings into play one setback: the language.  
Often the language of open source development is English but this is not the language of all involved in the development. It is not uncommon that not all member of open source project is a native English speaker and therefore language would become a barrier. This could lead to communication difficulties. This leaves many talented and expert developers out of the loop, as they might be uncomfortable in English.
- **Configuration** – This is one area where some open source products fall behind the commercial counterparts. It can sometimes be a challenge to get an open source product up and running; as it could be difficult to configure. Installation often requires technical knowledge of sorts, an example would be Axis – this requires knowledge of Java, HTTP and a little XML where configuration is needed. Commercial products are often easier to use as they often come with user interface and normally require less prior knowledge and expertise.
- **Bad Guys** – There is an increased security concern when using open source products. This I refer to as the “bad guy” effect. “Bad guys” are individual (both male and female) that set out to maliciously attack other users.  
The bad guy effect is particularly real when using open source product because all have access to the source code and freedom to study and modify the code. The bad guys with malicious intent and time can study the code to the point where they become very familiar with the flow of the program and all its underlying function. With such

in-depth knowledge they become aware of any pitfall or shortcomings of the product. Armed with such knowledge they set out to attack other users of the product by exploiting the shortcomings they are aware of.

## 4.6 Development Tools

An important component in any software project is the tool(s) used in the development process. Within the confines of this section a concise introduction is given on the development tools used for this project.

### 4.6.1 The Apache Project

The Apache Project's main endeavour is to provide a HTTP (Web) server. The project is a joint software development effort aimed at producing a robust, commercially approved, feature-filled and freely available source code implementation of a Web server. It is a collaboration of developers the world over with communication, planning and development of the server and its related documentation is done using the Web and Internet. The volunteers are known as the Apache Group and alongside them stand hundreds of users that contribute ideas, codes and documentation to the project [45].

With very large support from the industry and the wide use of the server, Apache server is the number one server on the Internet and the most widely used. This survey was done by Netcraft [46]. The popularity of the Apache project and the wide support made it an attractive choice for this project along with the other benefits of open source (see chapter 4.5.3, Pros and Cons of Open Source). Tomcat and Axis provides the Web server and Web Service engine respectively both of which are provided by the Apache project.

**Tomcat** – This is a free, open source JSP/Servlet container that is used in the implementation of Java Servlet and Java Server Pages technologies [47]. The Java Community Process of Sun developed the specification for the Java Servlet and Java Server Pages. Developers the world over cooperate in the development of Tomcat; this includes Apache, Sun and various companies and individuals. It is released under the Apache Software License [48] both in binary and source versions. The implementation is available for use in Web servers, development tools and to create Web sites.

**Axis** – This is considered by most as the Web service engine. It is an implementation of the SOAP submission to W3C [20]. SOAP (therefore Axis) is designed as a protocol for exchanging XML based information in a decentralized, distributed environment. For a detailed explanation of SOAP see chapter two section four, SOAP.

Axis also includes the implementation of Web Service Invocation Framework (WSIF). WSIF enables developers to interact with the WSDL descriptions rather than work directly with SOAP APIs. It allows for stubless or fully dynamic invocation of a Web service, by examining the WSDL at runtime. Because WSIF is closely based upon WSDL, it can invoke any service described with WSDL.



### 4.6.1 MySQL

“The MySQL (R) software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software. MySQL is a registered trademark of MySQL AB. [49]”

MySQL AB was established in Sweden and is the company of the MySQL founders and main developers. Developers are all employed by MySQL. The company is described as a “virtual company” with people in a dozen countries around the world. Communication with partners, supporters and users is done primarily via the Internet.

The MySQL database software is Dual Licensed. One can choose to use MySQL database as an Open Source/Free Software product under the terms of the GNU General Public License or can purchase a standard commercial license from MySQL AB [49, 50].

Some important values of MySQL that is dedicated to open source are:

- MySQL should be available and affordable to all.
- Partners share values and mindset.
- MySQL works against software patents.
- And importantly MySQL AB and its employees subscribe to the Open Source philosophy and support the Open Source community.

### 4.6.1 Eclipse

The Eclipse Platform is described by Eclipse Foundation as “a kind of universal tool platform [51]”, this is because the Eclipse platform relies a lot on internationalization. It is an Integrated Development Environment (IDE) that can be used to develop an array of applications including Web sites, embedded Java program, C++ programs and Enterprise JavaBeans. With all these capabilities inherit in Eclipse, the statement used by the foundation to describe the Eclipse Platform is: “... an IDE for anything and for nothing in particular. [52]”

Eclipse is very generic though it comes with a lot of built-in functionality. The full capability of Eclipse is seen through the ability to extend the Platform to work with new content types, do new things with existing content types and to make the generic specific [52]. Plug-ins are used to discover, integrate and run modules. For the most part Eclipse uses Java SDK and has no dependence on the underlying operating system.

## 5 Discussions

The system delivered at the end of the thesis can be thought of as a combination of applications, thus adding a level of complexity to the development process. The three components that makes up the system are: a prototype License Server, client-side code and the all important Web service. It is important to give a clear and concise view of the steps involved in developing such a system and the resulting architecture. This chapter provides exactly this.

### 5.1 Software Design

Central to the development of the system is the principle software engineering. This section examines the software process employed during the project.

Software engineering is a discipline which is concerned with all aspects of software or information systems production from the beginning stages of specification straight through to maintenance of system after it is delivered to the user [53]. As a discipline, software engineering has structured models for software development. The software development model that is used for this system is the Waterfall Model.

#### 5.1.1 Waterfall Model

This software development model is commonly referred to as: Classic Life Cycle Model, Linear Sequential Model or Software Development Life Cycle Model.

Figure 4 give a graphical representation of the activities involved in the Waterfall model. The figure resembles that of a waterfall, with each component in the model cascading into the next, thus the name Waterfall. The arrows on the left of each component that flows upwards depict the fact that at any stage of the development cycle one can perform testing or verification and if there are shortcomings it is possible to loop backwards to the previous component to re-evaluate with a view of correcting errors.

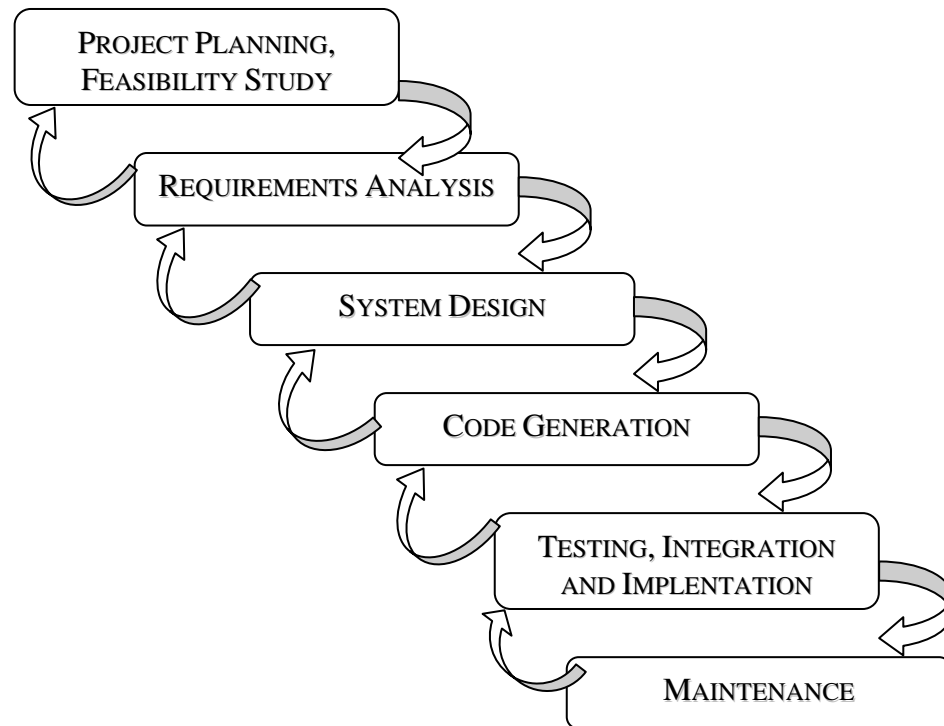


Figure 4: The Waterfall Model

The components identified in the model are by no means universal. Various organisations, institutions, individuals and texts present these components differently. But whether the model is given with different name or even more or less components with different names, the activities in the components above will be present.

- **Project planning, Feasibility study** – Every project must in some shape or form begin with this activity. Here a high-level view of the intended project is established and the goals determined. This stage is most important as it gives a system view of the project and a subset of the requirements. It is here where the life-cycle feasibility is performed.

The feasibility study is critical as it forms the basic of deciding as to whether or not the project or system is realistic with the resources available. Here a decision could be taken to rethink the overall project or to abort the project all together.
- **Requirements analysis** – If a decision is taken to proceed with the project in whatever form, than the next step is to perform an in-depth requirements analysis. In this phase an analysis is carried out on the end-user (customer's) information needs. The project goals are defined in terms of the intended functions and operations of the system. The main purpose of this level is to find the needs and define the problem that needs to be solved.

By the end of this stage a document that specifics the recommendation of the system should be available. An understanding of the nature of the program(s), understanding of the domain for the system, specific function and interface requirements and performance of the system are attended at this stage.
- **System Design** – At this stage the documentation produced in the prior stage is examined and used to describe the system's overall structure. An in-depth and

complete specification of the system architecture, business rules, process diagram, pseudo code, draft user's manual and test plans are produced.

Much care must be taken here as this is a pivotal point in the development process, error here can be very costly. Here a logical system is present that fulfills all the needs of the user.

- **Code Generation** – Here the logical system is taken to another level and the real code is generated: a complete, verified set of program components. The effect put in at the design phase directly affects this stage; if the design stage was thorough and concise then the code generation should not be very complicated. The “right” programming language and platform is used to develop the code.
- **Testing, Integration and Implementation** – Once code has been written the testing begins. There are a myriad testing methodologies and testing tools available for unearthing the bugs that are in the code written. If the system included several components then all the pieces are carefully put together over several stages. A fully functioning operational system is the result. This includes such objectives as program and data conversion, installation, and training.
- **Maintenance** – At this stage the system has been delivered to the end-users (customer) and is being used accordingly. Until the system goes obsolete and is removed there exists a need for on-going maintenance of the system. Any changes introduced into the system, corrections, updates are handled at this stage. Change is inevitable and unavoidable therefore a system should be written to handle changes as smoothly as possible. Maintenance could take one back to any stage of the life cycle and back. It is a very necessary and important phase.

### 5.1.2 Project's Software Processes

Having set the stage with the previous sub-section, it is now time to discuss the software processes of this project.

Knowing that this development is part of a larger, even more exciting and ground breaking project as well as a Master's thesis, it is important that a disciplined approach be taken for the start. Therefore, an attempt is made to adhere to the phases set forth in the development process under the Waterfall model.

With this sub-section, an attempt is made to describe the work in terms of the development life cycle of the Waterfall model. Figure 5 shows the development process of project, it closely resembles figure 4 of the Waterfall model except for an additional phase at the bottom of the falls and some of the flows upwards.

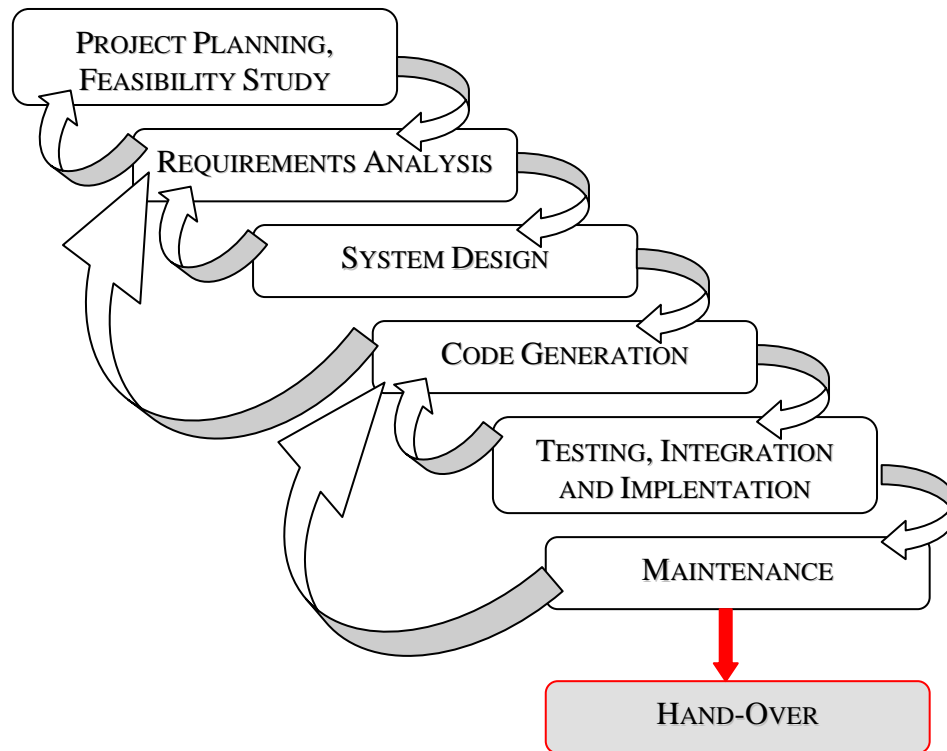


Figure 5: Development life cycle of this Project

With each phase goes particular activities and the following explains the activities carried out within each phase. The project's development life cycle unfolds as follows:

**Project planning, feasibility study** – This phase marks the beginning of the project. The first activity within this phase is gaining an overview of the project. For such information contact with StruSoft is necessary. Information from StruSoft gives a clear picture of the overall system (for an overview of the system, please see Chapter 1.2, VIP Energiberäkningssystem). See Appendix C for a copy of the System Overview provided by StruSoft.

The next activity in this phase is to form a high-level view of the paper's intended contribution. That is to say; get a system view of the project and the subset requirements. Once again, StruSoft provided the necessary information. See Appendix C for a copy of the Project Overview provided by StruSoft.

The Project Overview offers great freedom as to the focus of the paper. After discussion with my supervisor, a decision is taken to focus on solving the problem presented by distributed computing. Other possible areas that are available included: e-commerce Web site development and mobile device coding.

As with any project of this nature, there is a need to provide all parties involve with an idea of the duration of the project and the various activities involved. To this end, a project time table was created. See Appendix C for a copy of the Project Time Table.

Figure 6 give a simple view of the various activities involved in this stage.



Figure 6: Flow of activities within project planning and feasibility study phase.

**Requirements analysis** – At this stage a more detailed analysis of the intended system has to be performed. Having a thesis overview is not enough to proceed; a more detailed description of what functions and operations were to be performed by the system is needed. For such information contact is once again made with StruSoft.

After the analysis is carried out, a document with a concise definition of the needs of StruSoft and detailed information about the problem to be solved is created. See Appendix C for a copy the Final Project Proposal provided by StruSoft.

Within this step there is a need to revise requirements presented in the previous stage and therefore the need for an upward flowing arrow is seen in figure 5.

**System Design** – This stage starts with a discussion between my supervisor, my examiner and myself. Here, process diagram of the overall flow of the system is presented and test scenarios discussed. Further requirements are imposed (thus the upward flow in figure 5), these requires mainly focus on security. It is also this stage where a lot of decisions are taken about the way forward; decision such as choice of programming language and platform.

At the end of this phase, the overall architecture of the system is agreed on and presented to StruSoft for approval. Section 4 of this chapter gives the overall architecture of the system.

**Code Generation** – It is at this phase that all the requirements and needs get converted to computer codes. Here all code generated are either Java or XML.

During the coding additional requirements were imposed by StruSoft that made it necessary to revise documents from the “Requirements Analysis” phase. In figure 5 an arrow upward from “Code Generation” to “Requirements Analysis” shows this event.

**Testing, Integration and Implementation** – The work done on the project produce codes for a server-side application, client-side codes and a Web service. These codes are first individually debugged and tested to ensure that all function as intended with no programming errors. During the debugging and testing stage errors found are corrected.

Once individual components are error-free, all pieces are brought together to form a single system and testing of the system is performed.

The system is implemented and presented to StruSoft

**Maintenance** – After demonstrating the system at StruSoft, recommendations are made for updates. This recommended meant the certain section of codes needed to be rewritten. The arrow in figure 5 that flows from “Maintenance” to “Code Generation” accounts for this recommendation.

**Hand-Over** – This phase is introduced because my involvement in the project ends and therefore the all codes, documents and manual produced are handed over.

## 5.2 License Server Architecture

The creation of a License Server is an important requirement of the project. Recall that StruSoft had the following requirements:

- A prototype License Server should be created that holds the license of all customers.
- Implementation could use hardware or database solution.
- One should be able to add, delete and search for customer.
- When a customer license near the expiry date, the server should notify the customer by email, send a BBC to StruSoft and write an entry to a log file.
- License Server determines if customers has rights to run VIP Energiberäkningssystem.

A database solution is used to implement the License Server, which is created with the Java programming language (see Chapter 3.2, Design Decisions). Figure 7 below gives a view of the architecture of the License Server.

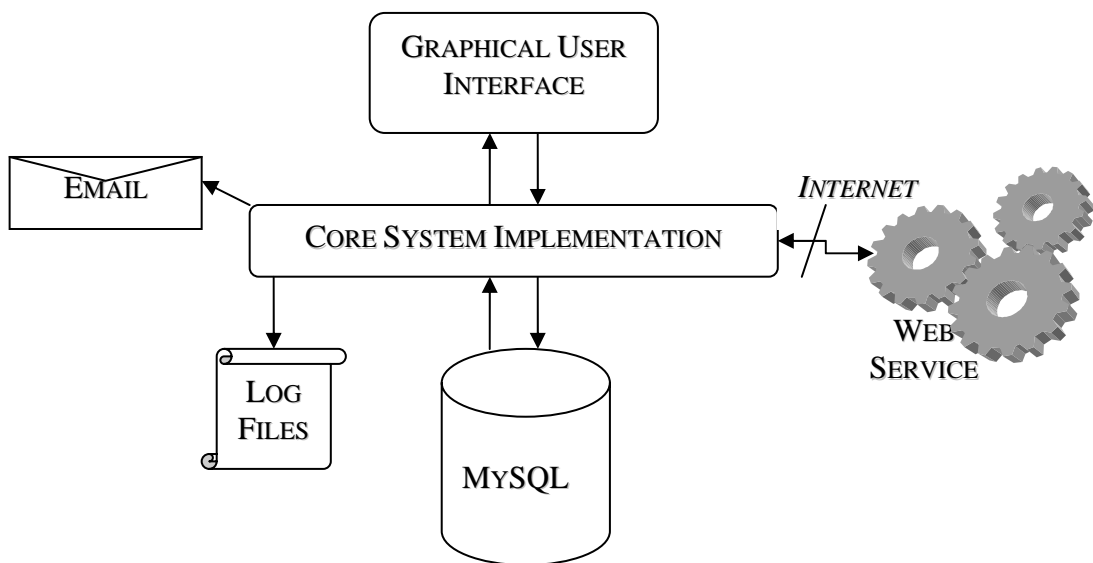


Figure 7: Architecture of License Server.

The “backbone” of the License Server application is a MySQL database. The database is at the bottom tier of the system. Through the use of the database, the server is able to store customer and license information, this information can later be deleted and also a search can be performed to find information stored.

The License Server application also makes uses of log files. The system uses log file to store information about emails sent. The log files for email store data about: the customer id and the date and time the email was sent. A log file is also kept for the Web service. This log stores data about which customer connection to the server and the date and time they connect.

In middle tier lies the core implementation of the License Server. At this level the system interacts with the underlying database and performs required database operations; write, read and search. All log operations are coded here. It is at this level of the tier where the email operations also take place, once triggered the system will perform weekly checks of all customer license and generate email automatically to customers where necessary. The core is where all interaction with the VIP Energiberäkningssystem occurs – via the Web service of course. It is within the core where implementation of cryptology techniques and user verification can be found. Last and not least there is an interaction with the user interface.

The top tier of the application holds the Graphical User Interface (GUI). The GUI gives user a familiar menu-drive approach to the system. Using the GUI one has the ability to add a new customer to the database; to delete a customer from the database and to search for customer(s). It allows users to start and or stop the email operators; view the logs for both Web service and for email and also to monitor the Web service. A help menu is also available with documentation as to how to use the system. For screen shots of the GUI see appendix a.

## 5.3 Web Service Architecture

The second and most important piece of the project is the creation of a Web service; for this Tomcat and Axis are used. (For more on Design Decisions see Chapter 3.2.)

The best and simplest architecture for implement this Web service solution is the Service Oriented Approach (SOA).

### 5.3.1 Why Service Oriented Approach

“The only absolute certainty in life/time is ... CHANGE. Life's/time's only constant ...CHANGE. Soh nuh fight it, work wid it, and embrace it” (The later part of the quote is written in Jamaican Patios but is not the part what is important, so there is no need to translate entirely.) Within the words of this quote one can find the reason for choosing SOA: “change.”

Most business's main aim is to make revenue grow by introducing a product and or service - new or improved. They have to do this in a global marketplace that is ever changing and dynamic and therefore a business must increase their quickness to respond to changing conditions. The challenges presented by the ever changing business environment requires IT solutions that are based on a SOA. SOA lends itself to change easily and is extensible and scalable.

### 5.3.2 Service Oriented Approach Defined

For the concept of service oriented architecture to be clearly understood, the term “service” must be put into perspective. A service is a function that is well defined and self contained that achieves a desired result.

SOA is basically an assortment of services. Services communicate with each other, this communication can involve either simple data passing or it could involve two or more services coordinating some activity. The goal of the architecture is to achieve loose coupling among interacting software agents.



The definition of SOA introduces the concept of agents. What are agents? An agent can be defined as a concrete piece of software or hardware that sends and receives units of data. The concept of agents can be further broken down into: Requestor and Provider. A provider agent is the one responsible for implementation of the logics of a service – owns the service. Requestor agents wish to use or exchange message with those services provided by the provider.

The SOA is not a new thing. One can look at other distributed solutions starting with Object Request Brokers (ORBs) based on the CORBA specification or the use DCOM. For more on CORBA and DCOM, see Chapter 2.1, Distributed System.

### 5.3.3 Service Oriented Approach Operations

The sub-section before introduced two roles that are important to the operations in a SOA. A third roles needs to be introduced: service broker or simply broker. The broker stores information about all services within a particular Web service.

The roles or players in SOA (and in Web service in general) – Provider; Requestor and Broker – performs three basic operations. The operations are: publish; find and bind. Figure 8 shows the three roles and their operations.

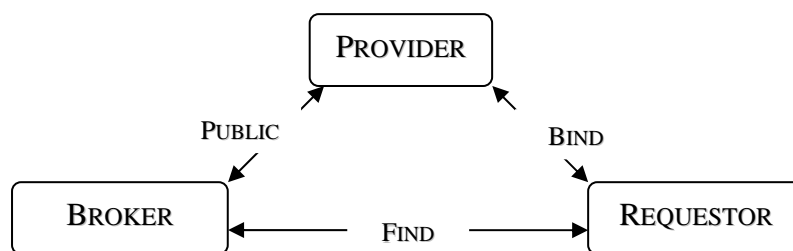


Figure 8: Service Oriented Approach Web Service Model

In SOA the Provider publishes the Web service to the broker. Information about the Web service is registered, this includes: the location of the Web service; the transport protocol to be used; the name of function and parameters and return types. The Requestor finds the Web service required by using or contacting the Broker. The Broker gives the requestor the service description it has for the service. After getting the service description there no further contact between the Requestor and the Broker. The Requestor then uses the information gained to contact or interact with the Provider.

The three roles identified do not have to involve three separate components. A component can play any or all of these roles.

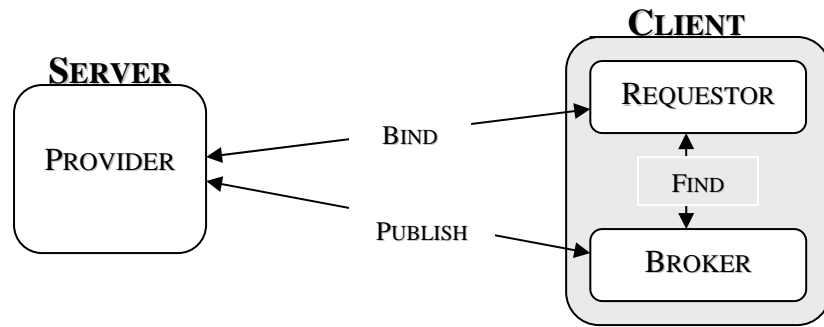


Figure 9: Project's Service Oriented Approach Model

The SOA model for the project is shown in figure 9. The figure shows the job of Requestor and Broker being undertaken by the Client. In this scenario, the Provider - Server has a service that it offers to select few Requestors – Clients. Here there is no need for the Provider to publicly publish the service description via a third party broker. The Provider makes the service description privately available the Requestor - Client. The Client uses the find operation to retrieve the service description locally. The service description is when used to bind with the Provider –server and invoke or interact with the Web service implementation.

## 5.4 Overall System Architecture

The two pervious sections gave the architecture of the License Server and the Web service respectively. This section is aimed at giving an integrated view of the system as a whole. Figure 10 shows the different pieces: License Server, Web service and client application working together.

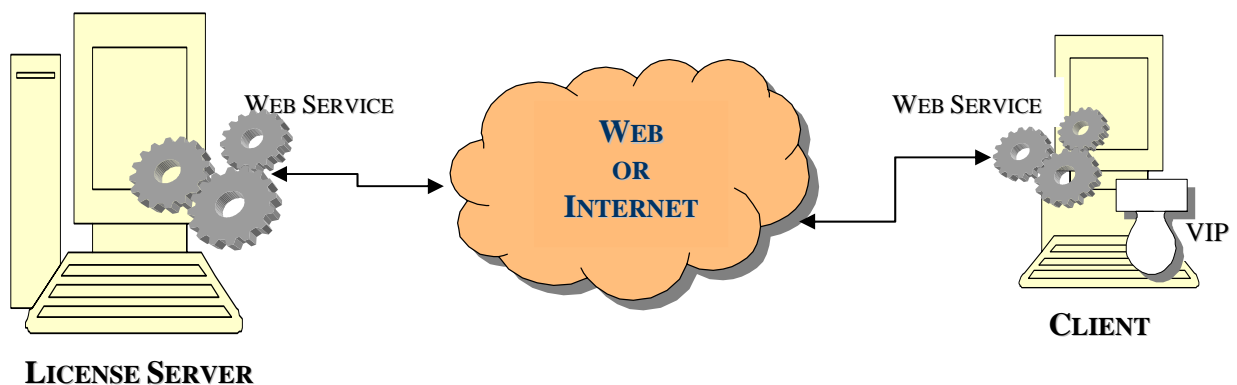


Figure 10: System Architecture

To explain the system with more clarity lets look at a scenario where a customer of StruSoft has the VIP Energiberäkningssystem and wishes to perform energy calculations.

When the VIP Energiberäkningssystem is launched, the system first checks to see if the customer has a valid license or permit. The check is performed examining the license that resides locally or via interaction with the License Server. Let's look at the case where the License Server is contacted, which is located at StruSoft. Interaction between the VIP Energiberäkningssystem and the License Server takes place via the Web or Internet.

These are the steps that take place:

1. Firstly, the VIP Energiberäkningssystem triggers the communication by starting up the Web service. The application asks the question: Is the license of this customer valid?
2. Secondly, the Web service on the Client interacts with the Web service on the server over the Web or Internet using a particular transport protocol, example HTTP.
3. Thirdly, the server's Web service takes the question to the License Server. The License Server performs various operations to provide an answer to the question. If the customer has a valid license then permission is granted otherwise it is denied. If the customer license near the expiry dates a warning message sent. Possible answers are: ACCEPT REJECT or NOTICE.
4. Next, the server's Web service returns the answer it has received from the License Server to the client's Web service.
5. In the fifth and final step, the client's Web service delivers the answer to the VIP Energiberäkningssystem. Based on the response received the system acts accordingly.

## 6 Security Issues

It is common to find applications and systems that are built without any regard for security, in other cases security becomes an afterthought and is added when everything else is in place and working. It is frequent that security holes are reported or the applications and sites are hacked then action is taken to introduce security. The practise of putting security on the backburner may result from: the developers lack of knowledge about security; security is seen as an expensive venture with no obvious financial returns; or security seen merely as an inconvenience, example requiring users to remember passwords.

A major focus of this thesis is security. The security needs results from the fact that communication between the loosely coupled applications occurred over the Web and Internet. This introduces a risk of possible security breach, which can lead to extensive and extensive damages. This chapter looks at the measures that are present in system to combat security breaches and also highlight the views of some industry experts as it pertains to open source security.

### 6.1 Product Security

The previous Chapter on System Discussion showed the final system can be thought of as three separate pieces: the License Server, the Web service and the client-side. In looking at the security implemented within the product it is useful to separate the two and examine the security that is implemented in each.

#### 6.1.1 License Server Security

The License Server is essence to the success of the overall system as it holds the customer and license information. Any breach in security would leak information that could be used by malicious individuals or organization to: perform operations with the VIP Energiberäkningssystem with valid customer id; delete a customer's license from the system, modify a license – example extending the life time of a license; or creating a license.

Obviously such undesired treats should be prevented or at least made unfeasible. A security breach could result in not only monetary loss for StruSoft but also the loss of their customers' confidence and damage to their reputation.

The treats posed to the License Server cannot be ignored, and they are not ignored. The server is build with a strong focus on security from the very first stage. Figure 11 shows a view of the License Server architecture with security added.

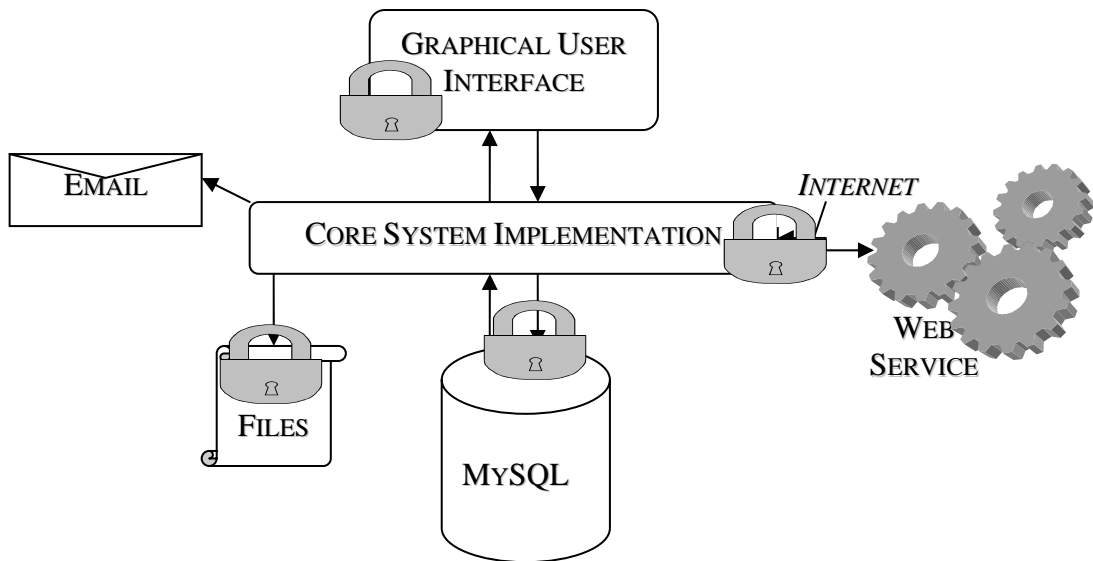


Figure 11: Security and the License Server

The image of the padlock had become associated with security and in figure 11 the padlock has the same meaning and is placed at the points within the License Server where security features are inherent. Each of these padlocks will be examined in further detail to give a complete understanding of the security that is present in the License Server. No claims are made to say that the measures implemented are complete and full-proof but they offer a level of security that would otherwise not be available.

### Database – MySQL

A database management system allows multiple users to access the database at the same time. This means that the License Server can interact with MySQL while authorized personnel at the company uses the command prompt to interact with the database.

- **Restricting Unauthorized Access** – With the offer of multiple users come the problem of unauthorized access to sensitive information in the database. MySQL provides a security and authorization subsystem. This enables the creation of accounts and the ability to specify account restrictions.

The project makes use of the security and authorization subsystem for MySQL by using a user account. When any database action is to be performed username and password must first be provided. If an incorrect username and or password are provided access to the MySQL database is denied.

By using the security and authorization subsystem of MySQL the system ensures that only authorized applications and individuals can perform database operations. This allows for: confidentiality and user validation, two important requirements of security.

### Log Files

A log file captures information on actions that have occurred. The log is able to capture such information as security failures, performance errors and application bugs and more.

- **Vital Security Information** – Logs provides a very effective way for analysing activities. Analysis can yield vital information, for example, the Web service maintains log files listing of every request made to the server. With log file analysis tools, it's possible to get a good idea of where visitors are coming from – IP address, who is requesting the service – customer ID, and what time they requested the service. From a security prospective this information is vital because any attempt to interact with the License Server via the Web service is recorded. In the event that an invalid or unauthorized attempt was made, say a malicious hacker tried to hack the system, this attempt would be written to the log whether or not it was successful. Log files are often implanted as read-only; this would prevent the content of the log from being modified and thus ensuring its integrity.

A log does not directly offer any of the CIA of security nor does it offer user validation. Nonetheless it provides a means by which one can check to see if any of the fore mentioned are being violated. It also provides a means by which one can track malicious users or hackers. In cases where the IP address is recorded in the log, the address can be use to trace the origins of the user or hacker.

### Core Implementation

In Chapter 2.3.2, Realizing Security, it is noted that the age old solution of cryptology is the means by which CIA of computer security is attained. The core implementation of the system relied heavily of cryptographic techniques to deliver security. The module is written in Java. The Java platform, both the language and library extensions, provides an excellent foundation for writing secure applications [54].

The security implementations makes an attempt to provide CIA and user validation.

- **User validation** – The user validation at this level deals with the application interaction with the MySQL database. The core module has to validate itself to the database by providing a username and a password.
- **Confidentiality** – The module guarantees confidentiality by using symmetric key cryptology. Here, the application shares a unique key with each individual user that it uses to encrypt and decrypt all messages being sent to and received from that user. When the License Server receives the question – “Is this customer valid?” it first checks the database to see if the customer has a valid license and prepare the approximate response. To guarantee the confidentiality of the response the server then encrypt the message with an agreed upon symmetric algorithm. This ensures that only the intended user with knowledge of the symmetric key can decrypt and read the answer.
- **Integrity** – Integrity ensures that unauthorized modification is prevented. This introduces the concept of a message digest (sometimes called fingerprint). To ensure integrity of the message, the system uses copy of the encrypted result and passes it through a message digest algorithm. This fingerprint produce will later be used on the client side to guarantee the integrity of the message.
- **Availability** – To make the service available we provide a Web server - Tomcat. Users can take advantage of the service via the Internet.

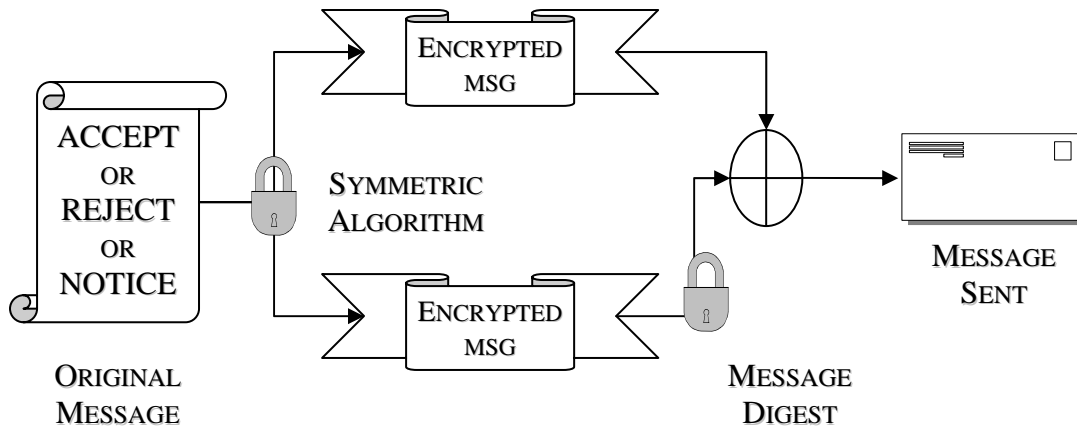


Figure 12: Implementing Confidentiality and Integrity.

Figure 12 gives a view of the how CI of CIA is implemented in the system. First the License Server checks the validity of a customer's license and creates a message. The message is then encrypted with a symmetric key algorithm. A copy of the encrypted cipher is then passed through a message digest algorithm to create a fingerprint of the message. The resulting fingerprint is then added to the encrypted message and sent to the client.

On the client-side, the operations need to be reversed. First, the fingerprint and the encrypted message must be separated. A copy is created of the encrypted message and a fingerprint produced. If the new fingerprint matches the one sent then the encrypted message is decrypted with the user's copy of the symmetric key. Once the answer is received, the VIP Energiberäkningsssystem will act accordingly.

### Graphical User Interface

Some might think why place security at the GUI level? The GUI is the point at which users interact with the underlying License Server and therefore it needs to protect the License Server for accidental or deliberate acts.

- Input Validation** – Not all inputs from users can be trusted. Input could be from malicious users, so all inputs must be filtered (validated) before being accepted and used by the system. This means that the program should determine what is considered legal input and reject anything that does not match this definition [55]. Particular emphasis is paid to this principle of secure software. All inputs that are accepted by the License Server GUI are validated before use otherwise an error message is given when the input is invalid. An example of input validation performed by the License Server is the submitting of an Email address when a new customer is being added. Code 7 show the extract of the code that is used to validate all email addresses. Further validation could be performed to ensure a more rigid adherence to rules of a particular individual, company or institution. With the above code the License Server ensures that all codes received have a standard email format, example `davru088@student.liu.se` or `123xy@someaddress.com`.

---

```

// validate email address
if ( ! email.equals("") ){
    if ( !email.startsWith(".") ){
        Pattern format = Pattern.compile( "^[a-zA-Z0-9_\\.\\-]+
        +\\@((([a-zA-Z0-9_\\.\\-])+\\.)+([a-zA-Z]){2,3})$" );
        if ( !format.matcher(email).find() ){
            dowrite = false;
            errmsg += "\nInvalid Email Format!";
        }
    } else {
        dowrite = false;
        errmsg += "\nCannot Email Start With '.'";
    }
} else {
    dowrite = false;
    errmsg += "\nInvalid Email Address!";
}

```

---

Code 7: Email Validation

Secure software development practices that involve the user(s) get tremendous consideration at the development stage. Users are not unnecessarily required to provide security information [56]. This eliminates the need for repeated login. Users are always not prompted to make security decisions [56]. All security decisions are taken by the License Server.

### 6.1.2 Web Service Security

It has been alluded to earlier that Web service introduces increased security risk to the system. This is because the communication between loosely coupled applications takes place across the boundaries of a trusted network. This introduces the risk that messages can be read by unauthorized users, modified by unauthorized users, deleted or lost. The applications cannot guarantee that they communicate with the intended application.

The first measure taken to make the Web service more secure is to keep it private. The idea behind this is, by not publicly advertising the existence of the Web service this would in turn reduce possibility of attacks. In other words, the more public the existence of a service, the more people knows about the service and therefore the higher the likelihood of an attack or attempted attack. This by no means says that malicious users cannot obtain or find the address of the Web service rather it is made more difficult. Also those who know of the Web service could also have malicious intent.

This means that further measures need to be introduced to further secure the Web service. To combat the security risks introduced by using Web Service, the system used HTTP BASIC-AUTH and HTTPS.

**HTTP BASIC-AUTH** – This is a simple mechanism used in HTTP to protect Web resources from unauthorized access. To access resources protected by HTTP BASIC-AUTH, an application or individual must provide a username and a password.

At invocation time the Web service uses HTTP as the transport protocol, this means that BASIC-AUTH can be used to restrict Web Service access. To use HTTP BASIC-AUTH the Web server must be configured with a list of valid users and resources that a user can access.



The Web server used for the project was Tomcat. The changes necessary for HTTP BASIC-AUTH to work with Tomcat will now be given.

**Step 1: Define user credentials** – To add new user credentials, from the %TOMCAT\_HOME%\conf directory edit “tomcat-users.xml” and add a new user as shown in code 8.

---

```
<tomcat-users>
...
  <!-- Define new user name and password with the role -->
    <user name="wsuser" password="wspwd" roles="wsuser" />
</tomcat-users>
```

---

Code 8: Define new User Credentials

Save changes to the files. Adding these lines to the file creates a new user.

**Step 2: Define security constraint** – Having defined a new role, it is time to assign security constraint to the roles. To add a security constraint, edit “Web.xml” in %TOMCAT\_HOME%\Webapps\axis\WEB-INF and insert the lines shown in code 9 immediately before the end of the element <Web-app>.

---

```
...
<Security-constraint>
  <Web-resource-collection>
    <Web-resource-name>Protected</Web-resource-name>
    <!-- specify the directory for restricted Web Service
         application -->
    <url-pattern>/axis/*</url-pattern>
  </Web-resource-collection>
  <auth-constraint>
    <!-- specify the role name of new user added in step 2 -->
    <role-name>wsuser</role-name>
  </Web-resource-collection>
  <auth-constraint>
</security-constraint>

<!-- Define the Login Configuration for this Application -->
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Protected Web Service</realm-name>
</login-config>
```

---

Code 9: Define Security Constraint

For the changes to take effect Tomcat must be restarted. Any attempt to access a URL that includes the axis directory, for example, the URL <http://localhost:8080/axis/> will cause the browser to request authentication of the user. The users must then enter a valid username and password.

**Step 3: Java client program accessing the Web service** – Code 9 shows a sample Java client program that uses BASIC-AUTH.

---

```

import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import javax.xml.namespace.QName;

public class WSClient {
    public static void main(String[] args) {
        try {
            String endpoint =
                "http://localhost:8080/axis/EchoService.jws";
            Service service = new Service();
            Call call = (Call) service.createCall();
            call.setTargetEndpointAddress( new Java.net.URL(endpoint) );
            call.setOperationName(new QName("echoString"));

            call.setUsername("wsuser");
            call.setPassword("wspwd");

            String ret = (String) call.invoke(new Object[] {"Hello!"});
            System.out.println("Sent 'Hello!', got '" + ret + "'");
        } catch (Exception e) {
            System.err.println(e.toString());
        }
    }
}

```

---

Code 9: Client code invoking Web service with HTTP BASIC-AUTH [34]

By using HTTP BASIC-AUTH, the Web service has been equipped with user validation.

**HTTPS** – Using BASIC-AUTH allows user validation. But, is user validation enough for the security of the system? What security is offered to the message while it is on the wire? The answer is none. BASIC-AUTH only validates users at the point of entry but once the message is sent there is no further protection added. To add security to the message while it is on the wire, HTTPS is used.

Secure socket layer (SSL) is a technology which allows Web browsers and Web servers to communicate over a secure connection. This means that the data being sent are encrypted by one side, transmitted, and decrypted by the other side before being processed. This is a two-way process, meaning that both the server and the browser encrypt all messages before sending out data.

To develop a secure Web Service application using Axis and HTTPS do the following:

**Step 1: Create SSL Certificate** - The SSL certificate is used for authenticating and encrypting data. To create a SSL certificate use the Java “keytool” utility. It generates a self-signed certificate. To create a SSL for Tomcat using keytool execute the following from the command line:

```
%Java_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA
```

Upon executing the command a new file called “.keystore” is created in the home directory. To complete the operation a password (default password is “changeit”) and other certificate information (company, contact name, country, etc) must also be entered. Enter a password for the key store when prompted.

**Step 2: Configure Web server for SSL** - Edit the “server.xml” file located in the '%TOMCAT\_HOME%\conf' directory by uncommenting the lines shown in code 10.

---

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
  <Connector port="8443" maxThreads="150" minSpareThreads="25"
    maxSpareThreads="75" enableLookups="false"
    disableUploadTimeout="true" acceptCount="100"
    debug="0" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS" />
```

---

Code 10: Client code invoking Web service with HTTP BASIC-AUTH [57]

After changing these configuration, you must restart Tomcat, and you should have a working HTTPS. You should now be able to connect to the URL <https://localhost:8080/axis/>.

**Step 3: Java client program for accessing the Web Service using HTTPS** - Code 11 shows a sample Java client program that uses HTTPS.

---

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import javax.xml.namespace.QName;

public class WSClient {
    public static void main(String [] args) {
        try {
            String endpoint =
                "https://localhost:8443/axis/EchoService.jws";
            System.setProperty("javax.net.ssl.trustStore",
                "C:\\Documents and Settings\\Administrator\\.keystore");
            Service service = new Service();
            Call call = (Call) service.createCall();
            call.setTargetEndpointAddress( new Java.net.URL(endpoint) );
            call.setOperationName(new QName("echoString"));

            String ret = (String) call.invoke(new Object[] {"Hello!"});
            System.out.println("Sent 'Hello!', got '" + ret + "'");
        } catch (Exception e) {
            System.err.println(e.toString());
        }
    }
}
```

---

Code 11: Client code invoking Web service with HTTPS [34]

When code 11 is compiled and executed, the data sent between the two applications are encrypted using the RSA asymmetric key algorithm.

Figure 13 shows both HTTP BASIC-AUTH and HTTPS working together to add a level of security to Web service.

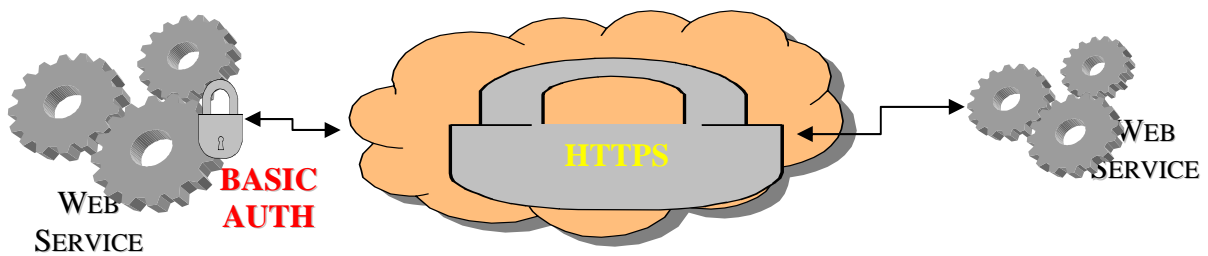


Figure 13: HTTP BASIC-AUTH and HTTPS

By using both HTTP BASIC-AUTH and HTTPS the Web service attempts to ensure Confidentiality using RSA cryptographic technique and to prevent unauthorized access to Web resources.

## 6.2 Open Source Security

The popularity of open source solutions today has re-opened the door and re-kindled flame on a lot of issues that are related to open source. One issue that is being hotly debated is that of security. The central boiling point is the fact that open source exposes source code to everyone. This section examines the views of industry experts about the impact of open source on security.

### 6.2.1 Expert Opinions

Many individuals and companies have had (and still do) something to say about security and open source. Here only a few opinions are presented.

**Bruce Schneier** [58] – A world-renowned cryptographer and security expert argues in favour of open source. From his point of view, “...open source (is) necessary for good security... Public security is always more secured than proprietary security.” He continues by saying “It's true for cryptographic algorithms, security protocols, and security source code. For us, open source isn't just a business model; it's smart engineering practice.”

Using public algorithms and protocols are central to cryptology, as it allows others to examine the algorithms and protocols to ensure they are correct. Schneier argues that open source is vital, because security has nothing to do with functionality. Take for instance two algorithms, one insecure and the other secure, and both work perfectly; never crashing, encrypting and decrypt as desired with efficiency. The only true way to tell the good cryptography is for the algorithm to be examined by experts. Open source offers this, having an algorithm examined by expert – and accepted – gives confidence in the algorithm's security. Schneier makes the claim that “a proprietary algorithm, no matter who designed it and who was paid under NDA to evaluate it, is much riskier than a public algorithm.”

The monetary value involved in security cannot be understated. Do you pay to develop your own in-house security protocols or do you use publicly available protocols? As an example take the Internet IP security protocol (IPSec) and Microsoft's Point-to-Point Tunnelling Protocol (PPTP). IPSec is an open source implementation that has been subject to public scrutiny from the start. The result, based on years of public analysis, is a strong protocol that is trusted by many. On the other hand, Microsoft's PPTP to do much the same thing but they

invented their own authentication protocol, their own hash functions, and their own key-generation algorithm. PPTP was released with Windows NT and Windows 95 for use with their Virtual Private Network (VPN) products. Upon publishing their protocol it was found that PPTP was severely flawed. Here again, the point is made that the only way to tell a good security protocol is to have experts evaluate it in the open.

A sentiment echoed by Schneier is that a smart security engineer should insist on open source code for anything related to security. He goes on to say that one of the greatest benefits of the open source is the positive-feedback effect. The open source model is by no means perfect, but is considered by Schneier to be superior to the other option.

For a detail read on Schneier's view of open source read his Crypto-Gram Newsletter dated September 15, 1999 [58].

**Michael H. Warfield's** [59] – Is a senior researcher at Internet Security Systems Inc. He is a Unix systems engineer, Unix consultant, security consultant, and network administrator on the Internet for well over a decade. Michael has been involved in computer security for over 23 years. In his paper "Musings on open source security" he defended open source security.

Michael started his article by making reference to the cryptographic circles. He makes the argument that, similar to cryptography where the security of an algorithm should not depend on its secrecy the same is true for security software in general. The view of security through secrecy is largely a myth.

In order for software to be secure there is a guideline that should be followed. He gives the following guideline:

- "Secure systems require quality software utilizing secure coding techniques implemented and installed in a manner consistent with security guidelines and policy."

The article attacks some of the myths that surround open source software. He explains that:

1. The nature of open source makes it necessary, not optional, to have control in open source software. This results for the large and diverse development teams that are normally involved.
2. Some are of the opinion that though codes are made available it is not examined. This is of course not the case; example in point is Pretty Good Privacy (PGP) where after the release of version 2.6 someone reported an error in the code which was then corrected.
3. Instead of saying hackers are going to find all the security holes in open source software the real myth is that security holes in closed source will not be found.

Another argument put forth was that the quality of code in open source software was better. This resulted for a sense of personal accountability exists in open source development that doesn't exist in closed source. An open source programmer puts his or her reputation on the line with every line of code he writes. Poor code quality can result in ridicule or worse. In contrast to closed source where the identity is kept secret, the mistakes and foul-ups are not attributed to a programmer but to the company that released the software.

The article goes on further to make claims that, open source promotes secure coding techniques. This he says is as a result of a programmer's involvement in a community of developers that exchange ideas, thoughts, critiques and solutions. This exposes an open source programmer to new ideas and cutting-edge theory regarding coding practice and security.

Michael believes that, fear is a driving factor. People fear what is known and embodied in the open source model, and choose to embrace what is hidden, unknown, and uncertain in the closed source model. With the view that at least no one knows what the source code is under the hood.

**Vincent Rijmen** – The developer of the popular Advanced Encryption Standard (AES) encryption algorithm. Vincent believes that the open source provides a superior means of making security vulnerabilities more easily detected and patched. This results not only for the fact that the code is examined by thousands, but more importantly, because open source force people to write clear code, and to adhere to standards [55].

To this point only those with arguments in favour of open source have been presented, but what for those who are not convinced by these arguments and believe that open source is not secure, simply not totally convinced. The following are some such persons:

**Kenneth Brown** [60] – He is president of the Alexis de Tocqueville Institution (ADTI). In his paper entitled “Opening the Open Source Debate” Kenneth dismissed idea behind most of open source. The paper made particular reference to open source licensing, mainly GPL. Kenneth makes the following analogy; source code is to software like a map is to buried treasure, given the source code the software can be created.

One of the arguments put forward in the paper looks at the economic damage of open source. The paper states that, the fight by GPL to commonplace exchange of open source and proprietary systems has the potential to harmfully impact the research and development budgets of companies. He reasons that, where ownership of intellectual property is affected and with no rights to own the product generated by research and development programs, dollars spent on research and development would be at risk as well and such program could become obsolete.

Any widespread acceptance of GPL copyleft policy would present a radical change to an industry sector responsible for billion dollars in sales annually worldwide. Kenneth further argues that open source with GPL can coexist status quo. GPL open source is a “gift that keeps taking” and therefore has no place with traditional open source and close source. Kenneth describes Stallman as “an extremist and fanatic... a fallen hero in the open source world...”

Important points made by the article can be summarised as follows:

- Open source would change the incentive to develop software and would result in a corresponding change in the quality and efficiency of the software.
- The GPL open source community is headed for doom. The future that is in store for GPL is either some form of change or discontinuity.

- The use of open source GPL software by the government is an area of national concern (national security concern that is) and therefore such use in the public domain (for example by government) is extremely risky.
- Reverse engineering is a bad idea that threatens the software industry.
- The courts have yet to weigh in on the General Public License. Without legal interpretation, the use of the GPL could be perilous to users in a number of scenarios.

**Ian Tien** [61] – In a paper entitled “Open Source and Assurance” Ian looked at open source and discussed the pros and cons. He presented the following as some of the problems that plague open source security.

The paper was of the view that, the lack of infrastructure is the major barrier to open source security. System security is a holistic art. Open source system cannot provide the security assurance because of the lack of automated tools to rigorously test huge volumes of complex codes.

Accountability is missing for open source. Ian raises the point that individuals and institutions cannot risk using ill-fated and untested software that provides no accountability, warranties or liability and some uses of open source are unprepared to support security issues in their software.

Another limitation of open source is its lack of infrastructure. Open source community cannot easily compete with proprietary designs that often come with architects, analysis tools and automated testing environments.

The paper makes the claim that bugs are located and fixed at a faster rate in proprietary software than in open source. This argument is based on the fact that the open source relies on volunteer networks which operate at a slow pace.

Ian argues that there are benefits in security by obscurity. There are some vulnerabilities that cannot be defended, possibly due to economic reason maybe, and therefore the only security against these types of vulnerabilities is security through obscurity.

**Fred Schneider** – Is a professor and expert in the computer industry. Fred doesn't believe that open source helps security, with the view that there is no reason to believe that the many eyes inspecting source code would be successful in identifying bugs that allow system security to be compromised. Schneider further states that bugs in the code are not the dominant means of attack [55].

## 6.2.2 Further Open Source Precautions

This short sub-section accounts for further measures that can be taken to secure the open source solutions, in particular, it focuses on the MySQL database.

Installing the MySQL database creates several accounts. These default accounts (ghost accounts) are a potential security risk if not removed. To remove ghost accounts go to %MySQL\_HOME%\bin and do the following:

- Open MySQL
  - C:\www\mysql\bin> mysql -u root -p
- Display all accounts.
  - mysql> SELECT User, Host, Password FROM mysql.user;
- Select the 'mysql' database.
  - mysql> USE mysql;
- Delete all initial accounts, except root@localhost.
  - mysql> DELETE FROM mysql.user WHERE User = '' OR Host = '%';
  - mysql> DELETE FROM mysql.db WHERE User = '';
  - mysql> FLUSH PRIVILEGES;
- Exit.
- mysql> quit;

MySQL accounts that do not have passwords should either be removed or assigned passwords. The root@localhost does not have a password and therefore must be assigned one to do this type the following:

- mysql> SET PASSWORD FOR 'user\_name'@'host\_name'=PASSWORD('new\_pass');

The changes recommended remove ghost account and sets a password for the root@localhost account, both of which are possible security holes.



## 7 Testing

The project started with an aim of finding a solution to the problem of a loosely coupled system. Web service is used in order to provide an answer. By using Web service, the aim is to provide a solution application-to-application interaction that has secure and interoperability.

At this point there is one question that has gone unanswered; what happens when the system is built and tested? This chapter describes two scenarios under which the system is tested. The scenarios focused on testing the interaction between the VIP Energiberäkningssystem and the License Server. By performing these tests knowledge is gained as to whether or not the Web service and the License Server do indeed work.

### 7.1 Scenario Description

This scenario depicts the standard way in which the system is intended to run.

A customer has purchased a copy of the VIP Energiberäkningssystem to run as a standalone application – VIP+. The system is installed and along with the system the customer receives a Customer ID and a unique fingerprint and a key for cryptographic purposes.

When the application starts it checks for the existence of a valid fingerprint. At pre-defined interval the application must interact with the License Server to determine if the customer's license is still valid. If the License Server validates the customer then the application will execute as intended, but if the Server rejects the customer the applications will not execute.

When the application requires interaction with the License Server, the application then uses Web service to send a request to the License Server. The License Server performs the action required by the service and returns a response using Web service.

To simulate the above scenario, two separate testing environments are created. The test environments were: testing on a standalone system and testing in a network environment.

In an attempt to create a close to realistic working environment, there is a need to populate databases with test data of customers. With the help of the GUI that accompanied the License Server application two databases are created and populated. One database houses information on Customer, example name, address and customer id. The other database stores the license detail for each customer, example expiration date.

During the test phase the database stores information on thirty five to fifty customers at any given time. Some customers have licenses that were valid; some had invalid licenses while others held no licenses.

## 7.2 Standalone Testing

Standalone environment is built around a notebook running Windows XP on an Intel Pentium 4 2.9 GHz processor with 512 MB of RAM. In order to run the system Java 2 SDK 1.4 is installed to provide the platform on which to install the other components. MySQL database is also installed. Tomcat is used to provide the Web server and Axis 1.1 provides the Web service engine.

The system used at this stage of testing is the same system with which the project was developed. Throughout each step of development the pieces are tested individual once created but no account will be made of this early modular test. The test here therefore refers to those conducted under the scenario described above. Under the standalone environment tests were conducted for a period of one week with an average of twenty tests day.

Additional measures are also taken to make the database more secure by deleting ghost accounts and assigning password to the remaining account.

Under this test environment, HTTP is the chosen protocol used to transport the Web service messages.

HTTP BASIC-AUTH and HTTPS are added to HTTP to provide a measure of security to the Web resources and the data on the wire. HTTP BASIC-AUTH provided user validation while HTTPS offered encryption. The License Server also added another layer of security before sending a message or response to the client.

Before a client is permitted to use the service offered, the client is first required to authenticate itself. Failure to do so results in denial of the service. Also, HTTPS is used to ensure that all messages sent between the two parties are encrypted and thus private (this includes the user credentials). Prior to sending a message the License Server performs encryption as well as creating a message digest, after which the result is then sent over HTTPS.

The client-side code refers only to the pieces of codes that would later be integrated in the VIP system and used to perform Web service call and hand-over an answer to the VIP Energiberäkningssystem.

Within this test environment where the system is evaluated on a single machine, the address used to represent the Web server and simulate interaction over the Web or Internet is localhost. The address of the License Server is therefore <http://localhost:8080>.

The system is then tested in the following stages:

1. Interacting between the License Server and the VIP Energiberäkningssystem.
2. Using HTTP BASIC-AUTH.
3. Using HTTP BASIC-AUTH and HTTPS

### 7.2.1 Evaluation

#### 1. Interacting between the License Server and the VIP Energiberäkningssystem.

The first aim of testing is to determine whether or not it is possible to achieve the desired result of interaction between the License Server and the VIP Energiberäkningssystem. At first, the VIP Energiberäkningssystem was unable to communicate with the Server. After several attempts and restarting of the system, the database and the Web server, communication between the two was achieved. From this point forth there has been a one hundred percent connectivity rate between the two applications.

To test the correctness of the License Server's response, a mix of customer data is sent by client. In all case the Server responded with the correct answer. Table 1 shows how the License Server reacts to customer data.

CUSTOMER DATA	PERMISSION GRANTED
Valid Customer ID + Valid License	YES – 100%
Valid Customer ID + Invalid License	NO – 100%
Invalid Customer	NO – 100%

Table 1: License Server Decision

A valid customer is one that has purchased a license to run the VIP system and the data about the customer can be found in the License Server database. A valid license is one where the date of the license has not expired.

When a valid client with a valid license requests permission the Server grants permission to the client. If the client is valid but the license is not, then permission will be denied. If the client is invalid permission is also denied.

#### 2. Using HTTP BASIC-AUTH.

After achieving the first milestone, it is now time to add additional security to the system. BASIC-AUTH provides simple user validation and attempts to prevent unauthorized access to the Web service.

Once the appropriate changes are made in the client and Web server, the system is then tested using BASIC-AUTH. The results are very satisfying. No problems are encountered during this stage. Access to resources protected by BASIC-AUTH is denied unless a correct username and password is provided.

The system continued to function as before with interaction between the two applications yielding the expected results when given mixed customer data.

#### 3. Using HTTP BASIC-AUTH and HTTPS

HTTPS can be used in two ways; server-side authentication or server-side/client-side authentication. In server-side authentication the server authenticates itself to the client during the initial handshake process by showing its certificate. While in server-side/client-

side authentication both parties authenticates themselves to each other by showing certificates.

This final layer of security ensures the wire encryption. After making the changes to accommodate HTTPS, the results are mixed.

When HTTPS is used with server-sided authentication the system continued to function. It is possible for the VIP Energiberäkningssystem to communicate with the License Server using HTTPS. The applications are able to interact in a very secure manner.

The result is not the same when server-side/client-side authentication is used. Once server-side/client-side authentication is added interaction between the two applications seize. The problem resulted from the Web server's refusal to accept the client's certificate. Attempts to recreate the certificates do not solve the problem.

At the end of testing, achieving interaction with HTTPS using server-side/client-side authentication is not possible.

### 7.3 Network Testing

Testing is also conducted within a network environment; a small home network. The network consists of two computers, both Intel base processors.

The server is a personal computer with Windows XP as the operating system running an Intel III 500 MHz processor having 128 MB of RAM. As server, the machine needed to run the License Server application. Therefore Java 2 SDK 1.4 is installed to provide the platform. MySQL database is also installed. After doing this, the License Server is functional as a standalone application. To enable client application(s) to local the server, Tomcat is used to provide a Web server and Axis 1.1 gives the Web service engine needed for application-to-application communication.

The client machine is a notebook running Windows XP with an Intel Pentium 4 2.9 GHz processor with 512 MB of RAM. Java 2 SDK 1.4 is installed to provide the platform on which the client code would run. And once again Axis 1.1 is installed to provide the Web service engine. The client-side code is then placed on the notebook.

The client-side codes do not include the actual VIP Energiberäkningssystem. It refers to lines of code to be integrated into the VIP system that would perform Web service calls and hand-over response and control to the VIP Energiberäkningssystem.

The network had an ADSL connection to the Internet, an ADSL modem and a hub.

Network testing is conducted for a period of one week, again with an average of twenty tests day.

Once again security is at the forefront and measures are taken to make the database more secure by deleting ghost accounts and assigning a password to the remaining account.

HTTP is the protocol chosen to transport the Web service messages. With HTTP BASIC-AUTH and HTTPS a layer of security is added to HTTP. HTTP BASIC-AUTH provided user

validation while HTTPS offered encryption. Firstly, HTTPS is used to ensure that all messages sent between the two parties are encrypted and thus private (this includes the user credentials). Next, before a client is permitted to use the service on offer the client is required to authenticate itself. Failure to do so results in denial of the service. Thirdly, before sending a message the License Server performs encryption as well as creating a message digest, the result is then sent over HTTPS.

In order for the applications to communicate within the network, both the client and the server need a unique IP address. Address is issued by the DHCP server within the ADSL modem. The address of the Web server was the most important as it is with this address that the client would locate and interact with the service.

Testing is once again carried out in the following stages:

1. Interacting between the License Server and the VIP Energiberäkningssystem.
2. Using HTTP BASIC-AUTH.
3. Using HTTP BASIC-AUTH and HTTPS

### 7.3.1 Evaluation

Most of the results in all three categories mirrors the results obtained during Standalone Testing, with one exception.

1. Interacting between the License Server and the VIP Energiberäkningssystem.

Once again the first and foremost goal of testing is to determine whether it was possible to achieve application-to-application communication between the License Server and the VIP Energiberäkningssystem. Communication between the two worked without any problems; there is a one hundred percent connectivity rate between the two applications.

Extensive testing is performed on the License Server to determine its response to incorrect data. In all case where incorrect or invalid data is sent the License Server acted accordingly.

Table 1 show that in all instances where a license is invalid or where a customer is invalid, access to the service was denied. In case where a valid client with a valid license request permission the Server grants permission to the client.

2. Using HTTP BASIC-AUTH.

Communication between the applications works as desired and all data sent between the two are encrypted, however there is a need for additional security. BASIC-AUTH provided simple user validation that is aimed at preventing unauthorized access to the Web service.

Then communication is tested using BASIC-AUTH no problems are encountered. Access to resources protected by BASIC-AUTH is denied, unless a correct username and password is provided.

The system continued to function as before with interaction between the two applications yielding the expected results when giving mixed customer data.

### 3. Using HTTP BASIC-AUTH and HTTPS

BASIC-AUTH sends user credentials in the clear and offers no additional security to the communication process. HTTPS is used to ensure on the wire privacy. The necessary changes were made to facilitate the use of HTTPS. It is here where results differed from those in Standalone Testing.

With one-way authentication (server-sided authentication) the system continues to function. It is possible for the VIP Energiberäkningssystem to interact with the License Server using HTTPS offering a more secure communication.

The difference between the results comes when testing using two-way authentication (server-side/client-side authentication) is performed. With two-way authentication, the applications do interact with each other.

The fact that HTTPS two-way authentication functions in this scenario and not in Standalone Testing could be attributed to various reasons. One reason could be the fact that with Network Testing the certificates are housed on separate machines while with Standalone Testing both certificates are on the same machine. So failure could be from the fact that the system is not able (for unknown reasons) read or switch between both certificates when required.

Testing conducted under the above mentioned two scenarios shows that Web service does in fact provide application-to-application interaction, but is these tests enough? Frankly, no!

The system needs move rigorously large scale testing. This includes testing scenarios were several users connect to the Web service simultaneously. This would provide valuable information on Web service's ability to handle a large number of simultaneous ongoing connections. Also, test needs to be conducted where the Web service is exposed to different communication/connection speed. How will the Web service respond if the connection speed was reduced?

These and other similar tests should be conducted before the system that uses Web service is placed on the market.

## 8 Delivery

The effort put forth in this assignment does not stand by itself; it is part of a much larger, groundbreaking and existing project. The result of the work carried out during this thesis will therefore closely examined with a view of integrating the solution or the premise on which the solution was based into the wider project. Before this thesis can be considered complete, there need to be an explanation as to how the solutions developed are ported to StruSoft machines and also consider what needs to take place on the customer side; this includes software update and configuration changes.

### 8.1 StruSoft

Solutions presented at the end of the thesis are developed on an IBM ThinkPad running Windows XP and needs to be ported to StruSoft's computers. Several issues must be considered and addressed at this critical stage, some of the main issues include:

- **Installation** – This is a very important step. There needs to be a clean, clear and quick process to get the software up and running. The first step in adoption is the installation and testing of the solutions by StruSoft to determine whether they meet necessary criteria.

Detailed explanations about how to install the necessary components are given in Chapter 4, Development Environment. Once these instructions are followed (and installation manuals read) there should be very little or no problem getting all the necessary components in place. Additional configuration relating to security should also be carried out, for this, visit Chapter 6 on Security Issues. Follow the instructions to have a more secure system.

Full source code for the License Server application will be handed over to StruSoft. To get the License Server up and running simply compile and execute the source code. The License Server application provides among other things a graphical user interface to certain database activities and it can be used to input data for testing.

All necessary files for Web service will also be made available. For speed and simplicity a Java code is written that handled the deploying and un-deploying of the service.

A running License Sever and Web service depends upon the installation process, once this process is correctly done, StruSoft can then perform testing on the two as they see fit.

- **Integration** – How to fit the solutions into the overall project? What are the activities that are necessary for integration?

The License Server is developed as a standalone application and therefore need not be integrated with any other software. It provides the project with a means by which customer information and license information are maintained using an underlying database. The application also performs email, logging and Web service functionality. These functionalities are carried out independent of other software and therefore there is no need for the License Server to interact with other components in the system.

The Web service on the other hand must be integrated into existing software. The client-side codes generated must be integrated into the VIP Energiberäkningssystem, as the codes provide the means by which the VIP Energiberäkningssystem interacts with the License Server to determine whether or not to run the system.

The integration requires that certain lines of codes be placed within the codes for the VIP Energiberäkningssystem. Once this is done the two applications can interact using the Web service.

- **Functionality** – Will the solutions continue to work when ported to a different machine, there could be a difference in operation systems for example?  
One of the major benefits gained from choosing the Java platform is that it gives the project platform independence. Java gives the project a “write once run anywhere” feel. This means that if there is a change in the underlying operating system there will be no effect on the solutions present.  
If the operating system is different then StruSoft need only to follow the instructions provided by the various user manuals as to how to install Java and the other components on that system. There exists manual and extensive resource on how to install the required component on several different operating systems. Once all the components are installed and configured as instructed the License Server and Web service will function in the correct manner.  
There is also no restriction on the database used. A decision can be made to the change the database; this would only require changes in a few lines of codes. All database operations are performed with SQL which has no relation to the underlying database.  
Web service as examined before is built using industry standard protocol and therefore it offers platform independence as well.
- **Scalability** – Bearing in mind that development was done in a single user environment and when integrated into the bigger project, will it fit into a multiple user environment?  
Though the system is developed and tested primarily in a single user environment, it has in no way affected the scalability of the final result. Web Service offers a business solution that is extremely scalable. One of the most important features of Web service is its scalability.

Because most of the component used to generate the solutions for this project is platform independent and relied on industry standard the process of delivery to StruSoft is for the most part getting the installation and configurations correct.

## 8.2 Customers

In order for the system to function in its totality, consideration must be given to the client-side. The client-side changes needed are not as many but nonetheless very important. These changes are:

- **Code Update** – In order for the VIP Energiberäkningssystem to communicate with the License Server the client-side codes generated must be added to the VIP system. This is a simple but very important requirement.
- **Installation**- On the client-side only two components are required, Java platform and Axis. Installation of Java is no different but there are differences in the Axis installation.

In order for the application to use Axis the following needs to be redistributed [62]:

- axis.jar



## 8 • Delivery

- commons-logging.jar
- commons-discovery.jar
- log4j.jar or simply use the logging facilities that is shipped with Java1.4.
- xerces XML Parser or crimson that is shipped with Java1.4.

Set the class path to these files and restart the system. It is also possible to place these files in the applications home directory without setting the class path but this approach is less preferred.

Once the necessary changes are made and the installation complete, it should be possible to run the VIP Energiberäkningssystem. The application will now be able to interact with the License Server via Web service technology.

## 9 Conclusions

StruSoft has embarked on a project that involves the development of a complex and loosely coupled system. The software required applications to communicate beyond the boundaries of a single network. This paper makes two contributions to the project.

The first contribution is the use of Web Service technology to present a solution to the problem faced by loosely coupled distributed application. Web Service is a relatively new technology that allows sharing of services between communicating applications. Not only is Web Service used but, the technology is used in a secure manner. Testing showed that it is possible to use this technology to solve the problem faced by distributed applications. The results demonstrated that with Web Service the project benefited from a solution that is scalable, interoperable and secure while allowing loosely coupled applications to interact.

The second contribution made by this thesis is the development of a License Server application and client-side code. The application solves some very important needs of StruSoft and allowed a controlled environment where customer and license data can be stored and maintained. Testing confirmed that all the functionalities required have been built into the application. And further non-functional requirements such as security were also built into the application.

Web Service is definitely not the answer to all distributed computing problems, but they definitely solve the problem presented within this project. The result gained from testing gave evidence that Web Service application-to-application communication can take place in a secure manner.

The thesis also relied heavily on open source tools and solutions. It showed that open source solutions can be used to deliver solutions that are secure. There is no clear decision on the use of open source, in the end it boils down to making an informed decision. The final decision on open source will not be in for some time yet as the industry will continue to debate the topic; of this I am sure.

### 9.1 Future Works

The project uses the Java programming language and platform to provide a platform neutral solution. The client-side code generated during this thesis must be incorporated into the VIP Energiberäkningssystem in order for the application to interact with the License Server. The fact is the client-side code is written in a different language from the VIP Energiberäkningssystem. The process of incorporating the two is not difficult but it is certainly not the most ideal. It would be much more appropriate and efficient if the client-side code was converted to the language of the VIP system. This would therefore remove the need to install Java SDK on the client-side.

The security solutions implemented within the License Server application and the client-side uses the Java security class. If the decision is taken to convert the client-side code future work must then be done to determine how and if it is at all possible to encrypt and decrypt message on both application using the same key and encryption algorithm but different language. This would certainly be an interesting venture.

## 9 • Conclusions

The security measures incorporated into the entire system are thought to be enough but the question remains, how much security is secure enough? It would also be an area that could be looked at, adding future security to the system. XML encryption and XML Digital Signature are just two possibilities.

Last but not least, the use of UDDI as the popularity of the system becomes a reality. Apart from excitement about the growth of the system, it would also be interesting to see how additional Web Service technology can be used to bring the system to a wider audience.

## 10           References

- [1] Vivek Chopra, Zaev Zoran; Gary Damschen, Chris Dix, Patrick Cauldwell, Rajesh Chawla, Kristy Saunders, Glenn Olander, Francis Norton, Tony Hong, Uche Ogbuji, Mark A. Richman, "Professional XML Web Service", Wrox Press Inc., ISBN 1-861005-09-1
- [2] <http://www.strusoft.com/OmStruSoft/OmStruSoft.asp?link=About%20StruSoft> 2004-12-06
- [3] <http://www.strusoft.com/produkter/produkter.asp?link=Products> 2004-12-06
- [4] <http://www.strusoft.com/tjanster/tjanster.asp?link=Services&Hilite=0> 2004-12-06
- [5] Mats Ola Rasmusson, VIP Energiberäkningssystem StruSoft 2004-06-24
- [6] Leslie Lampart, DEC SRC 1980
- [7] <http://www.hrm.uct.ac.za/uct-year2000/glossary.htm> 2004-12-07
- [8] Kurt Cagle, Dave Gibbons, David Hunter, Nikola Ozu, Jon Pinnock, Paul Spencer, "Beginning XML", Wrox Press Inc., ISBN 1-861003-41-2
- [9] <http://www.omg.org/gettingstarted/corbafaq.htm> 2004-12-07
- [10] <http://Java.sun.com/developer/onlineTraining/corba/corba.html> 2004-12-07
- [11] <http://www.corba.org> 2004-12-07
- [12] <http://Java.sun.com/products/jdk/rmi/> 2004-12-07
- [13] <http://www.cisco.com/univercd/cc/td/doc/product/voice/evbugl4.htm> 2004-12-07
- [14] [http://searchvb.techtarget.com/sDefinition/0,,sid8\\_gci213883,00.html](http://searchvb.techtarget.com/sDefinition/0,,sid8_gci213883,00.html) 2004-12-08
- [15] <http://www.microsoft.com/com/> 2004-12-09
- [16] Feilding et. Al, "http/1.1", <http://www.rfc-editor.org/rfc/rfc2616.txt> 2004-12-08
- [17] <http://www.wmo.ch/Web/www/WDM/Guides/Internet-glossary.htm> 2004-12-08
- [18] Elliotte Rusty Harold, "XML Bible" 2<sup>nd</sup> Edition, Wiley Publishers, ISBN 0-7645-4760-7
- [19] <http://www.w3.org/TR/xml-infoset/> 2005-02-02
- [20] Box, D et al., "Simple Object Access Protocol (SOAP) 1.1", W3C Note May 8, 2000, <http://www.w3c.org/TR/SOAP/> 2004-12-08
- [21] <http://www.w3.org/2003/05/soap-envelope> 2004-12-08
- [22] <http://www.w3.org/TR/wsdl> 2004-12-08
- [23] UDDI.org, UDDI Technical White Paper, URL: [http://www.uddi.org/pubs/Iru\\_UDDI\\_Technical\\_White\\_Paper.pdf](http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf), 2000. 2004-12-09
- [24] <http://www.w3.org/TR/ws-arch> 2005-01-26
- [25] Gollmann, Dieter, "Computer Security", John Wiley & Sons Ltd, 2002, ISBN 0-471-97844-2
- [26] <http://www.w3.org/Protocols/HTTP/1.0/draft-ietf-http-spec.html>, 2004-12-09
- [27] <http://Java.sun.com/developer/onlineTraining/Security/Fundamentals/Security.html>, Security 2004-12-09
- [28] <http://Java.sun.com>. 2004-12-14
- [29] [http://www.cs.ucsb.edu/~teliot/Path\\_and\\_Classpath.htm](http://www.cs.ucsb.edu/~teliot/Path_and_Classpath.htm) Setting the PATH and CLASSPATH on Windows XP 2005-01-06
- [30] <http://dev.mysql.com/downloads/index.html> 2004-12-15
- [31] <http://www.mysql.com/products/connector/j/> 2004-12-15
- [32] <http://jakarta.apache.org/tomcat/index.html> 2004-12-14
- [33] <http://ws.apache.org/axis/index.html>. 2004-12-15
- [34] Installing and deploying Web applications using xml-axis, Version 1.1
- [35] [http://www.disinfopedia.org/wiki.phtml?title=Open\\_Source\\_Software](http://www.disinfopedia.org/wiki.phtml?title=Open_Source_Software), Open Source

- Software 2004-12-16
- [36] Peter H. Salus, "A Quarter Century of UNIX", Addison-Wesley Professional, 1994 ISBN 0-201-54777-5
  - [37] <http://www.bell-labs.com/history/unix/> 2004-12-16
  - [38] <http://www.opensource.org/docs/history.php> 2004-12-16
  - [39] [www.gnu.org/gnu/thegnuproject.html](http://www.gnu.org/gnu/thegnuproject.html) 2004-12-16
  - [40] <http://www.linux10.org/history/> 2004-12-16
  - [41] <http://www.qsl.net/kd2bd/linux.html> 2005-01-03
  - [42] <http://www.e-envoy.gov.uk/assetRoot/04/00/28/41/04002841.pdf> 2005-01-06
  - [43] <http://www.fsf.org/gnu/thegnuproject.html> 2005-01-03
  - [44] <http://www.opensource.org/docs/definition.php> 2005-01-05
  - [45] <http://apache.org>. 2004-12-14
  - [46] <http://www.netcraft.co.uk/survey/> 2004-12-14
  - [47] <http://jakarta.apache.org/tomcat/> 2004-12-14
  - [48] <http://www.apache.org/licenses> 2004-12-15
  - [49] MySQL Reference Manual for version 4.0.21.
  - [50] <http://www.fsf.org/licenses/> 2004-12-16
  - [51] <http://www.eclipse.org/> 2005-01-06
  - [52] <http://www.eclipse.org/whitepapers/eclipse-overview.pdf> Eclipse Platform Technical Overview 2005-01-06
  - [53] Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli, "Fundamentals of Software Engineering", Prentice Hall, Second Edition, ISBN 0-13-305699-6
  - [54] [http://ibm.com/developerWorks/Java/Security, Part 1: Crypto basics](http://ibm.com/developerWorks/Java/Security/Part1/CryptoBasics) 2004-10-06
  - [55] David A. Wheeler, "Secure Programming for Linux and Unix HOWTO", <http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO.html> 2004-10-10
  - [56] Razvan Peteanu, "Best Practices for Secure Development", <http://members.home.net/razvan.peteanu> 2004-10-10
  - [57] SSL Configuration HOW-TO, <http://jakarta.apache.org/tomcat/tomcat-4.0-doc/ssl-howto.html> 2004-09-15
  - [58] Bruce Schneier, Crypto-Gram Newsletter, September 15, 1999 <http://www.schneier.com/crypto-gram-9909.html> 2005-01-12
  - [59] Michael H. Warfield's "Musings on open source security" <http://www.br.fgov.be/SCIENCE/INFORMATICS/doc/ramparts.html> 2005-01-12
  - [60] Kenneth Brown, "Opening the Open Source Debate" 2002, [http://parrhesia.com/old\\_opensource\\_whitepaper.pdf](http://parrhesia.com/old_opensource_whitepaper.pdf), 2005-01-12
  - [61] Ian Tien, "Open Source and Assurance" <http://www.cs.cornell.edu/courses/cs513/2002fa/opt1.soln.it33.pdf> 2005-01-12
  - [62] "Client Side Axis" <http://ws.apache.org/axis/Java/client-side-axis.pdf> 2005-01-18

## 11 Glossary

Axis	Axis is an implementation of the SOAP submission to W3C.
BASIC-AUTH	Basic Authentication. Used to introduce a measure of security into the otherwise insecure HTTP protocol.
CIA	Confidentiality, Integrity and Availability. Three fundamental principles of computer security.
Copyleft	Policy or law applied to open source product.
CORBA	Common Object Request Broker Architecture. It is described as an open, vendor-independent framework that permits distributed computer applications to interoperate over networks.
DCOM	Distributed Component Object Model. The DCOM solution enables applications to interact directly over a network
DES	Digital Encryption Standard. Symmetric key algorithm used for encrypting and decrypting messages.
ECLIPSE	The Eclipse Platform is a kind of universal tool platform, an IDE for anything and for nothing in particular.
GNU	GNU's Not Unix. An open source project that offers a complete operating system and accompanying software.
GPL	General Public License. License use for distribution of open source products.
HTTP	Hypertext Transfer Text Protocol. HTTP is a transport level protocol that often carries SOAP messages.
HTTPS	HTTP with security. See TLS.
IDE	Integrated Development Environment. Set of development tools such as editors, debuggers and executing environment all tied together in a single software.
IDL	Interface Definition Language. The method used to expose services that are put on offer a service provider.
IIOP	Internet Inter-ORB Protocol. Allows CORBA-based program can interoperate with another CORBA-based program.
IPSec	Internet Protocol Security. A specification for encryption of messages.
Java	A platform neutral programming language and environment.

LINUX	Open source, freely available operating system.
MD5	Message Digest 5. Algorithm for generating the fingerprint of a message. Used for Integrity.
MySQL	The MySQL software delivers a very fast, multi-threaded, multi-user, and robust Structured Query Language database server.
OMG	Object Management Group. OMG is responsible for defining CORBA standard.
ORB	Object Request Broker. The ORB acts as a bus that finds the remote object, communicates the request, waits for the result(s) and then sends back the result to the client.
RSA	Rivest Shamir Adelman: name after its founders. Asymmetric key algorithm used for encrypting and decrypting of messages.
RMI	Remote Method Invocation. Enables software distributed computing by facilitating object communication between distributed applications in which calls are made between objects.
SHA-512	Secure Hash Algorithm 512. Algorithm used for creating fingerprint of a message.
SOAP	Simple Object Access Protocol. SOAP is a XML based protocol that is simple and lightweight used to facilitate information exchange in a decentralized distributed environment
SMTP	Simple Mail Transport Protocol. Used for sending and receiving messages, commonly used for email communication.
SOA	Service Oriented Architecture. This is basically an assortment of services that communicate with each other.
TLS	Transport Layer Security. Provides authentication and communication privacy between communicating applications over the internet.
UDDI	Universal Description Discovery and Integration. It is a platform-independent framework for describing services, discovering businesses and integrating business service by using the internet
UNIX	Open source operating system.
W3C	World Wide Web Consortium. One of the governing bodied of the Web that sets standards and polices the Web.

WSDL	Web Service Definition Language. WSDL provides documentation for distributed systems and serve as a recipe for automating the details involved in application communication.
WSIF	Web Service Invocation Framework. WSIF is a Java API for invoking Web Service.
WSS	Web Service Security. Specification for encrypting Web Service messages.
XML	eXtensible Markup Language. It is a meta-language, allowing designers the ability to represent data in a self-describing manner.

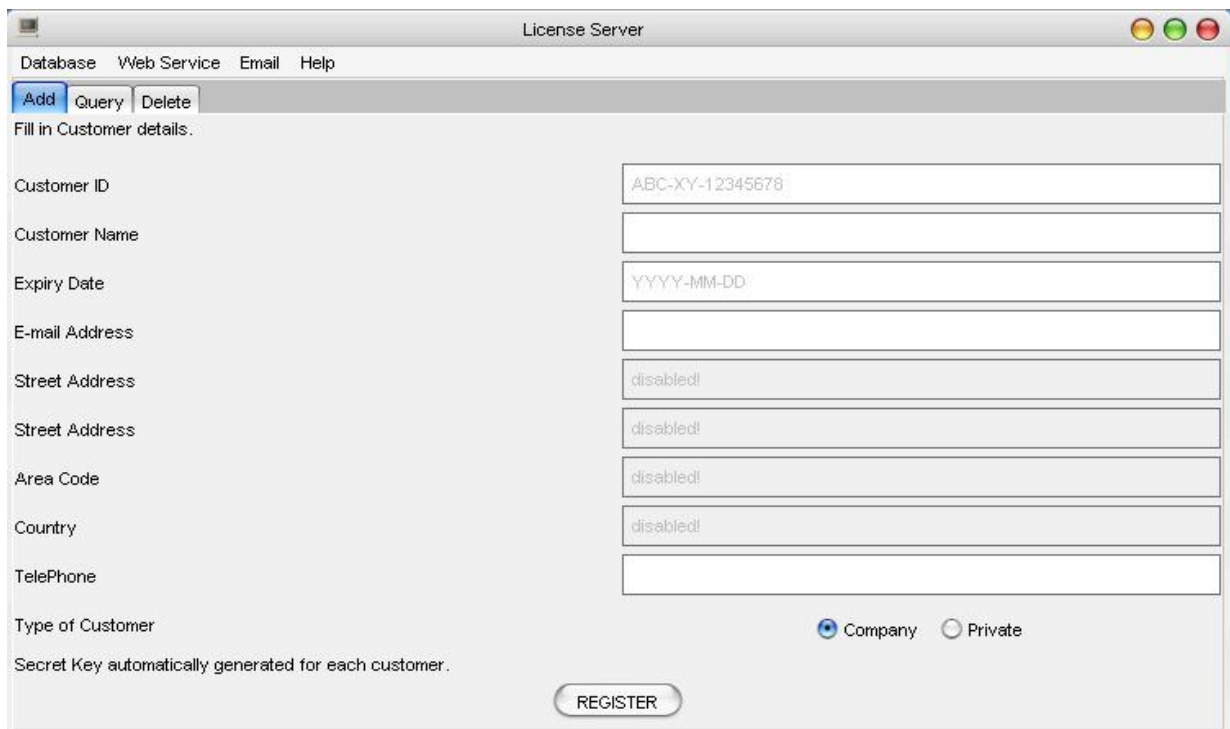


## A License Server Screenshots

The License Server application provides a graphical user interface for performing certain database operations as well as allowing a user to view the logs, start and stop email process plus monitoring of the Web service.

### A.1 Adding Customer to Database

Shot 1 show the view of the interface that is provided when a customer is to be added to the database. Strict input validation is performed on all entered. Once completed the user is added to the database, a symmetric key created and license information stored.



The screenshot shows a window titled "License Server" with a menu bar containing "Database", "Web Service", "Email", and "Help". Below the menu bar are three buttons: "Add" (highlighted in blue), "Query", and "Delete". The main area is titled "Fill in Customer details." and contains the following fields:

- Customer ID: ABC-XY-12345678
- Customer Name: (empty)
- Expiry Date: YYYY-MM-DD
- E-mail Address: (empty)
- Street Address: disabled!
- Street Address: disabled!
- Area Code: disabled!
- Country: disabled!
- TelePhone: (empty)
- Type of Customer:  Company  Private

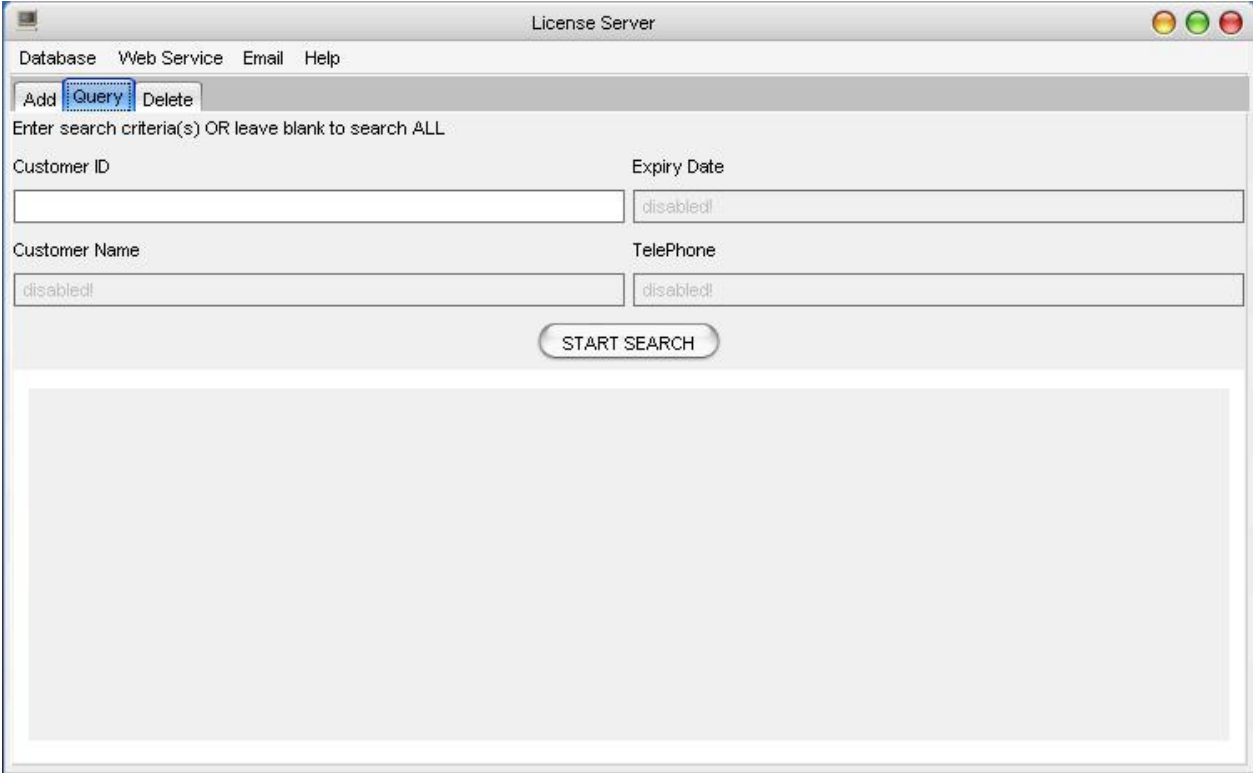
Below the fields, it says "Secret Key automatically generated for each customer." and there is a "REGISTER" button.

Shot 1: Add Customer

### A.2 Search Database for Customer

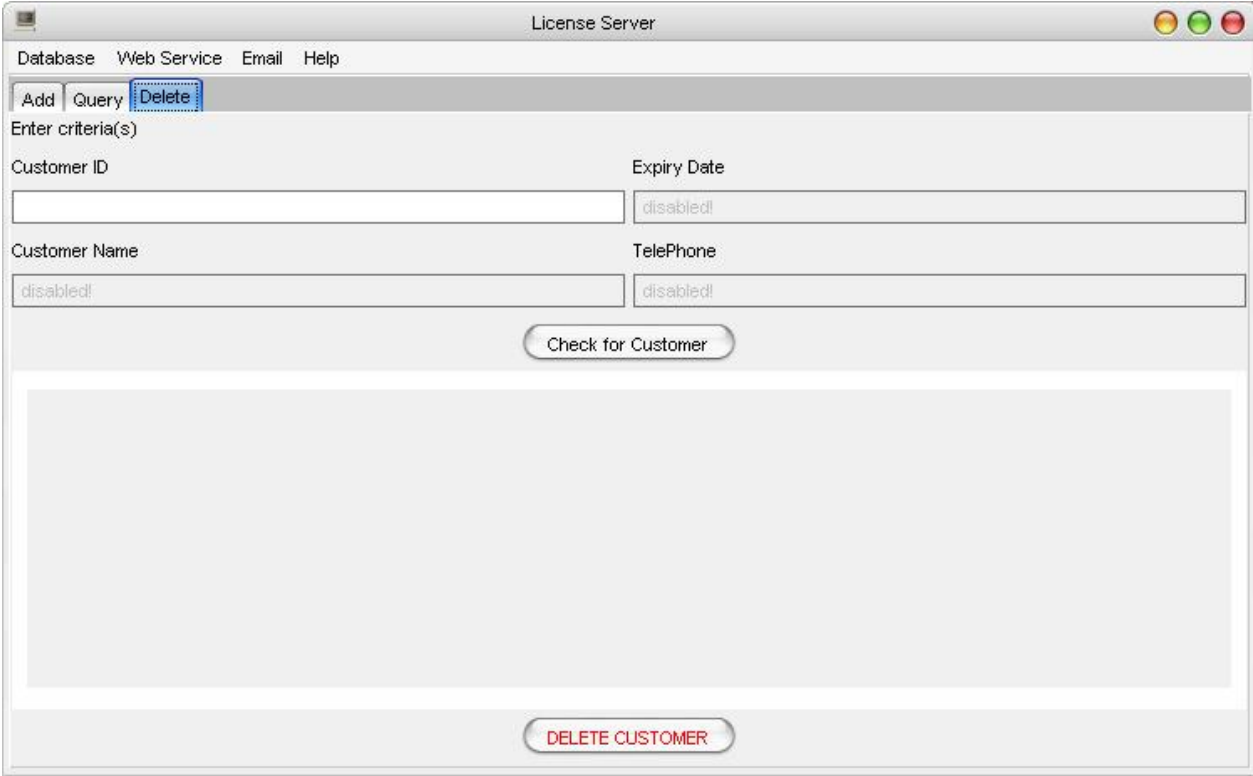
Shot 2 show the view of the screen presented when a user wishes to search the database.

A wide search can be performed on all customers by leaving the fields blank or a more specific search performed by filling in data such as Customer ID.



Shot 2: Search Database

### A.3 Delete Customer from Database

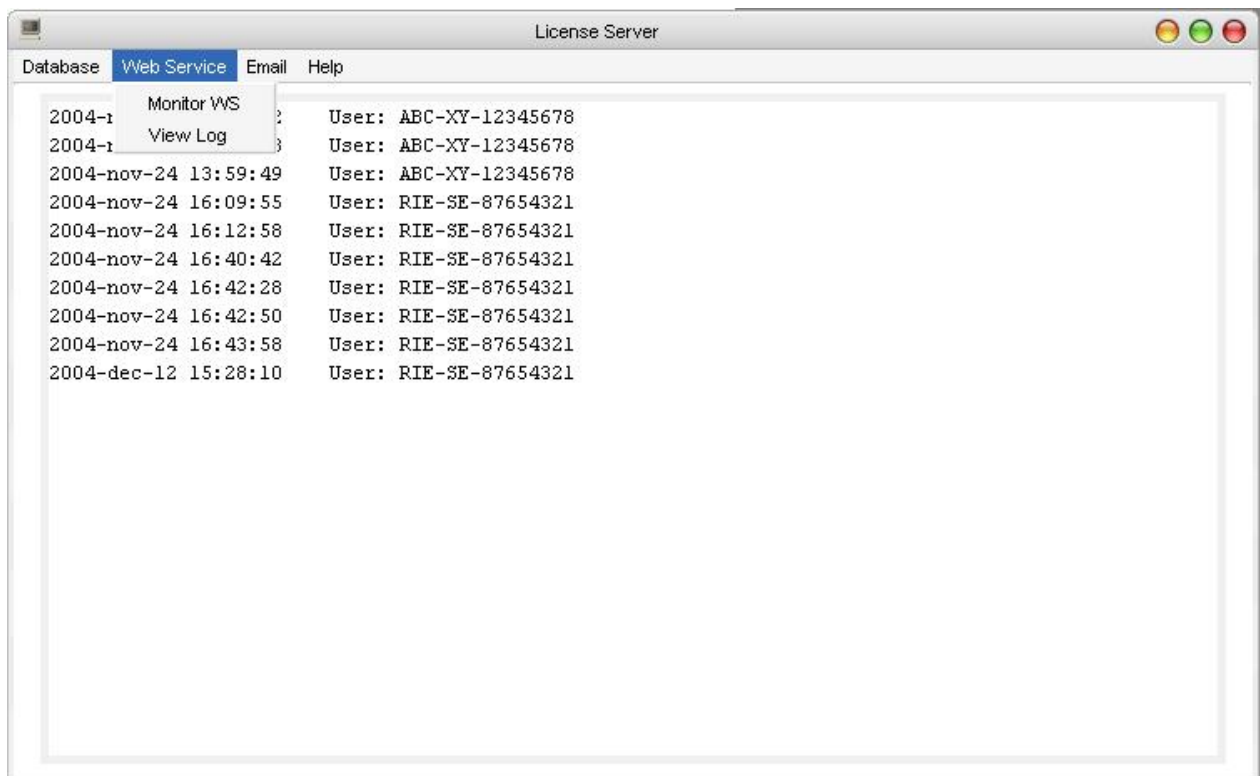


Shot 3: Delete a Customer

Shot 3 gives the interface that is shown when a delete operation is performed. The delete interface is similar to that of query. But here the user is asked to click two separate buttons, the first click will show the customer(s) that are identified for deletion and the second will delete the customer(s). The use of two buttons is an attempt to ensure that the user wants to in fact delete customer(s).

## A.4 Web Service

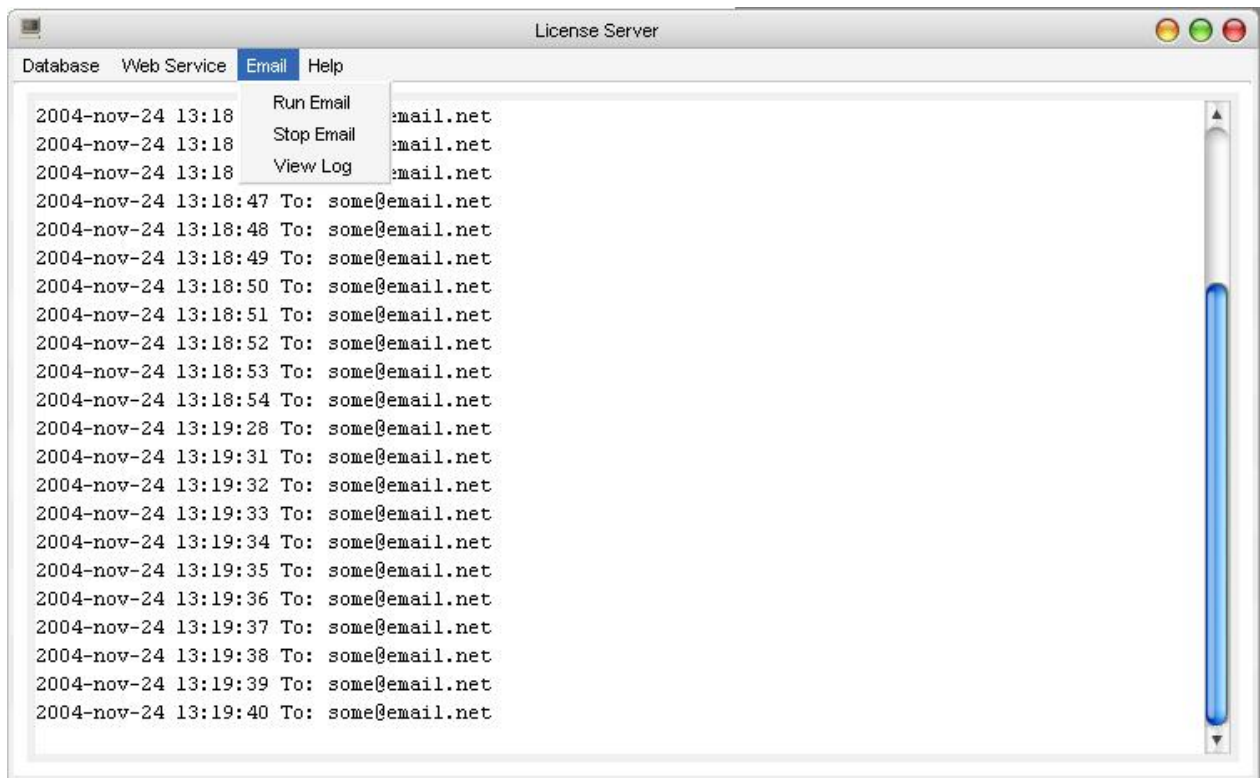
The screen shot provided by shot 4 shows the Web service functionality incorporated into the License Server GUI. One can choose to view the Web service log or monitor to on going Web service traffic.



Shot 4: Web Service Screen

## A.4 Email

The application perform an on going check of customer database to ensure that license are up to date and where a customer license is coming to an end then the application informs the customer. The menu allows the user to start and or stop an email process. It is also possible to view the email log to see a listing of all customers that was sent an email. Shot 4 shows the menu and a view of the email log.



Shot 4: Email Screen

## B Web Service WSDL

Below is a sample of a WSDL file that describes the service used for this project.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace=" urn:licenseServer" xmlns:impl="
urn:licenseServer" xmlns:intf=" urn:licenseServer"
xmlns:apachsoap="http://xml.apache.org/xml-soap"
xmlns:wsdlssoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:message name="checkLicenseRequest">
    <wsdl:part name="custId" type="xsd:string"/>
  </wsdl:message>

  <wsdl:message name="checkLicenseResponse">
    <wsdl:part name="checkLicenseReturn" type="xsd:base64Binary"/>
  </wsdl:message>

  <wsdl:portType name="LicenseServer">
    <wsdl:operation name="checkLicense" parameterOrder="custId">
      <wsdl:input name="checkLicenseRequest"
        message="impl:checkLicenseRequest"/>
      <wsdl:output name="checkLicenseResponse"
        message="impl:checkLicenseResponse"/>
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="licenseServerSoapBinding" type="impl:LicenseServer">
    <wsdlsoap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="checkLicense">
      <wsdlsoap:operation soapAction=""/>

      <wsdl:input name="checkLicenseRequest">
        <wsdlsoap:body use="encoded"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace=" urn:licenseServer"/>
      </wsdl:input>

      <wsdl:output name="checkLicenseResponse">
        <wsdlsoap:body use="encoded"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace=" urn:licenseServer"/>
      </wsdl:output>

    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="LicenseServerService">
    <wsdl:port name="licenseServer"
      binding="impl:licenseServerSoapBinding">
      <wsdlsoap:address location="
        http://localhost:8080/axis/services/licenseServer"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

## C Related Documents

The documents provided in this section are an extract from actual email contact that took place between myself and StruSoft and the university. Original emails were in Swedish and therefore a translation was performed.

### C.1 System Overview

Extract from email that served to give an overview of the system.

VIP is a series of programs that are mutually compatible and integrated for the control and analysis of energy consumption. The entire series of programs utilize the same calculation module and climatic data to guarantee identical results regardless of where calculation is performed.

**VIP+** A complete standalone application for analysing energy consumption of detached houses.

**VIP WEB** A system that is highly adaptable to accept minimum input data for analysing and calculating results for a number of houses.

**VIP\*** VIPSTAR a project between StruSoft, Skanska, Cementa and LTH-financed by SBUF and FORMAS/BIC for handling of energy calculation and inner temperature for property during the whole planning, building and management phase.

### C.2 Project Overview

Extract from email that described the project.

Quick description of VIP system:

- GUI is separate from calculating kernel.
- GUI generates as output a XML file.
- Between the GUI and the calculating kernel is a pre-processor that handles the XML file.
- Converts the output XML file into a VIP data file.
- Pre-processor then starts the calculating kernel and converts the results into a data file or html file with detailed results.
- Results are also stored in a database.

Possible areas of work for thesis:

- Communication technique between different devices and technology, example PDA and mobile devices.
- E-Commerce.
- Direct access to calculating kernel.
- Offline running of input data.
- Structural handling of XML file and menu files.

### C.3 Final Project Proposal

Extract from email the described the final proposal of the project.

StruSoft is at present developing a ground breaking and exciting product. This product will enable its customers (individuals, groups, companies, etc) to perform energy calculations and determine the energy consumption of a room, apartment, house, building or complex. There is no such product available on the market at the moment! At its heart is an Energy Calculating Module (ECMod). The system uses a Web interface that produces an XML file. A Pre-processor reads and converts the XML file to a binary file and passes it to the ECMod, which performs the necessary calculations. The Pre-processor is expected to interact with a License Server that will determine whether or not to run a request based on the validity of the customers license.

The product could and will be scattered across different sites, as the Pre-processor, Web server and other components could be at the client site BUT the License Server MUST reside at StruSoft.

The intent of this Masters thesis is to develop and implement a prototype Web service plus License Server and allow components to interact regardless of where components are physically located.

Requirements and comments

- Pre-processor MUST communicate with License Server (once per week for example).
- Communication must guarantee security.
- License Server implementation can be (1) a traditional License Server with hardware lock or (2) a database.
- Information as to who is running the Pre-processor and where, COULD be capture.
- License Server issues notification when license near expiration date.
- System must be extensible with the possibility to add component.
- The License Server should/ will be developed as part of this thesis (if database approach is chosen).


### C.4 Project Time Table

WEEK	ACTIVITY	MILESTONE	TASK
1	Background study		1
2			
3			
4		Problem formulation, components identified	
5	Configuration	Required components installed and configured.	2
6 [15/10 - 2004]	Meet the University supervisor and examiner.	Agree on design issues and the way ahead for the thesis paper.	3
	Develop packages: □ Security		4

Appendix C • Related Documents

7	<ul style="list-style-type: none"> <li>❑ Database</li> <li>❑ File handling</li> </ul>	Functional packages for handling the mentioned issues should be complete and working	
8	GUI programming		5
9		Functional prototype that with a GUI that ties the functionality of the packages amount other WS functions (developed later).	
10	Create Web service <ul style="list-style-type: none"> <li>❑ XSD</li> <li>❑ WSDL</li> <li>❑ SOAP</li> <li>❑ Other pieces</li> </ul>		6
11		Achieve the coupling of system. (Possibly adding WS function to GUI).	
12	Develop client side module - C <sup>++</sup> OR Java	The piece needs for the client should be developed and working – WS + C <sup>++</sup> OR Java.	7
13	Prototype		8
14		Fully functional prototype.	
15	Meet the University supervisor and examiner.	Agree on content of paper...	9
	Begin thesis writing		10
16			
17			
18		Draft of final paper.	
19	Update paper		
20		Final version of thesis paper.	



 <b>LINKÖPINGS UNIVERSITET</b>	<b>Avdelning, Institution</b> Division, Department	<b>Datum</b> Date 2005-02-28
	Institutionen för datavetenskap 581 83 LINKÖPING	

<b>Språk</b> Language Svenska/Swedish X Engelska/English	<b>Rapporttyp</b> Report category Licentiatavhandling Examensarbete C-uppsats D-uppsats Övrig rapport	<b>ISBN</b>	
		<b>ISRN</b> LITH-IDA-EX--05/019--SE	
		<b>Serietitel och serienummer</b> Title of series, numbering	<b>ISSN</b> _____
<b>URL för elektronisk version</b> <a href="http://www.ep.liu.se/exjobb/ida/2005/dt-d/019/">http://www.ep.liu.se/exjobb/ida/2005/dt-d/019/</a>			

<b>Titel</b> Title	Developing A Secure Web Service for License Management in StruSoft.
<b>Författare</b> Author	Dave Alfanso Russell

**Sammanfattning**  
Abstract

As software increases in complexity and relies more on Internet and Web technology, the challenge of enabling interaction and communication between loosely coupled applications becomes increasingly vital.

Distributed computing presents challenges to loosely coupled applications that require means with which to interact and communicate. There exist technologies that are aimed at solving these problems; Web service is one such technology. Web service is a relatively new and rapidly maturing technology in the area of distributed computing; it offers a standards-based way to exchange information in an interoperable manner.

This thesis is done in partnership with StruSoft and attempts to provide a solution to their problem of distributed computing, by using Web service technology. The paper looks at distributed systems and various solutions to the problems associated with distributed computing. A comprehensive insight into Web service technology is provided, along with rationale as to why it is chosen for the project. In addition, there are guidelines as to how the necessary components of Web service are installed.

Development of License Management Software is also a part of this thesis. The software offers a means with which to store and maintain data about customers and their licenses.

Security is a major focus of this paper and thus extensively mentioned throughout. A detailed explanation of computer security is presented, along with the necessary configurations that are needed to make the Web service and the License Management Software more secure.

**Nyckelord**  
Keyword

Web service, XML, SOAP, WSDL, Computer Security, Distributed Systems, Open Source, Tomcat, Axis and Software Development Life Cycle.

## På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>

## In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: <http://www.ep.liu.se/>