

Web Services: Application Development, Interoperability, Mobility and Security Issues

THESIS

Submitted in partial fulfilment
of the requirements for the degree of
DOCTOR OF PHILOSOPHY

by

SUJALA ALVA

Under the Supervision of
Prof. Dr. S. Vadivel



**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN) INDIA**

2010

Web Services: Application Development, Interoperability, Mobility and Security Issues

THESIS

Submitted in partial fulfilment
of the requirements for the degree of
DOCTOR OF PHILOSOPHY

by

SUJALA ALVA

Under the Supervision of
Prof. Dr. S. Vadivel



**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN) INDIA**

2010

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)**

CERTIFICATE

This is to certify that the thesis entitled **Web Services: Application Development, Interoperability, Mobility and Security Issues** and submitted by Sujala Alva ID No **2005PHXF005P** for award of Ph. D. Degree of the Institute embodies original work done by her under my supervision.

Signature in full of the Supervisor:

Name in capital block letters: **Prof. DR. S. VADIVEL**

Designation: Professor and Head of
Department (CS),BITS, PILANI – Dubai,
DIAC, Dubai, U.A.E.

Date: November 7, 2010

Acknowledgements:

I would like to express my sincere gratitude to Prof. Dr. R. K. Mittal, Director BITS, Pilani – Dubai for his support. I would also like to express my sincere gratitude to Prof. Dr. M. Ramachandran, Former Director BITS, Pilani - Dubai for being the motivating factor for pursuing my PhD. He was the person who initially instilled in me the importance of getting a doctorate. I would like to thank my guide Prof. Dr. S. Vadivel who has been a pillar of support right through my research work. He is always ready to share his vast resource of knowledge, provide help and guidance at every step.

I would like to express my gratitude to the members of my Doctoral Advisory Committee Dr. Santhosh Kumar, Dr. B. Vijaykumar and Dr. Ramachandran Nair for their valuable suggestions during the course of my work. I would also like to thank Dr. G. Vijaya, Dean Research and Consultancy, BITS, PILANI - Dubai and Dr. K.K. Singh in charge of Research and Consultancy activities for overall monitoring of my work.

I would like to thank the Research and Consultancy Division at BITS, PILANI, Rajasthan for monitoring my work very closely and setting deadlines and targets which have enabled me to accomplish my work. My special thanks to Dean Research and Consultancy, Prof. Ashis Kumar Das , members of the research board Mr Dinesh Kumar, Ms Monica Sharma and Mr Sharad Shrivatsava.

My special thanks to my parents who are always supportive in my every endeavor and for having provided me with the education to go ahead in life. They are my motivators. My thanks to my in-laws who have stood by me at every step. Last but not the least I would like to appreciate the help and co-operation of my husband Dr. Deepak S. Shetty and my children. I have taken up a lot of time which was rightfully theirs, but they have always stood by me and have been my strength.

Abstract:

Web Services create a platform for applications to communicate with each other across different platforms. Web services enable machine to machine interaction, this finds great relevance in today's business world where processes can be automated hence minimizing human intervention.

WEKA is a very powerful desktop data mining application. A step by step procedure has been developed to convert the popular data mining application to a web service. Tests are conducted where it is proved that there is no loss of accuracy when the application is converted to a web service with the added benefit of being accessible to a number of users. Users need not install the application on their system instead they can utilize the data mining facilities by invoking the service through the internet.

In today's world people would want to consume web services using their mobile device because of the anytime anywhere connectivity of mobile devices. In this thesis an attempt has been made to create a mobile enabled web service such that only registered users can consume the available web services and get benefited. It creates a level of abstraction for a person using this application in which he need not know how to programmatically invoke the web service. He only has to register online to the application and fill up a form which allows him to connect to the database and perform his search. Mobilink uses MySQL for the database which records and stores customer details. Form validation is done via PHP and entries like the date field where the number of days in a month are dependent on the month as well as the year (in case of a leap year) are filled in using JavaScript. Checking the username availability is done using AJAX. The web service as well as the mobile client is developed using Netbeans IDE.

Interoperability is the keyword of web services, but in spite of a large amount of research done in this field it is seen that some issues remain unresolved when web services and clients across different platforms interact. Until these issues are identified and sorted out web services may not be able to achieve wide spread acceptance. These issues have been identified and programmatically tested and proved in this thesis.

Since web services have access to secure information like credit card numbers, personal information and other sensitive information, It is important that any transaction with web services occurs in a secure manner. Secured invocation across different platforms as seen commonly in web services is a matter of great concern as compared to secured invocation across the same platform. Web service interoperability across .NET and Java is a difficult task the task gets further complicated when we consider security. An attempt has been made to develop a platform using which secured invocation of a .NET client by a Java web service can be done. The platform ensures that certificates get exchanged irrespective of the Java / .NET platform. Further the platform developed has the means to verify the security certificate exchange by logging the request response message.

Table of Contents

Acknowledgements	iv
Abstract	v
Table of Contents	vii
List of Tables	ix
List of Figures and Listings	x
List of Abbreviations	xii

Chapter 1: Introduction to the Thesis

1.1. Introduction	1
1.2. Web services and Service Oriented Architecture	1
1.3. Web services	4
1.4. M-services	7
1.5. Application developed to convert a desktop data mining application into a web service	12
1.6. Interoperability in web services	21
1.7. Security in Interoperable web services	23
1.8. Scope and Limitations of research	27
1.9. Organization of the thesis	28

Chapter 2: State of art in web services

2.1. Introduction	29
2.2. Working of web services	29
2.3. Evolution of web services	30
2.4. Introduction to Service Oriented Architecture	32
2.5. Types of web applications	36
2.5. Building blocks of web services	39
2.7. Building and developing SOA applications / Web services	54

Chapter 3: Case study of converting a desktop data mining application into a web service using Weka

3.1. Introduction	55
3.2. Overview of work done	55
3.3. Java Data Mining (JDM API)	56
3.4. Need for web service	57
3.5. Weka web service creation	58
3.6. Verification of output	59
3.7. Conclusion and future extension of work	74

Chapter 4: Comparison of J2EE and .NET development platforms used in web services and tests conducted on the commonly seen interoperability issues

4.1.	Introduction	75
4.2.	Definition of Interoperable Web Services	75
4.3.	Why interoperability?	75
4.4.	Examples of Interoperability	77
4.5.	The two common platforms used for web service development	78
4.6.	Comparative analysis between J2EE and .NET	81
4.7.	Test results of the platform developed for testing Interoperability issues	91
4.8.	Conclusion	106

Chapter 5: Mobile web services

5.1.	Introduction	108
5.2.	Evolution of Java	108
5.3.	Introduction to J2ME	109
5.4.	Mobilink	112
5.5.	Mobilink Architecture	115
5.6.	Mobilink on the Netbeans emulator	124
5.7.	Conclusion	130
5.8.	Future work	130

Chapter 6: Security issues in web services and web service security interoperability platform development

6.1.	Introduction	131
6.2.	Concept of interoperability	131
6.3.	WS-I Basic Profile	132
6.4.	.NET Framework	134
6.5.	Windows Communication Foundation(WCF)	135
6.6.	Web Service Security	137
6.7.	Work Done	147
6.8.	SOAP Messages Log	154
6.9.	Conclusion	160

Conclusions and Future Scope of work	161
Specific Contributions	164
List of References	165
Appendix A	174
Appendix B	176
Appendix C	177

Appendix D	181
Appendix E	183
Appendix F	184
List of Publications	186
Brief Biography of Candidate	187
Brief Biography of Supervisor	188

List of Tables

2.1:	Sample HTML and XML documents used to describe the same data	40
4.1:	Comparison of JAX-RPC and JAXM	87
4.2:	Table showing output of web methods of the two Java web services	96
4.3:	Compiled table	98

List of Listings

2.1:	Unencrypted XML purchase order document	38
2.2:	Encrypted XML purchase order document with only payment details encrypted	38
2.3:	Sample WSDL document for stock quote service	47
3.1:	Output obtained on running the J48 Classifier	60
3.2:	SOAP Request for J48 classifier	60
3.3:	SOAP Responses for J48 Classifier	60
3.4:	WSDL generated for J48 Classifier	62
3.5:	Output for client code	63
3.6:	Method returned on invoking the clustering algorithm	64
3.7:	SOAP Request for clustering algorithm	64
3.8:	SOAP Response for clustering algorithm	65
3.9:	WSDL generated for clustering algorithm	66
3.10:	Output for client code of clustering algorithm	67
3.11:	Output obtained for the text classifier	68
3.12:	SOAP Request for text classifier	68
3.13:	SOAP Response for text classifier	69
3.14:	WSDL generated for text classifier	72
3.15:	Output for the client code of text classifier	73

5.1:	Ajax script for checking username availability	117
6.1(a):	A Plain SOAP message without WSS	139
6.1(b):	The Plain SOAP message with WSS Encryption	140
6.2:	SetCertificateMethod()	152
6.3:	The Identity to the server certificate is provided in the app.config file	153
6.4:	SOAP Request log	154
6.5:	SOAP Response log	158

List of Figures

1.1:	Transformation of data to knowledge	13
2.1:	Example web a service application	30
2.2:	Client server application	31
2.3:	Service Oriented Architecture	34
2.4:	Interaction in web services	39
2.5:	Structure of a SOAP message	44
2.6:	Architecture of UDDI	50
2.7:	UDDI data structure	51
2.8:	W3C Web Services reference Model	53
4.1:	Architecture layers of Interoperability	76
4.2:	Output of an array with null elements when invoked by a Java client	92
4.3:	Output of an array with null elements when invoked by a .NET client	92
4.4:	Precision testing with a Java client	93
4.5:	Precision testing with a .NET client	93
4.6:	Browser window for float data type web service	95
4.7:	Browser window for double data type web service	96
4.8:	Double data type client window showing the result of float data type Web Service	99
4.9:	Double data type client window showing result of double data type Web Service	99
4.10:	Initial screen to enter student details(Java Clinet)	102
4.11:	Result displayed after successful invocation (Java client)	102
4.12:	Output showing hash map contents of student details (Java client)	102
4.13:	Initial screen for entering student details (.NET client)	103
4.14:	Result displayed after successful invocation (.NET)	104
4.15:	Output showing hash map contents of student	

details(.NET client)	104
5.1: J2ME Architecture	110
5.2: Screenshot of registration page	116
5.3: Sample database values of registered users	117
5.4 Web service creation page	118
5.5: Screenshots of data fields of web service	118
5.6: Adding data to a web service	119
5.7: Screenshot of web service with the data	119
5.8: A J2ME client invoking the web service	121
5.9: Interaction of web service stub with J2ME midlet and the web service	122
5.10: Typical JAX-RPC application	122
5.11: A wireless carrier's network ensures XML encoding of SOAP messages	123
5.12: Login screen with sample input values	124
5.13: Login successful screen	125
5.14: Web services screen with the list of available web services	126
5.15: Search field screen with searching parameter	127
5.16: Search screen with given input keyword	128
5.17: The result screen	129
6.1: Overview of Common Language Infrastructure	135
6.2: WCF service and client	137
6.3: XML security standards	138
6.4: How certificates are used	142
6.5: The Conceptual relationship between XML and web service security standards	145
6.6: Working of Symmetric key Encryption	148
6.7: Microsoft Management Console Utility	149
6.8: JKS2PFX tool	150
6.9: TCPMonitor utility	154

List of Abbreviations

AJAX	Asynchronous Java and XML
ARFF	Active Resource File Format
B2B	Business to Business
B2C	Business to Consumer
CA	Certification Authority
CDC	Connected Device Configuration
CIL	Common Intermediate Language
CLI	Command Line Interface
CLDC	Connected Limited Device Configuration
CLR	Common Language Runtime
CRL	Certificate Revocation List
CTL	Certificate Trust List
CSV	Comma Separated Vector
DOM	Document Object Model
DTD	Document Type Definition
ebXML	Enterprise Business XML
FTP	File Transfer Protocol
GPL	General Public License
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
JAXP	Java API for XML Parsing
JAXM	Java API for XML Messaging
J2EE	Java 2 Platform Enterprise Edition
J2SE	Java 2 Standard Edition
J2ME	Java 2 Micro Edition
JCA	Java Connector Architecture
JDBC	Java Data Base Connectivity

JDM	Java Data Mining
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KVM	Kilo Virtual Machine
MIDP	Mobile Information device Profile
MSMQ	Microsoft Messaging Queue
OASIS	Organization for the Advancement of Structured Information Standards
PDAP	Personal Digital Assistant Profile
PHP	hyPertext Processor
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
RPC	Remote Procedure Call
SAX	Simple API for XML
SSL	Secure Socket Layer
SOAP	Simple Object Access Protocol
SMTP	Simple Mail Transfer Protocol
TCP	Transfer Control Protocol
UBL	Uniform Resource Locator
UBR	UDDI Business Registry
UDDI	Universal Discovery Description Integration
W3C	World Wide Web Consortium
WEKA	Waikato Environment for Knowledge Analysis
WSCI	Web Service Choreography Interface
WSDL	Web Services Description Language
WS-I	Web Service Interoperability
WSIT	Web service Interoperability Technologies
WCF	Windows Communication Foundation
WSE	Web Services Enhancement
WS	Web Service
WSS	Web Service Security

XML	eXtensible Markup Language
XKMS	XML Key Management Specification
X-KRSS	XML Key Registration Service Specification
X-KISS	XML Key Information Service Specification
XSD	XML Scheme Definition

Acknowledgements:

I would like to express my sincere gratitude to Prof. Dr. R. K. Mittal, Director BITS, Pilani – Dubai for his support. I would also like to express my sincere gratitude to Prof. Dr. M. Ramachandran, Former Director BITS, Pilani - Dubai for being the motivating factor for pursuing my PhD. He was the person who initially instilled in me the importance of getting a doctorate. I would like to thank my guide Prof. Dr. S. Vadivel who has been a pillar of support right through my research work. He is always ready to share his vast resource of knowledge, provide help and guidance at every step.

I would like to express my gratitude to the members of my Doctoral Advisory Committee Dr. Santhosh Kumar, Dr. B. Vijaykumar and Dr. Ramachandran Nair for their valuable suggestions during the course of my work. I would also like to thank Dr. G. Vijaya, Dean Research and Consultancy, BITS, PILANI - Dubai and Dr. K.K. Singh in charge of Research and Consultancy activities for overall monitoring of my work.

I would like to thank the Research and Consultancy Division at BITS, PILANI, Rajasthan for monitoring my work very closely and setting deadlines and targets which have enabled me to accomplish my work. My special thanks to Dean Research and Consultancy, Prof. Ashis Kumar Das , members of the research board Mr Dinesh Kumar, Ms Monica Sharma and Mr Sharad Shrivatsava.

My special thanks to my parents who are always supportive in my every endeavor and for having provided me with the education to go ahead in life. They are my motivators. My thanks to my in-laws who have stood by me at every step. Last but not the least I would like to appreciate the help and co-operation of my husband Dr. Deepak S. Shetty and my children. I have taken up a lot of time which was rightfully theirs, but they have always stood by me and have been my strength.

Abstract:

Web Services create a platform for applications to communicate with each other across different platforms. Web services enable machine to machine interaction, this finds great relevance in today's business world where processes can be automated hence minimizing human intervention.

WEKA is a very powerful desktop data mining application. A step by step procedure has been developed to convert the popular data mining application to a web service. Tests are conducted where it is proved that there is no loss of accuracy when the application is converted to a web service with the added benefit of being accessible to a number of users. Users need not install the application on their system instead they can utilize the data mining facilities by invoking the service through the internet.

In today's world people would want to consume web services using their mobile device because of the anytime anywhere connectivity of mobile devices. In this thesis an attempt has been made to create a mobile enabled web service such that only registered users can consume the available web services and get benefited. It creates a level of abstraction for a person using this application in which he need not know how to programmatically invoke the web service. He only has to register online to the application and fill up a form which allows him to connect to the database and perform his search. Mobilink uses MySQL for the database which records and stores customer details. Form validation is done via PHP and entries like the date field where the number of days in a month are dependent on the month as well as the year (in case of a leap year) are filled in using JavaScript. Checking the username availability is done using AJAX. The web service as well as the mobile client is developed using Netbeans IDE.

Interoperability is the keyword of web services, but in spite of a large amount of research done in this field it is seen that some issues remain unresolved when web services and clients across different platforms interact. Until these issues are identified and sorted out web services may not be able to achieve wide spread acceptance. These issues have been identified and programmatically tested and proved in this thesis.

Since web services have access to secure information like credit card numbers, personal information and other sensitive information, It is important that any transaction with web services occurs in a secure manner. Secured invocation across different platforms as seen commonly in web services is a matter of great concern as compared to secured invocation across the same platform. Web service interoperability across .NET and Java is a difficult task the task gets further complicated when we consider security. An attempt has been made to develop a platform using which secured invocation of a .NET client by a Java web service can be done. The platform ensures that certificates get exchanged irrespective of the Java / .NET platform. Further the platform developed has the means to verify the security certificate exchange by logging the request response message.

Table of Contents

Acknowledgements	iv
Abstract	v
Table of Contents	vii
List of Tables	ix
List of Figures and Listings	x
List of Abbreviations	xii

Chapter 1: Introduction to the Thesis

1.1. Introduction	1
1.2. Web services and Service Oriented Architecture	1
1.3. Web services	4
1.4. M-services	7
1.5. Application developed to convert a desktop data mining application into a web service	12
1.6. Interoperability in web services	21
1.7. Security in Interoperable web services	23
1.8. Scope and Limitations of research	27
1.9. Organization of the thesis	28

Chapter 2: State of art in web services

2.1. Introduction	29
2.2. Working of web services	29
2.3. Evolution of web services	30
2.4. Introduction to Service Oriented Architecture	32
2.5. Types of web applications	36
2.5. Building blocks of web services	39
2.7. Building and developing SOA applications / Web services	54

Chapter 3: Case study of converting a desktop data mining application into a web service using Weka

3.1. Introduction	55
3.2. Overview of work done	55
3.3. Java Data Mining (JDM API)	56
3.4. Need for web service	57
3.5. Weka web service creation	58
3.6. Verification of output	59
3.7. Conclusion and future extension of work	74

Chapter 4: Comparison of J2EE and .NET development platforms used in web services and tests conducted on the commonly seen interoperability issues

4.1.	Introduction	75
4.2.	Definition of Interoperable Web Services	75
4.3.	Why interoperability?	75
4.4.	Examples of Interoperability	77
4.5.	The two common platforms used for web service development	78
4.6.	Comparative analysis between J2EE and .NET	81
4.7.	Test results of the platform developed for testing Interoperability issues	91
4.8.	Conclusion	106

Chapter 5: Mobile web services

5.1.	Introduction	108
5.2.	Evolution of Java	108
5.3.	Introduction to J2ME	109
5.4.	Mobilink	112
5.5.	Mobilink Architecture	115
5.6.	Mobilink on the Netbeans emulator	124
5.7.	Conclusion	130
5.8.	Future work	130

Chapter 6: Security issues in web services and web service security interoperability platform development

6.1.	Introduction	131
6.2.	Concept of interoperability	131
6.3.	WS-I Basic Profile	132
6.4.	.NET Framework	134
6.5.	Windows Communication Foundation(WCF)	135
6.6.	Web Service Security	137
6.7.	Work Done	147
6.8.	SOAP Messages Log	154
6.9.	Conclusion	160

Conclusions and Future Scope of work	161
Specific Contributions	164
List of References	165
Appendix A	174
Appendix B	176
Appendix C	177

Appendix D	181
Appendix E	183
Appendix F	184
List of Publications	186
Brief Biography of Candidate	187
Brief Biography of Supervisor	188

List of Tables

2.1:	Sample HTML and XML documents used to describe the same data	40
4.1:	Comparison of JAX-RPC and JAXM	87
4.2:	Table showing output of web methods of the two Java web services	96
4.3:	Compiled table	98

List of Listings

2.1:	Unencrypted XML purchase order document	38
2.2:	Encrypted XML purchase order document with only payment details encrypted	38
2.3:	Sample WSDL document for stock quote service	47
3.1:	Output obtained on running the J48 Classifier	60
3.2:	SOAP Request for J48 classifier	60
3.3:	SOAP Responses for J48 Classifier	60
3.4:	WSDL generated for J48 Classifier	62
3.5:	Output for client code	63
3.6:	Method returned on invoking the clustering algorithm	64
3.7:	SOAP Request for clustering algorithm	64
3.8:	SOAP Response for clustering algorithm	65
3.9:	WSDL generated for clustering algorithm	66
3.10:	Output for client code of clustering algorithm	67
3.11:	Output obtained for the text classifier	68
3.12:	SOAP Request for text classifier	68
3.13:	SOAP Response for text classifier	69
3.14:	WSDL generated for text classifier	72
3.15:	Output for the client code of text classifier	73

5.1:	Ajax script for checking username availability	117
6.1(a):	A Plain SOAP message without WSS	139
6.1(b):	The Plain SOAP message with WSS Encryption	140
6.2:	SetCertificateMethod()	152
6.3:	The Identity to the server certificate is provided in the app.config file	153
6.4:	SOAP Request log	154
6.5:	SOAP Response log	158

List of Figures

1.1:	Transformation of data to knowledge	13
2.1:	Example web a service application	30
2.2:	Client server application	31
2.3:	Service Oriented Architecture	34
2.4:	Interaction in web services	39
2.5:	Structure of a SOAP message	44
2.6:	Architecture of UDDI	50
2.7:	UDDI data structure	51
2.8:	W3C Web Services reference Model	53
4.1:	Architecture layers of Interoperability	76
4.2:	Output of an array with null elements when invoked by a Java client	92
4.3:	Output of an array with null elements when invoked by a .NET client	92
4.4:	Precision testing with a Java client	93
4.5:	Precision testing with a .NET client	93
4.6:	Browser window for float data type web service	95
4.7:	Browser window for double data type web service	96
4.8:	Double data type client window showing the result of float data type Web Service	99
4.9:	Double data type client window showing result of double data type Web Service	99
4.10:	Initial screen to enter student details(Java Clinet)	102
4.11:	Result displayed after successful invocation (Java client)	102
4.12:	Output showing hash map contents of student details (Java client)	102
4.13:	Initial screen for entering student details (.NET client)	103
4.14:	Result displayed after successful invocation (.NET)	104
4.15:	Output showing hash map contents of student	

details(.NET client)	104
5.1: J2ME Architecture	110
5.2: Screenshot of registration page	116
5.3: Sample database values of registered users	117
5.4 Web service creation page	118
5.5: Screenshots of data fields of web service	118
5.6: Adding data to a web service	119
5.7: Screenshot of web service with the data	119
5.8: A J2ME client invoking the web service	121
5.9: Interaction of web service stub with J2ME midlet and the web service	122
5.10: Typical JAX-RPC application	122
5.11: A wireless carrier's network ensures XML encoding of SOAP messages	123
5.12: Login screen with sample input values	124
5.13: Login successful screen	125
5.14: Web services screen with the list of available web services	126
5.15: Search field screen with searching parameter	127
5.16: Search screen with given input keyword	128
5.17: The result screen	129
6.1: Overview of Common Language Infrastructure	135
6.2: WCF service and client	137
6.3: XML security standards	138
6.4: How certificates are used	142
6.5: The Conceptual relationship between XML and web service security standards	145
6.6: Working of Symmetric key Encryption	148
6.7: Microsoft Management Console Utility	149
6.8: JKS2PFX tool	150
6.9: TCPMonitor utility	154

List of Abbreviations

AJAX	Asynchronous Java and XML
ARFF	Active Resource File Format
B2B	Business to Business
B2C	Business to Consumer
CA	Certification Authority
CDC	Connected Device Configuration
CIL	Common Intermediate Language
CLI	Command Line Interface
CLDC	Connected Limited Device Configuration
CLR	Common Language Runtime
CRL	Certificate Revocation List
CTL	Certificate Trust List
CSV	Comma Separated Vector
DOM	Document Object Model
DTD	Document Type Definition
ebXML	Enterprise Business XML
FTP	File Transfer Protocol
GPL	General Public License
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
JAXP	Java API for XML Parsing
JAXM	Java API for XML Messaging
J2EE	Java 2 Platform Enterprise Edition
J2SE	Java 2 Standard Edition
J2ME	Java 2 Micro Edition
JCA	Java Connector Architecture
JDBC	Java Data Base Connectivity

JDM	Java Data Mining
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KVM	Kilo Virtual Machine
MIDP	Mobile Information device Profile
MSMQ	Microsoft Messaging Queue
OASIS	Organization for the Advancement of Structured Information Standards
PDAP	Personal Digital Assistant Profile
PHP	hyPertext Processor
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
RPC	Remote Procedure Call
SAX	Simple API for XML
SSL	Secure Socket Layer
SOAP	Simple Object Access Protocol
SMTP	Simple Mail Transfer Protocol
TCP	Transfer Control Protocol
UBL	Uniform Resource Locator
UBR	UDDI Business Registry
UDDI	Universal Discovery Description Integration
W3C	World Wide Web Consortium
WEKA	Waikato Environment for Knowledge Analysis
WSCI	Web Service Choreography Interface
WSDL	Web Services Description Language
WS-I	Web Service Interoperability
WSIT	Web service Interoperability Technologies
WCF	Windows Communication Foundation
WSE	Web Services Enhancement
WS	Web Service
WSS	Web Service Security

XML	eXtensible Markup Language
XKMS	XML Key Management Specification
X-KRSS	XML Key Registration Service Specification
X-KISS	XML Key Information Service Specification
XSD	XML Scheme Definition

Chapter 1: Introduction to the Thesis

1.1 Introduction

This chapter gives an overview of service oriented architecture and web services. The different applications where web services are used in various walks of day to day life is looked into. A brief introduction is given to the research carried out in the following areas.

- i) Applying web service technology to a desktop data mining application.
- ii) Interoperability issues seen in web services across Java and .Net platforms.
- iii) Mobile enabled publishing of web service, locating the web service and invocation of the web service.
- iv) Web service security issues.

1.2 Web Service and Service Oriented Architecture

In the current scenario, where there is large scale globalization, having a standalone application will not suffice. In order to build a large application it is very important to integrate a number of systems which could be distributed all over the world. The software industry is coming to terms with the fact that integrating software applications across multiple operating systems, programming languages, and hardware platforms is not something that can be solved by any one proprietary environment. Traditionally the problem has been one of tight coupling, where one application that calls a remote network is tied strongly to it by the function call it makes and the parameters it requests. In most systems before web services, this is a fixed interface with very little flexibility or adaptability to the changing needs of the environment [1].

Web services and Service Oriented Architecture (SOA) have changed the definition of distributed computing they provide a new paradigm for building distributed computing applications. Web services have an advantage over conventional distributed computing platforms such as Remote Procedure Calls

(RPC), Remote Method Invocation(RMI), Distributed Component Object Model(DCOM), Common Object Request Broker Architecture (CORBA) etc which were bound to a particular platform, in contrast web services have a loosely coupled architecture, combined with their standardized interoperability led to a new computing paradigm that supports the construction of more flexible and dynamic distributed applications [2]. Traditional means of building distributed computing applications have been dominated by various technologies which were popularly called “Middleware Technologies” which were basically vendor dependent and they were typically implemented using the client server technology. Some of the common technologies are:

- i) RMI – uses Java to create distributed applications
- ii) DCOM – Is a ActiveX software component technology which was developed by Microsoft Corporation
- iii) CORBA – Managed by the Object Management Group (OMG) which is a consortium of over 800 companies [3].

A SOA is designed to help developers overcome many distributed enterprise computing challenges including application integration, transaction management and security policies. While allowing multiple platforms and protocols and leveraging numerous access devices and legacy systems. The driving goal of SOA is to eliminate these barriers so that applications integrate and run seamlessly.

A service in SOA is an exposed piece of functionality with three essential properties. Firstly a SOA based service is self contained i.e. the service maintains its own state. Secondly services are platform independent assertions. Lastly, the SOA assumes that services can be dynamically located, invoked and recombined.

Thus with SOA, an enterprise can create, deploy and integrate multiple services and choreograph new business functions by combining new and existing application assets into a logical flow.

The use of SOA is gradually capturing the market because SOA has several advantages such as it reuses the existing resources and executes tasks by composing services at runtime based on specific parameters. Another major advantage of SOA is negotiation transparency, applications can delay binding services until deployment or execution. SOA is also cost effective. The requestor can pay for services on the basis of use rather than having access to the application. An attractive feature of web services is that if a service is upgraded, it needs to be updated only on the server, no changes need to be done on the client code. Most importantly SOA provides means for service interoperability in networks where heterogeneity is a major obstacle; the architecture allows developers to construct new services on demand through dynamic service construction at run time. Overall SOA is an extremely flexible and extensible architecture. As testimony to the appeal of SOA, it is used widely in government organizations like the Canadian government which uses SOA to facilitate management growth of large scale applications as quoted by [4] and by the European Union for standardizing their online documentation formalities and also by the German government in their postal application as explained by [5]. In essence SOA = Semantic integration + Loose coupling + Managed Evolution. Semantic integration is the major prerequisite and challenge, loose coupling is the distinct feature of SOA, and managed evolution represents both a purpose and an implementation approach [5].

Businesses would generally involve a series of tasks involving different systems and if the SOA approach can automate this process and make it more efficient and faster it is a technological boon to the business world and it can be seen that SOA is very relevant to today's business world. Service-Oriented Architecture (SOA) is an IT architectural style that supports the transformation of the business

into a set of linked services, or repeatable business tasks that can be accessed when needed over a network. This may be a local network, it may be the Internet, or it may be geographically and technologically diverse, combining services in New York, London, and Hong Kong as though they were all installed on a local desktop. These services can coalesce to accomplish a specific business task, enabling the business to quickly adapt to changing conditions and requirements. When SOA implementation is guided by strategic business goals, it ensures the positive transformation of the business and can realize the chief benefits on an SOA, as follows:

- i) Alignment of IT with the business
- ii) Maximal reuse of IT assets

Together, these help assure that investment in expensive IT projects result in lasting value to the business [1].

When creating such business processes it is important to remember an important business process called choreography which is useful for complex automation services. Choreography provides a set of rules that explain how different components can act together and in what sequence, giving a flexible, systematic view of the process. Using a travel package analogy, choreography allows reservations to recognize that airfare must be booked first and then hotel reservations can be made [6].

The Universal Business Language (UBL) is an initiative to develop common business document schemas for interoperability. However businesses operate in different industry, geopolitical and regulatory contexts and have different rules and regulations for the information they exchange. Hence several trading communities are tailoring the UBL schemas to their needs, requiring that these schemas translate to each other [5].

1.3 Web Services

Web services are an instance of SOA. Web services can be explained briefly as a framework of software technologies designed to support interoperable machine to machine interaction over the network [6]. A web service is a software interface that describes a collection of operations that can be accessed over the network through standardized XML messaging. It uses protocols based on the XML language to describe an operation or to exchange data with another web service. A group of web services integrating together in this manner defines a particular web service application in a SOA. The success of web services is mainly attributed to its use of the highly versatile XML, XML allows the separation of grammatical structure (syntax) and the grammatical meaning (semantics), and how data is processed and understood by each service and the environment it exists in.

Web services allow the following

- i) Interaction between services written on any platform and on any language.
- ii) Allows for loose coupling, in which service applications may not break if there is a change in the way in which one or more services are designed or implemented.
- iii) Conceptualize application function into tasks. This allows higher abstraction of software so that even less technical people can use it for business analysis.
- iv) Adapt existing applications or legacy applications to changing business and customer needs.
- v) Legacy applications can be interfaced with other applications without changing the original application.
- vi) Provide other administrative or management functions like reliability, accountability and security [1].

Web services can be used in a host of applications from weather forecast to ticket booking. Some applications where they are used very effectively as seen in the literature survey is discussed below.

Online labs with web services

In the present day distance learning programs are becoming very popular with students, an area which is slowly growing in popularity is online laboratories. To clearly understand the concepts learned in a course labs would have to be used but many institutions may not be able to afford setting up multiple labs, in such cases online labs could be accessed, or in cases of distance learning these online labs would have to be used. These online labs could be of any one of these categories

- i) Virtual laboratories which provide a simulation environment in which students conduct experiments.
- ii) Remote laboratories – students use a GUI to operate actual instruments via remote control.

The difficulty of creating an effective laboratory using remote control is to use scattered computational resources and instruments across platforms. Normally to ensure interoperability, systems use equipment from the same company like Agilent or National Instruments. They even use the same operating system like Microsoft. Users must then install additional software to use these devices using remote control, thereby involving a lot of constraints on users. One solution to these problems is to base online experiment systems on web services, which can support interoperable machine to machine interaction over the network and can also integrate heterogeneous resources [7].

Integrated automobile design

A scenario is considered as discussed by [8] where an automaker in Germany may be required to cooperatively design a new automobile with an automaker from Japan, however their software components that remote clients invoke for designing the automobile may be in different technologies, such as Java 2

Enterprise Edition (J2EE), .Net or other distributed object technologies. Accessing such different components with a unified method is difficult for a remote client. Web services are used in such applications because they can implement loosely coupled web applications across platforms and program languages. Moreover developers can use some tools to transform most existing components to web services. A framework using AJAX and web services can be used for cooperative image editing [9].

Distributed Healthcare System

A distributed health care system built on SOA and web services can be used to integrate a number of entities like physicians, nurses, pharmacists, patients as well as medical devices used to monitor patients. Multimedia input and output with text, images and speech makes systems less computer like and easily accessible to non computer savvy people. Using this system a patient can book an appointment either on their mobile or the desktop with a particular doctor, the nurse can check this appointment list, the doctor can prescribe medicines which are sent directly to the pharmacist to avoid tampering of the medicine list or to avoid misinterpretation of the list due to illegible handwriting. Medical equipment like blood pressure monitoring devices and glucose monitoring devices can also be connected on this framework. Web services are the most ideal platform to connect these devices because of its ability to connect devices on different platforms and implemented on different programming languages [10].

Hence considering the relevance of web services in today's world and the wide range of applications where web services can be used. This thesis considers web services to be a technological innovation of this century in terms of distributed computing. This research work is based on this very relevant and highly evolving field. Since it is a new area there is scope for a lot of research and innovation in this field.

Along with all the benefits of web services there are a few downsides also. One of the major factors could be the different standardizing agencies like W3C, OASIS, Liberty Alliance Project and the Web Services Interoperability (WS-I) Organization. Different companies are involved with web service technologies like IBM, Microsoft, Sun etc. The uncoordinated web services standards process has resulted in some companies “predeveloping” a standard and then turning it over to a standards organization. This is a good business move because it helps to create a new mass of applications which interoperate with their own applications and tools. Companies like IBM and Microsoft believe in developing their products in a closed process and then turning it over to a standard body. Sun is working hard on developing applications in a more open environment [1].

1.4 M-services

The significant advances in wireless technologies open a promising application area for Web services. As a key extension of Web services, *mobile services* (M-services) cater to the increasing population of mobile users. Specifically, M-services are a special set of web services that can be accessible by mobile hosts over wireless networks [11, 12, and 13]. They work together with mobile devices to offer *anytime/anywhere* accessible services. In contrast to Web services with wired infrastructures, M-services are more suitable for time and location critical tasks. For instance, a stock quote service can help users make quick response by providing timely quote prices. However, users may prefer desktops to cell phones or Personal Digital Assistants (PDA's) when carefully preparing a travel package for vacations. The mobile environment poses great challenges for providing and consuming M-services. Mobile devices have low CPU and memory capacities, limited power supply, small screen size, and restricted input mechanisms. Wireless networks are limited by their small bandwidth. They also suffer from link outages, which result in temporary unavailability. These limitations hinder existing web service technologies from directly working with M-services. For instance, the limited bandwidth may not be enough to convey SOAP messages [11]. In addition, SOAP is expensive for mobile hosts in terms

of both power consumption and waiting time, they impose more overhead because of the SOAP requests and responses, and parsing XML code also adds extra computing costs. An option which is available is compression and decompression of the SOAP messages, though this process can add an overhead it is still effective since it reduces CPU computations [14].

In mobile environment, users' context, such as location and activity, may change rapidly. M-services need to track these changes and provide *context-aware* functionalities. However, service description techniques, such as Web Service Description Language (WSDL), have not provided support to model context. UDDI enables service discovery in the web service framework. However, the multiple costly round trips required by UDDI lookup are troublesome form of service discovery [11]. The frequent unavailability of wireless network may cause failures in service discovery processes.

But in spite of the shortcomings the landscape of software development is changing rapidly, a decade ago it was sufficient to develop a desktop version of a product to reach a majority of customers and to meet their expectations but now most applications are developed to be compliant with mobile devices. Though like the earlier web based applications, mobile services have technical and physical limitations they have the advantage of location independence whereas existing PC based systems can offer at best a nomadic context.

The evolution of mobile devices is so significant now that given an option of a stationary option or a mobile option, users would any day prefer the mobile option especially in applications like telephone conversations, taking digital pictures, listening to music, reading, e-mail, managing personal information, banking and a wide range of other consumer and employee activities [15].

The current challenge of software companies is to provide a uniform and integrated user experience across desktop, web and mobile applications. The

natural architectural driver is to maximize that part of the software system common to all three platforms while minimizing the platform specific part. In addition the interface between the generic and platform specific parts should be as simple and stable as possible, SOA offer an excellent technical solution for achieving this architectural goal.

However when mobile devices are considered, the following challenges arise.

- i) Different mobiles offer various subsets of mechanical, hardware and software capabilities.
- ii) Each device is available in the market for a very short period generally months not even years
- iii) Any software designed for mobile devices must be tested and validated on each device since it can affect both user experience and the software's ability to execute properly
- iv) Any application that is developed must be forward compatible so that users can use this application even if they upgrade to a new mobile device.

Since customers demand and expect a uniform integration of services and products, SOA is bound to become the platform of choice for an increased set of everyday tasks [5].

As the mobile internet grows, the number and types of services will increase, researchers should focus on empowering service users, programmers and non programmers alike to create and share mobile services, this will enable the kind of bottom-up creativity on the mobile internet that has served the conventional internet so well [16].

1.4.1 Work Done

In Chapter 5 of the thesis an application has been developed which can be accessed by mobile devices which is called Mobilink. The application creates a

web service which enables a data search to be performed on the back end data base, the interesting aspect of Mobilink is that it assumes that the user of this application would like to tailor make a web service to perform a search as per the desired specifications but the user may not be tech savvy and may not be able to code this application. Mobilink is designed with such people in mind who might have a specific demand but may be lacking in technical knowledge to develop this application. Mobilink provides a front end GUI to specify the parameters of the search, the web service which is created would then perform the search accordingly in a fixed set of databases. Mobilink provides a mobile web service client which allows the mobile device to be used for performing a search in the database. This application has tremendous potential, it can be extended to search for image files, video files, audio files or it can be interfaced with context aware GPS systems to give specific information to users.

1.4.2 Related Work from literature survey

It is very important that mobile service creation should not be limited to professional computer programmers, who constitute a fraction of the world's population, rather anyone who is able to browse the internet or perform simple jobs online like online banking should be able to create mobile services, an easy way to accomplish this is to create "service templates" that nonprogrammers can fill in to configure and create services. Authors of [16] have developed a prototype system called Streamspin available at: (<http://streamspin.com>). This prototype allows users to receive content from a range of services through a single interface. Access to the system is both through the Streamspin website as well as a mobile client. The users can register to services through the website and they can receive content from the system like photos, video and data on their mobile. Users can create services simply by filling up an online form , these services once created can be put up to be shared with friends or with the general public, for e.g. a person can create the desired travel plan which could be shared with friends. Streamspin offers a level of abstraction to the user by hiding all the implementation details. This system allows access to geospatial and

social networks. But this system is developed as a web application and not a web service, unlike the application Mobilink which is developed as a web service.

During the course of literature survey it was observed that most applications designed for mobile services were used to provide either directions, video files or text data like news, stock prices etc. The great advancements in mobile services have opened the possibility of mobile web based services that tailor content to user preferences, user locations, and device capabilities. As explained by [17] where an application is created which stores the user's personal data like choice of food, the direction of travel etc. When a search for a restaurant is performed, the database storing the preferences is searched and the direction of travel is obtained from the GPS system and the person is directed to the nearest restaurant serving his favorite type of food.

The common fields in which mobile web services are used widely could be categorized as

- i) News: Could include information portals such as online news papers or news broadcasting sites.
- ii) Social: Could be in the form of blogs, online gaming, YouTube etc.
- iii) Travel: Online reservations like Expedia or tourist and travel information.

Online games and Youtube videos require heavy graphics. Processing these graphics intensive applications on the fly is processor intensive. Hence it is better to pregenerate this data and cache it for quick future reference as explained in [18].

In [19] the authors have developed an application for a vehicle rental service for mobile users, this service allows the user to make a vehicle reservation and also to make the payment through a mobile device. Response time analysis was done using General Packet Radio Service (GPRS) and Wi-Fi networks separately.

Mobilink fills the research gap by providing a user friendly GUI which gives the flexibility of creating the web service to the client as per the requirements. It also creates a mobile web service client which allows access to this created web service from a mobile phone.

1.5 Application developed to convert a desktop data mining application into a web service

1.5.1 Overview of Data Mining

One thing that has caught everyone's attention in the age of data explosion is the huge volume of data that surrounds us.

- i) International retailers like Carrefour, Wal-Mart etc. have huge volumes of data
- ii) Organizations performing research on human cells may store tons of data on DNA, protein sequence and gene expression data
- iii) Sites like Google, Yahoo, and YouTube etc may have to store terabytes of data.

What is Data? Data can be any fact, number or text which can be processed by the computer. These include:

- i) Transactional or operational data like sales, payroll, accounting and cost.
- ii) Non-operational data like forecast data, industry sales and macro economic data.
- iii) Meta data which is the data about itself such as data dictionary definitions.

It becomes pointless having this huge volume of data if there are no efficient data mining tools to extract patterns in this data. Data can be any fact, number, or text which can be processed by the computer. Data mining uses sophisticated algorithms to extract useful data and identify trends and relationships in data that are beyond simple analysis. In data mining, there are several fundamental functions and associated algorithms. These mining functions include classification, regression, clustering and association and form a core supporting many common data mining solutions [62].

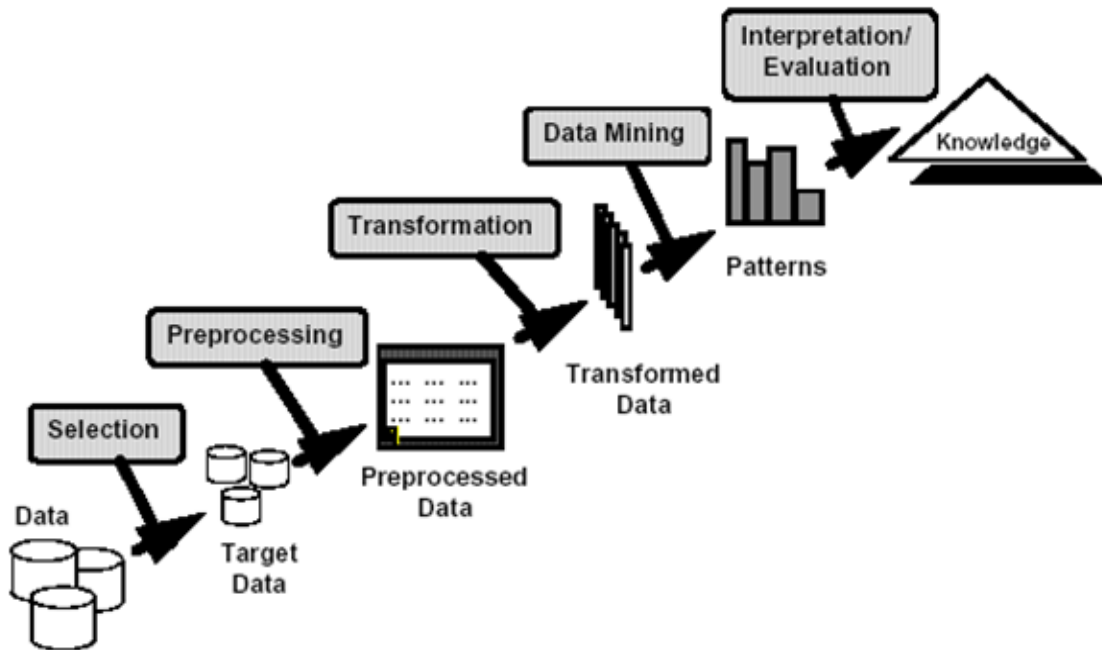


Figure 1.1 Transformation of data to knowledge

Data Mining has evolved from areas like statistics, machine learning, pattern recognition, databases and high performance computing [61]. As represented in Figure 1.1 [110] the process of converting data to knowledge is called Knowledge Discovery in Databases (KDD). KDD is an iterative process which involves data selection, preprocessing the data, applying algorithms on the data and finally viewing or analyzing the patterns in the data [23].

The technology of data mining is not new. It finds wide use in various domains like-

Science and engineering:

- i) Educational research
- ii) Changes in DNA sequence for detecting diseases [66].
- iii) Condition monitoring of high voltage electrical equipments.
- iv) Dissolved Gas Analysis (DGA) on power transformers.

Business:

- i) Customer Relationship Management (CRM) to find prospects with high likelihood of responding.
- ii) Market Basket Analysis for retail sales [70]

Mail Mining:

- i) Parse and store e-mails by senders/subject.
- ii) Separate spam mail

Building a web portal (like MyYahoo):

- i) Collect data from more than one source page and present it as a single page by adding some value to the data.

Building a search engine (e.g. Google):

- i) Data mining helps Web search engines find high quality web pages and enhances web click stream analysis.
- ii) Web search engine crawls the web, indexes web pages and builds and stores huge keyword based indices that help to locate sets of web pages to a query.
- iii) Return relevant pages to the query [70].

Personalized B2C E-Commerce (e.g. Amazon.com):

- i) A host of web mining techniques, e.g. associations between pages visited, click-path analysis, wish lists, instant recommendations etc., are used to improve the customer's experience during a 'store visit'
- ii) Also with the help of data mining techniques, the various patterns of customer web-usage can be found.
- iii) Depending on these usage patterns, users can be classified and categorized and accordingly promotions and discounts can be sent to the appropriate user groups [70].

1.5.2 Work Done

In chapter 3 a standalone desktop data mining application is converted into a web service using the Weka software and tests are performed to see if the data mining output obtained for three data mining operations which are clustering, classification and text classification are the same when used as a standalone

desk top application and as a web service. The results obtained showed that web services give the same output as a standalone application thus proving that a standalone application can be converted to a web service with no loss of precision but with the added advantage of having the application invoked by any number of users who would require data mining results. This is the whole idea of SOA to have code reuse and the usage of applications as services which can be invoked over the network by other applications. Another major advantage of converting the application to a web service is that the data mining application need not be loaded on the client's system, this service can be invoked as a web service thus saving on precious memory space on the system. A added benefit would be any upgradation of the service will involve only the server side code without affecting the client program.

1.5.3 Literature survey of the algorithms used in the data mining application

The three algorithms considered in the data mining application are

- i) J48 Classifier
- ii) EM Classification
- iii) IBk Lazy algorithm for text classification

1.5.3.1 J48 Classification

The notion of classification is to classify cases according to a fixed set of categories. In simple words, classification is a machine learning (data mining) technique to predict group membership of instances. Decision trees represent a supervised approach to classification. A decision tree is a simple structure where non-terminal nodes represent tests on one or more attributes and terminal nodes reflect decision outcomes [67]. Classification can either be binary where there are only two categories like will a person vote? Yes/ No, or classification can be of multiclass like the different categories of income. Classification models may use probability like the probability that people vote is 80%. Tools that support

classification in JDM include decision trees, Naïve Bayes, support vector machines and feed forward neural networks [72].

Regression is used for classifying continuous values like value of a property, congestion level in the atmosphere etc, algorithms that support regression includes neural networks, decision trees and vector machines.

The J48 Decision tree classifier follows a simple algorithm. In order to classify a new item, it first needs to create a decision tree based on the attribute values of the available training data. So, whenever it encounters a set of items (training set) it identifies the attribute that discriminates the various instances most clearly [72].

The general approach can be summarized as given below:

- i) Choose an attribute that best differentiates the output attribute values.
- ii) Create a separate tree branch for each value of the chosen attribute.
- iii) Divide the instances into subgroups so as to reflect the attribute values of the chosen node.
- iv) For each subgroup, terminate the attribute selection process if:
 - a. All members of a subgroup have the same value for the output attribute,
terminate the attribute selection process for the current path and label the
branch on the current path with the specified value.
 - b. The subgroup contains a single node or no further distinguishing attributes
can be determined. As in (a), label the branch with the output value
seen
by the majority of remaining instances.
- v) For each subgroup created in iii) that has not been labeled as terminal, repeat the above process.

1.5.3.2 EM Clusterer Working

Clustering works by grouping data or documents on its similarity and not on previous knowledge, Weka provides a number of clustering tools [58]. Clustering analysis identifies clusters that exist in a given dataset, where a cluster is a collection of cases that are more similar to one another than cases in other clusters. A set of clusters is considered to be of high quality if the similarity between clusters is low, yet the similarity of cases within a cluster is high [73].

The Expectation-Maximization (EM) algorithm is part of the Weka clustering package. The EM algorithm assumes all attributes to be independent random variables. The parameters are re-computed until a desired convergence value is achieved [64].

In the simplest case, the probability distributions are assumed to be normal and data instances consist of a single real-valued attribute. Using the scenario, the job of the algorithm is to determine the value of five parameters, specifically:

- i) The mean and standard deviation for cluster 1
- ii) The mean and standard deviation for cluster 2
- iii) The sampling probability for cluster 1 (or the probability for cluster 2)

The general procedure is :

- i) Guess initial values for the five parameters.
- ii) Use the probability density function for a normal distribution to compute the cluster probability for each instance. In the two-cluster case, the two probability distribution formulas each having differing mean and standard deviation values.
- iii) Use the probability scores to re-estimate the five parameters. [68]

1.5.3.3 IBK Lazy

Data is of two types either it is unstructured i.e. created by humans for humans like books, e-mails, web pages etc or structured form for e.g. a database or an XML document which is meant for the computer. Text mining is a specialized application of data mining where it extracts useful information from unstructured

text or it uses it in the unstructured form by what is called a bag of words model [58].

Text mining can be a very complex task, the tools available should provide a layer of abstraction and they should keep the user unaware of how text mining actually works.

Generally text mining could involve the following categories

- i) Language Identification – Language profiles can be compared using n-gram frequencies, n-gram is a chunk of continuous characters from a single word for e.g. the word hello can have six grams –he, hel, ell, llo, lo_, o_ _
- ii) Clustering – Using K-nearest neighbor the centroid of the cluster of words is found and it is decided into which cluster a word belongs.
- iii) Similarity – Different functions like Hamming, Manhattan etc can be applied to look for similarity.

Plagiarism detection is an application of text mining [58].

Generally to identify spam the following is done

- i) Source Analysis – Looks at the identities of object contributors
- ii) Text Analysis – Looks for words or phrases which could be categorized as spam.
- iii) Link or behavior analysis – looks at networks of objects or users.

Many sites like Wikipedia manually train spam filters to identify spam mail. Text analysis in e-mail can generally happen with Bayesian Networks which are trained to identify spam mail. There are three actions which can be taken when spam is identified.

- i) the spam mail is deleted
- ii) the mail is given but with a warning

- iii) the link with spam would have a lower rating this is generally effective in a search engine portal like Google which would rank pages with spam lower in its list [59].

The given code for training and testing a Text Classifier can be implemented with two algorithms – IBk Lazy and Naïve Bayes. This code can be extensively used for the purpose of mail mining and depending on the content of the mail it can be classified as SPAM or NO SPAM.

The IBK Lazy is Weka's implementation of the K – Nearest Neighbor Classification algorithm [67]. The k-nearest neighbor's algorithm (k-NN) is a method for classifying objects based on closest training examples in the feature space. It is one of the simplest algorithms that support a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors, k is a positive integer, typically small. If $k = 1$, then the object is simply assigned to the class of its nearest neighbor. In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes. The neighbors are taken from a set of objects for which the correct classification (or, in the case of regression, the value of the property) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required [74].

1.5.4 Related work from literature survey

Various works has been carried out where Weka is converted to a distributed data mining application on the grid environment. Weka4WS is a framework where Web Services Resource Framework (WSRF) is used for accessing remote data mining algorithms and managing distributed computations. On every computing node, a WSRF compliant web service is used to expose all the data mining algorithms as a web service. The authors of [78] use the Globus Toolkit to

develop the framework; Globus Toolkit is a middleware system which runs on top of existing operating systems and networks it provides a Java WSRF library on the Unix platform.

A very similar application to Weka4WS is discussed in [81] where a web services based toolkit for providing distributed data mining is provided. A workflow engine is provided within the toolkit to enable a user to compose web services to implement particular point solutions. Three types of web services are provided: classifiers, clustering algorithms and association rules.

Another framework developed along the similar lines is Weka4GML as described by [89] where the WSRF technology is used for developing meta learning methods to deal with data sets distributed among data grids.

In all the above three cases it is seen that Weka is converted to a distributed data mining application on the grid and offered as a web service, the Globus Toolkit framework is used in all the above cases. Another disadvantage is that the use of the Globus toolkit makes all the three applications dependent on the Unix environment. Managing the grid environment is a complex task and it would require a workflow engine to distribute the tasks and to compute the result. On the other hand the application developed and discussed in the thesis reduces the scale of complexity and offers the service by performing the data mining application at one node. This application will be simple to implement on any system and it provides the data mining requirement for a small application by eliminating the complexity of grid computing.

Related work in this field was conducted by [20] where they have used web services for integrating incompatible applications in bioinformatics, an area which is becoming increasingly critical to meaningful biological research. Web services could be used in any research field that requires analyzing volumes of data and conducting complex data mining. While analyzing the research gap it was

identified that most applications use data mining techniques to mine data like in [21] where a database to simulate the world wide web has been created and this data is mined using data mining tools created as web services to perform operations like clustering, association rules etc.

A very interesting web service mining framework has been developed in [22] which allow unexpected and interesting service components to automatically emerge in a bottom up fashion. Different mining techniques aiming at discovery of such service compositions are used, an evaluation measure of the usefulness and interestingness of this model is also made. A case study has been designed to work on biological processes and to study the discovery of interesting and useful patterns.

A scalable, extensible, and easy to use data mining application that relies on web services to achieve extensibility and interoperability has been developed in [23]; Ant eater provides simple abstractions for users, and supports computationally intensive processing on large amounts of data through massive parallelism.

Weka is a very powerful desktop data mining tool which is used for numerous data mining applications, in this thesis Weka is converted into a web service and tests are conducted on the output generated. The tests performed compares the output of a desktop application to that of a web service, hence this research fills the gap of testing and proving that any application can be converted to a web service without any performance drop. And with the added advantage of being accessible to many more people over the network. The only limiting factor here is the network delay.

1.6 Interoperability in web services

Interoperability is defined by IEEE as the ability of two or more systems or components to exchange information and to use the exchanged information [24].

Interoperability would require an application to interact with other applications written using different languages or operating in different platforms. In web services the possible scenarios for web service and web service clients can be endless. Some of the main issues seen in interoperability are

- i) Usage of complex data types – Whenever complex data types are specified, they would ultimately be represented in XML. But the XML schema may not be implemented to its fullest; the degree of the schema implementation may vary from vendor to vendor. So support for complex data type may vary from vendor to vendor. Therefore it is best to use primitive data types which are supported by all vendors.
- ii) Interoperability issues of wire protocol – SOAP sits over the same wire protocol like HTTP, SMTP, and Jabber etc. So the different interoperability issues related to these protocols are also escalated to web service interoperability
- iii) Document/Literal or RPC encoding – Some web services like Microsoft follow the document/literal encoding while some web services like Apache web service follow the RPC encoding. SOAP messages have many optional parts, this leads to a number of interoperability issues.
- iv) Security and related issues – There are stringent requirements in security specifications. Each of them follows their own security policy. Specifications like WS-Reliability and WS-Specification are built on top of SOAP and they provide security features [25].

Document schemas create dependencies among documents because any change to the document like any data added or removed may directly affect existing applications. In this sense RPC message exchange is much more transparent. The problem of standardizing structure and semantics of the document is reduced to a much more manageable task of standardizing service interfaces [26].

When discussing about interoperability, an important issue is to follow specifications or follow a standard. If each developer uses their own technique it

would be very difficult to put all this together, hence a more homogenous approach is needed to put together all applications to interact with each other. There is a lot of research going on in increasing automation in service interoperability i.e. the client application will read the requirements to bind to the web service by going through the WSDL, the WSDL is more for machine consumption but most WSDL documents come with a documentation file which contains details of the WSDL document, such that it is easy for humans to understand. The SOAP document is also semiformal and it is meant for humans to understand, but only such that they can develop middleware tools to support it [27].

1.6.1 Work Done

Chapter 4 explains the state of the art in interoperability issues and it explains about the platform which has been developed to test commonly seen interoperability issues across two platforms which are the most commonly used tools for web service development that is Microsoft .NET and the Java platform. Interoperability issues arise when the server and the client are developed using different platforms. The commonly seen issues are with the transfer of primitive data types, complex data types, namespace issues and with representation of date and time.

1.6.2 Related work from literature survey

The author of [28, 29, and 30] discusses about the different interoperability issues, all these issues have been programmatically developed and tested during the course of this research. A platform for testing the various interoperability issues has been developed. The different issues which are tested are interoperability issues when an array with null elements is passed. Passing of different primitive data types like unsigned numbers, float values etc. Passing complex data types like objects between the platforms. The results obtained after performance of the tests is clearly explained in chapter 4 of this thesis report.

[31] discusses work is done with both basic and complex data types and the authors have encountered interoperability issues, in their paper they propose a business model where data binding tools not only generate the WSDL, but also provide portable binding extensions for manipulating the XSD types, these binding extensions can be integrated into any other binding tool, overcoming their limitations. Other work in web service interoperability as observed during literature survey are more of analytical work where current standards are assessed and specifications for identifying new opportunities are suggested as the work carried out by [32] indicate. Similarly [33] study the interoperability issues involved in e-government issues and they highlight the fact that interoperability needs to focus on both technical and non-technical issues. In [34] a smart home environment has been designed where home entertainment, home surveillance, energy management, assistive computing and healthcare are all interfaced using web service technology and they discuss about interoperability issues and present a simple SOAP solution to solve interoperability issues.

Interoperability issues can be solved to some extent by having some similarity among several service registries as suggested by [35] crawler engines can also be designed such that they could aggregate the web service references, resources and description documents. Since one of the main beneficiaries of the web service technology happens to be business organizations, web services can solve interoperability to some extent by making use of the Universal Business Language (UBL) such that the schemas of the UBL documents can translate to each other.

1.7 Security in Interoperable web services

As explained previously web services are loosely coupled applications which can interact across multiple platforms and across multiple programming languages and operating systems. Web services make use of the SOAP protocol for data transfer, SOAP is nothing but XML documents. These XML documents flow as clear text and it is not difficult for an intruder to intercept and retrieve sensitive

information like credit card numbers, therefore to enable security of data transaction XML digital signatures and XML encryption is used. Both XML digital signatures and XML encryption use Public Key Infrastructure (PKI) for encrypting, decrypting and signing the various documents. For an application to use XML encryption and/or digital signature, the application must use or integrate with a PKI solution. Various PKI solutions are available, such as X.509 (widely used), Pretty Good Privacy (PGP), Simple Public Key Infrastructure (SPKI) and Public Key Infrastructure X.509 (PKIX). To enable one application to talk to another application either both of them should use the same PKI solution or they should be aware of each other's PKI solutions [36]. For example, if organization A uses an X.509 PKI solution and sends encrypted documents to organization B, which uses an SPKI PKI solution, then organization B won't be able to decrypt and use the document sent by A. For A and B to work together, one of them has to understand the other's PKI solution. If this scenario is extrapolated to a situation where multiple partners are involved, it becomes clear that all of the partners will have to be aware of each other's PKI solution, thus increasing each application's complexity many times. In order to abstract the user from all these complexities the XML Key Management Specification (XKMS) takes care of maintaining the keys and certificates and lets enterprises exchange their digital signature or their certificates.

1.7.1 Work Done

Web service security interoperability platform has been developed using Java and .Net. The web service is developed in Java using the Netbeans IDE; the web service performs simple addition and simple interest calculation. A C# client is developed to invoke this web service. The web service developed is a secured web service which uses username authentication with symmetric key. The client has to authenticate itself to the web service before it can access the service offered by the web service, this is done by certificate exchange, and the client's certificate is matched with the issued certificates by the glassfish server. The glassfish server is used to deploy the web service. Once the certificate matches

than a session key is enabled and there is than flow of data between the web service and the web service client.

The important work that is accomplished by the developed web service security interoperability platform can be specified as given below.

- i) It is observed that the certificates to be exchanged by the two different platforms Java and .Net have different formats which are represented by the different file extensions.
- ii) The certificate stores are converted to a format which could be understood by each other.
- iii) The transfer of data after certificate verification is verified. The verification is done by logging the SOAP messages which travel back and forth between the web service and the client by using a logging tool like the TCP monitor.

1.7.2 Related Work: Setting up Web Service Security (WSS) can be a very complex and difficult task even for somebody associated with security leave alone somebody new to security, [37] have proposed an API which allows non security experts to easily configure and enable WSS. This API makes use of a six step programming model to configure and test WSS easily. This API which is developed is than compared with similar API's like JSR, WSS4J, WSE, and WSSAPI in terms of lines of code, number of classes and cyclomatic complexity.

When security is added in web services, the performance of WSS remains a concern due to the additional security contents added to the SOAP message and extra service time for processing these security contents. In [38] the authors conduct a performance evaluation of WSS. A simple web service is designed and used for performance testing with a variety of WSS policies and message sizes. The test results are categorized, compared and analyzed to work out the overheads for individual security setting.

WS-FESec is a framework which is developed and it highlights the importance of end to end security in web services [39].

Most of the existing tools give a technology oriented view and only assist in choosing the data to encrypt and selecting an encryption algorithm, the users would have to design their own security model and decide on how they would relate to their business policies. The authors of [40] have described a tool which gives a simplified business-policy-oriented view. It models the messaging with customers and business partners, lists various threats and presents best practice security against the threats.

Just checking the access control rights of a person trying to access a web service may not be sufficient but it may be required to check the web access history of the person, data mining operations like association rules are also performed simultaneously to predict the web services the person could possible invoke. A match is done to see if the request matches with the analysis done using data mining; depending on the outcome web access is either permitted or denied [41].

A large number of papers have been written which explain about the current state of the art of XML and web service security. This is important because of the need of WSS to constantly evolve with time such that the technology is ahead of hackers and security breaches. A very good reference in this area is [42].

During literature survey it was observed that a platform developed for web service security interoperability like the one developed during the course of this research does not exist, hence this research bridges this gap. This developed platform allows different interacting systems to exchange authentication information prior to data exchange; the platform is an excellent base for young researchers to experiment with different security *specifications*.

1.8 Scope and Limitations of Research

1.8.1 Scope

Web Services are a very important technological innovation. They enable services to be hosted such that a service consumer can simply invoke a service without bothering about which programming language is used and in what platform the application is developed. Web services can be composed into large applications by using Business Process Execution Language (BPEL). This enables large services to be developed by combining smaller services. Work carried out in this thesis is divided into four main areas.

- i) A case study is developed for converting a desktop data mining application to web service. It is programmatically verified that the the output of the desktop application is the same as a web service application.
- ii) Mobile enabled publishing, location and invocation of web service is developed.
- iii) A GUI based platform is developed to check interoperability issues across Java and .Net .
- iv) A platform for studying secured communication between a .Net and J2EE web service has been developed and verified.

1.8.2 Limitations

- i) An alternate form of web services is REST (REpresentational State Transfer) web services. REST is a design idiom that embraces a stateless client-server architecture in which web services are viewed as resources and can be identified by their URL's. This research is confined to SOAP based web services, and in future it can be extended to REST web services.
- ii) Semantic web services is an emerging area for discovering a particular preferred web service among similar web services by adding semantic

information to the description of web services. In future this work can be extended to semantic web services by adding semantic information to the WSDL document.

iii) Scalability is an important issue to be considered, when multiple clients invoke a particular web service at the same instant of time. This research can be extended to scalability issues.

iv) Response time analysis of invoking the different tools developed during this research is not addressed.

1.9 Organization of the Thesis

Chapter 2 Discusses the state of art in web services.

Chapter 3 Deals with applying web services to a data mining application. Tests are performed to check the performance of the desk top data mining application as compared to a web service.

Chapter 4 Discusses the issues seen in web service interoperability across diverse platforms.

Chapter 5 Presents an innovative web framework for developing mobile enabled web service .

Chapter 6 Explains the development of a framework for studying web service security across .Net and J2EE platform.

Chapter 2: State of Art in Web Services

2.1 Introduction

This chapter discusses the evolution of web services and SOA. The different building blocks of web services like UDDI , WSDL, SOAP and XML are explained in depth.

2.2 Working of Web services

A Web service is a software system designed to support interoperable machine to machine interaction over a network. It has a interface which is described in a machine understandable format (Web Services Description Language).Other systems interact with the web service in a manner prescribed by SOAP messages, typically conveyed using HTTP with XML serialization in conjunction with other web related standards.

An important point to remember with web services is that they need not be available on the world wide web , they can be anywhere from the internet to the intranet, infact web services have little to do with browser focused and HTML focused World Wide Web [43].

In a web service the invocation method is very important, how the web service is deployed and implemented is not of any concern this is very similar to the web browser interacting with a web application. The browser just does not care about the application because all that the browser is concerned about is the wire level protocol HTTP which is used to transfer the data and the wire level format HTML used to present the data. As long as the site supports the protocol and this format it would continue working [44].

In the traditional approach if a client had to make a booking for a business trip they would have to log on to multiple sites and make separate bookings for their air ticket, the hotel and pay separately through their credit card. But using a web

service to do the booking would considerably reduce the effort of the client as explained below.

Figure 2.1 [110] explains how a web service works. A client who is planning a business trip logs onto a client application which in turn communicates with a travel agent web service which communicates with other web services like the hotel reservation web service, the airline reservation web service and the credit card company web service for the payment. Here all the interactions are machine to machine with no human interaction. Finally the travel agent web service combines the response of all the other web services and gives the final itinerary to the client.

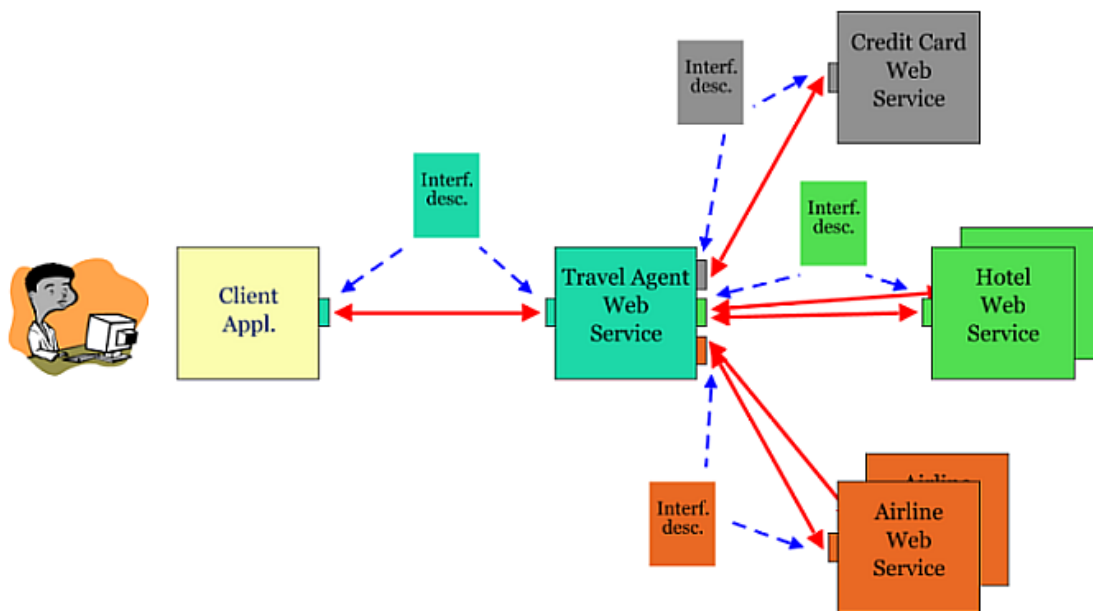


Figure 2.1 Example of a web service application

2.3 Evolution of Web services

i) **Mainframe architecture:** Used in the yesteryears where computers were just being used in applications. The applications were large, monolithic and expensive. Very few companies owned computers.

ii) Client Server application: N number of clients could interact with a single server; the server would process the request and send back the response to the client as illustrated in Figure 2.2 [110] . Now if there is a change in the client code than this code would have to be redeployed and installed on all the client machines. Now if this application is within an office where the clients are accessing a database server in the office itself this would be possible. But over the internet where there are a number of clients located world wide this may not be possible. This was the reason for client server applications not being used in large distributed applications.

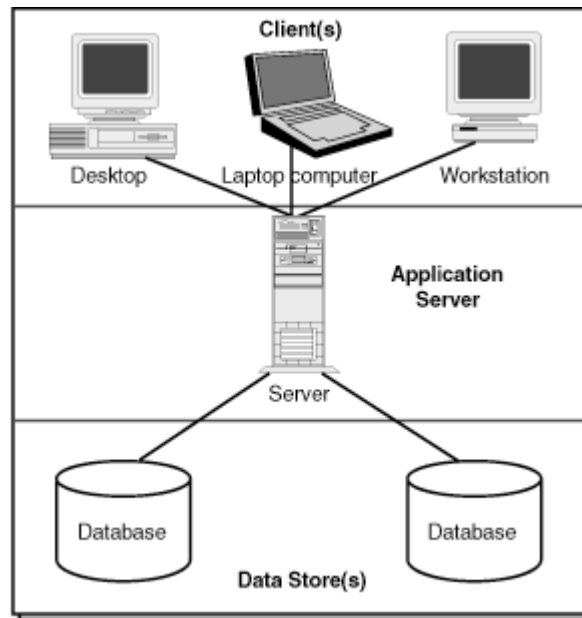


Figure 2.2 Client Server applications

iii) Distributed Architecture: Here individual computers are geographically located in different places and they are connected over the network. This was an efficient method of connecting computers because individual computers could perform computations simultaneously, without blocking for the services of the server .The computers used TCP/IP as their communication protocol.

iv) Web based application: The internet and the World Wide Web gave rise to a lot of business opportunities, they led to the globalization of a number of companies. Businesses could either be business to business B2B or business to consumer B2C. B2B allows electronic exchange of documents like purchase orders or invoices without any human interaction, B2B could be used for a small application like credit card validation to a large application like the full automation of a supply chain management. In B2C there was human to computer interaction, and this would be in the form of a request response interaction. Where the user would send a request for a particular website and the server would send a response. Here this interaction between components was browser based, if the browser did not support a particular format a document could not be viewed. Commonly used applications include web shopping portals, online banking application, distance education and stock trading application.

- a) In a B2C application the consumer would interact with java servlets or enterprise java beans while B2B applications can interact with straight java code which can be hosted on web service engines.
- b) B2C handle data over the HTTP protocol input comes in the form of GET parameters or POST parameters from forms. While B2B applications can use internet protocols like HTTP, FTP, and SMTP.
- c) B2C uses HTML which only allows string data types to be transmitted, even numbers are encoded as strings during transmission, B2B uses XML for data transmission, XML is programming language and platform neutral so any form of data can be transmitted.
- d) B2C needs a user interface because a human would be interacting with the system, B2B does not need a user interface because of a machine interacting with the application.

V) Service Oriented Architecture (SOA): With the usage of the internet businesses expanded, but business to business applications became rare. SOA offers a platform for services to be available and these services to be invoked by other applications over the internet.

2.4 Introduction to Service oriented Architecture (SOA)

In SOA all software components (functional units) which can be consumed over a network are modeled as services. The focus here is on the interface because other applications can invoke this process.

For e.g. a banking application which already has a loan processing application can be considered. The bank might decide to offer this application as a service to be accessed by other banks. Similarly an application may be designed to give the best rate of interest among a number of banks, so an application may be designed which just compares the rate of interest of different banks and gives the best interest possible so here the concept of a service oriented architecture where a number of independent services are collaborated together to generate a bigger service.

Another application of a service oriented architecture could be restaurant finder in a particular area, this is a service which can be invoked by any person, now this service may be interfaced with a map quest service which could give the map of a particular area and the route or direction to a particular restaurant so here again a combination of services giving rise to a service oriented architecture can be seen.

Another interesting application could be a doctor using a hand held device to access patients medical history, the doctor could then write out a prescription which is sent online directly to the pharmacy, the pharmacy could then courier the medicines to the patient if he so desires.

From the above examples it can be seen that services can be something which is as discrete or trivial as a currency converter or a language translator used for simple text conversions between English and French. Or a service can be

something as complicated as handling a complete supply chain process for a large company can be done with SOA [45].

2.4.1 SOA Architecture

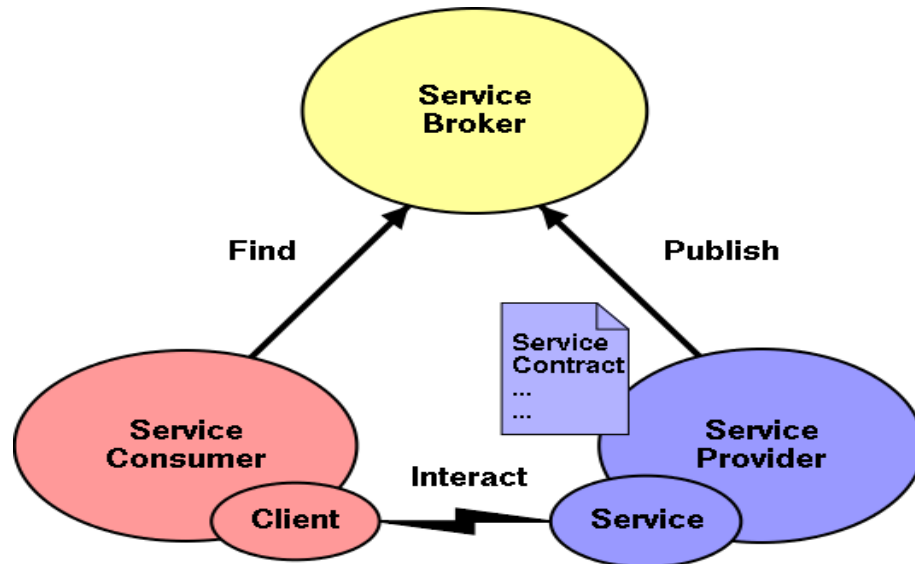


Figure 2.3 Service Oriented Architecture

The interaction of the different components of SOA are shown in Figure 2.3 [110] and they are defined as:

Service Provider: Is responsible for developing and deploying the service, the service provider also publishes the service with the service broker

Service Broker: Is more commonly called the service registry, contains information about the different services, their description, and their location.

Service Consumer: Is responsible for invoking the service.

These different roles can be played by any computer on the network for e.g. a computer which is invoking a service may be providing a service to a downstream computer.

SOA also includes three operations publish, inquire and bind.

Publish : Publish is the act of advertising the services available, this can be done by moving the service description into a web application server's directory structure or in a UDDI it can be a more sophisticated operation

Find: Is an operation of querying the service registry. How the result is displayed would depend on how the registry is organized. In a simple registry a simple HTTP GET would give all the web services available, but a UDDI has many powerful find capabilities.

Bind/Interact: It represents the relationship in a client server model, it can be sophisticated and dynamic such as on the fly generation of a client side proxy used to invoke the web service or it can be a static model where the developer hand codes the way a client application invokes a web service [43].

A scenario of a travel service provider who exposes the available business applications as services is considered, these services could be obtained from other businesses like an airline company, hotel reservation system, car rental system, credit card system etc. A service requester could send a request either from a desktop or from a mobile device to the travel registry which will in turn interact with the other services. This type of application is seen in the web sites of certain airline companies for e.g. South West airlines has a tie up with Dollar rent a car and if bookings are made through the airline web site the customer is offered a better rate, similarly a customer can book into a hotel from the airlines web site itself here a collaboration of a number of services together forming a service oriented architecture is seen.

2.4.2 Importance of SOA

- i) SOA changed the way that software was developed; software is built as loosely coupled mix and match processes.
- ii) It becomes easy to scale an application, for e.g. if an application has just 2-3 processes it can be scaled to a big application.
- iii) Having a large monolithic, tightly coupled, inflexible software is difficult to maintain and modify, this will slowly give rise to the SOA approach
- iv) Any changes to the application can easily be modeled, for e.g. the arrival of a new supplier, merger of two divisions etc.

- v) Another important feature of SOA is the merger of IT concepts with business rules.
- vi) Code reuse is an important feature, if code to perform currency conversion is already available, a system can simply invoke this service rather than build this service from scratch leading to a more specialized industry.

2.4.3 SOA and Web services are distinct but different

SOA is an architectural concept which focuses on building systems which are built as loosely coupled components or services which can be dynamically composed. Web services on the other hand are one approach to building a SOA application.

Web services provide a standard of a particular set of XML related technologies that can be used to build SOA applications [43]. Web services are instances or implementations of SOA. Other instances of SOA deployments, other instances include Distributed Component Object Model (DCOM) and the Common Object Request Broker Architecture (CORBA) [4]. Web services seem to become the preferred implementation technology for realizing the SOA promise of maximum service sharing, reuse and interoperability. Web services and SOA reduce complexity of enterprise application eco-systems by encapsulation and minimizing the requirements for shared understanding by defining service interface in an unambiguous and transparent manner [46].

2.5 Types of Web applications

Web applications can be anyone of the two types.

- i) Browser based Web application
- ii) Web services where there is an exchange of SOAP messages

Bowser based Web applications

- i) The browser acts like a thin client and it invokes a service which is hosted on the server.

- ii) The output seen is dependent on the browser used.
- iii) It is easy to upgrade the server application, without affecting browser's access to the application.

Web services based applications

- i) There is an exchange of information with the flow of XML messages.
- ii) Web services can interact across different platforms, for e.g. the service and the client can be hosted on computers with different operating systems and they can be coded with different programming languages.
- iii) Web services can be developed on different programming languages like .NET, Java, C# etc.
- iv) Web services are not meant for handling presentations like HTML, they are developed using XML, such that they can be accessed by any software application, any device and from any platform [47].

2.5.1 How are Web services different from Web applications?

- i) In web applications interaction is only with the browser, in web services methods are invoked directly and requests can come from any source or from any server.
- ii) Web applications interact with the browser, the browser is now dependent on the operating system and the hardware of the computer, the browser would be able to communicate only with HTML documents and it could understand the HTTP protocol only while web services interact through other XML based protocols also.
- iii) In a traditional web application, the actual user is on the other end, so in a single sign on application a user can be asked to authenticate himself if required whereas in a web service the originator of the request may not be available for authentication as and when required.
- iv) All browser based web applications make use of the Secure Socket layer (SSL) for secure transfer of data. SSL is a secure and reliable protocol which provides end to end security between two parties. Web services

make use of XML encryption. XML encryption provides two very important additional features which are not addressed by SSL.

- a. Encrypting only part of the data being exchanged
 - b. Secure sessions between more than two parties.
- v) SSL encrypts the data completely between two parties, thereby increasing the overhead at both ends, in case of XML encryption selective encryption of data is possible as indicated in Listing 2.2 [48].

Listing 2.1 Unencrypted XML purchase order document

```
<purchaseOrder>
  <Order>
    <Item>book</Item>
    <Id>123-958-74598</Id>
    <Quantity>12</Quantity>
  </Order>
  <Payment>
    <CardId>123654-8988889-9996874</CardId>
    <CardName>visa</CardName>
    <ValidDate>12-10-2004</ValidDate>
  </Payment>
</purchaseOrder>
```

Listing 2.2 Encrypted XML purchase order document with only payment details encrypted

```
<?xml version='1.0' ?>
<PurchaseOrder>
  <Order>
    <Item>book</Item>
    <Id>123-958-74598</Id>
    <Quantity>12</Quantity>
  </Order>
  <Payment>
    <CardId>
      <EncryptedData
Type='http://www.w3.org/2001/04/xmlenc#Content'
xmlns='http://www.w3.org/2001/04/xmlenc#'>
        <CipherData>
          <CipherValue>A23B45C564587</CipherValue>
        </CipherData>
      </EncryptedData></CardId>
    <CardName>visa</CardName>
```

```
<ValidDate>12-10-2004</ValidDate>
</Payment>
</PurchaseOrder>
```

From Listing 2.2 and 2.2 we can conclude that XML permits selective encryption where only credit card information is encrypted, and all other data is sent in clear text

2.6 Building blocks of Web services

The building blocks of web services are XML, SOAP, WSDL and UDDI. The working and the significance of each of the following is explained in Figure 2.4 [110].

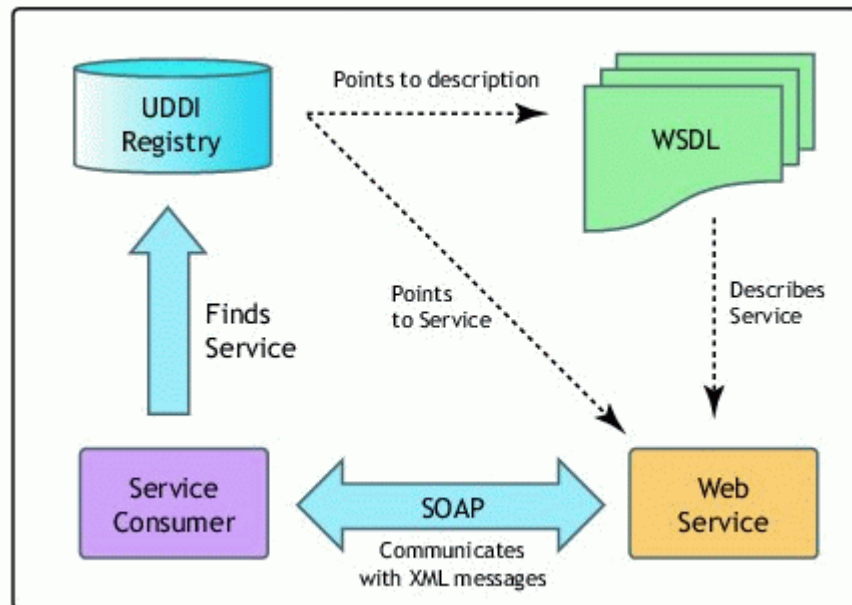


Figure 2.4 Interaction in web services

2.6.1 XML Extensible Markup Language

XML has changed the way of structuring, exchanging and describing data. For web services the significance of XML is paramount. All key web service technologies are based on XML. Many specifications are built on top of XML to

extend its capabilities and use it in a broader range of scenarios. XML is used to describe structured and semi structured data in textual form. Semi structured data can be in the form of a doctor's prescription and a structured document can be in the form of representation of scientific data from manuals and reports.

Table 2.1 Sample HTML and XML documents used to describe the same data (address of John Doe).

Sample HTML document	Sample XML document
<pre><html> <body> <h2>John Doe</h2> <p>2 Backroads Lane
 New York
 < 045935435
 john.doe@gmail.com
 </p> </body> </html></pre>	<pre><?xml version=1.0?> <contact> <name>John Doe</name> <address>2Backroads Lane</address> <country>New York</country> <phone>045935435</phone> <email>john.doe@gmail.com</email > </contact></pre>

The HTML document describes more about how the document is to be displayed and does not explain about the content of the document; it is difficult for a machine to extract information from the HTML document. The XML document describes the content of the document and it is easy for a machine to understand.

XML Separates Data from HTML

If dynamic data in an HTML document needs to be displayed, it will take a lot of work to edit the HTML each time the data changes. With XML data can be stored in separate XML files. This way more concentration can be on using HTML for layout and display, and be sure that changes in the underlying data will not

require any changes to the HTML. With a few lines of JavaScript, an external XML file can be read and updated.

XML Simplifies Data Sharing

In the real world, computer systems and databases contain data in incompatible formats. XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data. It makes it much easier to create data that different applications can share.

XML Simplifies Data Transport

With XML, data can easily be exchanged between incompatible systems. One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the internet. Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

XML Simplifies Platform Changes

Upgrading to new systems (hardware or software platforms), is always very time consuming. Large amounts of data must be converted and incompatible data is often lost. XML data is stored in text format. This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data

XML Makes Data More Available

Since XML is independent of hardware, software and application, XML can make data more available and useful. Different applications can access data, not only in HTML pages, but also from XML data sources. With XML, the data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc), and make it more available for blind people, or people with other disabilities.

XML is used to create new internet languages

A lot of new Internet languages are created with XML.

Here are some examples:

- i) XHTML the latest version of HTML
- ii) WSDL for describing available web services
- iii) WAP and WML as markup languages for handheld devices
- iv) RSS languages for news feeds
- v) RDF and OWL for describing resources and ontology
- vi) SMIL for describing multimedia for the web

The building blocks of XML are

- i) XML Instances: Are used to create syntactically correct documents.
- ii) XML Schema: A standard that allows detailed validation of XML documents as well as the specification of XML data types.
- iii) XML Namespaces: used to combine many XML documents from different sources to produce a single XML document.
- iv) XML Processing: is used for creating, manipulating and processing XML documents from programming languages as well as mapping Java data structures into XML.

XML documents are broadly divided into two categories

- i) Document centric XML
- ii) Data centric XML

i) Document centric XML – used to represent semi structured data like technical manuals, legal documents and product catalogues which are meant mainly for human consumption.

ii). Data centric XML – used to represent highly structured data like textual information of a database system, financial transactions and programming language data structures. It is generally generated by machines for machine consumption and used to represent highly structured and repetitive data [43].

An important feature of XML is its self describing ability; an XML document contains a XSD (XML Schema Definition) which defines the schema of the language. The use of XSD makes XML more resilient to changes in data, because when there is a change in the XML document it can be reflected in the schema definition making it easy for the parser to parse the given document [49] There is another form of defining data i.e. DTD (Document Type Definition) but this form is not used with web services because the structure of the DTD document is totally different from an XML document. A large number of DTD documents are still found on the web, but the growing demand for XML and the shortcomings of DTD led to alternate description schemas like RELAX, TREX and XSD [50].

There are two types of parsers available in XML

- i) SAX parser
- ii) DOM parser

- i) SAX (Simple API for XML) parser transforms the XML document to a string, giving the programmer a temporal sequence of tokens.
- ii) DOM (Document Object Model) Represents the XML document as a node labeled tree in which the nodes correspond to the distinguished components of the format [51].

2.6.2 SOAP Simple Object Access Protocol

- i) SOAP is a wired protocol, which means that SOAP is used to define how data is serialized and transmitted over the network, for

e.g. if a data 56 is transferred on the wire it specifies how the data is transmitted. In SOAP the data is transmitted in XML whereas in other transport protocols data is transferred as binary.

- ii) SOAP uses XML framework to define extensible messaging framework which works with a number of other protocols like HTTP, FTP and also MSMQ. SOAP is by nature platform neutral and vendor neutral. These characteristics allow for a loosely coupled relationship between requester and provider which is especially important over the internet where two parties may reside in different organizations or enterprises [46].
- iii) SOAP is put on HTTP, most systems which are behind the firewall use port 80 to communicate with the internet, so SOAP also travels through this port.
- iv) SOAP permits complex data to be put on HTTP, unlike HTTP which only allows name value pairs
- v) SOAP protocol supports a rich set of data types like enumeration, structures and arrays.
- vi) SOAP is a light weight protocol used for exchanging structured information in a decentralized, distributed environment.

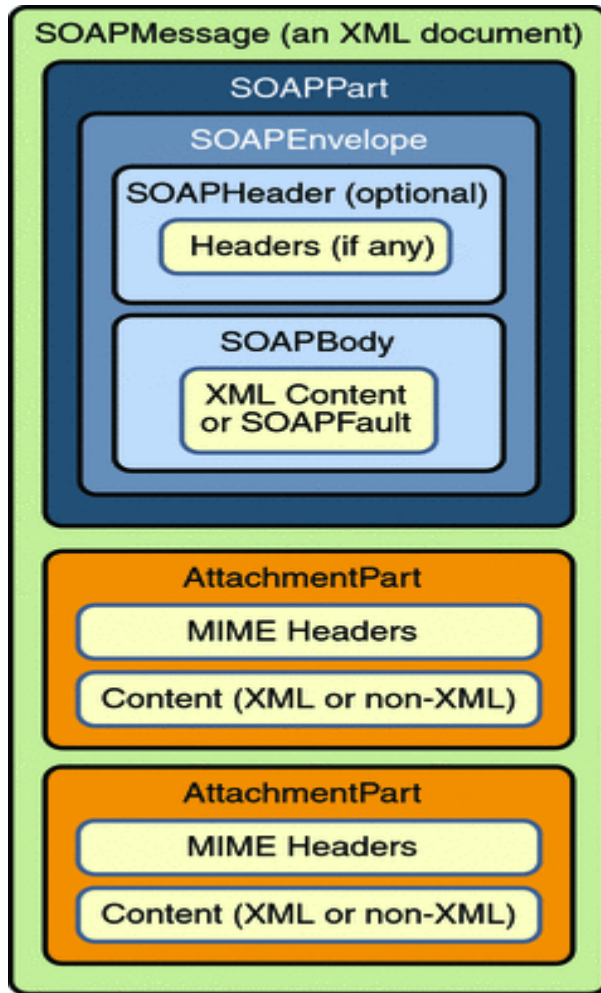


Figure 2.5 Structure of a SOAP message

The components of a SOAP message are represented in Figure 2.5 [110]. A raw XML document is converted to a SOAP message by the addition of a SOAP envelope which in turn contains a SOAP header(optional), and the SOAP body(compulsory).The SOAP message may also contain attachments, these attachments could be XML or non-XML files (i.e. image files, text files etc.).The SOAP header provides a mechanism to extend the SOAP message by adding functionalities like security, transactions, priority etc

SOAP Body: Is the mandatory part of the message, more than one body may be contained in the SOAP message; it contains the message which is to be transferred to the receiver. The body block of the SOAP message can contain any one of the following:

- i) RPC method and its parameters
- ii) Target application (specific data for the receiver)
- iii) SOAP fault for reporting any errors and status information.

2.6.3 Web Service Description Language (WSDL)

WSDL is an XML document which is used to describe web services. With a large number of communication formats and protocols it becomes necessary to define a standard. WSDL provides a standardized format for all the necessary information about services like name of the service, location of the service, parameters exchanged, the communication protocol used etc. The WSDL should be easily accessible to clients, in order to make the web service known the service provider registers the WSDL in a registry. A client wanting to access the web service retrieves the WSDL details from a registry like a UDDI and it can then invoke the web service by sending a SOAP request to the specified web service.

In most of the cases application tools or Integrated Development Environment (IDE) like Netbeans can generate the WSDL document for a service during the development of the service. In some cases it is possible to generate the WSDL document for code, which has already been developed and deployed by using tools to perform this operation, for example there could be a courier tracking application, weather reporting service or a stock quote service which is already developed but it is to be accessed as a web service here the tools will have to go through the source code and generate the WSDL. In some cases the WSDL document can be created and tools can be invoked to create the matching J2EE base component to create the web service.

Structure of the WSDL document

The WSDL contains four important components.

- i) Interface information, which contains information on publically available functions

- ii) Data type information about incoming messages (request) and outgoing information (response)
- iii) Binding information about the protocol used to invoke the Web service
- iv) Address information for invoking the specified Web service

The services description is accomplished through a set of seven special XML elements as illustrated in Listing 2.3. These elements are specific to WSDL and are governed by the XML namespace convention. The seven elements are : <types>, <messages>,<operations>, <portType>, <binding>, <port> and <service> .

Definitions are the root element that contains all seven elements that describe web services.

- i) <types> element - defines the data type definitions using some type of system like the XML Schema Definition(XSD). XML namespaces are also used when required, they define the data types of the information used in the message element.
- ii) <message> element - this defines the message that will be exchanged between the two ports. It specifies the name of the message, the input or request data type and the output or response data type. There could be many messages exchanged and each message is identified by a part name
- iii) <operations> element – is an abstract definition of the actions supported by the web service, it supports two types of elements
 - <input> - supports server side action
 - <output> - supports client side action
- iv) <portType> element - Describes one or more abstract set of operations supported by one or more endpoint. In order to distinguish between one or more operations a name attribute is associated with the operation.
- v) <binding> element – specifies a concrete protocol and a data format specification for a <portType>

- vi) <port> element – provides a single endpoint which is defined by a combination of <binding> and network address to specify where a web service is hosted.
- vii) <service> element - Identifies the web service since multiple web services can be hosted. It models the multiple services through a list or collection of <port> elements [52].

Listing 2.3 Sample WSDL documents for a stock quote service

```

<?xml version="1.0"?>
<definitions name="StockQuote"
  targetNamespace="http://example.com/stockquote.wsdl"
  xmlns:tns="http://example.com/stockquote.wsdl"
  xmlns:xsd1="http://example.com/stockquote.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
      xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TradePriceRequest">
        <complexType>
          <all>
            <element name="tickerSymbol" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="TradePrice">
        <complexType>
          <all>
            <element name="price" type="float"/>
          </all>
        </complexType>
      </element>
    </schema>
  </types>

  <message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"/>
  </message>

  <message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice"/>
  </message>

```

```

<portType name="StockQuotePortType">
  <operation name="GetLastTradePrice">
    <input message="tns:GetLastTradePriceInput"/>
    <output message="tns:GetLastTradePriceOutput"/>
  </operation>
</portType>

<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
  <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetLastTradePrice">
    <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>

<service name="StockQuoteService">
  <documentation>My first service</documentation>
  <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
    <soap:address location="http://example.com/stockquote"/>
  </port>
</service>
</definitions>

```

Limitations of WSDL

WSDL is not capable of describing complex business web services, where a web service consists of multiple fine grained web services. This is due to the fact that WSDL does not support workflow descriptions in WSDL. To overcome this eXtensible Business using XML(ebXML) and Web Service Choreography Interface(WSCI) are provided. The information provided in the WSDL may be too generic and inadequate to properly invoke a target web service. Some extra semantic information may be provided to help construct input parameters [24].

2.6.4 Universal Description Discovery Interface (UDDI)

The UDDI is a directory which contains the description of the different web services which are available. It is an electronic medium in which other services can discover web services registered with this UDDI, it is also a medium where web service developers can register their services to be invoked by others. When services have to be discovered among a small number of businesses it becomes easy to identify the service manually, but when the number of business is large and distributed all over the world, it becomes important to have a registry where these services can be searched for. Companies like Microsoft, IBM etc maintain UDDI repositories.

Technical architecture: The UDDI is a single system built from multiple nodes which have their data synchronized through data replication. All the nodes together form the UDDI Business Registry (UBR). Data can be added at any of these nodes but any editing or deletion of data is allowed only at the node where the data was added. Figure 2.6 explains the architecture of the UDDI [110].

Two API's are described by the UDDI specification.

- i) Inquiry API – used to enquire the UDDI, users need not be authenticated hence the protocol used is HTTP.
- ii) Publishing API – used to create, store or edit information, users must be authenticated to use the registry hence the HTTPS protocol is used.

Registries are of two kinds

- i) Public registries
- ii) Private registries

i) Public registries are available for anyone to publish/query their service information. Public registries could generally be distributed, but they are synchronized, so if changes are made at any site the changes will be synchronized. Such registries are called UBR. Content in these registries can be

uploaded at any site but if the data is to be edited it can only be done at the site where data was added.

- ii) Private Registries are owned privately and a user must be authenticated to be allowed access to these registries [47].

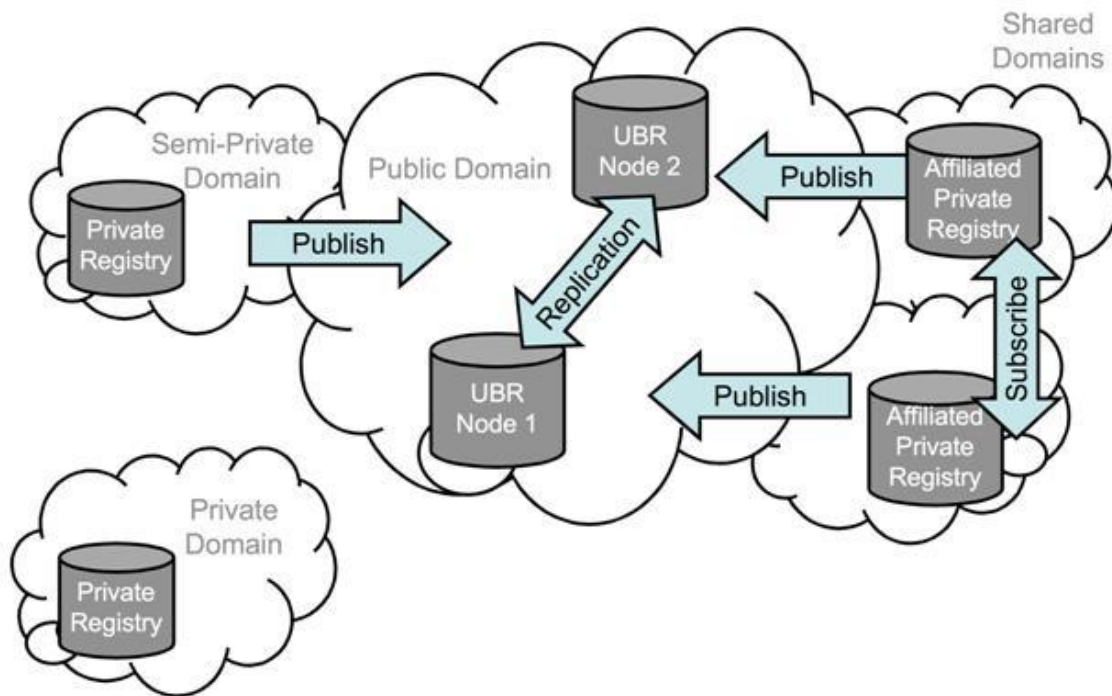


Figure 2.6 Architecture of UDDI

Depending on how a UDDI is used it is classified into one of these categories

- i) White pages: Here companies register information like name, address, contact information etc. White pages will be more or less like the regular telephone directory where information is stored according to alphabetical order. Searching in this directory can be difficult.
- ii) Yellow pages: Businesses classify their information based on different categories of business like automobiles, share brokers etc very much like the yellow pages available in telephone directories. This classification of information makes searching for a particular data easy.

- iii) Green pages: Technical information about the web services like the behavior and supported functions is published here.

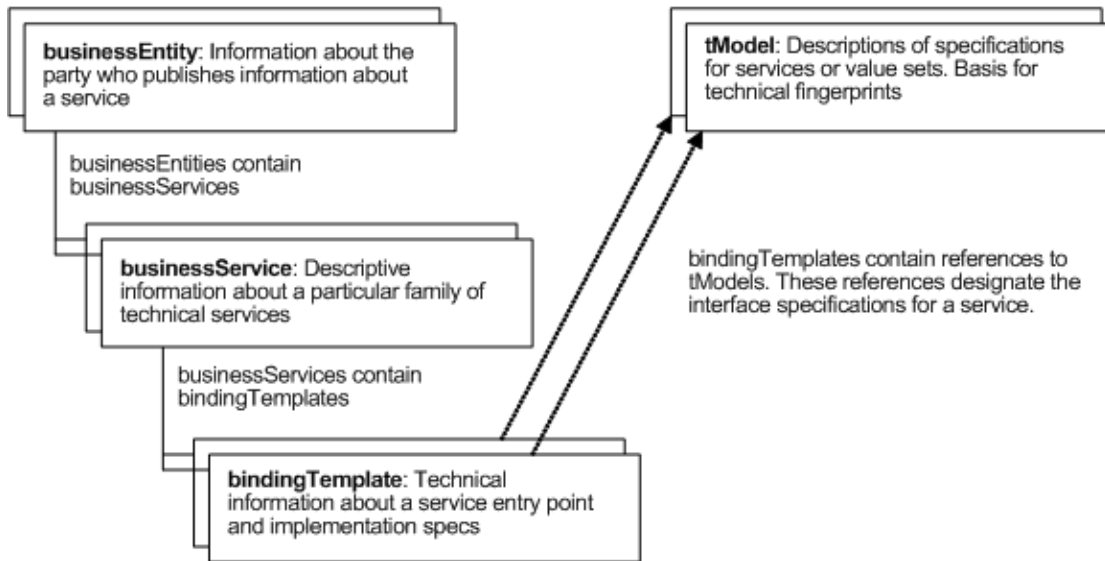


Figure 2.7 UDDI data structures

UDDI Data Structures

The different data structures in the UDDI are illustrated in Figure 2.7 [110].

- i) **businessEntity:** contains general information about the company like contact information, categorizations, identifiers and relationship with other companies.
- ii) **publicAssertion:** Is used to establish public assertion with another company, only if both companies specify the same relationship with each other than they can be seen by the public.
- iii) **businessService:** Is a single logical service classification, it describes the set of services provided by the business, these services can be manual services or they can be web services for e.g. if a college has a faculty web service this can be accessed by the administration web service or the admissions web service, it can contain one or more binding template.
- iv) **T-Model:** Is the technical model which has the digital fingerprint which specifies how to interact with a particular web service, it provides pointers to a location which gives these specifications.

As explained in [53] a browser can also be used to access web services. During the early years of web services, browsers did not have the ability to send SOAP messages, hence an intermediary was required to transform a HTTP request to a SOAP request and vice versa, this was done by components within the web portal the only disadvantage of this method is that end to end connectivity was broken.

In recent time's browsers like Mozilla starting from version 1.0 and Internet Explorer 5 support SOAP. They offer some form of SOAP API which can be used with scripts embedded in HTML to send SOAP messages directly from browser, the only disadvantage is that the API's are browser specific.

UDDI suffers from several shortcomings:

- i) Provides no guarantee for the quality of the registered service.
- ii) Registration to UDDI is voluntary, therefore public registries could contain passive and outdated information.
- iii) Lacks business oriented integration capabilities like service provider validation, pricing models, security, trust establishment etc.

In some cases businesses could maintain a separate registry like ebXML which allows a business to business framework which allows organizations to advertise and discover information about themselves. It contains XML documents which specify about the manner in which businesses exchange information.

Because of the possibility of service providers advertising their services on different registries, the data may be duplicated, therefore searching for relevant information may be difficult. It is a good idea to have a web service crawler engine that can facilitate the aggregation of web service references, resources and description documents [54].

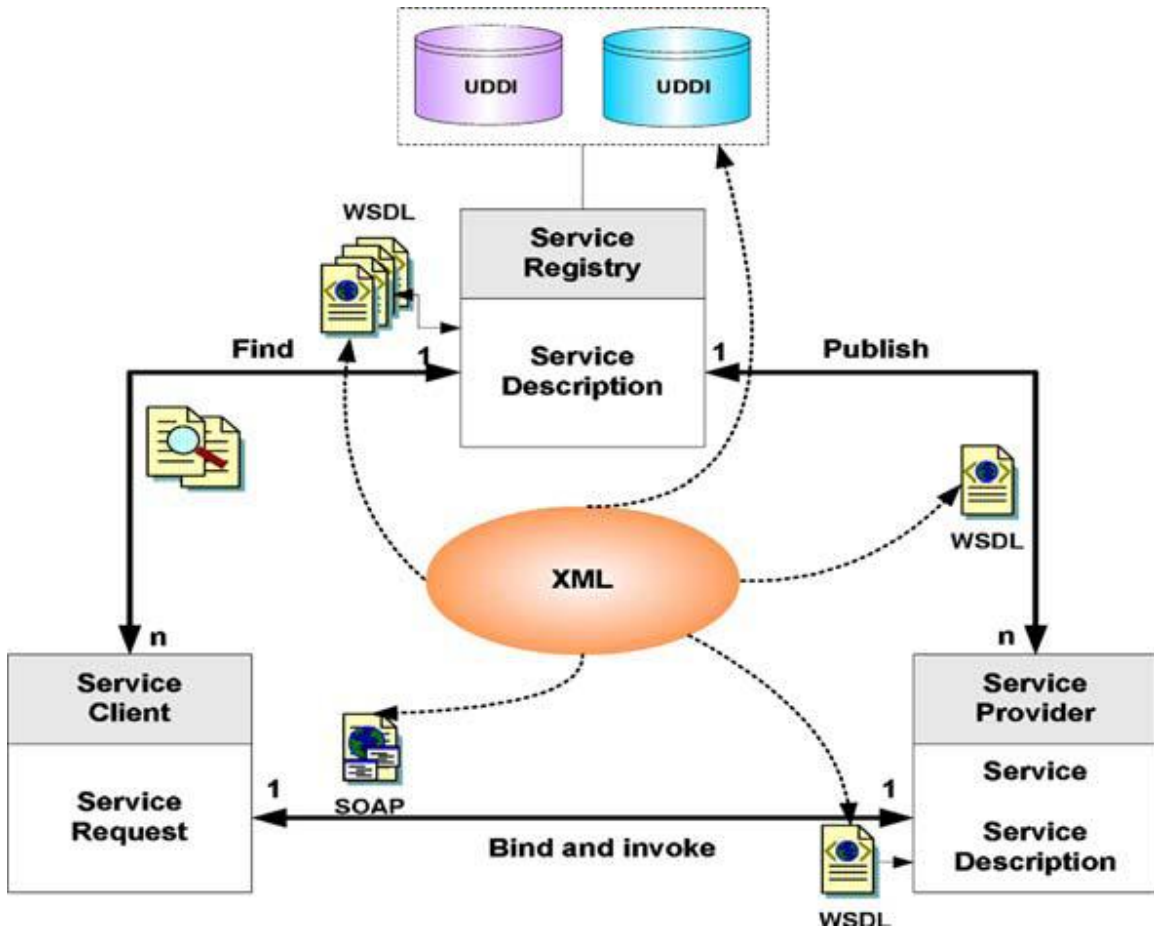


Figure 2.8 W3C Web Services reference model

Now that the different building blocks of web services is discussed, Figure 2.8 [76] shows that the existing technologies such as SOAP for Web services communication, WSDL for web services interface description, and UDDI for web services publishing and searching and so on are all based on XML centric approach to web services and can implement loosely coupled distributed web applications across different platforms [55].

Web services interact with each other by exchanging messages encoded using XML. In its basic form, the web service architecture consists of a simple RPC model in which a client invokes operations exported by a service provider using SOAP, a standard communication protocol for transmitting XML messages. Each web service publishes its invocation interface (for example, network address, ports, operations provided, and the expected XML message formats to invoke the service) using the WSDL. The WSDL specification serves as a contract between the client and the server that (in part) defines the valid interactions [56].

2.7 Building and developing SOA applications / Web services

To successfully build and deploy a distributed SOA, there are four primary aspects to be addressed

- i) Service enablement: Each discrete application needs to be exposed as a service.
- ii) Service orchestration: Distributed services need to be configured and orchestrated in a unified and clearly defined distributed process
- iii) Deployment Emphasis: should be shifted from test to the production environment, addressing security, reliability and scalability concerns.
- iv) Management Services: must be audited, maintained and reconfigured. The latter requirements require that corresponding changes in processes must be made without rewriting the services or underlying application.

Services can be programmed using application development tools like (Microsoft.NET, Borland JBuilder, or BEA WebLogic Workshop) which allow new or existing distributed applications to be exposed as web services. Technologies like J2EE Connector Architecture (JCA) may also be used to create services by integrating packaged applications (like ERP systems), which would then be exposed as web services [46].

Now that web services and the technology behind them have been discussed and understood. The next chapter discusses about a case study of how a standalone data mining application is converted into a web service, such that it can be invoked by any client who requires data mining services, all that the client has to do is to pass the URL of the file which is to be mined. This would save memory on the client's machine and if it is required to extend the features of the data mining application it can be done only at the server end without affecting the client application.

Chapter 3: Case study of converting a desktop data mining application into a web service using Weka

3.1 Introduction

Data mining is the process of sifting through large volumes of data, analyzing data from different perspectives and summarizing it into useful information. One of the widely used desktop applications for data mining is the Weka tool which is nothing but a collection of machine learning algorithms implemented in Java and open sourced under the General Public License (GPL). A web service is a software system designed to support interoperable machine to machine interaction over a network using SOAP messages. Unlike a desktop application, a web service is easy to upgrade, deliver and access and does not occupy any memory on the system. Weka software requires about 20MB of system memory along with memory space required for storing the tables. Another benefit is when the desktop application is installed on the desktop the entire suite of algorithms will have to be installed, though most of these algorithms may never be used. Unlike this a web service can invoke only the required mining algorithms. Keeping in mind the advantages of a web service over a desktop application, this chapter demonstrates how this Java based desktop data mining application can be implemented as a web service to support data mining across the internet.

3.2 Overview of work done

The open source code available on downloading the Weka 3.6 serves as the basis for converting the desktop application into a web service. Netbeans IDE 6 has been used to create the web service. Three data mining tasks are considered as case studies and for each of these tasks a data mining algorithm has been selected. The work is divided into different phases.

Phase 1: Libraries available in the open source code of Weka is used in the Java code

of data mining.

Phase 2: Each of these data mining codes is converted into a web service.

Phase 3: Clients are developed to consume each of these three web services.

3.3 Java Data Mining (JDM API)

Is the first attempt to create a standard Java API to access data mining tools from Java applications. It is similar to JDBC for Java applications. JDBC is an API which allows Java applications to interact with databases similarly JDM is an API which allows Java API to interact with data mining tools.

JDM uses packages to group together similar classes which form a package. Not all vendors will support all functions, algorithms or features. Java packages provide an effective way to modularize the API by mining functions and algorithms so that vendors can choose packages that they want to implement [75].

3.3.1 Weka for Data Mining

Weka (Waikato Environment for Knowledge Analysis) is an ensemble of data mining algorithms written in Java. These algorithms can either be applied directly to a dataset using the Weka explorer or called from the modified Java code. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization and can be used to develop new machine learning schemes [71].

Weka provides different environments like the Simple CLI (command-line interface) for direct execution of Weka commands, Explorer for exploring data with Weka and Experimenter for performing experiments and conducting statistical tests between learning schemes [65]. It is a desktop application and needs to be downloaded from the official web site. It requires that a specific Java version be installed on the system and is supported by Windows, MAC and Linux operating systems. The datasets for Weka should also be saved on the system

and formatted according to the ARFF format. Converters included in WEKA can convert other file formats like Comma Separated Vector (CSV) to Active Resource File Format (ARFF) [63].

3.4 Need for Web Service

So far data mining applications which are run from the desktop are considered. The user has to install the data mining engine in Oracle or the data mining software like WEKA or RAPID MINER and then solve the data mining tasks. This not only occupies a lot of memory on the system for storing the data repository and the software but is also not easily accessible from a system other than the system on which the software is installed. When WEKA is downloaded as a standalone application it requires 20MB of memory in contrast the size of the client code of a web service application would require about 700KB of memory. Also huge volumes of data available across the net cannot be mined successfully from the desktop.

Web services are easy and inexpensive to deliver, upgrade and access. So if a data mining application is converted into a web application, the data mining application can be hosted and accessed from any client in the world. Any modifications can be made to the code without the need of downloading the updated code. Also the data on the net can be mined without the need to download the application, only the file URL needs to be sent to the web service. Then depending on the algorithm used in the web enabled data mining application, the application will return the results of the mined data. It also helps in easy interaction of the web service with large corporations who need to mine their data and get back the result anytime anywhere [69]. Web services can be a boon for any organization which requires analyzing large volumes of data. Organizations which use data mining for only specific tasks would have to spend a considerable amount of time performing time consuming data analysis that is required in data mining, instead it would be worthwhile for the company to concentrate on their competencies and allow other groups to perform tasks like

data mining and whenever these special services are required they can invoke these services through web services and get back the response they require [57].

3.5 Weka Web Service Creation

The libraries used in the open source code of Weka is considered to implement certain data mining algorithms of J48 classification, EM clustering and text classification as web services. Clients are created to invoke these applications.

3.5.1 Convert J48 Classifier into Web Service

The J48 classifier algorithm explained in 1.5.3.1 is applied to the training data. The code for the above as a web service with the necessary comments is as shown in Appendix A .The web service makes use of the method 'execute' which takes as an input a string parameter specifying the name of classifier algorithm, the filter and also the file location of the dataset. It then trains the iris dataset using the J48 decision tree algorithm and outputs the result as shown in 3.6.1.

3.5.2 Convert EM Clusterer into Web Service

The EM Clustering algorithm explained in 1.5.3.2 is applied to the training data. The code for the above as a web service with the necessary comments is as shown in Appendix B. Here the web service makes use of the method 'execute' whose input parameter is the file URL of the dataset. It then trains the weather dataset using the EM Clustering algorithm and outputs the result as shown in 3.6.2.

3.5.3 Convert Text Classifier into Web Service

The IBK Lazy algorithm is explained in 1.5.3.3. The code for converting the algorithm to a web service for text classification with the necessary comments is as shown in Appendix C. Here the web service makes use of the 'execute' method which does not have any input parameters. The training and testing dataset are stored as string arrays and are classified into 'spam and 'no spam' by making use of the Lazy IBk algorithm and the results are as shown in 3.6.3.

3.6 Verification of Output

The output generated by the three sample codes of web services described above is found to be same as that of the java data mining code for the desktop application. This shows that the datasets provided as inputs are correctly mined by the above given sample web services with the same precision and accuracy.

The input, output on testing these three web services with the SOAP request and response messages, the WSDL document and the code for the client is given in the sections below.

3.6.1 J48 Classifier as Web Service

The training data set used here is the IRIS DATASET and the location of the file iris.arff is given as input to the code along with the string parameter specifying the name of the algorithm. The IRIS dataset is a table with 150 records describing four characteristics of flower petals i.e. sepal length, sepal width, petal length, petal width and the categorization of the flower.

Output of the web service with its corresponding SOAP request and SOAP response is as shown below. The input to the execute method indicates that a J48 classifier is used and the dataset is the iris.arff file available in the local directory. When required to be invoked as a web service, the path of the file need not be specified only the URL of the specified file has to be supplied to the service.

execute Method invocation

Method parameter(s)	
Type	Value
java.lang.String	CLASSIFIER weka.classifiers.trees.J48 -U FILTER weka.filters.unsupervised.instance.Randomize DATASET C:/Users/s1/Documents/NetBeansProjects/JavaApp/build/classes/ iris.arff

Listing 3.1 Output obtained on running the J48 Classifier

Method returned on running the application

```
java.lang.String : "Weka - Demo ===== Classifier...:
weka.classifiers.trees.J48 -U -M 2 Filter.....:
weka.filters.unsupervised.instance.Randomize -S 42 Training file:
C:/Users/s1/Documents/NetBeansProjects/JavaApp/build/classes/iris.arff J48
unpruned tree ----- petalwidth <= 0.6: Iris-setosa (50.0) petalwidth > 0.6
| petalwidth <= 1.7 | | petallength <= 4.9: Iris-versicolor (48.0/1.0) | | petallength >
4.9 | | | petalwidth <= 1.5: Iris-virginica (3.0) | | | petalwidth > 1.5: Iris-versicolor
(3.0/1.0) | petalwidth > 1.7: Iris-virginica (46.0/1.0) Number of Leaves : 5 Size of
the tree : 9 Correctly Classified Instances 142 94.6667 % Incorrectly Classified
Instances 8 5.3333 % Kappa statistic 0.92 Mean absolute error 0.043 Root mean
squared error 0.1854 Relative absolute error 9.6778 % Root relative squared
error 39.3217 % Total Number of Instances 150 == Confusion Matrix == a b c
<-- classified as 49 1 0 | a = Iris-setosa 0 46 4 | b = Iris-versicolor 0 3 47 | c = Iris-
virginica == Detailed Accuracy By Class == TP Rate FP Rate Precision Recall
F-Measure Class 0.98 0 1 0.98 0.99 Iris-setosa 0.92 0.04 0.92 0.92 0.92 Iris-
versicolor 0.94 0.04 0.922 0.94 0.931 Iris-virginica "
```

Listing 3.2 SOAP Request for J48 classifier

SOAP request indicates that the classification algorithm is invoked.

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:execute xmlns:ns2="http://demo1/">
      <input>CLASSIFIER weka.classifiers.trees.J48 -U FILTER
weka.filters.unsupervised.instance.Randomize DATASET
C:/Users/s1/Documents/NetBeansProjects/JavaApp/build/classes/iris.arff</input>
    </ns2:execute>
  </S:Body>
</S:Envelope>
```

Listing 3.3 SOAP Response for J48 classifier

SOAP Response- Shows the mined data generated by the web service.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
```

```
<S:Body>
  <ns2:executeResponse xmlns:ns2="http://demo1/">
    <return>Weka - Demo
```

=====

```
Classifier...: weka.classifiers.trees.J48 -U -M 2
Filter.....: weka.filters.unsupervised.instance.Randomize -S 42
Training file:
C:/Users/s1/Documents/NetBeansProjects/JavaApp/build/classes/iris.arff
```

J48 unpruned tree

```
petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
| petalwidth <= 1.7
| | petallength <= 4.9: Iris-versicolor (48.0/1.0)
| | petallength > 4.9
| | | petalwidth <= 1.5: Iris-virginica (3.0)
| | | petalwidth > 1.5: Iris-versicolor (3.0/1.0)
| petalwidth > 1.7: Iris-virginica (46.0/1.0)
```

Number of Leaves : 5

Size of the tree : 9

Correctly Classified Instances	142	94.6667 %
Incorrectly Classified Instances	8	5.3333 %
Kappa statistic	0.92	
Mean absolute error	0.043	
Root mean squared error	0.1854	
Relative absolute error	9.6778 %	
Root relative squared error	39.3217 %	
Total Number of Instances	150	

=== Confusion Matrix ===

```

a b c <-- classified as
49 1 0 | a = Iris-setosa
0 46 4 | b = Iris-versicolor
0 3 47 | c = Iris-virginica
```

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
---------	---------	-----------	--------	-----------	-------

0.98	0	1	0.98	0.99	Iris-setosa
0.92	0.04	0.92	0.92	0.92	Iris-versicolor
0.94	0.04	0.922	0.94	0.931	Iris-virginica

```

</return>
  </ns2:executeResponse>
</S:Body>
</S:Envelope>

```

Listing 3.4 WSDL generated for J48 classifier

```

<?xml version="1.0" encoding="UTF-8" ?>
<!--
Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI
2.1.3.1-hudson-417-SNAPSHOT.
-->
<!--
Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI
2.1.3.1-hudson-417-SNAPSHOT.
-->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://demo1/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://demo1/"
name="NewWebService12Service">
<types>
<xsd:schema>
<xsd:import namespace="http://demo1/"
schemaLocation="http://localhost:13699/WebApplication12/NewWebService
12Service?xsd=1" />
</xsd:schema>
</types>
<message name="execute">
<part name="parameters" element="tns:execute" />
</message>
<message name="executeResponse">
<part name="parameters" element="tns:executeResponse" />
</message>
<message name="Exception">
<part name="fault" element="tns:Exception" />
</message>
<portType name="NewWebService12">
<operation name="execute">
<input message="tns:execute" />

```

```

<output message="tns:executeResponse" />
<fault message="tns:Exception" name="Exception" />
  </operation>
</portType>
<binding name="NewWebService12PortBinding" type="tns:NewWebService12">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
    style="document" />
<operation name="execute">
  <soap:operation soapAction="" />
<input>
  <soap:body use="literal" />
</input>
<output>
  <soap:body use="literal" />
</output>
<fault name="Exception">
  <soap:fault name="Exception" use="literal" />
</fault>
</operation>
</binding>
<service name="NewWebService12Service">
<port name="NewWebService12Port"
  binding="tns:NewWebService12PortBinding">
  <soap:address
    location="http://localhost:13699/WebApplication12/NewWebService12Service" />
  </port>
</service>
</definitions>

```

Listing 3.5 Output for client code:

Servlet NewServlet at /WebApplicationClient12

```

Result = Weka - Demo ===== Classifier...: weka.classifiers.trees.J48 -U -
M 2 Filter.....: weka.filters.unsupervised.instance.Randomize -S 42 Training file:
C:/Users/s1/Documents/NetBeansProjects/JavaApp/build/classes/iris.arff J48
unpruned tree ----- petalwidth <= 0.6: Iris-setosa (50.0) petalwidth > 0.6
| petalwidth <= 1.7 | | petallength <= 4.9: Iris-versicolor (48.0/1.0) | | petallength >
4.9 | | | petalwidth <= 1.5: Iris-virginica (3.0) | | | petalwidth > 1.5: Iris-versicolor
(3.0/1.0) | petalwidth > 1.7: Iris-virginica (46.0/1.0) Number of Leaves : 5 Size of
the tree : 9 Correctly Classified Instances 142 94.6667 % Incorrectly Classified
Instances 8 5.3333 % Kappa statistic 0.92 Mean absolute error 0.043 Root mean
squared error 0.1854 Relative absolute error 9.6778 % Root relative squared
error 39.3217 % Total Number of Instances 150 === Confusion Matrix === a b c
<-- classified as 49 1 0 | a = Iris-setosa 0 46 4 | b = Iris-versicolor 0 3 47 | c = Iris-

```


virginica === Detailed Accuracy By Class === TP Rate FP Rate Precision Recall
 F-Measure Class 0.98 0 1 0.98 0.99 Iris-setosa 0.92 0.04 0.92 0.92 0.92 Iris-
 versicolor 0.94 0.04 0.922 0.94 0.931 Iris-virginica

3.6.2 EM Clusterer as Web Service

The training data set used here is the WEATHER DATASET and the location of the file weather.arff is given as input to the code. Output of the web service and its corresponding SOAP request and SOAP response is given below:

execute Method invocation indicates that weather.arff file is provided as input to the EM clustering algorithm.

Method parameter(s)	
Type	Value
java.lang.String	C:/Users/s1/Documents/NetBeansProjects/ClusteringDemo/build/classes/weather.arff

Listing 3.6 Method returned on invoking the clustering algorithm

```
java.lang.String : "Weka - Demo ===== --> normal EM == Number of
clusters selected by cross validation: 1 Cluster: 0 Prior probability: 1
Attribute: outlook Discrete Estimator. Counts = 6 5 6 (Total = 17) Attribute:
temperature Normal Distribution. Mean = 73.5714 StdDev = 6.3326 Attribute:
humidity Normal Distribution. Mean = 81.6429 StdDev = 9.9111 Attribute:
windy Discrete Estimator. Counts = 7 9 (Total = 16) Attribute: play Discrete
Estimator. Counts = 10 6 (Total = 16) === Clustering stats for training data
=== Clustered Instances 0 14 (100%) Log likelihood: -9.4063 --> manual # of
clusters: 1 --> density (CV) # of clusters: 0 "
```

Listing 3.7 SOAP Request for clustering algorithm

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:execute xmlns:ns2="http://demo/">
```

```
<input>C:/Users/s1/Documents/NetBeansProjects/ClusteringDemo/build/classes/
weather.arff</input>
  </ns2:execute>
</S:Body>
</S:Envelope>
```

Listing 3.8 SOAP Response for clustering algorithm

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:executeResponse xmlns:ns2="http://demo/">
      <return>Weka - Demo
```

```
=====
```

```
--> normal
```

```
EM
```

```
==
```

```
Number of clusters selected by cross validation: 1
```

```
Cluster: 0 Prior probability: 1
```

```
Attribute: outlook
```

```
Discrete Estimator. Counts = 6 5 6 (Total = 17)
```

```
Attribute: temperature
```

```
Normal Distribution. Mean = 73.5714 StdDev = 6.3326
```

```
Attribute: humidity
```

```
Normal Distribution. Mean = 81.6429 StdDev = 9.9111
```

```
Attribute: windy
```

```
Discrete Estimator. Counts = 7 9 (Total = 16)
```

```
Attribute: play
```

```
Discrete Estimator. Counts = 10 6 (Total = 16)
```

```
=== Clustering stats for training data ===
```

```
Clustered Instances
```

```
0 14 (100%)
```

```
Log likelihood: -9.4063
```

```
--> manual
```

```
# of clusters: 1
```

```
--> density (CV)
```

```
# of clusters: 0
```

```
</return>
```

```
    </ns2:executeResponse>
  </S:Body>
</S:Envelope>
```

Listing 3.9 WSDL generated for clustering algorithm

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI
 2.1.3.1-hudson-417-SNAPSHOT.
-->
<!--
Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI
 2.1.3.1-hudson-417-SNAPSHOT.
-->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://demo/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://demo/"
  name="NewWebService2Service">
<types>
<xsd:schema>
  <xsd:import namespace="http://demo/"
    schemaLocation="http://localhost:13699/WebApplication4/NewWebService2
    Service?xsd=1" />
  </xsd:schema>
</types>
<message name="execute">
  <part name="parameters" element="tns:execute" />
</message>
<message name="executeResponse">
  <part name="parameters" element="tns:executeResponse" />
</message>
<message name="Exception">
  <part name="fault" element="tns:Exception" />
</message>
<portType name="NewWebService2">
<operation name="execute">
  <input message="tns:execute" />
  <output message="tns:executeResponse" />
  <fault message="tns:Exception" name="Exception" />
```

```

    </operation>
  </portType>
<binding name="NewWebService2PortBinding" type="tns:NewWebService2">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
    style="document" />
  <operation name="execute">
    <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
  <fault name="Exception">
    <soap:fault name="Exception" use="literal" />
  </fault>
</operation>
</binding>
<service name="NewWebService2Service">
<port name="NewWebService2Port" binding="tns:NewWebService2PortBinding">
  <soap:address
    location="http://localhost:13699/WebApplication4/NewWebService2Service"
  />
</port>
</service>
</definitions>

```

Listing 3.10 Output for client code of clustering algorithm

Servlet NewServlet at /WebApplication5

Result = Weka - Demo ===== --> normal EM == Number of clusters selected by cross validation: 1 Cluster: 0 Prior probability: 1 Attribute: outlook Discrete Estimator. Counts = 6 5 6 (Total = 17) Attribute: temperature Normal Distribution. Mean = 73.5714 StdDev = 6.3326 Attribute: humidity Normal Distribution. Mean = 81.6429 StdDev = 9.9111 Attribute: windy Discrete Estimator. Counts = 7 9 (Total = 16) Attribute: play Discrete Estimator. Counts = 10 6 (Total = 16) === Clustering stats for training data === Clustered Instances 0 14 (100%) Log likelihood: -9.4063 --> manual # of clusters: 1 --> density (CV) # of clusters: 0

3.6.3 Text Classifier as Web Service

The testing and training data set used here are provided in the code itself.

Output of the above web service with **Lazy-IBk algorithm** and its corresponding SOAP request and SOAP response is given below:

execute Method invocation

Method parameter(s)

Type	Value
Java.lang.String	

Listing 3.11 Output obtained for text classifier

Method returned

```

java.lang.String : "dataset: DATA SET: @relation 'data set' @attribute text
string @attribute class {'no spam','?',spam} @data 'hey, buy this from
me!',spam 'do you want to buy?',spam 'I have a party tonight!','no spam'
'today it is a nice weather','no spam' 'you are best',spam 'I have a horse','no
spam' 'you are my friend','no spam' 'buy, buy, buy!',spam 'it is spring in the
air','no spam' 'do you want to come?','no spam' INFORMATION ABOUT THE
CLASSIFIER AND EVALUATION: classifier.toString(): IB1 instance-based
classifier using 1 nearest neighbour(s) for classification
evaluation.toSummaryString(title, false): Summary Correctly Classified
Instances 10 100 % Incorrectly Classified Instances 0 0 % Kappa statistic 1
Mean absolute error 0.1026 Root mean squared error 0.1088 Relative
absolute error 29.4118 % Root relative squared error 26.9191 % Total
Number of Instances 10 evaluation.toMatrixString(): === Confusion Matrix
=== a b c <-- classified as 6 0 0 | a = no spam 0 0 0 | b = ? 0 0 4 | c = spam
evaluation.toClassDetailsString(): Details TP Rate FP Rate Precision Recall
F-Measure Class 1 0 1 1 1 no spam 0 0 0 0 0 ? 1 0 1 1 1 spam
evaluation.toCumulativeMarginDistribution: -1 0 0.768 100 CHECKING ALL
THE INSTANCES: Class values (in order): 'no spam' '?' 'spam' Testing:
'hey, buy this from me!' predicted: 'spam (2.0)' real class: 'spam (2.0)' ==>
OK!! Testing: 'do you want to buy?' predicted: 'spam (2.0)' real class: 'spam
(2.0)' ==> OK!! Testing: 'I have a party tonight!' predicted: 'no spam (0.0)'
real class: 'no spam (0.0)' ==> OK!! Testing: 'today it is a nice weather'
predicted: 'no spam (0.0)' real class: 'no spam (0.0)' ==> OK!! Testing: 'you
are best' predicted: 'spam (2.0)' real class: 'spam (2.0)' ==> OK!! Testing: 'I
have a horse' predicted: 'no spam (0.0)' real class: 'no spam (0.0)' ==> OK!!
Testing: 'you are my friend' predicted: 'no spam (0.0)' real class: 'no spam
(0.0)' ==> OK!! Testing: 'buy, buy, buy!' predicted: 'spam (2.0)' real class:
'spam (2.0)' ==> OK!! Testing: 'it is spring in the air' predicted: 'no spam
(0.0)' real class: 'no spam (0.0)' ==> OK!! Testing: 'do you want to come?'
predicted: 'no spam (0.0)' real class: 'no spam (0.0)' ==> OK!! NEW CASES

```

CHECKING ALL THE INSTANCES: Class values (in order): 'no spam' '?' 'spam' Testing: 'you want to buy from me?' predicted: 'spam (2.0)' real class: '?' (1.0)' ==> NOT OK!! Testing: 'usually I run in stairs' predicted: 'spam (2.0)' real class: '?' (1.0)' ==> NOT OK!! Testing: 'buy it now!' predicted: 'spam (2.0)' real class: '?' (1.0)' ==> NOT OK!! Testing: 'buy, buy, buy!' predicted: 'spam (2.0)' real class: '?' (1.0)' ==> NOT OK!! Testing: 'you are the best, buy!' predicted: 'spam (2.0)' real class: '?' (1.0)' ==> NOT OK!! Testing: 'it is spring in the air' predicted: 'no spam (0.0)' real class: '?' (1.0)' ==> NOT OK!! "

Listing 3.12 SOAP Request for text classifier

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:execute xmlns:ns2="http://demo/">
  </S:Body>
</S:Envelope>
```

Listing 3.13 SOAP Response for text classifier

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:executeResponse xmlns:ns2="http://demo/">
      <return>dataset:
```

DATA SET:

@relation 'data set'

@attribute text string

@attribute class {'no spam','?',spam}

@data

'hey, buy this from me!',spam

'do you want to buy?',spam

'I have a party tonight!','no spam'

'today it is a nice weather','no spam'

'you are best',spam

'I have a horse','no spam'

'you are my friend','no spam'

'buy, buy, buy!',spam

'it is spring in the air','no spam'

'do you want to come?','no spam'

INFORMATION ABOUT THE CLASSIFIER AND EVALUATION:

classifier.toString():

IB1 instance-based classifier

using 1 nearest neighbour(s) for classification

evaluation.toSummaryString(title, false):

Summary

Correctly Classified Instances 10 100 %

Incorrectly Classified Instances 0 0 %

Kappa statistic 1

Mean absolute error 0.1026

Root mean squared error 0.1088

Relative absolute error 29.4118 %

Root relative squared error 26.9191 %

Total Number of Instances 10

evaluation.toMatrixString():

=== Confusion Matrix ===

a b c <-- classified as

6 0 0 | a = no spam

0 0 0 | b = ?

0 0 4 | c = spam

evaluation.toClassDetailsString():

Details

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
---------	---------	-----------	--------	-----------	-------

1	0	1	1	1	no spam
---	---	---	---	---	---------

0	0	0	0	0	?
---	---	---	---	---	---

1	0	1	1	1	spam
---	---	---	---	---	------

evaluation.toCumulativeMarginDistribution:

-1 0

0.768 100

CHECKING ALL THE INSTANCES:

Class values (in order): 'no spam' '?' 'spam'

Testing: 'hey, buy this from me!'

predicted: 'spam (2.0)' real class: 'spam (2.0)' ==> OK!!

Testing: 'do you want to buy?'
predicted: 'spam (2.0)' real class: 'spam (2.0)' ==> OK!!

Testing: 'I have a party tonight!'
predicted: 'no spam (0.0)' real class: 'no spam (0.0)' ==> OK!!

Testing: 'today it is a nice weather'
predicted: 'no spam (0.0)' real class: 'no spam (0.0)' ==> OK!!

Testing: 'you are best'
predicted: 'spam (2.0)' real class: 'spam (2.0)' ==> OK!!

Testing: 'I have a horse'
predicted: 'no spam (0.0)' real class: 'no spam (0.0)' ==> OK!!

Testing: 'you are my friend'
predicted: 'no spam (0.0)' real class: 'no spam (0.0)' ==> OK!!

Testing: 'buy, buy, buy!'
predicted: 'spam (2.0)' real class: 'spam (2.0)' ==> OK!!

Testing: 'it is spring in the air'
predicted: 'no spam (0.0)' real class: 'no spam (0.0)' ==> OK!!

Testing: 'do you want to come?'
predicted: 'no spam (0.0)' real class: 'no spam (0.0)' ==> OK!!

NEW CASES

CHECKING ALL THE INSTANCES:

Class values (in order): 'no spam' '?' 'spam'

Testing: 'you want to buy from me?'
predicted: 'spam (2.0)' real class: '?' (1.0)' ==> NOT OK!!

Testing: 'usually I run in stairs'
predicted: 'spam (2.0)' real class: '?' (1.0)' ==> NOT OK!!

Testing: 'buy it now!'

predicted: 'spam (2.0)' real class: '? (1.0)' ==> NOT OK!!

Testing: 'buy, buy, buy!'

predicted: 'spam (2.0)' real class: '? (1.0)' ==> NOT OK!!

Testing: 'you are the best, buy!'

predicted: 'spam (2.0)' real class: '? (1.0)' ==> NOT OK!!

Testing: 'it is spring in the air'

predicted: 'no spam (0.0)' real class: '? (1.0)' ==> NOT OK!!

```
</return>
  </ns2:executeResponse>
</S:Body>
</S:Envelope>
```

Listing 3.14 WSDL generated for text classifier

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI
2.1.3.1-hudson-417-SNAPSHOT.
-->
<!--
Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI
2.1.3.1-hudson-417-SNAPSHOT.
-->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://demo/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://demo/"
name="TextClassifierServiceService">
<types>
<xsd:schema>
<xsd:import namespace="http://demo/"
schemaLocation="http://localhost:13699/TextClassifierService/TextClassifier
ServiceService?xsd=1" />
</xsd:schema>
</types>
<message name="execute">
<part name="parameters" element="tns:execute" />
</message>
<message name="executeResponse">
```

```

    <part name="parameters" element="tns:executeResponse" />
  </message>
<portType name="TextClassifierService">
  <operation name="execute">
    <input message="tns:execute" />
    <output message="tns:executeResponse" />
  </operation>
</portType>
<binding name="TextClassifierServicePortBinding"
  type="tns:TextClassifierService">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
    style="document" />
  <operation name="execute">
    <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
</operation>
</binding>
<service name="TextClassifierServiceService">
  <port name="TextClassifierServicePort"
    binding="tns:TextClassifierServicePortBinding">
    <soap:address
      location="http://localhost:13699/TextClassifierService/TextClassifierService
        Service" />
    </port>
  </service>
</definitions>

```

Listing 3.15 Output for the client code of text classifier

Servlet NewServlet at /TextClient

```

Result = dataset: DATA SET: @relation 'data set' @attribute text string
@attribute class {'no spam','?',spam} @data 'hey, buy this from me!',spam 'do
you want to buy?',spam 'I have a party tonight!','no spam' 'today it is a nice
weather','no spam' 'you are best',spam 'I have a horse','no spam' 'you are my
friend','no spam' 'buy, buy, buy!',spam 'it is spring in the air','no spam' 'do you
want to come?','no spam' INFORMATION ABOUT THE CLASSIFIER AND
EVALUATION: classifier.toString(): IB1 instance-based classifier using 1 nearest
neighbour(s) for classification evaluation.toSummaryString(title, false): Summary
Correctly Classified Instances 10 100 % Incorrectly Classified Instances 0 0 %
Kappa statistic 1 Mean absolute error 0.1026 Root mean squared error 0.1088
Relative absolute error 29.4118 % Root relative squared error 26.9191 % Total

```

Number of Instances 10 evaluation.toMatrixString(): === Confusion Matrix === a
b c <- classified as 6 0 0 | a = no spam 0 0 0 | b = ? 0 0 4 | c = spam
evaluation.toClassDetailsString(): Details TP Rate FP Rate Precision Recall F-
Measure Class 1 0 1 1 1 no spam 0 0 0 0 0 ? 1 0 1 1 1 spam
evaluation.toCumulativeMarginDistribution: -1 0 0.768 100 CHECKING ALL THE
INSTANCES: Class values (in order): 'no spam' '?' 'spam' Testing: 'hey, buy this
from me!' predicted: 'spam (2.0)' real class: 'spam (2.0)' ==> OK!! Testing: 'do
you want to buy?' predicted: 'spam (2.0)' real class: 'spam (2.0)' ==> OK!!
Testing: 'I have a party tonight!' predicted: 'no spam (0.0)' real class: 'no spam
(0.0)' ==> OK!! Testing: 'today it is a nice weather' predicted: 'no spam (0.0)' real
class: 'no spam (0.0)' ==> OK!! Testing: 'you are best' predicted: 'spam (2.0)' real
class: 'spam (2.0)' ==> OK!! Testing: 'I have a horse' predicted: 'no spam (0.0)'
real class: 'no spam (0.0)' ==> OK!! Testing: 'you are my friend' predicted: 'no
spam (0.0)' real class: 'no spam (0.0)' ==> OK!! Testing: 'buy, buy, buy!' predicted:
'spam (2.0)' real class: 'spam (2.0)' ==> OK!! Testing: 'it is spring in the
air' predicted: 'no spam (0.0)' real class: 'no spam (0.0)' ==> OK!! Testing: 'do
you want to come?' predicted: 'no spam (0.0)' real class: 'no spam (0.0)' ==> OK!!
NEW CASES CHECKING ALL THE INSTANCES: Class values (in order): 'no
spam' '?' 'spam' Testing: 'you want to buy from me?' predicted: 'spam (2.0)' real
class: '?' (1.0)' ==> NOT OK!! Testing: 'usually I run in stairs' predicted: 'spam
(2.0)' real class: '?' (1.0)' ==> NOT OK!! Testing: 'buy it now!' predicted: 'spam
(2.0)' real class: '?' (1.0)' ==> NOT OK!! Testing: 'buy, buy, buy!' predicted: 'spam
(2.0)' real class: '?' (1.0)' ==> NOT OK!! Testing: 'you are the best, buy!' predicted:
'spam (2.0)' real class: '?' (1.0)' ==> NOT OK!! Testing: 'it is spring in
the air' predicted: 'no spam (0.0)' real class: '?' (1.0)' ==> NOT OK!!

3.7 Conclusion and future extension of work

Weka is a widely used tool for data mining. It provides an open source code that can be used for machine learning. Not only that the interface provided by the software can also be used for mining large datasets. However keeping in mind the benefits of a web service as compared to a desktop application, it is demonstrated how Weka (or any other software) can be ported as a web service. The results obtained from the web application are the same as that of the desktop application, thereby highlighting the fact that the data can be mined by the Weka web service in a similar fashion as can be mined by Weka desktop application, with easier access to large datasets and less memory consumption of the system. This work has been published in World Academy of Science, Engineering and Technology, International Conference, ICCESSE/'10 held in Rome, Italy April 28-30, 2010.

Data mining applications involve fixed algorithms like association rules, Bayesian learning or clustering it is possible to scale up these regular algorithms such that they work in parallel in distributed data mining which is a fast developing extension of data mining [60]. Distributed data mining also has the advantage that all the data is not available in one centralized site thus increasing the reliability of the site [61]. Another important aspect which could be added to this work is the inclusion of security during the transfer of the SOAP request or SOAP response. In chapter 6 the details of web service security are looked at and discussed.

Chapter 4: Comparison of J2EE and .NET development platforms used in web services and the tests conducted on the commonly seen interoperability issues.

4.1 Introduction

Web services are designed to bring together applications from geographically distributed and heterogeneous environment and provide interoperability among them [76]. In this chapter, a comparison between the two choices that businesses have for building XML-based web services: the Java 2 Platform, Enterprise Edition (J2EE), built by Sun Microsystems and Microsoft.NET, built by Microsoft Corporation is considered. Some of the features on which the comparison is made include tools available for web service generation, support for UDDI registries, and support for RPC and message driven web services and since security is a major concern in web services the security offered in both the platforms is looked into.

Because of the heterogeneous nature of the internet it is mandatory that a J2EE client should be able to invoke a .Net web service or the other way round. But in reality the feasibility of the above is not always possible. The state of art in web services interoperability issues which have been extensively discussed.

4.2 Definition of Interoperable Web Services

Interoperability is defined by IEEE as the ability of two or more systems or components to exchange information and to use the exchanged information [77]. The goal of interoperability in web services is to provide seamless and automatic connections from one software application to another [31].

4.3 Why Interoperability?

Once achieved, the ability to seamlessly integrate Java Enterprise Edition (Java EE) and .NET environments will help developers create applications on a diverse

range of operating systems including the Solaris operating system, Windows and Linux, that can co-exist and interoperate across heterogeneous computing environments. Seamless integration will also enable greater collaboration for enterprises, by allowing them to leverage a larger ecosystem of partners in application development. Additionally, interoperability between the two platforms will help pave the way for greater adoption of web services and SOA based application development by reducing the associated cost, complexity and risk.

4.3.1 Architecture layers for interoperability

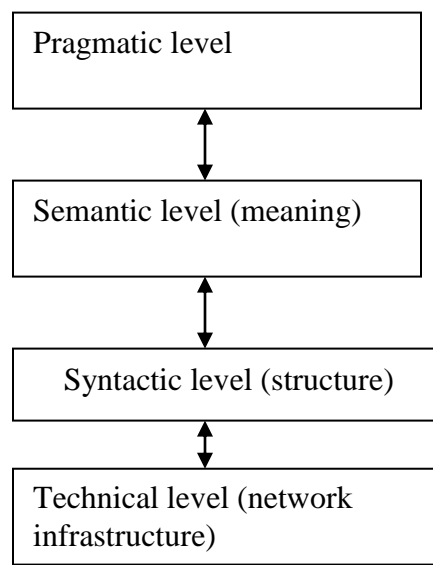


Figure 4.1 Architecture layers of Interoperability

As shown in Figure 4.1 [33] the different levels at which interoperability can occur are

- i) Technical level – includes transport, protocol, security, reliability and authentication features.
- ii) Syntax level – ensures that data has the same structure and follows the same format.

- iii) Semantic level – ensures that communicating systems interpret the information in the same way, this is very important to be achieved in a heterogenous environment [24].
- iv) Pragmatic level is split into organizational level and business process. Organizational level consists of responsibilities, agreements etc. Process level gives importance to process driven integration.

4.4 Examples of Interoperability

Example 1

In a real life scenario it is seen that organizations comfortable in .NET would develop their web service using a .NET platform. Now the client which invokes the web service could be a ASP.Net client, a perl client or a Java client, similarly organizations comfortable in Java could have a web service developed in Java and this service could be invoked by a client developed on any programming language platform. As a simple example a web service for currency conversion which is developed in Java with Axis which is consumed by an ASP.Net client created in C# is considered.

Example 2

Multiple web services interacting with each other is considered, as an example a supply chain application where a supplier, warehouse manufacturer and the retail stores interact with each other. Developers could be using a Java2 Enterprise Edition (J2EE) based web service deployed on Oracle 9i application server with web services deployed on other platforms such as .Net and J2EE application servers from other vendors.

Example 3

A purchase order scenario is considered. In this scenario, a potential buyer (that is, the "Customer") retrieves a catalog of products offered by a particular buyer and selects which products will be purchased in what quantities. When a purchase is complete, an order is sent to the supplier with the types and amounts of products that the buyer requested. The supplier will then check if the requested products are available to be shipped to the customer. This is achieved

by querying the warehouse. If there is insufficient stock to fulfill the order, the order's status is set appropriately and an error is returned. If the available stock is sufficient, the supplier will then check the customer's credit to execute the purchase. Each buyer is associated with a particular bank that can provide information about the account status, as well as deduct the required funds from that account. Finally, assuming that the bank account status is good, the supplier sends a request to the warehouse to ship the order. The customer, in the meantime, can check on the status of a particular order with the supplier, or retrieve an invoice for a confirmed shipment. Each of the participants in the application scenario described above, namely customer, supplier, warehouse and bank, is implemented as a web service. These services can run on the same machine or on different machines, with different implementations, showing the value of web services technology in a heterogeneous environment.

Which implementation of a role is used when running the application is determined by a set of XML files that describe combinations, for example, a particular warehouse service with a particular supplier. This way, any permutation of role implementations by the different vendors can be configured. The XML configuration files are kept in a central place and, typically, cannot be changed [79].

4.5 The two common platforms used for web service development

4.5.1 J2EE

The Java 2 Platform Enterprise Edition (J2EE) is a set of coordinated specifications and practices that together enable solutions for developing, deploying, and managing multi-tier server-centric applications. Multi-tier server applications are those, where a client's request to a server generates request to other servers that are connected together through a backbone network. Building on the Java 2 Platform Standard Edition (J2SE), the J2EE platform adds the capabilities necessary to provide a complete, stable, secure, and fast Java

platform to the enterprise level. It provides value by significantly reducing the cost and complexity of developing and deploying multi-tier solutions, resulting in services that can be rapidly deployed and easily enhanced [80].

The J2EE camp's goal is to give customers choice of vendor products and tools, and to encourage best-of breed products to emerge through competition. To secure buy-in, Sun collaborated with other vendors of eBusiness platforms, such as BEA, IBM, and Oracle, in defining J2EE. Sun then initiated the Java Community Process (JCP) to solicit new ideas to improve J2EE over time [81].

The Java 2 Platform, Enterprise Edition adds full support for Enterprise JavaBeans components, Java Servlets API, JavaServer Pages and XML technology. The J2EE standard includes complete specifications and compliance tests to ensure portability of applications across the wide range of existing enterprise systems capable of supporting the J2EE platform. In addition, the J2EE specification now ensures web services interoperability through support for the WS-I Basic Profile [80].

4.5.2 .NET

The Microsoft .NET Framework is a software component that is a part of modern Microsoft Windows operating systems. It provides a large body of pre-coded solutions to common program requirements. The .NET framework is included with Windows Server 2003, Windows Server 2008 and Windows Vista, and can be installed on most older versions of Windows. Microsoft.NET is a product suite that enables organizations to build smart, enterprise-class web services. .NET is the Microsoft web services strategy to connect information, people, systems, and devices through software. .NET-connected solutions enable businesses to integrate their systems more rapidly and in a more agile manner and help them realize the promise of information anytime, anywhere, on any device.

4.5.3 Similarity and Differences between Java and .NET

Similarities

- i) Both technologies provide a number of API's that serve a common purpose e.g. IO, reflection, serialization, networking etc.
- ii) Both technologies support primitive data types, e.g. integer, float, boolean, double, long etc. However since .NET supports multiple languages primitive data types have been mapped to a specific class in the .NET framework.
- iii) Both technologies do not support multiple inheritances.
- iv) Applications written in both technologies get compiled to an intermediate language.
- v) Both technologies have a garbage collector for managing their resources. The garbage collector deletes resources once they go out of scope. Though the garbage collector performs the same function in both the technologies their approach is very different.

Differences

The differences in approach between .NET and J2EE and the technical challenges accompanying them are sometimes significant hindrances to interoperability. The following are some of the differences between .NET and J2EE.

- i) J2EE is a set of open standards not a product .NET on the other hand, is a product suite with some features built on standards and other features that extend standards.
- ii) .NET provides runtime support for SOAP and UDDI as native .NET protocols.
- iii) Integrated support is provided in .NET to build and deploy XML based web services. J2EE vendors must provide integration between J2EE products and an IDE offering.
- iv) .NET provides business process management and e-commerce capabilities. These capabilities may be provided in J2EE implementation but are not part of the standard.

- v) J2EE is focused on application portability and connectivity between platforms supporting Java..NET targets application integration using XML.

Application and backend integration approaches differ. Java uses JCA (Java Connector Architecture) to connect to specific systems and applications. Connections across disparate applications is through JMS..NET provides integration through several mechanisms, the host integration server 2000, COM Transaction Integrator (COM TI), Microsoft Messaging Queue (MSMQ) and Biztalk Server 2000 [45].

4.6 Comparative Analysis between J2EE and .NET

4.6.1 Tools for Web service generation in J2EE

The following are the most popular software solutions commercially available for implementing Web services in J2EE.

- i) BEA systems products – BEA WebLogic Server 7.0 provides the infrastructure solution for Web services supporting the standards and protocols of Web Services. The BEA Weblogic Integration server also enables complex web services to be deployed with transactional integrity, security and reliability while supporting the emerging ebXML and BTP standards.
- ii) Cape Clear Products – Provides Web Services infrastructure and product solutions such as CapeConnect and CapeStudio, which enable the development of web service solutions based on industry standards such as XML, SOAP, WSDL and UDDI. Cape Clear also enables business applications from diverse technologies such as Java, EJB, CORBA and Microsoft.NET. These components can be composed as web services over the Internet.
- iii) Oracle Products – Oracle's 9i release 2 application servers provides the J2EE based infrastructure solution for web services supporting web services standards including SOAP, WSDL and UDDI. It also has the ability to integrate with legacy systems.

- iv) SUN products – SUN has released JAX (Java API for XML) Pack which supports the XML technology, it has also released JWSDP (Java Web services Developer Pack) [47].

Tool for web services generation in .NET

Microsoft has always been a strong tools vendor, and that has not changed. As part of its launch of .NET, Microsoft released a beta version of the Visual Studio.NET integrated development environment. Visual Studio.NET supports all languages supported by earlier releases of Visual Studio with the notable exception of Java. In its place the IDE supports C#, Microsoft's new object-oriented programming language, which bears a remarkable resemblance to Java. Visual Studio.NET has some interesting productivity features including Web Forms, a web-based version of Win Forms, .NET's GUI component set. Visual Studio.NET enables developers to take advantage of .NET's support for cross-language inheritance.

It is concluded that Microsoft has the clear win when it comes to tools. While the functionality of the toolset provided by J2EE community as a whole supersedes the functionality of the tools provided by Microsoft, these tools are not 100% interoperable, because they do not originate from a single vendor. Much more low-level hacking is required to achieve business goals when working with a mixed toolkit, and no single tool is the clear choice, nor does any single tool compare with what Microsoft offers in Visual Studio.NET. Microsoft's single-vendor integration, the ease-of-use, and the super-cool wizards are good to have when building web services [82].

4.6.2 Web services support in J2EE

The future of eBusiness is undoubtedly web services. For organizations that are preparing for the future of web services, their underlying eBusiness architecture must have strong web services support.

Today, J2EE supports web services through the Java API for XML Parsing (JAXP). This API allows developers to perform any web service operation today through manually parsing XML documents. For example, JAXP can be used to perform operations with SOAP, UDDI, WSDL, and ebXML. Additional API's are also under development. These are convenience APIs to help developers perform web services operations more rapidly, such as connecting to business registries, transforming XML-to-Java and Java-to-XML, parsing WSDL documents, and performing messaging such as with ebXML.

A variety of J2EE-compatible 3rd party tools are available today that enable rapid development of web services [82].

Web services support in .NET

Microsoft.NET also enables organizations to build web services. The tools that ship with Microsoft.NET also offer rapid application development of web services, with automatic generation of web service wrappers to existing systems. It can perform operations using SOAP, UDDI, and SDL (the precursor to WSDL). Visual Studio.NET provides wizards that generate web services.

It is concluded that:

- i) J2EE can be used to develop and deploy web services today using JAXP which is not the ideal way to build web services, since it requires much manual intervention.
- ii) .NET can be used to develop web services today using the partial .NET platform eCollaboration model, which is based on the UDDI and SOAP standards. These standards are widely supported by more than 100 companies. Microsoft along with IBM and Ariba are the leaders in this area.

4.6.3 Portability of web services

A key difference between J2EE and .NET is that J2EE is platform-agnostic, running on a variety of hardware and operating systems, such as Win32, UNIX,

and mainframe systems. This portability is an absolute reality today because the Java Runtime Environment (JRE), on which J2EE is based, is available on any platform.

There is a second more debatable aspect of portability as well, J2EE is a standard, and so it supports a variety of implementations such as BEA, IBM, and Sun. The danger in an open standard such as J2EE is that if vendors are not held strictly to the standard, application portability is sacrificed. To help with the situation, Sun has built a J2EE compatibility test suite, which ensures that J2EE platforms comply with the standards, it ensures portability of applications.

By way of comparison, .NET only runs on Windows, its supported hardware, and the .NET environment. There is no portability at all. It should be noted that there have been hints that additional implementations of .NET will be available for other platforms.

4.6.4 System Cost

A wide variety of implementations based on J2EE architecture are available for purchase, with price varying dramatically, enabling a corporation to choose the platform that meets its budget and desired service level. The important point is that low-cost solutions can be obtained with both Microsoft and J2EE architecture. Microsoft's solution has an aggressive price, whereas J2EE architecture allows the user to choose a service level. For example, J2EE can be used to develop a high-end, expensive solution (iPlanet running on Sun Solaris in an E-10000 server), or a low-end, inexpensive solution (jBoss running on Linux on a Cobalt RAQ server).

Therefore, it's recommend not to consider the price of the platform when selecting between J2EE, .NET, or any other platform, but rather consider the more important other factors [81].

4.6.5 Support for UDDI registry access support in J2EE

UDDI is several things

- A federated and open database
- A standard SOAP interface for querying and submitting information
- A repository that includes custom data for querying and finding web services

UDDI allows companies to register their electronic services – everything from an e-mail address for technical support to XML based web services for purchasing. They can do this from a web page by programmatically using the UDDI interface. This information is then replicated to IBM, Microsoft and several other database providers that run UDDI registries. The UDDI specification defines two XML based programming API's for communicating with the UDDI registry node the Inquiring API and the publishing API

Inquiring API: The inquiry API consists of XML messages defined using UDDI schemas, which can be used to locate information about a business such as the services a business offers and the technical specifications of these services.

A UDDI programmer uses the inquiry API to retrieve information stored in the registry, to retrieve information a registry user does not need to be authenticated. Some of the API functions used to find information include <find_business>, <find_service>, <find_binding> , <find_tmodel> to get further detailed information from the UDDI registry the API functions available are <get_business_detail>, <get_service-detail>, <get_tModelDetail>. Both .NET and Java use the same data structures for finding and querying the database.

Publishing API:

- i) Uses XML messages that can be used to create, update and delete the information found in the UDDI registry. To publish in the UDDI registry the user needs to be authenticated.

- ii) The publishing API defines functions that deal with authentication of the registry users that is needed to successfully execute the rest of the functions of this API.
- iii) XML messages constituting the UDDI programmer API's are defined using a UDDI XML schema, the XML messages are wrapped in a SOAP body element and sent as a HTTP post request to a UDDI registry.
- iv) The UDDI registry processes these SOAP messages and gets hold of the actual API function represented by the actual XML message, which further instructs the registry service to provide either publishing or inquiry services.
- v) Typically publishing access points use HTTPS because they need to be authenticated [80].

Microsoft and IBM jointly authored the WS-Inspection which is available in .NET for inspecting endpoints such as the root of a web server and finding out what services it offers. WS-Inspection is a simple XML grammar for gathering web services together. It has the ability to link to UDDI registries to give more information about the service offered. WSDL is also a document that describes the UDDI but WS-Inspection can create human readable documentation, such as HTML pages [83].

Support for UDDI registry access support in .NET

Microsoft offers UDDI client support through several tools including Visual Studio .NET, the Office XP Web Services Toolkit, and the UDDI SDK. Microsoft Visual Studio .NET (provides native support for UDDI services through the "Add Web Reference" feature, enabling developers to easily discover Web Services and other programmatic resources in UDDI Services for use in building dynamic applications. By navigating the `http://<machinename>/uddipublic/addwebreference`, a developer is presented with a user interface from which UDDI can be queried directly from Visual Studio

.NET. Microsoft Visual Basic for applications provides an add-in that supports UDDI discovery as well. This tool works slightly differently, in that the access point for UDDI services must be provided (that is, <http://<machinename>/uddipublic/inquire.asmx>)[84].

4.6.6 Support for RPC and Message Driven Web Services in J2EE

Java provides Java API for XML messaging (JAXM) which is an integral part of JWSDP which provides synchronous and asynchronous messaging capabilities in the web services environment and enables exchange of XML documents over the intranet or Internet. JAXM is an API framework based on the messaging protocols defined by SOAP 1.1 specifications and SOAP attachments. JAXM uses a standard messaging provider infrastructure and Java based API's to facilitate sending and receiving of XML based messages asynchronously in a web services environment. Core features of JAXM are as follows

- i) Has portable XML messaging applications
- ii) Has synchronous request/response and asynchronous (one-way) messaging.
- iii) Transmits and routes messages to many providers
- iv) Ensures message delivery through reliable messaging.
- v) Supports standard internet protocols like HTTP, SMTP, and FTP.

The role of JAX-RPC in Web Services

In a web service environment, JAX-RPC defines an API framework and a runtime environment for creating and executing XML based RPC's. The web services requestors invoke the service provider's methods and transmit parameters and receive return values as XML based requests and responses. The core features of JAX-RPC are as follows

- i) Providing API's for defining RPC based web services, hiding the underlying complexities of SOAP packaging and messaging.
- ii) Provides run time API's for invoking RPC based web services based on stubs and ties, dynamic proxy, and dynamic invocation [47].

A comparison of the key features of JAX-RPC and JAXM is provided in table 4.1 [47]

Table 4.1 Comparison of JAX-RPC with JAXM

SI No	JAX-RPC	JAXM
1	Synchronous RPC-based service interaction	Synchronous / Asynchronous message based service interaction
2	Message sent as XML based request and response	Message sent as document driven XML messages
3	Exposes internals to service requestors	Loosely coupled and does not expose internals to service requestors
4	Provides a variety of client invocation models	Does not provide a client-specific programming model
5	No reliability mechanisms	Provides guaranteed message delivery and reliability mechanism

Support for RPC and document centric web services in .NET

Microsoft provides messaging systems that allow store and forward communication, this type of communication is appropriate for one way communication like logging. Server based messaging products like MSMQ (Microsoft Message Queuing) provide rich features including transactions which complement rather than compete against web services [85]. XML-RPC.NET is a library for implementing XML-RPC Services and clients in the .NET environment.

Its features include:

- i) interface based definition of XML-RPC servers and clients
- ii) code generation of type-safe client proxies
- iii) support for .NET remoting on both client and server
- iv) ASP.NET web services which support both XML-RPC and SOAP

- v) client support for asynchronous calls
- vi) client support for various XML encodings and XML indentation styles (some other XML-RPC server implementations incorrectly only accept certain indentation styles)
- vii) built-in support for XML-RPC introspection API on server
- viii) dynamic generation of documentation page at URL of XML-RPC end-point
- ix) support for mapping XML-RPC method and struct member names to .NET compatible names
- x) support for Unicode XML-RPC strings in both client and server
- xi) support for optional struct members when mapping between .NET and XML-RPC [86].

4.6.7 Web services security support comparison in .NET and J2EE

Web services are widely used because of their ability to deliver integrated, interoperable solutions, web services can be potentially accessed by a complete stranger over the network hence without proper security infrastructure in place it would not be possible to adopt web services. A number of technologies and standards are being used to secure web services ensuring integrity, confidentiality and security. Some of the prominent technologies supported in XML include

- i) XML Encryption
- ii) XML Signature
- iii) Security Assertion Markup Language (SAML)
- iv) XML Access Control Markup Language (XACML)
- v) XML Key Management Services (XKMS) [47].

Netbeans an IDE available in J2EE offers the following methods for implementing security in Web services

- i) Mutual Certificates Security: Adds security via authentication and message protection that ensures integrity and confidentiality.

- ii) Transport Security (SSL): Is a point to point security mechanism that can be used for authentication, message integrity and confidentiality.
- iii) Message authentication over SSL: Attaches a cryptographically secured identity or authentication token with the message and uses SSL for confidentiality protection.
- iv) SAML authorization over SSL: Attaches an authorization token with the message and uses SSL for confidentiality protection.
- v) Endorsing Certificate: Uses symmetric key for integrity and confidentiality protection and uses an endorsing client certificate to augment the claims provided by the token associated with the message signature [87].

Web Service Security in .NET

Web Service Enhancement (WSE) 3.0 simplifies the development and deployment of secure Web services. It enables developers and administrators to apply security policies to web services running on the .NET Framework 2.0. Using WSE 3.0, web services communication can be signed and encrypted using Kerberos tickets, X.509 certificates and other custom binary and XML-based security tokens. In addition username/password credentials can be used for authentication purposes. An enhanced security model provides a policy-driven foundation for securing web services. WSE also supports the ability to establish a trust-issuing service for retrieval and validation of security tokens, as well as the ability to establish more efficient long-running secure communication via secure conversations [88].

Hence it is concluded that both platforms give sufficient support for security.

4.6.8 Support for Existing Systems

Most large corporations have existing code written in a variety of languages, and have a number of legacy systems, such as CICS/COBOL, C++, SAP R/3, and Siebel. It is very important that corporations be given an efficient path to preserve

and reuse these investments of a system. This legacy integration often is one of the most challenging tasks to overcome when building a web service.

There are several ways to achieve legacy integration using J2EE, including

- i) **Java Message Service (JMS)** to integrate with existing messaging systems
- ii) **CORBA** for interfacing with code written in other languages that may exist on remote machines.
- iii) **JNI** for loading native libraries and calling them locally.
- iv) **J2EE Connector Architecture (JCA)** is a specification for plugging in resource adapters that understand how to communicate with existing systems. If such adapters are not available, they can be written using an adapter. These adapters are reusable in any container that supports the JCA.

.NET also offers legacy integration through the Host Integration Server 2000.

- i) **COM Transaction Integrator (COM TI)** can be used for collaborating transactions across mainframe systems.
- ii) **Microsoft Message Queue (MSMQ)** can integrate with legacy systems built using IBM MQSeries.
- iii) **BizTalk Server 2000** can be used to integrate with systems based on B2B protocols, such as Electronic Data Interchange (EDI).

In conclusion, the legacy integration features offered by J2EE are superior to those offered by .NET. The JCA market is producing a marketplace of adapters that will greatly ease enterprise application integration.

4.7 Test Results of the platform developed for testing Interoperability issues

Results seen in implementation of Interoperability Issues

Web services exchange data by exchanging XML documents. As soon as data objects are pushed into the web service stack they are represented as XML documents. Thus the web service stack on the receiving end should know how to interpret the XML document sent by the sender. The XML Schema, which provides an outline of the XML document, helps the receiver to map the data which is represented in XML. But the implementation difference in the underlying technologies of J2EE and .NET results in different mappings between the schema and native data types on both the platforms. This may lead to information distortion and de-serialization failure [29]. To avoid interoperability issues the actual trend is to create web services that use simple data types i.e. atomic data types and arrays. More complex data types can only interoperate when both parties use the same underlying stack; here the purpose of using web services is lost [31].

Interoperability issues are tested by creating a reliable web service in Java using Netbeans IDE 6.5 and deploying it on Glassfish server (v2). A C# client in .NET and a Java web client are developed. In order to compare the performance of a Java web service and a Java client and a Java web service and a .NET client the client is made to invoke the web service and pass data to the web service the web service processes this data and sends it back to the client. The output for various cases like primitive data types, arrays with null elements, and complex data types are considered. The communication between the web service and the client is observed by the exchange of SOAP messages using the TCP Monitor.

4.7.1 An array with null elements

The XML representations of an array with null elements are different between .NET and Java. Consider a Java web service which returns an array with a null element. A Java client can correctly interpret the null string in an array. However,

a .NET client interprets the null string as a string of length zero or an empty string. Empty and null strings are completely different from each other in object oriented programming language [29]. The screen shots given below clearly indicate this fact when a array with three elements one of which is null is sent from the service provider to the client.

Output from Java and .NET clients

The screenshots in Figure 4.2 and 4.3 show the difference in the interpretation of null values by Java and .NET clients. The output of the Java client is Disha, null, Vinita. Hence, it is inferred that Java clients infer the null values correctly, whereas the .NET client displays null as an empty string and cannot deserialize null values correctly.

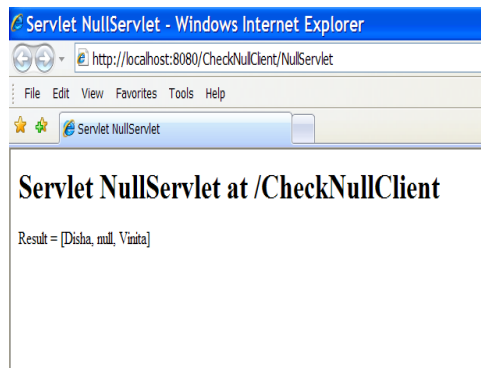


Figure 4.2 Output of an array with null element when invoked by a Java client

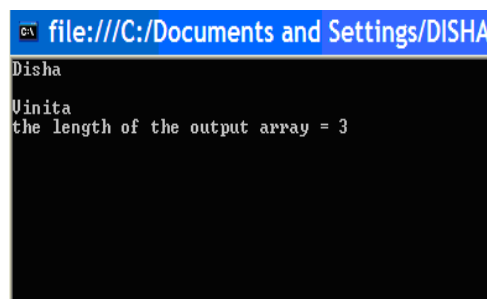


Figure 4.3 Output of an array with null element when invoked by a .NET client

4.7.2 Precision issues

For xsd:decimal, xsd:double, and xsd:float data types, each platform might have different precision support. This may lead to loss of precision. Let's consider the

following example in which a Java web service returns the sum of two float numbers. Java has a precision of 6 digits after decimal whereas .NET has a precision of 5 digits after decimal. Therefore, rounding off takes place in the .NET client and it loses precision [87] as illustrated by Figures 4.4 and 4.5.

Java Client

The Java client which calls the add method of the testprecision web service and passes the float values 4.111111 and 8.888888 to the web service and displays the sum that is returned by the web service i.e. 12.999999.

.NET client

The .NET client also passes the same values and displays the sum returned by the service. But because .NET is less precise the value 12.999999 gets rounded off to 13 and this is displayed.

Output from Java and .NET clients:



Figure 4.4 precision testing with a Java client

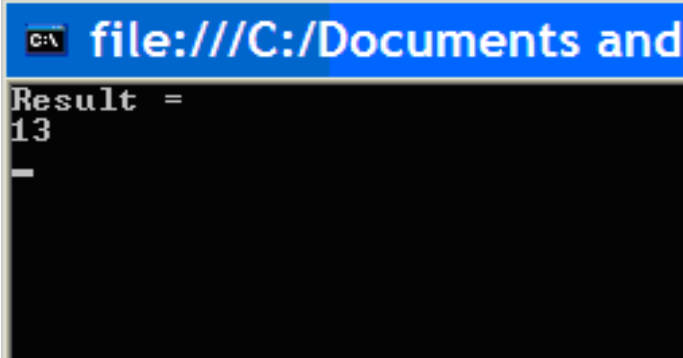
A screenshot of a Windows command prompt window. The title bar is blue and contains the text "file:///C:/Documents and Settings/...". The console output shows "Result =" followed by "13" on the next line, and a hyphen "-" on the line below that. The background of the console is black with white text.

Figure 4.5 precision testing with a .NET client

4.7.3 Primitive Types

Primitive data types can cause trouble. Each programming language has a set of native data types. A one-to-one mapping is not available between native data types and XSD data types. Therefore, information can be lost during the translation, or the receiver is not able to do the mappings for certain native data types [86].

i) Unsigned Numbers

For example, unsigned numerical types, such as `xsd:unsignedInt`, `xsd:unsignedLong`, `xsd:unsignedShort`, and `xsd:unsignedByte`, are the typical examples of XSD types. In .NET, the `uint`, `ulong`, `ushort`, and `ubyte` types map directly to the XSD types, in Java language unsigned types are not defined.

[WebMethod]

```
Public uint getUInt(uint ui)
```

```
{ Return ui; }
```

This is a .NET web service which returns the unsigned integer passed to it. Since unsigned types are not defined in Java, it leads to an interoperability issue when a Java client tries to call this web service.

To solve this, use the WebServicesAssembler tool to map the request input type to the Java native type long and then call the web service. Another thing to do is use wrapper methods to convert these unsigned data types to xsd:string type so that interoperability is achieved.

ii) Interoperability Issues seen with data types float and long

To test interoperability issues seen with Java and .NET two web services in Java which perform simple arithmetic operations like add, subtract, multiply and divide are considered. One web service takes the two input parameters as float data type, and it returns the result of the operation also as a float data type. The second web service takes parameters as type long and also returns the result as data type long.

Random values 2.1 and -1.2 are chosen for the add method, -5.36 and 3.82 for the subtract method, -.72 and 0.4 for divide method and 6.16 and -5.14 for the multiply method. The screen shots for the same are shown in Figure 4.6 and 4.7.



Figure 4.6 Browser Window for Float Data Type Web Service



Figure 4.7 Browser Window for Double Data Type Web Service

Upon testing the output of these methods the following outputs are obtained as shown in the table 4.2

Table 4.2 Table Showing Output of Web Methods of the Two Java Web Services

Web method	Add	Subtract	Divide	Multiply
Inputs	2.1,-1.2	-5.36,3.82	-0.72,0.4	6.16,-5.14
Float Data Type Web Service	0.89999986	-9.18	-1.8000001	-31.6624
Double Data Type Web Service	0.9000000000 000001	-9.18	- 1.7999999999 9998	- 31.6623999999 99
Actual Result	0.9	-9.18	-1.8	-31.6624

As can be seen from table 4.2 some of the results vary with the expected values by a very small precision and in the number of decimal digits.

Testing of a .NET client

A .NET client in VB.NET is developed to invoke the same two web services which have the parameters and the return type of float and long. The web service processes the results and sends the result back to the client where it is displayed. A point to be noted here is that .Net has no direct mapping of the 'Float' data type used in Java, when programmed in VB.NET. Hence the .NET client has been programmed to accept the float data type as a double.

As seen earlier that data type incompatibility can cause failure of interoperability or give us a situation where the SOAP messages are valid, interoperability is successful but the results are incorrect. This situation is highlighted in our implementation as when the float data type web service is consumed as a double data type, the results of the methods vary from the expected correct answer by a very small precision and in the number of decimal digits. Possible reasons for the incorrect results can be, firstly using different data types float and double, secondly since .Net and Java have different precision for decimal digits. This explains why the answers returned by the .Net client have more number of decimal digits.

When the double data type Java web service is consumed as a double data type, the results are closer to the correct value. This helps us infer that using same data types with both .Net and Java platforms leads to better results than otherwise. Table 4.3 shows us the final results after computing values entered by the .Net client. Clearly the best results were when both the web service and client used the same data type. In all other cases results were not completely correct varying in degree or precision and number of decimal digits. Figure 4.8 and 4.9 show output of .Net client when used with float data type web service and double data type web service.

Again for verification purposes the values 2.1,-1.2 are considered for add method, -5.36, 3.82 for the subtract method, -0.72, 0.4 for divide and 6.16, -5.14 for multiply method.

Table 4.3.Compiled table

Showing Output of Java Web Services and Java clients (first two rows) and Java web services and .Net clients (last two rows)

Web method	Add	Subtract	Divide	Multiply
Inputs	2.1,-1.2	-5.36,3.82	-0.72,0.4	6.16,-5.14
Float Data Type Web Service	0.8999998 6	-9.18	-1.8000001	-31.6624
Double Data Type Web Service	0.9000000 00000000 1	-9.18	-1.799999999999998	-31.6623999999999
Float Web Service-Double Client	0.8999998 56948853	- 9.1800003051757 8	-1.80000007152557	-31.6623992919922
Double Web Service-Double Client	0.9	-9.18	-1.8	-31.6624
Actual Result	0.9	-9.18	-1.8	-31.6624

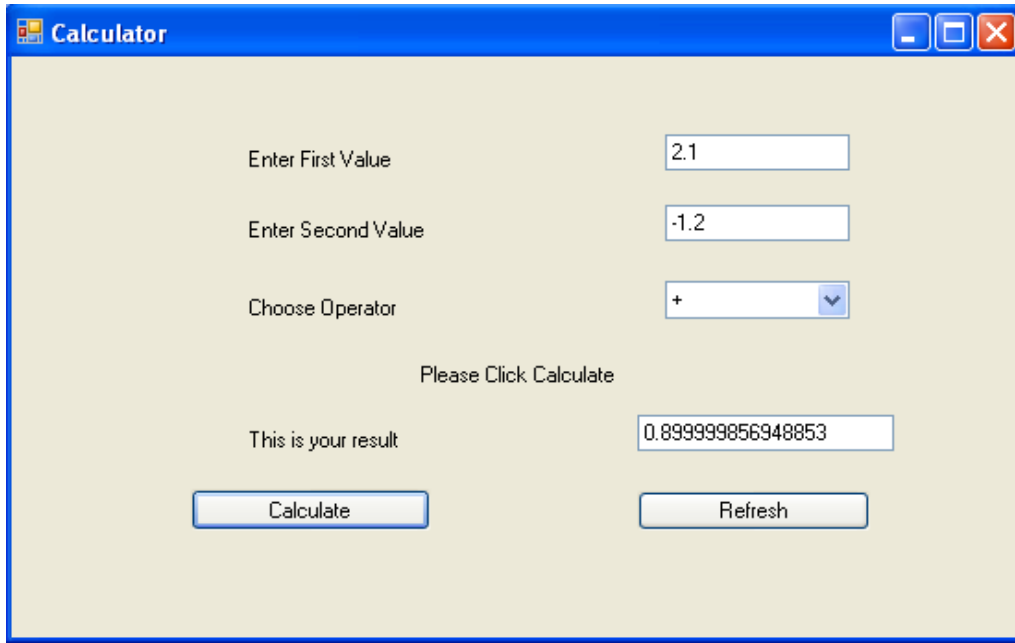


Figure 4.8 Double Data Type Client Window Showing Result of Float Data Type Web Service

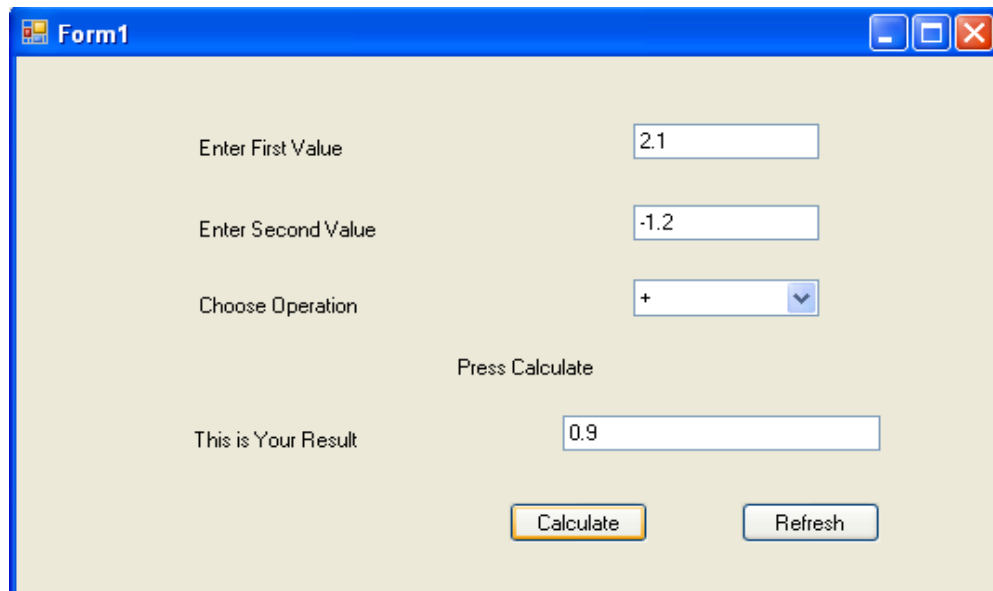


Figure 4.9 Double Data Type Client Window Showing Result of Double Data Type Web Service

Conclusion: Now .NET does not have an exact data type mapping for Java's float. Hence the client is programmed with data type as float. This gave us two situations, one, where a Java web service with data type float was consumed by a .Net client of data type double, two a Java web service with data type double consumed by a .Net client with data type double. Clearly, the best results were when the double data type web service was consumed by the double data type client. In situations where the web service and the client were developed in different platforms it is seen that the result is close to the expected value but not equal to the correct value. Also the number of decimal digits was large and differed between .Net and Java.

As recommendations for any successful cross platform interoperability application it is always best to have the same or almost similar data types in both client and web service. Achieving data type compatibility can be done by creating an XML schema document for the web service and developing a client that strictly complies with the Schema. WSDL of web service clearly defines all the web service methods and attributes such as data type, associated with input output parameters of the web service.

.Net platform is strongly recommended for developing web service applications which will be consumed on a small network such as within a business enterprise, university campus etc. .Net is more graphic, user friendly and easier to learn than Java. Java due to the presence to EJB'S and other highly specialized features is recommended to use for large scale applications such as banking , ATM web services, etc, where multitier data base applications are involved.

4.7.4 Collection of complex data types

In both Java and C# there are rich libraries of collection types. For example, Java supports collection types like java.util.Hashtable, vectors, Set, ArrayList, etc. Whereas in C# there's Systems.Collections.Hashtable, SortedList, Queue, Stack,

ArrayList, etc. These collection objects contain elements of different data types. Due to this, they may also be considered as weakly typed data structures.

When exposed across web services they create problems. The receiving side may not be able to understand the SOAP messages that contain weakly-typed object elements and native data types. For example, an ArrayList in a .NET web service is taken to be data of 'anytype' in the XML schema this makes it ambiguous. Therefore when the Java client sees the schema, he won't know which collection type to map the data to at the receiving side. This can be resolved by sticking to simple data types as much as possible and avoiding the use of complex data types [25].

a.Types of collection objects

Collection objects might contain elements of any data types. Thus, many consider them as weakly typed data structures. That makes them a wonderful programming tool. In object-oriented programming, there are rich libraries of collection types. In Java for example, there are:

- i) java.util.Hashtable
- ii) Vectors
- iii) Hashmap
- iv) Set
- v) ArrayList

While in .NET, there are:

- i) System.Collections.Hashtable
- ii) SortedList
- iii) Queue
- iv) Stack
- v) ArrayList

If exposed across web services, these collection types can cause insurmountable problems. The problem lies in how the receiving side is able to understand the serialized SOAP messages that contain the weakly-typed object elements and native data types. Even though some collection types look extremely similar between languages, such as System.Collections.ArrayList in .NET and java.util.ArrayList in Java, remember that the elements in the collections are

generic references. To accurately unmarshal the XML representation of a collection, consumers must have prior knowledge of the original concrete types. The burden is on the toolkit developers to interpret the XML schemas published by the web services providers and map the SOAP messages to the native data is not an easy task for the weakly-typed collections [85] .

An object of student was created, the object contained details like ID-No which was of type int, Name of type string, DOB of type date, Gender of type char, subjects an array of type string, marks an array to store float values and a Boolean value, the object containing the mixed data types could be successfully sent from the client to the web service and the object could be successfully displayed in the hash map table on the web service as shown in Figures 4.10 – 4.12.

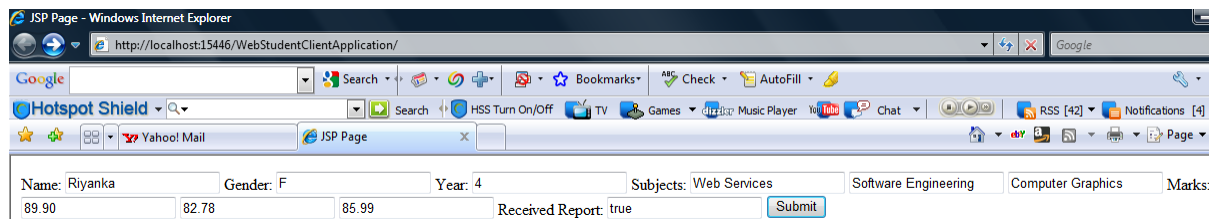


Figure 4.10 Initial Screen to enter Student Details (Java Client)

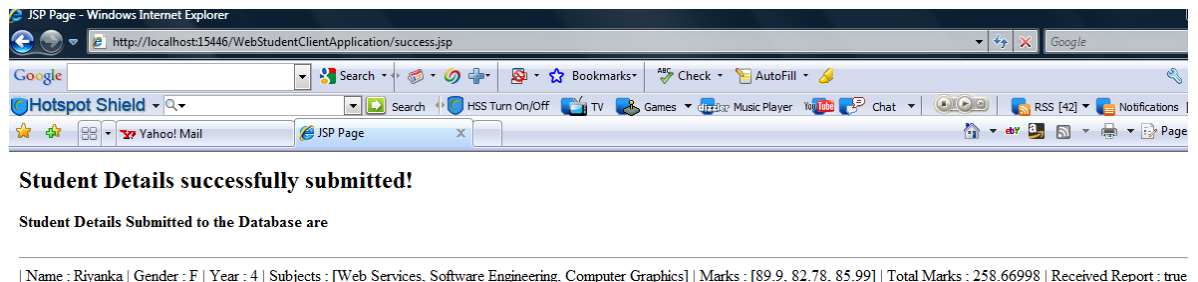


Figure 4.11 Result displayed after successful invocation (Java Client)

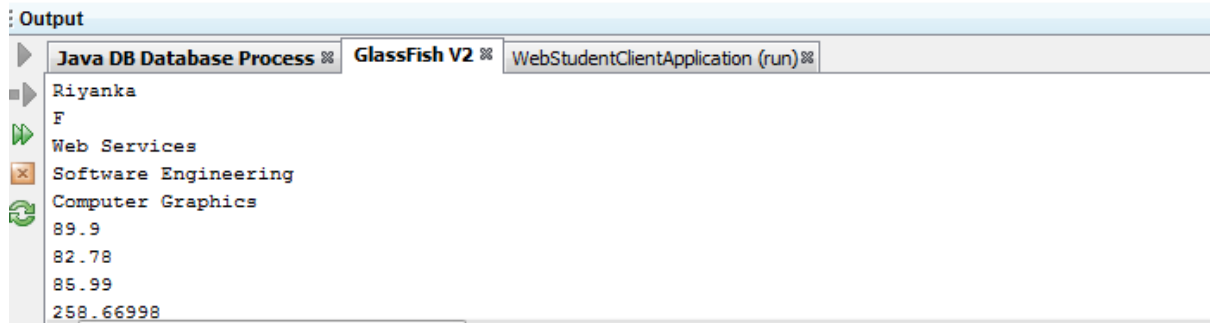


Figure 4.12 Output showing hash map contents of StudentDetails (Java Client)

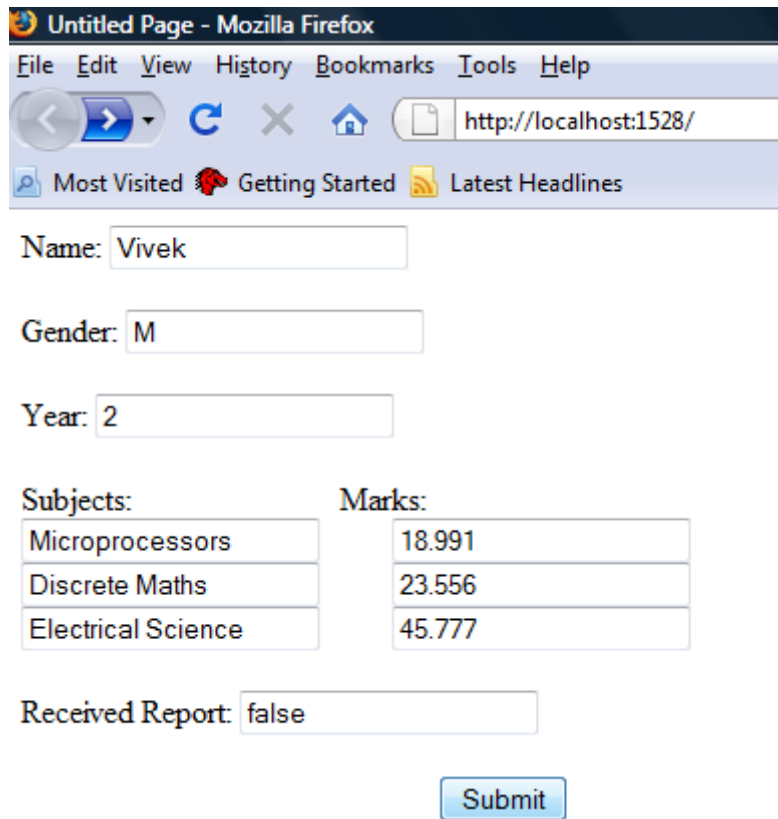
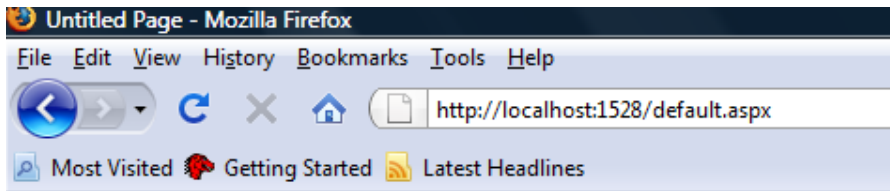


Figure 4.13 Initial Screen for entering Student Details(.NET Client)



Student Details successfully submitted!

The Student Details submitted to the database are:

Name: Vivek

Gender: M

Year: 2

Subjects: Microprocessors Discrete Maths Electrical Science

Marks: 18.991 23.556 45.777

Total Marks: 88.324

Received Report: False

Figure 4.14 Result displayed after successful invocation (.NET Client)

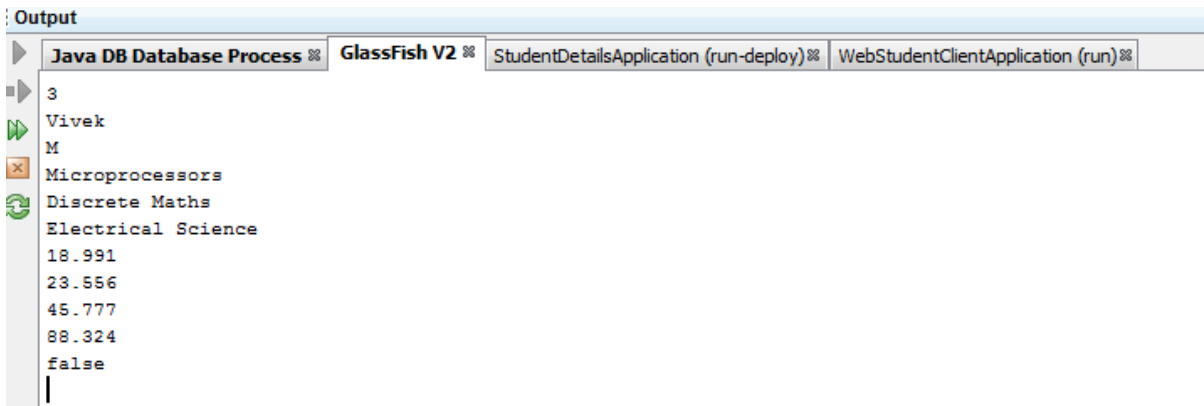


Figure 4.15 Output showing hash map contents of StudentDetails(.NET Client)

In Figure 4.13-4.15 the complex data type could be transmitted and interpreted correctly, but this may not always happen, hence it is best to use primitive data types which is supported by all vendors [89].

4.7.5 Relative URI reference as a namespace declaration in WSDL

XML namespaces help in creating universally unique URIs. They resolve naming conflicts in the XML documents. However, the way that URIs are interpreted and mapped in the native code differs between platforms. It is usually relative URIs which cause a problem. In Java, it's not a problem when the WSDL file is generated by the web service itself. This is because the target namespace is derived from the package and the tool automatically qualifies them with the schemas. But, when the web service is on .NET and it generates the WSDL then the target namespace comes directly from what is mentioned in the code. In .NET, the process of qualifying with the schema is not done and the relative URIs sometimes cause conflict when the target namespace is the same. Therefore, to avoid this, the best practice is to always make the namespace unique by qualifying it with its own organization domain name [86].

4.7.6 Date Time Issues

A schema data type called `xsd:dateTime` is available. This too is one of the primitive data types, but is discussed as a separate issue here due to a variety of problems occurring when this schema type is used, if not careful [87].

4.7.7 Null Values in Date data type

The communicating parties could pose problems if one of their data types is a reference type and the other is a value type. The `xsd:dateTime` is mapped to `System.dateTime` in .NET. This is a value type, whereas it is mapped to `java.util.Calendar` or `java.util.Date`, which is a reference type in Java. The object of a value type is in a stack and the object of a reference type is in a heap. Hence a null reference is allowed as it signifies that the object has a null pointer but a value type cannot have a null value. In Java when the reference type is not referencing any object, a null value can be assigned to it. Whereas .NET web services will throw a `System.FormatException` in case it receives a null value to its value type of data from a Java client.

If the Calendar or Date object is initialized with a null value in a Java client, then a null xsd:dateTime is sent in the SOAP message. When the web service built on the .NET platform receives the SOAP message, correct deserialization of the message is not possible. This is because the System.DateTime type is not nullable.

4.7.8 Precision problem in date data type

Different platforms use different precisions when interpreting the native dateTime types. When translating values of an XML dateTime simple type to different platforms, loss of precision can occur. The .NET platform uses four digits for the year value and seven digits for the milliseconds and the Java platform uses five digits for the year value and three digits for milliseconds. This can be clearly illustrated in the following example. Here is a .NET web method that returns a system MAX_VALUE of the DateTime data type.

The Java client then gets a SOAP Response from the .Net Web Method returning the MAX_VALUE of the DateTime datatype.

```
<?xml version="1.0" encoding="utf-8" ?>
  <dateTime          xmlns="http://tempuri.org/">9999-12-31T23:59:59.9999999
08:00</dateTime>
```

Since the Java platform uses only 3 digits for the milliseconds and the MAX_VALUE has seven digits, it rounds up the date. Therefore on the receiving side the output obtained is January 1, 10000.

4.8 Conclusion

Both Java and .NET are useful and both can lead to the same destination. When deciding about which platform is good for which application, it is recommended to concentrate on the larger business issues. Think about the existing developer skill sets, the existing systems, the existing vendor relationships, and the customers. Regardless of which platform is selected, new developers will need to be trained for J2EE or .NET. No matter which platform is chosen, both the

platforms can build web services, both the platforms offer low system cost and both offer single vendor solutions [14].

Sun's J2EE vision is based on a family of specifications that can be implemented by many vendors. One of J2EE's major disadvantages is that the choice of the platform dictates the use of a single programming language, and a programming language that is not well suited for most businesses.

There are several important advantages to the J2EE platform:

- i) J2EE offers absolute portability since its vendors support multiple operating systems.
- ii) J2EE has better legacy integration through the Java Connector Architecture (JCA) [15].
- iii) J2EE is a more advanced programming model, appropriate for well-trained developers who want to build more advanced object models and take advantage of performance features [14].

Microsoft's .NET platform vision, is a family of products rather than specifications. The major disadvantage of this approach is that it is limited to the Windows platform, so applications written for the .NET platform can only be run on .NET platforms[14].

There are several important advantages to the .NET platform:

- i) Web Services support is stronger, with industry standard eCollaboration built into the platform.
- ii) The cost of running applications is much lower, since .NET has a simpler programming model [14].
- iii) The ability to scale up is much greater, with the proved ability to support at leasten times the number of clients J2EE supports [15].

A series of tests are conducted and areas which lead to interoperability issues are explained and justified. The outcome of this work was published as one Journal paper “Interoperability Issues seen in Web Services”, IJCSNS International Journal of Computer Science and Network Security, VOL.9 No 8, August 2009 and as a conference paper “Comparison of Web Service Development Platforms”, Proceedings of the 2008 International Conference on Semantic Web and Web Services”, WORLDCOMP’08, July 14-17, 2008, Las Vegas Nevada, USA. In the next chapter a web service applications created for mobile devices is explained.

Chapter 5: Mobile Web Services

5.1 Introduction

In order to reap the full benefits of web services, not only desktop clients but even mobile clients should be able to interact with SOA. Mobilink the application designed and explained in this chapter allows mobile clients to publish, locate and invoke web services.

5.2 Evolution of Java

Because of the wide spread proliferation of the usage of mobile devices, and also because of the lifestyle of people where people are constantly on the go it may not be possible for a person to connect to his computer and stay in touch, but people generally carry their mobile phones around so it is very essential that mobile devices should be able to access web services to exploit the full potential of web services.

Due to the diversity and range in size of Java applications Sun Microsystems divided the Java technology into three categories as represented by Figure 5.1 [90]

- i) J2EE – used in large applications for developing server side scalable applications.
- ii) J2SE – Used for mid size applications used for desktop machines
- iii) J2ME – used for resource constrained mobile application. Here most of the features available in J2SE is available but on a much more scaled down form.

Each Java platform defines a set of technologies which can be used with a particular product. These technologies include.

- i) Java Virtual Machine – it fits inside a range of computing devices and is the reason for the popularity of Java which enables Java programs to have the ability to be built once but run any number of times from

any location, this term is referred to as WORA (Write Once Run Anytime).

- ii) Libraries and API's – are different for different types of computing devices.
- iii) Tools for deployment and device configuration [90].

5.3 Introduction to J2ME

J2ME is designed for small devices but there are a wide range of devices ranging from devices with high processing power like set top boxes which require constant network connection which is maintained using TCP/IP , to devices with low processing requirement like PDA's or pagers.

Unlike J2EE or J2SE where the range or application of devices is limited in J2ME the diversity of the applications is vast therefore it is not possible to have single software to cater to all the needs. Instead of being a single entity J2ME is a collection of specifications that define a set of platforms each of which is suitable for a subset of the total collection of consumer devices that fall within its scope. The configuration and the profiles that are appropriate for a device depend on both the nature of the hardware and the market to which the device is targeted.

5.3.1 Configurations

A configuration is a specification that defines the software environment for a range of devices and it generally depends on features like.

- i) amount of memory available
- ii) type of processor and its speed
- iii) the type of network connection available to a device

A configuration represents a minimum platform for a target device and it is not permitted to have any optional features. Vendors should implement this specification fully so that developers can rely on a consistent programming

environment and create applications which are as device independent as possible.

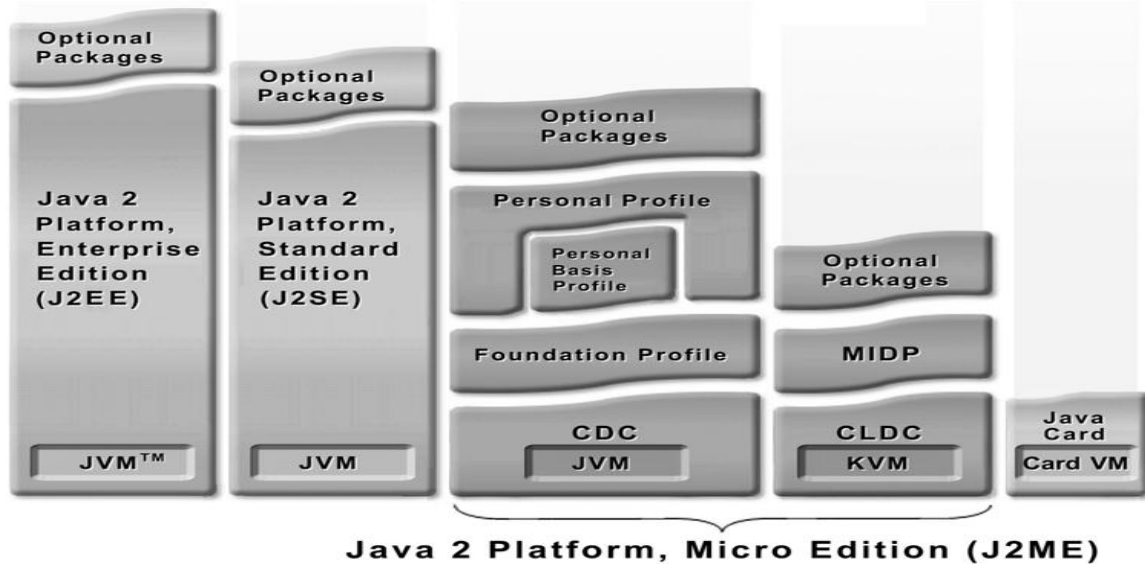


Figure 5.1 J2ME Architecture

J2ME currently has two types of configurations:

- i) Connected Limited Device Configuration (CLDC)
- ii) Connected Device Configuration (CDC)

Connected Limited Device Configuration (CLDC) – Is used for the low end of the consumer electronics range, used generally for devices like cell phones, pagers and PDA's, as the name indicates limited indicates that the devices have limited memory, limited processing power, limited display, limited battery life and limited network connection. The data rates are typically 9.6Kbps and network connection is generally intermittent and not very fast, it is generally costly accessing the internet from the phone and they are charged by the number of data packets exchanged.

The CLDC is designed for devices with 160KB to 512KB of total memory including a minimum of 160KB of ROM and 32KB of RAM available for the Java

platform. CLDC works on a smaller scaled down implementation of the JVM called the Kilo Virtual Machine (KVM) since the KVM is much smaller than the JVM it does not do everything performed by the JVM in the J2SE world. The KVM does not allow native methods to be added at run time, all native functionality is built into the KVM.

The KVM only has a subset of the standard byte code verifier, this means that the task of verifying the classes is split between CLDC devices and some external mechanism, this can have serious security drawbacks.

Connected Device Configuration (CDC) – Addresses the needs of devices like television set top boxes, car navigation systems, high end PDA's , web telephones etc which are between the CLDC and J2SE applications. These devices have a minimum of 512KB of Read Only Memory ROM, 256KB of Random Access Memory RAM and they have network connectivity maintained by TCP/IP and have much more capable processors.

In tune with keeping up with the upward compatibility of Java it is required that the core class libraries of Java should be based on the Java2 platform, wherever possible J2ME must use the classes and libraries available in J2SE this also reduces the learning curve for J2ME [91].

5.3.2 Profiles

A profile is layered on top of a configuration, adding the API's and specifications necessary to develop applications for a specific family of devices.

Foundation Profile: Is a specification for devices that support a rich networked J2ME environment. It does not support user interface, other profiles like Personal Basis Profile and Personal Profile are layered on top of the foundation profile and they produce user interface and other functionality. The combination of CDC + Foundation profile + Personal Basis Profile + Personal Profile is designed as the next generation of the Personal Java application runtime environment.

PDA Profile (PDAP) : Is built on CLDC , it is designed for palmtop devices which have better memory with a minimum of 512KB combined ROM and RAM and a maximum of 16MB and a better screen display.

Mobile Information Device Profile (MIDP) – This profile is of most important use to us because it deals with mobile devices which have wide usage and since mobiles are used widely in real life there is a lot of interest in developing useful mobile applications. This profile adds networking, user interface components and basic networking based on HTTP 1.1. There are two versions MIDP 1.0 and MIDP 2.0. MIDP 2.0 is backward compatible with MIDP 1.0 and it offers many new features which were not available in MIDP 1.0 such as support for multimedia, a new game user interface API, support for HTTPS connection. Applications developed on MIDP profile are also called midlets keeping in mind the other applications created by SUN like applets and servlets. The actual development process for midlets is more complicated than that of a J2SE it involves the following steps:

Edit Source Code -> Compile -> Preverify -> Package -> Test or Deploy [92].

5.4 Mobilink

The application Mobilink which is developed is a very important technological innovation. In Mobilink, web service publishing and user creation is done by the administrator using PHP based framework and Mysql. Details of registered users are stored in MySQL database. Checking correctness of data entered like month or day in a year is done by Javascript, AJAX is used to check availability of username and password. The web service and the J2ME client are developed using Netbeans IDE.

For the purpose of registration, the user has to log on to the web application and register with the specified username and password, consequent access of the

framework can either be on the desktop or by using the mobile interface. When accessing the web service through the mobile device the user is first prompted to enter the user name and password, Mobilink application validates the user using a method of the master web service, after this a list of available web services is shown, upon selection another method of the master web service can be invoked. The beauty of Mobilink is that it allows the user to specify the web services he would like to create to access the data stored in the back end database, this gives a level of abstraction to the user where he need not be concerned about how the web service is created.

5.4.1 Different software used for the development of Mobilink

JavaScript and AJAX

JavaScript - Despite its name, JavaScript and Java are unrelated except for the syntax, JavaScript was invented in 1995 as a scripting language for Netscape web browser to enable the web pages to be more dynamic. JavaScript supports only a few data types like numbers, strings, booleans, functions and objects. JavaScript objects are generally name-value pairs called properties.

AJAX (Asynchronous JavaScript and XML) – Is a powerful web development model for browser based web applications. Technologies that form the AJAX model, such as XML, JavaScript, HTTP and XHTML, are individually widely used and well known. However AJAX combines these technologies and lets web pages retrieve small amounts of data from the server without having to reload the entire page [93]. This capability makes web pages more interactive and lets them behave like local applications [94]. AJAX is responsible for enhancing the responsive nature and interactiveness of web pages. With AJAX web pages contain JavaScript that asynchronously invoke requests on a web server by creating an XMLHttpRequest request object, attaching it to a Java callback function to handle the response, and then invoking the request [95].

5.4.2 MySQL

MySQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout its history. With its superior speed, reliability, and ease of use, MySQL has become the preferred choice for Web, Web 2.0, SaaS, ISV, Telecom companies and forward-thinking corporate IT Managers because it eliminates the major problems associated with downtime, maintenance and administration for modern, online applications.

Many of the world's largest and fastest-growing organizations use MySQL to save time and money powering their high-volume Web sites, critical business systems, and packaged software — including industry leaders such as Yahoo!, Alcatel-Lucent, Google, Nokia, YouTube, Wikipedia, and Booking.com [96].

Reasons for selecting MySQL

A database is nothing but a collection of structured data, the data can be in the form of text, pictures, video or music files, to add process or manage the data in the database a database management system like MySQL server is needed.

- i) MySQL is open source software which means that it is possible for anyone to download the MySQL software from the internet and use it without paying anything.
- ii) MySQL is very fast, reliable and easy to use.
- iii) MySQL is a relational database management system, in a relational database the data is stored in separate tables rather than have all the data in a single file.
- iv) A large amount of contributed software is available.

5.4.3 PHP

PHP stands for Hypertext Preprocessor. PHP is a server side scripting language, which can be embedded with HTML or it can be used as a standalone.

Proprietary products in this niche include Microsoft's Active Server pages, Macromedia's Cold Fusion, Sun's Java Server Pages.

Reasons for choosing PHP

- i) Ease of use – PHP is easier to learn compared to other ways to achieve similar functionality. Unlike Java Server Pages or C based CGI, PHP doesn't require a deep understanding of a major programming language before a trivial database or remote server call.
- ii) HTML embeddedness – PHP is embedded in HTML in other words PHP pages are normal HTML pages that escape into PHP mode only when necessary.
- iii) Cross-platform compatibility – PHP and MySQL run on different platforms (Unix, Windows, MAC etc)PHP is compatible with three leading web servers: Apache HTTP server for Unix and Windows, Microsoft Internet Information Server and Netscape Enterprise Server it also works with several less well known servers like Microsoft's personal web server, AOL Server etc.
- iv) Open Source software- open source software is generally very flexible, it allows the programs to be extended and customized as per the customer's requirement.

5.5 Mobilink Architecture

Mobilink is divided into three main parts, a web service publisher which is a web form designed using PHP, MySQL and JavaScript used to define the web service and populate it before publishing it. Secondly the web service itself which is developed in Java using the Netbeans IDE and thirdly the web service client which is a J2ME application which accesses the web service.

5.5.2 Screenshots of Mobilink

Example of the registration form

Fill in all the values.

First Name:	Shoaib	
Last Name:	Khan	
Username:	shoaibk	Username available
Password:	*****	
Confirm Password:	*****	Passwords Match!
Gender :	<input checked="" type="radio"/> Male <input type="radio"/> Female	
Date of Birth :	1986 7 12	
Country :	India	
State:	Karnataka	
City:	Bangalore	
Zip Code:	682987	
Telephone No.:	009198802321209	
Fax no.:	009198802334232	

Submit and Proceed

Figure 5.2 Screenshot of Registration page

The registration page shown in Figure 5.2 accesses the Customer Table in the MySQL database and the values are recorded. Form validation is done via PHP and entries like the date field where the number of days in a month are dependent on the month as well as the year (in case of a leap year) are filled in using JavaScript. Checking the username availability is done using AJAX as shown in Listing 5.1.

Listing 5.1 Ajax script used for checking username availability

```
function check(data){
    if(XMLHttpRequestObject) {
        var obj = document.getElementById('mes1');
        XMLHttpRequestObject.open("POST", 'checkUser.php');
        XMLHttpRequestObject.setRequestHeader('Content-Type',
            'application/x-www-form-urlencoded');
    }
    XMLHttpRequestObject.onreadystatechange = function()
    {
        if (XMLHttpRequestObject.readyState == 4 &&
            XMLHttpRequestObject.status == 200) {
            if (XMLHttpRequestObject.responseText=="False"){
                obj.innerHTML="Username not available";
                document.getElementById('proceed').value="false";
            }
            else{
                obj.innerHTML="Username available";
                document.getElementById('proceed').value="true";
            }
        }
    }
    XMLHttpRequestObject.send("data=" + data);
}
```

The table structure and sample data are given in Figure 5.3

fname	lname	uname	pword	gender	dob	cc	state	city	zcode	tel	fax
Prabhakar	Reddy	prabhu	bits	m	1 Jan, 1987	IN	Maharashtra	Bangalore	5678	657	8567
Sangil	Vadivel	sangil	bits	m	3 Feb, 1973	IN	Maharashtra	Bombay	567	567	4567
Zamir	Parkar	zamir	bits	m	31 Mar, 1984	IN	Maharashtra	Bombay	5678567	45678	4567

Figure 5.3 Sample database values of registered users

The second step takes the user to the web service creation page

Web Service Creator

Fill in all the values.

Web Service Name:
Username available

No. of Fields:

Figure 5.4 WebService creation page

The web service name specifies the name of the web service that we are about to create and no of fields indicates the number of fields as shown in required in the web service as shown in Figure 5.4. The different fields are specified as shown in Figure 5.5.

Name:	<input type="text" value="Name"/>	Type:	<input type="text" value="String"/>	<input type="button" value="v"/>	Null:	<input type="text" value="No"/>	<input type="button" value="v"/>
Name:	<input type="text" value="Author"/>	Type:	<input type="text" value="String"/>	<input type="button" value="v"/>	Null:	<input type="text" value="No"/>	<input type="button" value="v"/>
Name:	<input type="text" value="ISBN"/>	Type:	<input type="text" value="Integer"/>	<input type="button" value="v"/>	Null:	<input type="text" value="No"/>	<input type="button" value="v"/>
Name:	<input type="text" value="Edition"/>	Type:	<input type="text" value="String"/>	<input type="button" value="v"/>	Null:	<input type="text" value="No"/>	<input type="button" value="v"/>

Figure 5.5 Screenshot of data fields of web service

Once the data is entered the final screen comes up as shown in Figure 5.6.

Name : Java
Author : sangil
ISBN : 6789
Edition : First

Figure 5.6 Adding data to web service

Name	Author	ISBN	Edition
Java	sangil	6789	First
CCNA	sangil	678	Second

Figure 5.7 Screenshot of web service with the data

In this way data is stored in the database as shown in Figure 5.7 according to the user's description. This structure is flexible and changes can be made when necessary.

5.5.3 Web Service

The web service is designed in Netbeans 6 IDE and deployed on the Sun Java System Application Server 9. The web service bridges the gap between the web service publisher, the database and the mobile J2ME application. In Netbeans the web service is coded using Java as follows.

```
@WebMethod(operationName="OperationName")
public returnType
OperationName(@WebParam(name="ParamaterName")String
ParameterName,@WebParam(name="ParameterName")String
ParameterName)
```

```
{  
    Code.....  
    Return returnType;  
}
```

The web service consists of a number of web methods which contain the parameter supplied and the returning value. The code inside the web method is standard java code and in this case it initializes an instance of DbCon.java which is the class responsible for accessing and carrying out transactions with the MySQL database.

The following Web Methods are used in this application

Login

Parameters: Username (string), Password (String)

Function: Validating the user

Return : True or False

number

Parameters: None

Function: Counts the number of web services

Returns the number of web services

Webservices

Parameters: index (int)

Function: Returns the web services listed at index

Return: The web service name

searchField

Parameters: name(String)

Function: Returns the number of datafields in webservice "name"

Return : Returns the total number of operations

searches

Parameters: index(int), name(string)

Function: Returns all the data fields in web service “name”

Return : The operation name at index

resCount

Parameters: name(String), field(String), entry(String)

Function: Returns the number of data entries matching the search criteria

Return : The number of results

finalResult

Parameters: name(String), field(String), pos(int),word(String)

Function: Searches for word in field in web service “name” at position pos

Return : the details matching the search keyword

5.5.4 Web Service Consumer

JAX-RPC (Java API for XML RPC) uses the popular concept of web service endpoints and clients. The clients invoke access, consume or make use of the services exposed by the end points. The details about web service endpoints are specified in the WSDL document. The J2ME client invokes the web service.

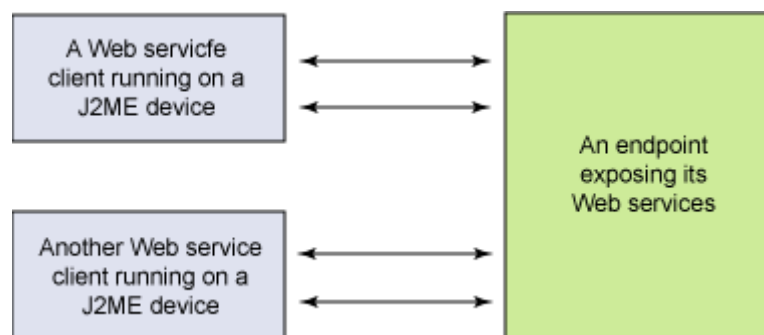


Figure 5.8 A J2ME client invoking the web service

A J2ME client uses remote method calls to invoke methods associated with web services as shown in Figure 5.8 [110], the midlet cannot directly call a method of the web service so it must create a proxy called a stub interface which can interact with the web service, and there are suitable classes available in J2ME which can generate the stub classes. The J2ME midlet to web service stub interaction is local and happens within the J2ME device as indicated in the Figure 5.10 [110]

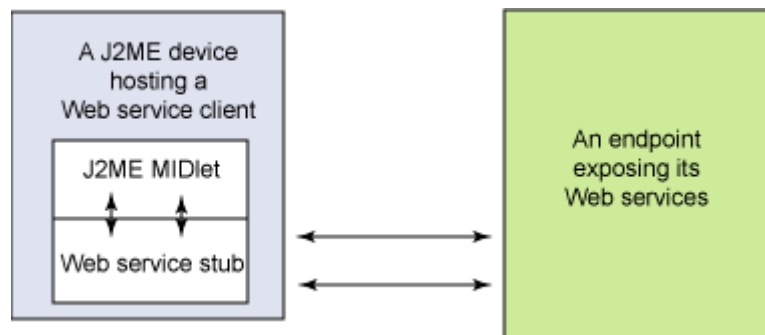


Figure 5.9 Interaction of a web service stub with J2ME midlet and the web service

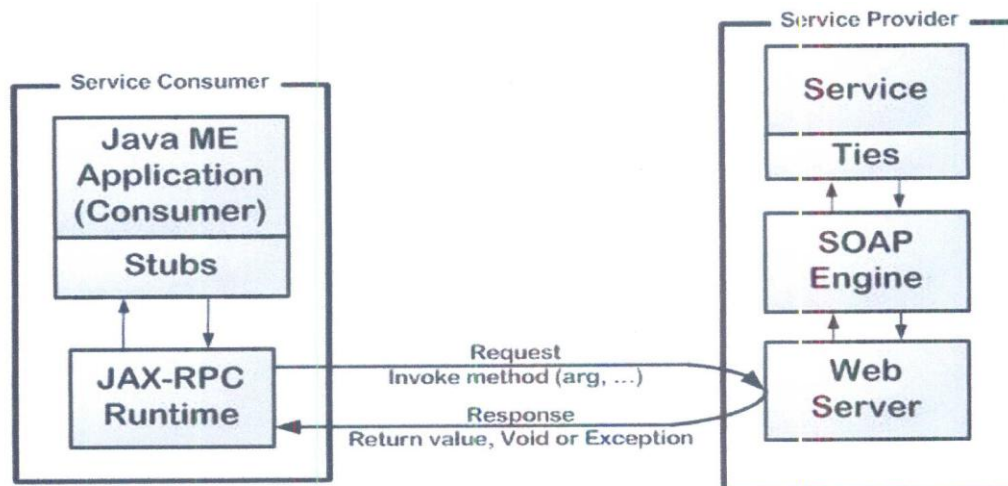


Figure 5.10 Typical JAX-RPC Application

The web service stub is a set of classes that act as a local agent or proxy of the actual web service endpoint, J2ME uses a subset of JAX-RPC 1.9 specification to provide a Java API to SOAP based web services this is appropriate for a J2ME platform because J2ME devices are generally used as web service clients and they are not likely to expose web services unless used in a peer to peer network where a mobile device can also act as a service provider in addition to requesting for web services, this is especially useful in ad-hoc networks [97], hence a subset of J2ME-RPC is available for J2ME devices. A JSR1.0compliant device may not support XML 1.0 encoding but it is the work of the network carrier to produce a SOAP representation that can be transferred to an interoperable XML 1.0 representation[98].

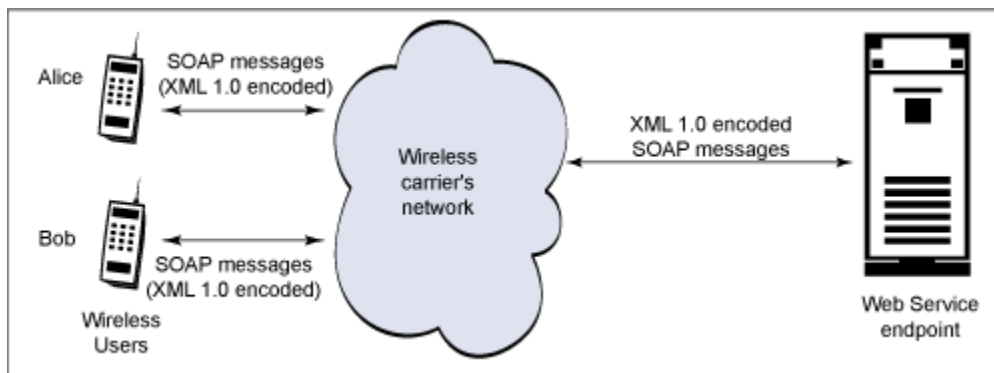


Figure 5.11 A wireless carrier's network ensures XML encoding of SOAP messages

The flow of events that occur on logging on to Mobilink and accessing the web service is shown in Figures 5.12 – 5.17.

5.6 Mobilink on the Netbeans emulator The Login screen

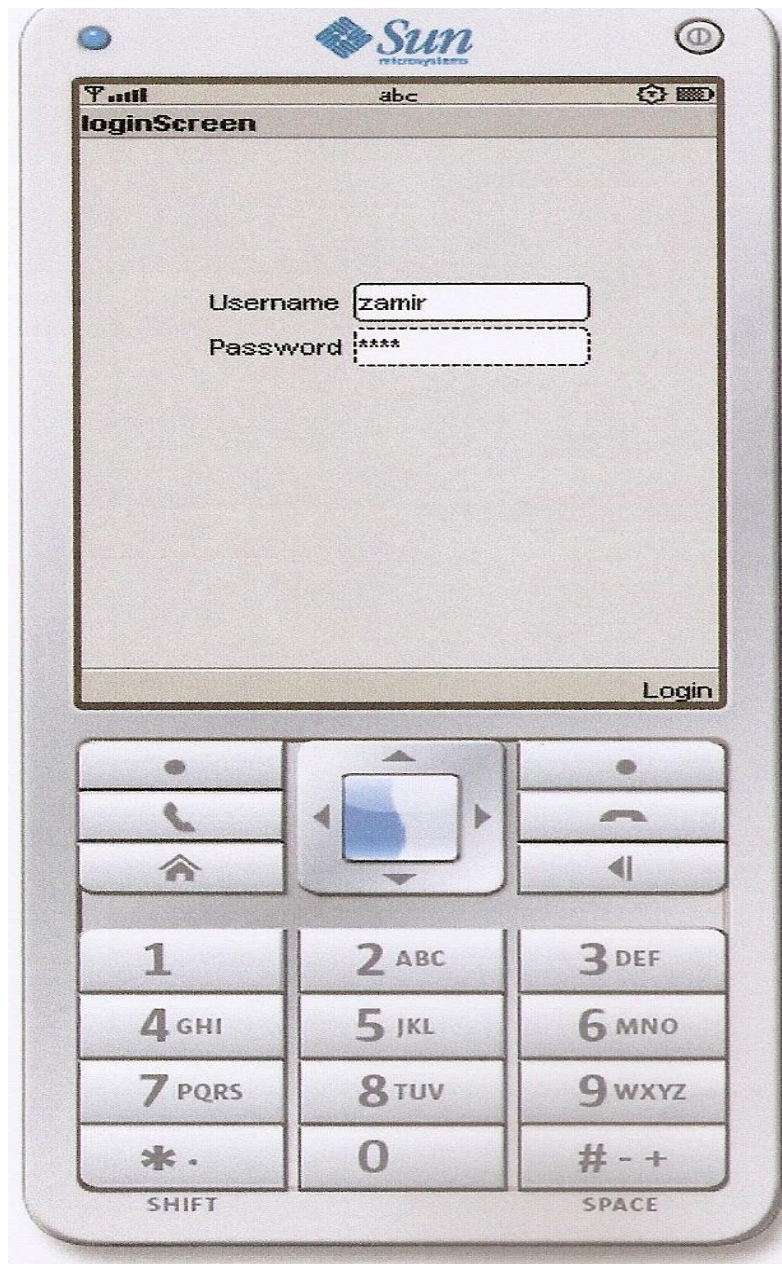


Figure 5.12 Login Screen



Figure 5.13 The Login Successful Screen

The Web Services Screen

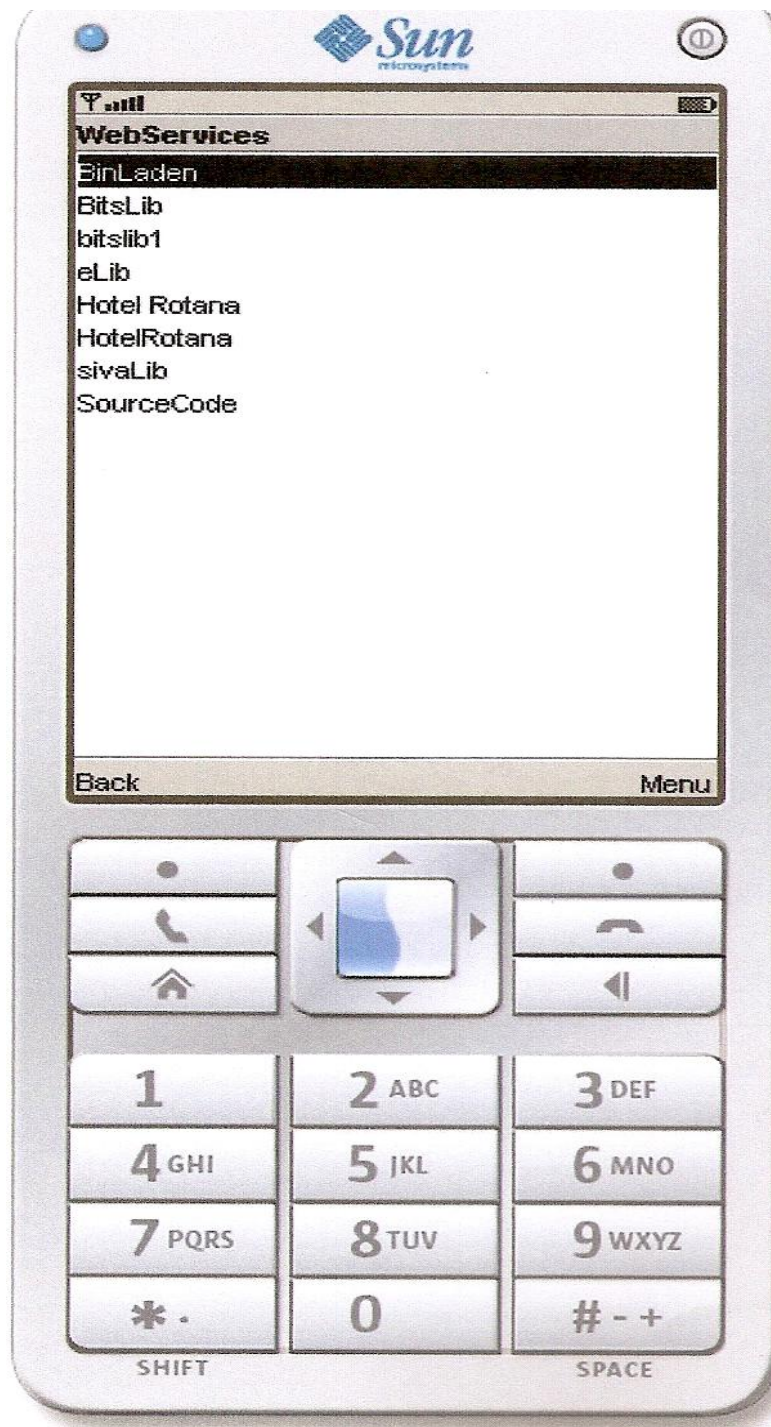


Figure 5.14 Web Services Screen with a list of available Web Services

Screen on selecting BitsLib

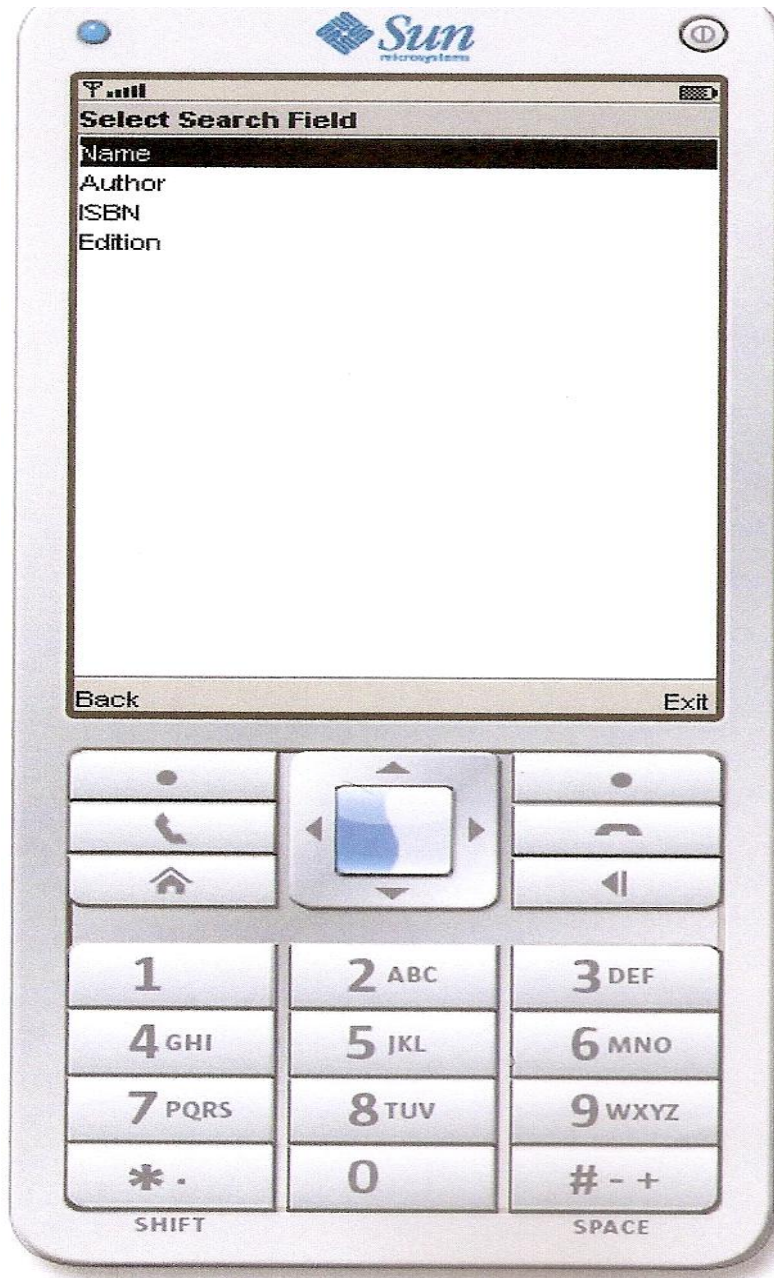


Figure 5.15 Search Field Screen with parameter to select for searching
Author is selected

Screen on selecting author

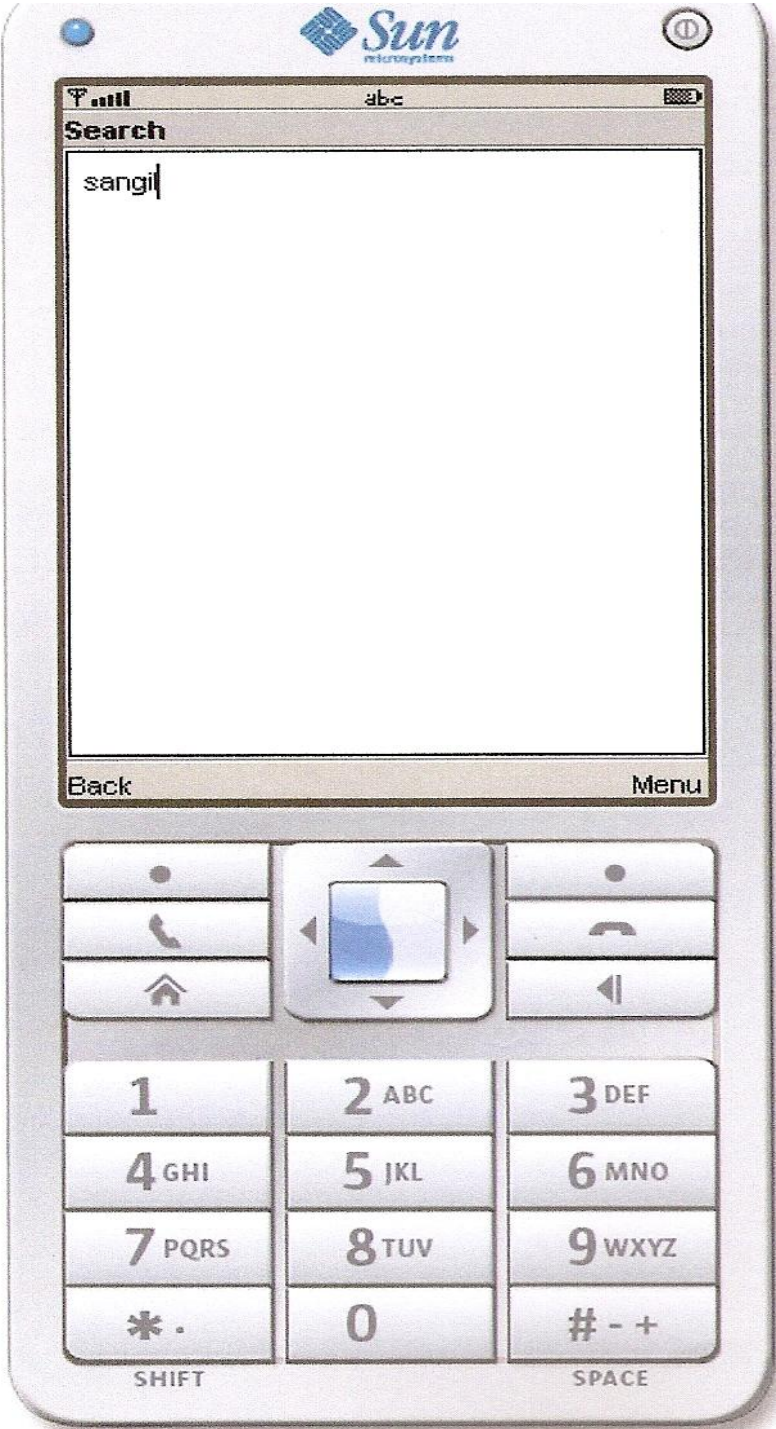


Figure 5.16 Search screen with the given input keyword

The Result Screen

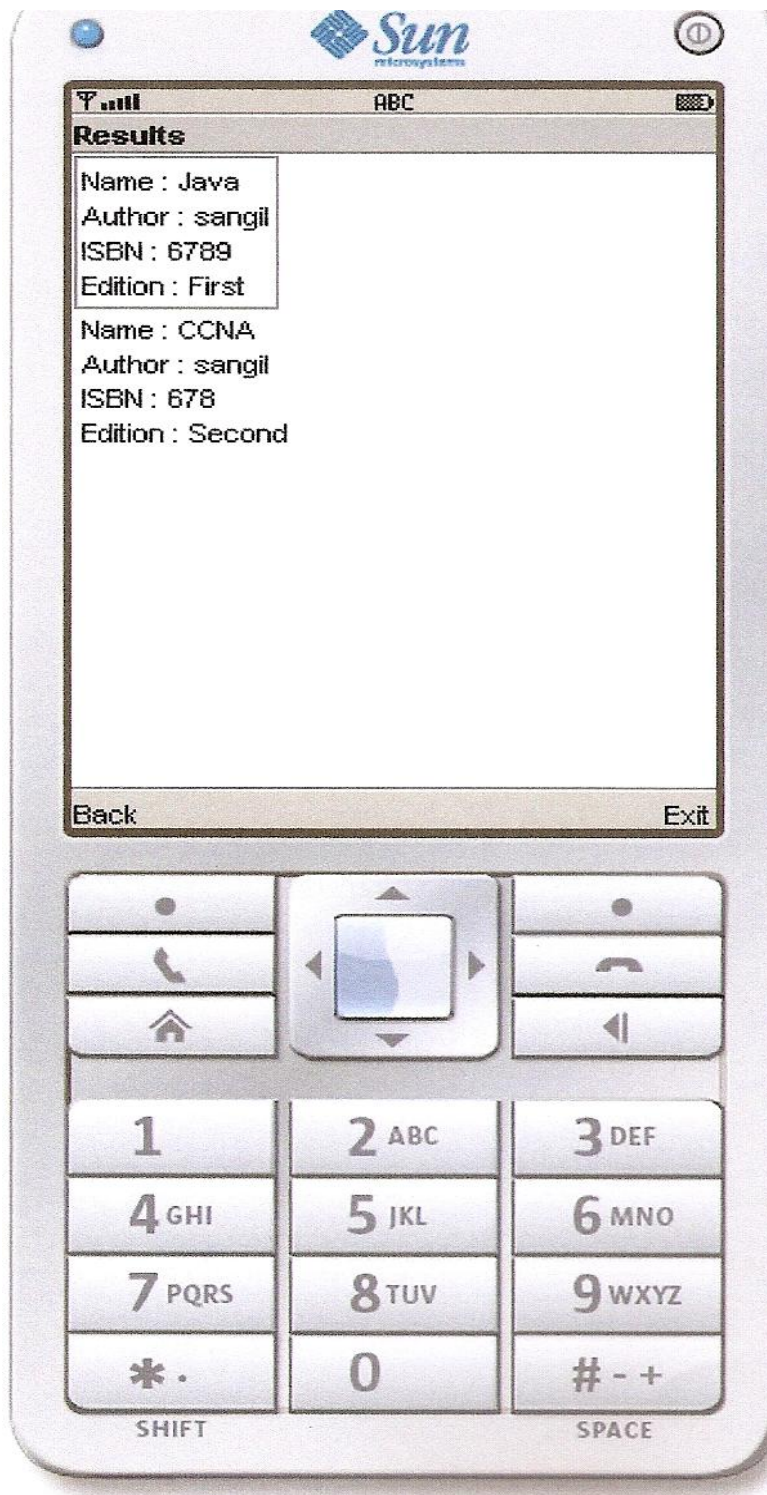


Figure 5.17 Result screen with results

5.7 Conclusion

Different web services were hosted on the server and a MySQL database contained the details of all the hosted web services, a J2ME client application was created which could access any of the registered web services. A search operation can be performed on the database by the selected web service, the interesting feature of this application is that there is a level of abstraction provided, which enables a user to create a web service as per the desired specification even if the user does not have programming knowledge, this is enabled by the availability of a GUI which provides the user with suitable forms to create the web service. This work is published as a paper “J2ME Based Tool for interaction With Web Services”, Proceedings of the 2008 International Conference on Semantic Web and Web Services”, WORLDCOMP’08, July 14-17, 2008, Las Vegas Nevada, USA.

5.8 Future work A simple search is performed in the database, a database which contains image files or video files can also be created and a search can be performed on the database, similarly more database operations like inserting, deleting and updating records can also be performed. The web service application can be extended to perform more complex operations which could involve banking transactions or online mobile purchasing which would require the user to specify the account number or the credit card details hence security features would need to be added here. In addition mobile phones are now used for more complex operations where content can be tailored to user preferences, user locations and device capabilities. Information could be provided on a location based context, for e.g. a user can be given directions to a restaurant as per the location, food preferences etc. On line video like YouTube which would require more efficient graphical processing ability can be viewed on the mobile so it is analyzed that the potential of mobile devices is large and the designed application can be designed to cater to all these possibilities [18]. In the next chapter the current status of web service security is considered and issues which could arise with security in interoperable web services is looked into.

Chapter 6: Security issues in interoperable web services and web service security interoperability platform development

6.1 Introduction

Web services provide a framework for interoperable machine to machine interaction over the network. Because of the large number of systems interacting, maintaining security specifications over the internet becomes a highly complex task. Since web services are loosely coupled, it becomes difficult to impose restrictions on the security mechanism followed by the different participating bodies. In order to have some uniformity in the manner of interaction the W3C and the OASIS have formulated certain specification standards like Web Service Interoperability Technology (WSIT) with Java and Web Services Enhancement (WSE) and Windows Communication Framework (WCF) with .NET [100].

Interoperability issues are dealt in detail in Chapter 4 of this thesis report. In this chapter the interoperability issues with respect to the security of web services is considered. A platform where participating entities can exchange security information prior to exchanging data is developed. This information is also transmitted in a secure fashion. The importance of security in web services cannot be underestimated. Security will be the prime deciding factor on the success and the wide scale adaptability of web services both by business houses as well as individual consumers.

6.2 Concept of Interoperability

Chapter 4 of this thesis report discusses interoperability issues in detail, just brushing through the concept of interoperability, interoperability can be defined as the ability of software and hardware systems of one computer network to communicate with other systems on other computer networks, each running different protocols and different technologies. It is a tool that allows systems running on different technologies to communicate and exchange information with

each other without requiring custom/intermediary coding to integrate the two technologies.

Interoperability with regards to web services means that a web service and a web service client are able to communicate and exchange important data with each other irrespective of the platform or the programming language on which they are running. Some major considerations for achieving interoperability in web services are:

- i) Both client side and web service should implement the same version of the specification. As not all specifications are totally backward compatible, this can pose a threat to achieving interoperability in web services.
- ii) Same versions of web services and Web Service Interoperability (WS-I) specification should be used.
- iii) The semantics used in web service communication must be agreed upon in advance.

The reason that web services are receiving considerable attention right now is because they provide interoperability. Web services are used these days in many B2B and B2C applications like banking, hotel management etc. All these web services and web service clients are built using different languages and run on different platforms. Hence, a technology is required that ensures integration of these applications without much effort.

6.3 WS-I Basic Profile

WS-I is an industry organization that includes members from IBM, Microsoft, Intel, Oracle etc. The WS-I organization aims at achieving web services interoperability. The WS-I Basic Profile is a profile that contains implementation guidelines to avoid interoperability issues. They are like a set of best practices that should be followed while developing web services.

Web Services Interoperability Technology (WSIT)

WSIT is used to ensure that interoperability is provided in the next generation web services enterprise technologies [101]. WSIT provides interoperability in various web service technologies like message optimization, reliable messaging, security etc. between the Java platform and Microsoft's .NET platform.

Message Optimization Technology

Various types of data are shared on the internet using web service applications. This data can be in any format like documents, images, music files etc. When these files are converted into XML formats for transmission via SOAP messages, even larger files are created. This leads to degradation in performance of web services over the network. Hence, XML messages that are encoded need to be optimized for web services. So message optimization technology encodes the binary object in such a manner, as to optimize the bandwidth required to send the SOAP message.

Reliable Messaging Technology

Reliability is defined as the system's ability to deliver messages from point A to point B without error. Reliable Messaging Technology is used to ensure that messages are delivered at least once to their desired destination. With the help of this technology, system can recover from a failure of message sequence being lost in transit. Sender sends the message repeatedly until an acknowledgement is received from the other end [101].

Reliable Message Technology has the following advantages and disadvantages:

- i) Ensures Messages are delivered exactly once in order.
- ii) Degradation of web service performance.
- iii) Web Service clients that do not support Reliable Message Technology are not interoperable with web services that have this technology.

Security Technology

WSIT provides another level of security on top of the existing transport-level security through WS-Security feature. It helps provide integrity and confidentiality of the messages being transmitted. WSIT enhances security in web services using the following implementations:

- i) **WS-Secure Conversation:** Both end parties are able to negotiate a shared security context when the first message is exchanged. Following messages use the derived session key, reducing the security processing overhead.
- ii) **Web Services Security Policy:** Web Services are able to use security assertions to emphasize the important security elements required for successful communication.

6.4 .NET Framework

Microsoft .NET Framework consists of a large in-built library that offers solutions to common programming problems. It also acts as a virtual machine on which programs written in certain specific programming languages can be executed. The pre-coded library is known as the Base Class Library using which other applications are built. The runtime environment of the .NET Framework is known as the Common Language Runtime (CLR). Because of the CLR, programmers can write applications irrespective of the underlying CPU, operating system etc. CLR also provides other services like exception handling, security and memory management. The Base Class Library and CLR together form the .NET Framework. The working of the CLR is represented in Figure 6.1 [110].

- i) **Common Language Infrastructure (CLI):** The Common Language Infrastructure provides a language independent platform for development and execution of applications. All .NET specific languages like VB, C# etc. compile into a second, platform independent language known as the Common Intermediate

Language (CIL). This is then converted into machine readable form with the help of .NET virtual machine CLR [102].

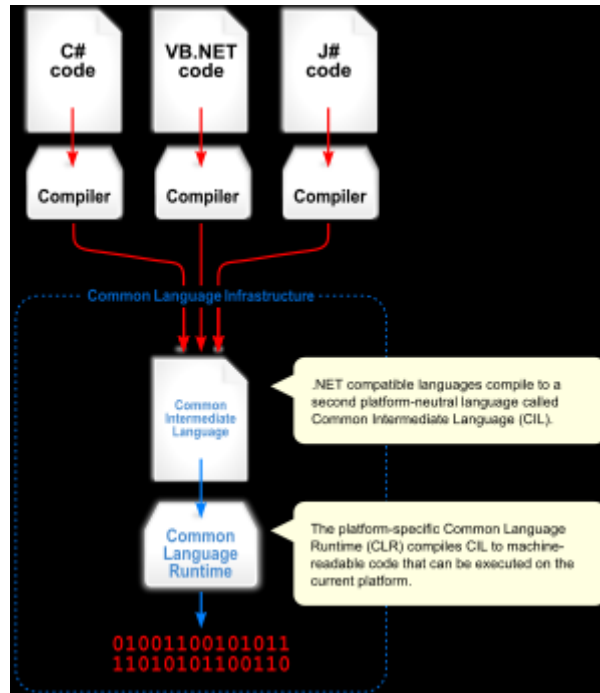


Figure 6.1: Overview of Common Language Infrastructure

- i) **Security:** .NET framework provides its own security mechanism. It limits the access that code has to critical resources and operations by using Code Access Security (CAS). It also implements role based security so that unauthorized access to resources is restricted. ASP.NET web application security prevents false access to a site by comparing authentication credentials to a list of authorized users contained in an XML file.
- ii) **Memory Management:** The .NET Framework Common Language Runtime manages memory on its own. The framework contains a garbage collector which runs periodically. As long as a reference to an object exists, the object is considered accessible. Once there is no reference to an object, it is considered garbage. The object is then destroyed freeing up the memory allocated to it.

6.5 Windows Communication Foundation (WCF)

Windows Communication Foundation is an Application Programming Interface (API) introduced with the .NET 3.0 Framework that is used to build applications that can communicate with each other. WCF uses Service Oriented Programming model for communication [45]. This model unifies mechanisms like web services, distributed transactions, .NET remoting and message queues all under WCF. The service model features a straightforward mapping of web services concepts to those of the .NET Framework common language runtime (CLR), including flexible and extensible mapping of messages to service implementations in languages such as Visual C# or Visual Basic [45]. It includes serialization facilities that enable loose coupling and versioning, and it provides integration and interoperability with existing .NET Framework distributed systems technologies such as Message Queuing (MSMQ), COM+, ASP.NET Web services, Web Services Enhancements (WSE), and a number of other functions. The ranges of issues addressed by the WCF technology are:

- i) Unification with existing .NET Framework communication technologies.
- ii) Support for Interoperability, including security and reliability.
- iii) Explicit Service Orientation.

WCF Service

A WCF Service consists of three parts: a service class, which implements the business logic of the service, host environment where the service is hosted and endpoints to which the client will connect. The endpoint of the service specifies the contract, which contains information regarding how the web service will be accessed. It also contains binding information which defines how a client will communicate with the web service.

WCF Client

WCF is designed according to the Service Oriented Architecture which support distributed computing. As a result a service can be consumed by more than one

client and vice versa. A WCF client uses the WSDL interface of the Web Service to communicate with the service [45]. WCF supports various advanced web service (WS) standards like WS-Security, WS-ReliableMessaging and WS-Addressing. Clients can communicate with the WCF Service using either RPC based or literal based encodings. The interaction between WCF service and client is shown in Figure 6.2 [110].

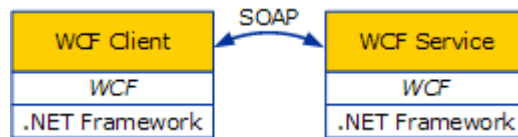


Figure 6.2: WCF Service and Client

6.6 Web Service Security

Some of the fundamental issues to be concerned in a distributed client server environment like web services where information travels over a wide open networking infrastructure are given below.

- i) Confidentiality: This ensures that any third party listening to the conversation cannot read and interpret the data.
- ii) Integrity: This provides receivers with the ability to detect any change of data, this prevents against intentional or unintentional change of data during transmission.
- iii) Authentication: This ensures that the client or user using the data is the correct person.
- iv) Authorization: This ensures that the client or user has the right to access the information.
- v) Non-Repudiation: This ensures that the client or the user cannot deny the use of the information at a later time.

6.6.1 Need for Web service Security

The architecture of web services allows information to be transmitted as plain text as XML/SOAP over HTTP which was the most preferred transport protocol. It is very easy to intercept and interpret this information. Security can be provided either at the transport level or at the application level. This leads to differences in

the various security mechanisms because all of a sudden security becomes tied down to the service provider. Security at the transport level is generally point to point security, they are based on a specific transport protocol/layer, such as TCP/IP for SSL and HTTP for HTTPS [102]. In applications like web services there could be intermediate nodes which could process or even modify (i.e. remove or insert a SOAP header) therefore there should be some mechanism which ensures message level security [100].

6.6.2 Web Services related security standards

Another way of securing web services as specified by Web Services Security (WSS) is to secure the data that is being transferred over the underlying non secure transport protocols like HTTP. Here the body of the SOAP message contains the encrypted data and the header of the SOAP message contains the session key encrypted with the private key of the message sender, the header could also contain information like security token, signature etc. At the receiving end the session key is extracted by using the public key of the sender and this session key is used to decrypt and extract the data contained in the SOAP body, this technique also ensures that the message comes from a particular user who has access to the private key that encrypted the session key [102].

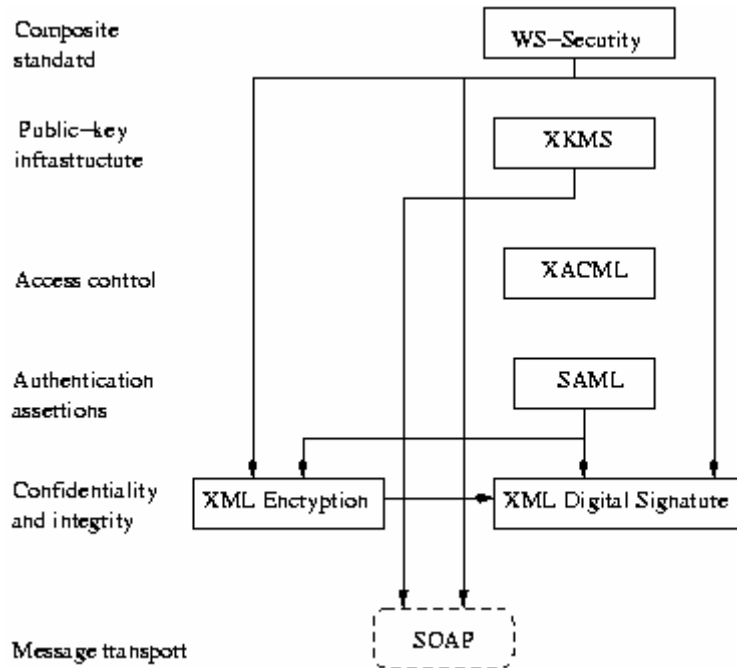


Figure 6.3 XML Security Standards

As indicated in Figure 6.3 SOAP is the core messaging protocol for web services. The SOAP message is constructed of the SOAP envelope which in turn consists of a SOAP header and a SOAP body [103].

XML Encryption and XML Digital Signature are used to provide confidentiality and integrity respectively. They are both based on encryption and digital signature and do not provide any new cryptographic algorithms. Instead they define how to apply well established digital signature/ encryption algorithms to XML.

XML Encryption is used to provide confidentiality to XML documents. Encryption generally follows the symmetric key encryption where the sender and the receiver use the same key for both encrypting and decrypting the data, hence great care must be taken during the transfer of the key and it may be difficult without person to person interaction. Hence public key encryption is used where there are two sets of keys a public key and a private key, when a sender has to send information he makes use of the receiver's public key which is well known

to everybody, the receiver then decrypts the data using the private key which is known to the receiver alone.

A combination of public key cryptography and symmetric cryptography together is a more efficient technique. The symmetric key is used to encrypt the data, and then the data along with the symmetric key is encrypted with the public key and sent to the recipient who uses the private key to decrypt the data and get the symmetric key which was used to encrypt the data.

Listing 6.1(a) A Plain SOAP message without WSS

```
<soap:Envelope>
<soap:Header>
</soap:Header>
<soap:Body>
  <createWorkOrdersResponse>
    <createWorkOrdersResult>
      <WorkOrder>
        <customerID>1</customerID>
        <customerName>Tang</customerName>
        <addressStreet>A Street</addressStreet>
        <addressCity>Sydney</addressCity>
        <addressState>NSW</addressState>
        <addressZip>2006</addressZip>
        <sourceCompany>EE</sourceCompany>
        <appointmentDate>210406</appointmentDate>
      </WorkOrder>
    </createWorkOrdersResult>
  </createWorkOrdersResponse>
</soap:Body>
</soap:Envelope>
```

Listing 6.1(b) The SOAP message with WSS Encryption

```
<soap:Envelope>
<soap:Header>
  <wsse:Security>
    <wsse:BinarySecurityToken ValueType="...">
      MIICnzCCAgigAwIBAgIQBHB1ZCwoIDXbdsxTrNLj
      AMAsGA1UECxEQ2VydDEMMAoGA1...
    </wsse:BinarySecurityToken>
    <xenc:EncryptedKey>
      <xenc:EncryptionMethod Algorithm="..." />
      <KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="... " .../>
        </wsse:SecurityTokenReference>
      </KeyInfo>
    </xenc:EncryptedKey>
  </wsse:Security>
</soap:Header>
```

```

        </wsse:SecurityTokenReference>
    </KeyInfo>
    <xenc:CipherData>
        <xenc:CipherValue>
            iHlscgQVO4uCwztyCBwFzH8ClekMAoG
            A1QBHB1gGjHa2GAKiaTaAgU...
        </xenc:CipherValue>
    </xenc:CipherData>
    <xenc:ReferenceList>
        <xenc:DataReference URI="..." />
    </xenc:ReferenceList>
</xenc:EncryptedKey>
</wsse:Security>
</soap:Header>
<soap:Body>
    <xenc:EncryptedData Id="..." ...>
        <xenc:EncryptionMethod Algorithm="..." />
        <xenc:CipherData>
            <xenc:CipherValue>
                QtzfuLYO/qh45yDxaypPhl/YdH4bJ...
            </xenc:CipherValue>
        </xenc:CipherData>
    </xenc:EncryptedData>
</soap:Body>
</soap:Envelope>

```

The significance of the different components of listing 6.1(b) [38] are explained, a pair of tags of `<wsse:Security>` element is added to the SOAP *header*, in which the required security elements are able to be embedded. The `<wsse:BinarySecurityToken>` element holds the information of the binary security token (e.g., X.509 certificate) used for encryption and the `<xenc:EncryptedKey>` element contains the key for encryption. Inside the `<xenc: EncryptedKey>` element, `<xenc: EncryptionMethod>` specifies the encryption algorithm applied to the encryption and the `<KeyInfo>` element keeps a reference to the `<wsse:BinarySecurityToken>` element. Since the encryption key of the message itself can be encrypted, and so can the content of the SOAP *body*, `<xenc: CipherData>` in the `<xenc: EncryptedKey >` in the *header* provides the encrypted data of the encryption key, and the one in `<xenc: EncryptedData>` in the *body*

contains the encrypted data of the *body* content. A < xenc:ReferenceList> is used as a reference list to the encrypted data of the *body* content. It can also be seen that the <wsse: Security> element and its descendants in the encrypted message make the SOAP message much larger in size than the original message.

While WSS enhances the security of web services, it may lead to performance overheads due to longer networking times to transport the larger SOAP messages and the CPU time involved in processing the SOAP message [38].

XML Digital Signature: Is important in electronic security and it can be used to ensure integrity, authenticity and non-repudiation of data. The important feature of XML digital signature is that it can be used to sign only some parts of an XML document, and also one XML document may have different parts signed for different users this feature distinguishes it from other forms of signing like PGP. There are three types of XML digital structures, namely enveloped signature, enveloping signature and detached signature. For enveloped and enveloping signature, the signed document and the signature are in the same document, in detached signature the signed document and the signature are in separate documents [103]. Digital signatures make use of the standard techniques like DSA-SHA1 and RSA-SHA1 [104].

How are certificates used ?

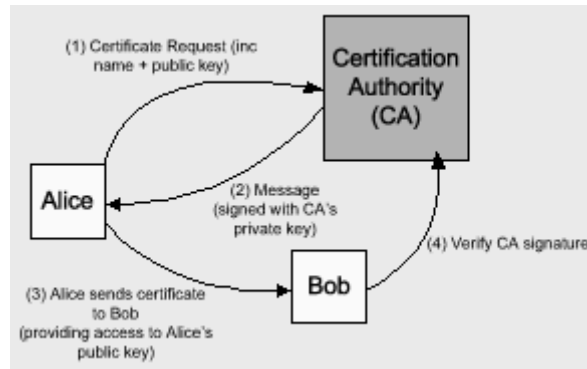


Figure 6.4 How certificates are used

The sequence of events shown in Figure 6.4 [105] is as follows:

- Alice sends a signed certificate request containing her name, her public key, and perhaps some additional information to a CA.
- The CA creates a message from Alice's request. The CA signs the message with its private key, creating a separate signature. The CA returns the message and the signature to Alice. Together, the message and signature form Alice's certificate.
- Alice sends her certificate to Bob to give him access to her public key.
- Bob verifies the certificate's signature, using the CA's public key. If the signature proves valid, he accepts the public key in the certificate as Alice's public key.

As with any digital signature, any receiver with access to the CA's public key can determine whether a specific CA signed the certificate. This process requires no access to any secret information. The preceding scenario assumes that Bob has access to the CA's public key. Bob would have access to that key if he has a copy of the CA's certificate that contains that public key.

X.509 Digital Certificate

The X.509 certificate includes not only a user's name and public key but also other information about the user like e-mail address an authorization to sign other documents etc.All certificates have a valid time duration. A certificate can expire and no longer be valid. The CA can revoke certificates. A list of the revoked certificates is maintained in the CRL (Certificate Revocation List), network users can access the CRL to find out about revoked certificates.

Certificate Stores in .NET and Java: Certificates are stored in safe locations called certificate stores, a certificate store can contain certificates, certificate trust lists(CTL) and certificate revocation list (CRL). In .NET each user has a personal store called "MY" store where that user's certificates are stored, the MY store can be a database, a directory service or another memory location[105].

Java Keytool is a key and certificate management utility. It allows users to manage their own public/private key pairs and certificates. It also allows users to cache certificates. Java Keytool stores the keys and certificates in what is called a keystore. By default the Java keystore is implemented as a file. It protects private keys with a password. A Keytool keystore contains the private key and any certificates necessary to complete a chain of trust and establish the trustworthiness of the primary certificate [106].

XML Key Management Specifications (XKMS): Since both encryption and digital signatures make use of keys, it is important to have a store for these keys and to manage them efficiently, this job is taken up by the XKMS.XKMS defines simple web services for retrieving, validating and registering public keys, thereby shielding clients from the complexity of the potentially underlying public key infrastructure (PKI). XKMS is divided into two main parts, the XML Key

Registration Service Specifications(X-KRSS) and the XML Key Information Service Specification(X-KISS).

X-KRSS defines services in order to register, revoke and reissue keys. In the case of registering a new public key, the key pair generation may either be performed by the client or by the offered service. In case the key pair is generated by the client, the client is required to prove possession of the private key in order to register the public key in either case the KRSS provides mechanisms for authenticating clients.

X-KISS defines two services namely locate and validate. The locate service enables a client to retrieve a public key, or information about a public key. The validate service provides the same functionality as the locate service but also assures that the returned information meets specific validation criteria (e.g. by validating the X.509 certificate).

Extensible Access Control Markup Language (XACML)

XAML is an XML specification for expressing fine grained information access policies in XML documents or any other electronic resource. It specifies the different access rights for different groups of people, the XML access control lists is generally 4 tuples: subjects, target objects, permitted action, provision.

Security Assertion Markup Language (SAML)

SAML defines an XML framework for exchanging authentication and authorization information. SAML addresses authentication and provides a mechanism for transferring authentication and authorization decisions between cooperating entities. SAML provides a very important feature of Single Sign On(SSO) where a user can log onto one service and make use of a host of other services for e.g. when a person logs onto their email account they can log onto chat, Facebook and a host of other applications. This facility is enabled by SAML which passes on the login information onto the other web sites.

All the above information is pertaining to XML security. WSS therefore takes care of message integrity and data confidentiality. WSS is flexible and designed to be used as the basis for securing web services within a wide variety of security models including PKI, Kerberos and Secure Socket Layer (SSL). Specifically this specification provides support for multiple security token formats, trust domains, signature formats and encryption technologies [107].

WS-SecurityPolicy		
Web Services Policy		
Web-SecureConversation		
WS-Trust		
WS-Security		
SOAP	XML Signature	XML Encryption
XML		

Figure 6.5 The conceptual relationship between XML and web services security standards

WS-Trust builds on WS-Security, and it provides functionality which can be used by WS-Security, WS-Secure conversation builds on WS-Security and WS-Trust, finally WS-Security Policy extends Web services Policy in order to facilitate the use of WS-Security, WS-Secure Conversation and WS-Trust as shown in Figure 6.5.

Development support for these standards can be found in the Web Services Interoperability Toolkit for Java and in the Windows Communication Foundation for .NET. These standards are also supported by XML firewalls. Environments such as Apache WSS4J [Apache Software Foundation 2006], IBM Web Sphere [IBM Corporation 2006], Microsoft Web services Enhancements (WSE) [Microsoft Corporation 2004] and Windows Communication Foundation (WCF)

[Microsoft Corporation 2006], provide tools and libraries for building web services that are secured via the mechanisms of WS-Security and related specifications.

Disadvantage of Security Specifications:

Properly designing and securing web service applications are important and it is not just about using security standards. Developers must understand both the limitations and drawbacks to security standards in order to fully secure their web services. The number of security standards available to users is large and confusing. SSL which is now replaced by TLS, WSS, Digital Signature Services, XML Encryption, XML-Signature, eXtensible Access Control Markup Language (XACML), SAML, XKMS etc. These specifications could be confusing for somebody new to web service standards and it could increase the effort for developers to build web services, while providing a thin veneer of actual web service security.

Just using these specifications will not assure complete web service security, these specifications will have to be used correctly. In many cases it is seen that it would be needed to combine more than one type of specification, but since most system designers are not security experts, they will not know intuitively what they are missing. Nor will they have any reason to suspect they need anything beyond the standard they know and use [108].

6.6.3.NET Framework architecture

Prior to .NET technologies, languages like C++ and C were directly converted to machine code for particular computer architecture. This was called unmanaged code because features like security, memory management and type checking was managed by the operating system itself. .NET applications are built with managed code, the source code is compiled into an Intermediate Language (IL). The Common Language Runtime (CLR) framework of .NET is responsible for converting the IL to machine code. CLR enables Code Access Security (CAS) which is a very different access model than the one that Windows OS has [109].

Security in the .NET environment

Web services in the .NET environment are hosted by IIS and thus the built in security features of IIS can be leveraged in this environment. SSL is enabled for HTTP to provide confidentiality and integrity of the data being transferred over HTTP. Non repudiation is provided by the use of client side X.509 certificates. Once SSL is enabled, all the data sent over the connection will be encrypted and signed.

IIS provides multiple authentication mechanisms: basic, digest, Integrated windows authentication or X.509 certificates. Any of these authentication mechanisms can be enabled for the particular directory that hosts the web service and this will allow the client to present the appropriate credentials and authenticate to the IIS. In addition the web.config file for web service needs to be modified to indicate that “windows” authentication should be used, also anonymous access needs to be disabled in IIS [102]. To provide authorization, “Code Access Security” mechanism is provided by .NET. This allows the service provider to check if the user is authorized to access this information.

Security in Java

- i) SSL security can be provided in Java environment also. This provides confidentiality and integrity of the data being transferred over HTTP.
- ii) The only authentication mechanism allowed in this environment is basic authentication. This requires appropriate usernames and passwords to be added to the web server.

6.7 Work Done

A platform for working with interoperable web services is created. A secure web service is built in Java using the Netbeans platform, the web service performs two simple operations, the first is to find the sum of two numbers and the second is to find simple interest given values for principal, time and rate of interest. The

client application is developed in C#. Security is enforced by using the simple username password authentication with symmetric key. Hence to access the web service the username and password are provided, as well as the certificate used for encrypting and decrypting the SOAP messages transported between the web service and the client.

6.7.1 Editing Web Service Attributes

Security can be added in Netbeans by changing the web service attributes. The web service can be configured to use secure service and then specify the service mechanism that is to be used. The following two mechanisms are used.

i) User Authentication with Symmetric Key: This mechanism only allows users specified in the Glassfish server to access the web service. Also, this mechanism uses the same key for encrypting and decrypting the messages.

ii) Mutual Certificate Security: In this mechanism the xws-security-server and the xws-security-client key pair are used for data confidentiality.

The security that is configured using netbeans is automatically stored in the sun.xml and sun-web.xml configuration files .

User Authentication with Symmetric Key: Encryption using symmetric key exchange works as represented in Figure 6.6. The client and the server use the same shared key to encrypt and decrypt information.

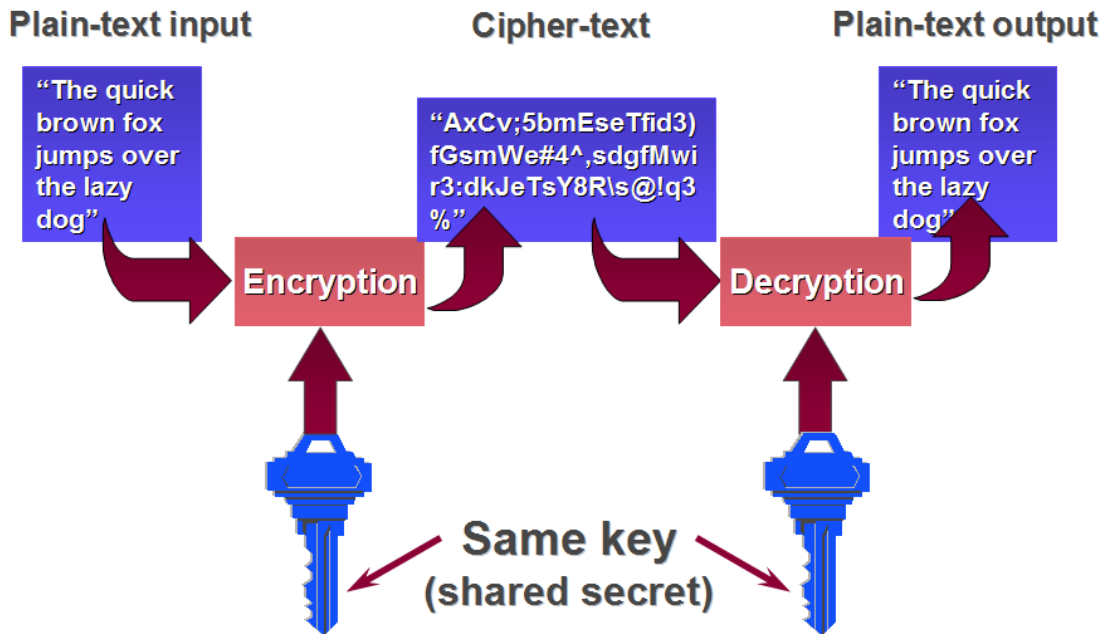


Figure 6.6 Working of Symmetric key Encryption

When the client wants to invoke the service, he sends his username and password encrypted using the shared symmetric key to the server. The server decrypts the username and password and only if the user is a registered user whose details are available in its directory than the client is allowed to access the services offered. The working of the symmetric key is shown in Figure 6.6 [110].

Mutual Certificate Security

A C# client is developed to invoke the web service since the web service that is invoked is a secure web service it is required that the client and the server exchange their certificates, but before the certificate can be used by the WCF client it must be installed on the windows machine.

The certificate is imported into the Microsoft Management Console (MMC). The MMC is a tool provided by windows that helps system administrators create flexible user interfaces and customize administration tools. There are two certificates required by the web service client they are xwsecurityserver and xwsecurityclient which are available in the *.JKS (Java Keystore) format. .NET identifies certificates only if they are in the .PFX or .PFB and (.SST) Microsoft

Serialized Certificate store, so before the certificate can be imported it must be converted from .JKS format to any of the acceptable .NET formats as shown in Figure 6.7.

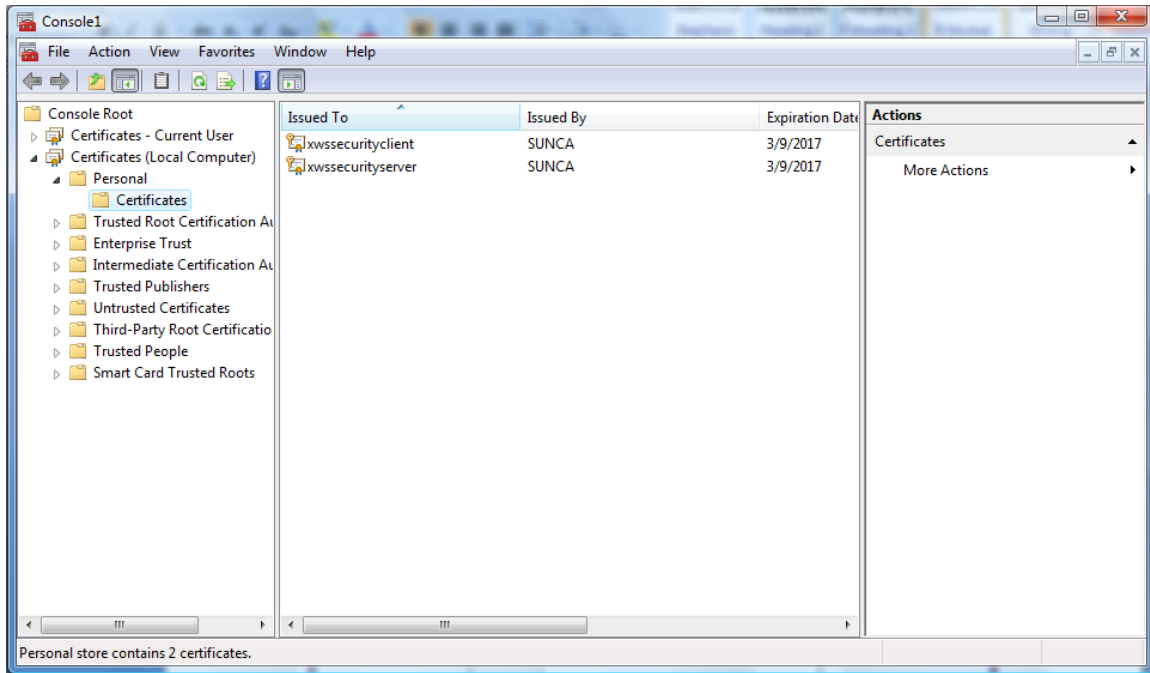
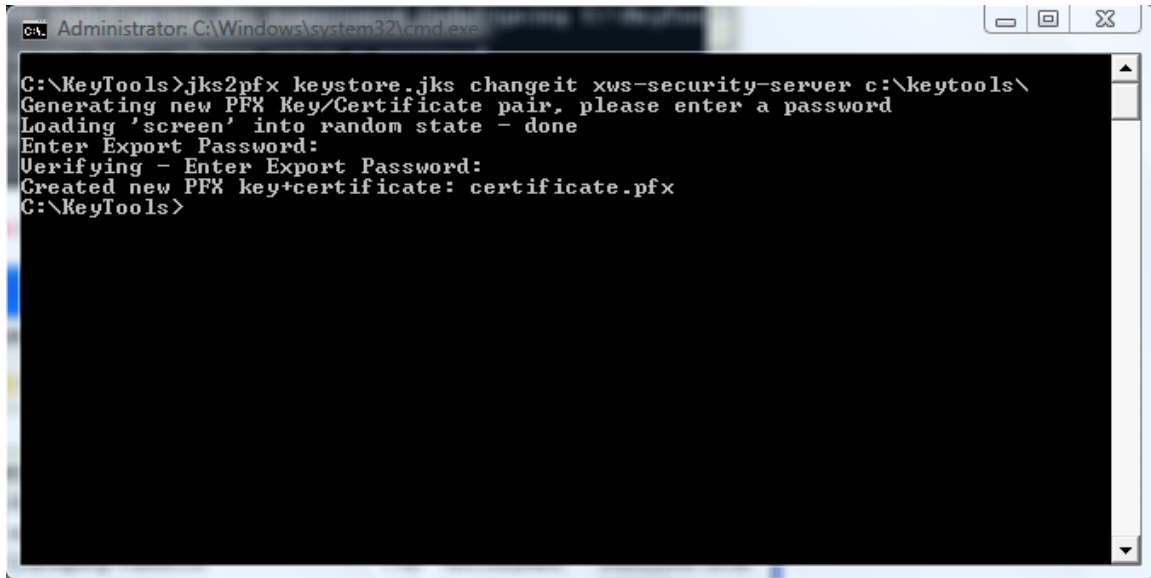


Figure 6.7 Microsoft Management Console Utility

This conversion is done using the tool JKS2PFX.bat as shown in Figure 6.8.



```
C:\KeyTools>jks2pfx keystore.jks changeit xws-security-server c:\keytools\  
Generating new PFX Key/Certificate pair, please enter a password  
Loading 'screen' into random state - done  
Enter Export Password:  
Verifying - Enter Export Password:  
Created new PFX key+certificate: certificate.pfx  
C:\KeyTools>
```

Figure 6.8 JKS2PFX Tool

After both the certificates are converted to the proper format they are imported into MMC. Once imported, the reference to these certificates in the web service client can be provided either programmatically or by configuring the security in the app.config file.

6.7.2 Using ClientCredentials.SetCertificate() Method

The client.ClientCredentials.SetCertificate() method is used to specify the certificate to be used to represent the web service client. The method Definition is as follows:

```
public void SetCertificate(  
    StoreLocation storeLocation,  
    StoreName storeName,  
    X509FindType findType,  
    Object findValue  
)
```

6.7.3 Configuring Security via app.config file

References to the client and server certificates can also be provided via the app.config file. The code snippets to be added to the app.config file to provide the necessary security measures are:

```
<behaviors>
  <endpointBehaviors>
    <behavior name="secureBehavior">
      <clientCredentials>
        <clientCertificate
          storeName="My"
          storeLocation="CurrentUser"
          findValue="xws-security-server"
          x509FindType="FindBySubjectName" />
      </clientCredentials>
    </behavior>
  </endpointBehaviors>
</behaviors>
```

This Behavior known as secureBehavior represents the certificate to be used by the client. An Identity needs to be provided to the certificate used by the web service before the application can be run.

```
<identity>
  <certificateReference findValue="xwssecurityserver"
    storeLocation="LocalMachine" storeName="My"
    x509FindType="FindBySubjectName" isChainIncluded="false"/>
</identity>
```

6.7.4 Source Code of the WCF Client in C#

Finally, a new Visual C# console application is created, adding a service reference. The WSIT endpoint provided in this case is:

Listing 6.2 SetCertificateMethod()

<http://localhost:8080/MyWebService/CalculatorWSService?wsdl>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```

namespace UserAuthenticationProject
{
    class Program
    {
        static void Main(string[] args)
        {
            ServiceReference1.CalculatorWSClient client = new
UserAuthenticationProject.ServiceReference1.CalculatorWSClient();
            client.ClientCredentials.UserName.UserName = "sujala";
            client.ClientCredentials.UserName.Password = "shetty";

            client.ClientCredentials.ClientCertificate.SetCertificate(System.Security.Cryptogr
aphy.X509Certificates.StoreLocation.LocalMachine, System.Security.Cryptograph
y.X509Certificates.StoreName.My, System.Security.Cryptography.X509Certificate
s.X509FindType.FindBySubjectName, "xwssecurityserver");
            Console.WriteLine("Sum:" + client.add(2, 3) + "\n Interest:" + client.interest(1000,
5, 2));
            Console.ReadLine();
        }
    }
}

```

As can be seen the certificate reference for the client is provided with the help of the SetCertificate() Method. The app.config file generated by the process is:

Listing 6.3 The Identity to the server certificate is provided in the app.config file.

```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <system.serviceModel>
        <bindings>
            <customBinding>
                <binding name="CalculatorWSPortBinding">
                    <security defaultAlgorithmSuite="Basic128Rsa15"
authenticationMode="UserNameForCertificate" requireDerivedKeys="false"
securityHeaderLayout="Strict" includeTimestamp="true"
keyEntropyMode="CombinedEntropy"
messageProtectionOrder="SignBeforeEncrypt"
messageSecurityVersion="WSSecurity11WSTrustFebruary2005WSecureConversa
tionFebruary2005WSSecurityPolicy11BasicSecurityProfile10"
requireSignatureConfirmation="false">
                        <localClientSettings cacheCookies="true"
detectReplays="true" replayCacheSize="900000" maxClockSkew="00:05:00"
maxCookieCachingTime="Infinite" replayWindow="00:05:00"
sessionKeyRenewalInterval="10:00:00"

```

```

sessionKeyRolloverInterval="00:05:00"
reconnectTransportOnFailure="true" timestampValidityDuration="00:05:00"
cookieRenewalThresholdPercentage="60" />
    <localServiceSettings detectReplays="true"
issuedCookieLifetime="10:00:00" maxStatefulNegotiations="128"
replayCacheSize="900000" maxClockSkew="00:05:00"
negotiationTimeout="00:01:00" replayWindow="00:05:00"
inactivityTimeout="00:02:00" sessionKeyRenewalInterval="15:00:00"
sessionKeyRolloverInterval="00:05:00"
reconnectTransportOnFailure="true" maxPendingSessions="128"
maxCachedCookies="1000" timestampValidityDuration="00:05:00" />
    <secureConversationBootstrap />
    </security>
    <textMessageEncoding maxReadPoolSize="64"
maxWritePoolSize="16" messageVersion="Soap11WSAddressing10"
writeEncoding="utf-8">
        <readerQuotas maxDepth="32" maxStringContentLength="8192"
maxArrayLength="16384" maxBytesPerRead="4096"
maxNameTableCharCount="16384" />
    </textMessageEncoding>
    <httpTransport manualAddressing="false"
maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
allowCookies="false" authenticationScheme="Anonymous"
bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
keepAliveEnabled="true" maxBufferSize="65536"
proxyAuthenticationScheme="Anonymous" realm="" transferMode="Buffered"
unsafeConnectionNtlmAuthentication="false" useDefaultWebProxy="true" />
    </binding>
    </customBinding>
</bindings>
<client>
    <endpoint
address="http://localhost:8080/MyWebService/CalculatorWSService"
binding="customBinding" bindingConfiguration="CalculatorWSPortBinding"
contract="ServiceReference1.CalculatorWS" name="CalculatorWSPort">
        <identity>
            <certificateReference findValue="xwssecurityserver"
storeLocation="LocalMachine" storeName="My"
x509FindType="FindBySubjectName" isChainIncluded="false"/>
        </identity>
    </endpoint>
</client>
</system.serviceModel>
</configuration>

```

6.8 SOAP Messages Log

SOAP messages that are transferred between the web service and the web service client can be logged using an utility called TcpMon. TcpMon is an open source utility developed by the Apache foundation to log HTTP or SOAP message traffic. It acts as a listener, waiting for inputstream at one port and

redirects it at another port, logging the messages being sent along the way as shown in Figure 6.9.

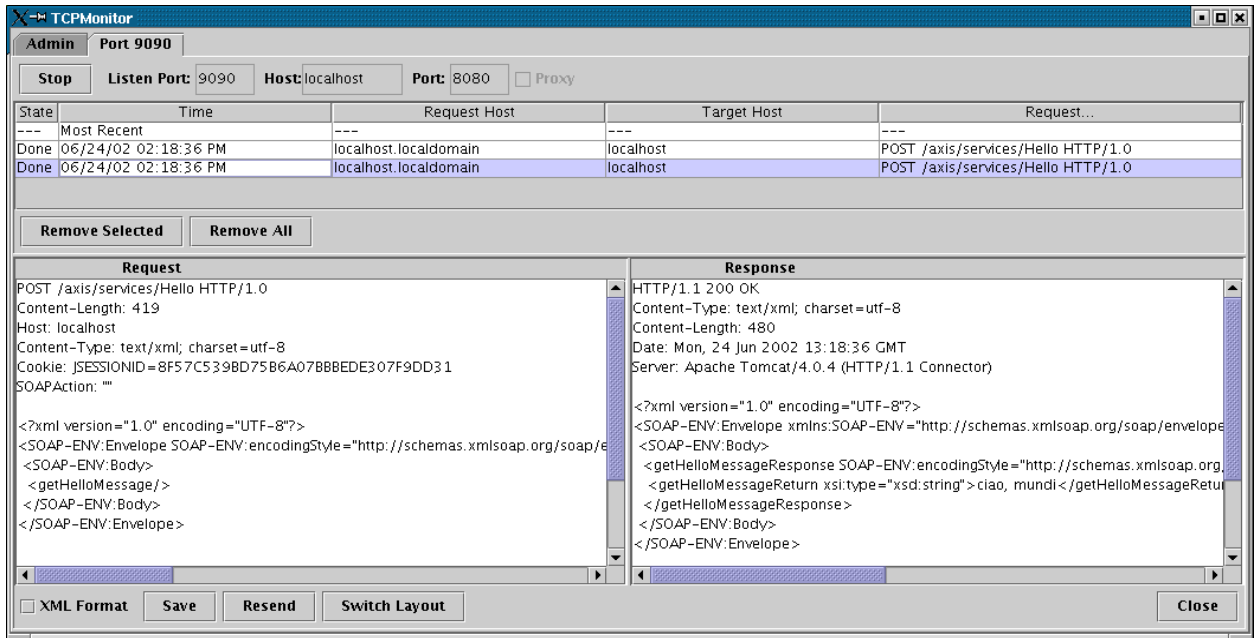


Figure 6.9 TCPMon Utility

Listing 6.4 SOAP Request Log

```
POST /MyWebService/CalculatorWSService HTTP/1.1
SOAPAction: "http://webservice.com.org/CalculatorWS/addRequest"
Accept: text/xml, multipart/related, text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Content-Type: text/xml;charset="utf-8"
User-Agent: JAX-WS RI 2.1.3.1-hudson-417-SNAPSHOT
Host: 127.0.0.1:8081
Connection: keep-alive
Content-Length: 7625
```

```
<?xml version="1.0" ?>
  <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
    1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
    1.0.xsd" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    xmlns:exc14n="http://www.w3.org/2001/10/xml-exc-c14n#">
    <S:Header>
      <To xmlns="http://www.w3.org/2005/08/addressing"
        wsu:Id="_5006">http://127.0.0.1:8081/MyWebService/CalculatorWSService</To>
      <Action xmlns="http://www.w3.org/2005/08/addressing"
        wsu:Id="_5005">http://webservice.com.org/CalculatorWS/addRequest</Action>
      <ReplyTo xmlns="http://www.w3.org/2005/08/addressing" wsu:Id="_5004">
        <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
```

```

</ReplyTo>
<MessageID xmlns="http://www.w3.org/2005/08/addressing"
wsu:Id="_5003">uuid:6876e04b-876b-4413-9e4a-9544b92d356b</MessageID>
<wsse:Security S:mustUnderstand="1">
  <wsu:Timestamp xmlns:ns10="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512" wsu:Id="_3">
    <wsu:Created>2008-11-13T01:18:51Z</wsu:Created>
    <wsu:Expires>2008-11-13T01:23:51Z</wsu:Expires>
  </wsu:Timestamp>
  <xenc:EncryptedKey xmlns:ns10="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512" Id="_5002">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
    <ds:KeyInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="keyInfo">
      <wsse:SecurityTokenReference>
        <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509SubjectKeyIdentifier" EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary">dVE29syFW/iD1la3ddePzM6IW0=</wsse:KeyIdentifier>
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
    <xenc:CipherData>

<xenc:CipherValue>SAPmpD1G7am4TCsVShGf0cAymvfMEd7ce2In8CKSc3zGqq3AmW8cf+D2
smIfYMFLU/SeLJ8XZF6ZpyWLorKNV46nuUGxldsrXyCfuE0XCt0LYIScByupZy+Q9McrGjssa7HX
2saNE3FMPTAqovr+E21oH5BxbZ8bBd2JLw26xSE=</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedKey>
  <xenc:ReferenceList xmlns:ns17="http://docs.oasis-open.org/ws-sx/ws-
secureconversation/200512" xmlns:ns16="http://www.w3.org/2003/05/soap-envelope" xmlns="">
    <xenc:DataReference URI="#_5008"></xenc:DataReference>
    <xenc:DataReference URI="#_5009"></xenc:DataReference>
  </xenc:ReferenceList>
  <xenc:EncryptedData xmlns:ns10="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"
Type="http://www.w3.org/2001/04/xmlenc#Element" Id="_5009">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
    <ds:KeyInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="keyInfo">
      <wsse:SecurityTokenReference>
        <wsse:Reference ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-
message-security-1.1#EncryptedKey" URI="#_5002"/>
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
    <xenc:CipherData>

<xenc:CipherValue>ZR1gaiwaW32MzdBcY2dKjR0pci/EvSAUFahBmbaFMQAeGUYPMzCMK+6i
uwIP3i74cx7GR+O+Lpyqv96x2kiX8ekL2iqPm1dDGO5+4eBjz/ozjK9shW12NI8FjjEw5WDWmsD
FT5Ysd0+z3SGrjVDxZDK8dws6L7ZuTOg8cR+sasrG/FiJlt+u/IAMXagE21+rf8UV/4BXqkbpHal7
BnglgA4WlhnuScZbb5WiGaetVKiGEft2GnREwfWkH47ehfNoZrFngDE4GtaxUC+0P7WeQdPOpt
EzPPq2wEzCsm4HABzdDtmp14YqQR4lzD/rHZMC225DfOP9TbqBwxboiASpvrR2C77VLneUQ+
0wyJGW/8xjlg2l/flid9uoJw0kQOTTaJ0FtpYH/3i4gb4bZYCeYdUqdAHjT8cafCjtbdgoLuZ6gHBdLLA
EX1WCihzL6ljvE288Mn5vuyqgUtSSI9SFngBmJO9sUJ44zdX08B/TPqoHu83YM0WUpyM43ORU
dpt</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>

```

```

<ds:Signature xmlns:ns10="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512" Id="_1">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
      <exc14n:InclusiveNamespaces PrefixList="wsse S"/>
    </ds:CanonicalizationMethod>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
    <ds:Reference URI="#_5003">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <exc14n:InclusiveNamespaces PrefixList="S"/>
        </ds:Transform>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>rqQBOAxFTbtazThQqvIM7ZinLm4=</ds:DigestValue>
    </ds:Reference>
    <ds:Reference URI="#_5004">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <exc14n:InclusiveNamespaces PrefixList="S"/>
        </ds:Transform>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>5Ab1ebo4/FraGgck/A8iDx1J9+I=</ds:DigestValue>
    </ds:Reference>
    <ds:Reference URI="#_5005">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <exc14n:InclusiveNamespaces PrefixList="S"/>
        </ds:Transform>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>CCmsv+bhlleGmSDIz3qdbPXoJ+k=</ds:DigestValue>
    </ds:Reference>
    <ds:Reference URI="#_5006">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <exc14n:InclusiveNamespaces PrefixList="S"/>
        </ds:Transform>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>2vKHblosw5rGfPsRRWYqVpqDv2s=</ds:DigestValue>
    </ds:Reference>
    <ds:Reference URI="#_5007">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <exc14n:InclusiveNamespaces PrefixList="S"/>
        </ds:Transform>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>FxqMoXAISeHNX38hCF1IqjRFotw=</ds:DigestValue>
    </ds:Reference>
    <ds:Reference URI="#_3">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <exc14n:InclusiveNamespaces PrefixList="wsu wsse S"/>
        </ds:Transform>
      </ds:Transforms>

```

```

        </ds:Transform>
    </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlsig#sha1"/>
    <ds:DigestValue>H5zXv2d8Eyd7cY//wIMFci2277o=</ds:DigestValue>
</ds:Reference>
<ds:Reference URI="#uuid_2096f1c1-fde1-46be-9bab-2aaf17b9075">
    <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <exc14n:InclusiveNamespaces PrefixList="wsu wsse S"/>
        </ds:Transform>
    </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlsig#sha1"/>
    <ds:DigestValue>d11zlh6pWtFmUJhHC3/g/Cd8sZo=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>dG75vk0PkoLf+Ua3Q7tWpwfGgxU=</ds:SignatureValue>
<ds:KeyInfo>
    <wsse:SecurityTokenReference wsu:Id="uuid_4c4e08b3-81b8-4bfd-b808-
e77be6448f22">
        <wsse:Reference ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-
message-security-1.1#EncryptedKey" URI="#_5002"/>
    </wsse:SecurityTokenReference>
</ds:KeyInfo>
</ds:Signature>
</wsse:Security>
</S:Header>
<S:Body wsu:Id="_5007">
    <xenc:EncryptedData xmlns:ns10="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"
Type="http://www.w3.org/2001/04/xmlenc#Content" Id="_5008">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
        <ds:KeyInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="keyInfo">
            <wsse:SecurityTokenReference>
                <wsse:Reference ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-
message-security-1.1#EncryptedKey" URI="#_5002"/>
            </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
            <xenc:CipherValue>6qBlcF23LRJ/idTkCra6uCtOTxl4wHXzoyLgU0fjhkhDYgx/2BiK2DagPagGtrw
M8IKfqMbXII/wYyZS/N49ZTxq5MeHfCkohEGR6as8rQFe2IkYOygKmDUQiROJSDxc</xenc:Ciph
erValue>
        </xenc:CipherData>
    </xenc:EncryptedData>
</S:Body>
</S:Envelope>

```

Listing 6.5 SOAP Response Log

HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
Content-Type: text/xml;charset="utf-8"
Transfer-Encoding: chunked
Date: Thu, 13 Nov 2008 01:18:53 GMT

14ca

```
<?xml version="1.0" ?>
  <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
xmlns:exc14n="http://www.w3.org/2001/10/xml-exc-c14n#">
  <S:Header>
    <To xmlns="http://www.w3.org/2005/08/addressing"
wsu:Id="_5005">http://www.w3.org/2005/08/addressing/anonymous</To>
    <Action xmlns="http://www.w3.org/2005/08/addressing"
wsu:Id="_5003">http://webservice.com.org/CalculatorWS/addResponse</Action>
    <MessageID xmlns="http://www.w3.org/2005/08/addressing"
wsu:Id="_5002">uuid:21236ef8-c995-4faf-b46f-e3c4f4c2ac48</MessageID>
    <RelatesTo xmlns="http://www.w3.org/2005/08/addressing"
wsu:Id="_5004">uuid:6876e04b-876b-4413-9e4a-9544b92d356b</RelatesTo>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp xmlns:ns10="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512" wsu:Id="_3">
        <wsu:Created>2008-11-13T01:18:53Z</wsu:Created>
        <wsu:Expires>2008-11-13T01:23:53Z</wsu:Expires>
      </wsu:Timestamp>
      <xenc:ReferenceList xmlns:ns17="http://docs.oasis-open.org/ws-sx/ws-
secureconversation/200512" xmlns:ns16="http://www.w3.org/2003/05/soap-envelope" xmlns="">
        <xenc:DataReference URI="#_5007"></xenc:DataReference>
      </xenc:ReferenceList>
      <ds:Signature xmlns:ns10="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512" Id="_1">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <exc14n:InclusiveNamespaces PrefixList="wsse S"/>
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
          <ds:Reference URI="#_5002">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <exc14n:InclusiveNamespaces PrefixList="S"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>gqgZkOQhK9+VvtchDOp5iYxXCsA=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#_5003">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
```

```

        <exc14n:InclusiveNamespaces PrefixList="S"/>
    </ds:Transform>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlsig#sha1"/>
<ds:DigestValue>dXy4TbqNvG4W/ltXHYZG6tpWd7w=</ds:DigestValue>
</ds:Reference>
<ds:Reference URI="#_5004">
    <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <exc14n:InclusiveNamespaces PrefixList="S"/>
        </ds:Transform>
    </ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlsig#sha1"/>
<ds:DigestValue>ATqT5G73L44ooJBSRt8ZCLbC2go=</ds:DigestValue>
</ds:Reference>
<ds:Reference URI="#_5005">
    <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <exc14n:InclusiveNamespaces PrefixList="S"/>
        </ds:Transform>
    </ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlsig#sha1"/>
<ds:DigestValue>Nd/8wVmBdLowQKMbIBRYK+6xcjA=</ds:DigestValue>
</ds:Reference>
<ds:Reference URI="#_5006">
    <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <exc14n:InclusiveNamespaces PrefixList="S"/>
        </ds:Transform>
    </ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlsig#sha1"/>
<ds:DigestValue>C5cQ9qlSuBacClfkOdpHrv7hBRY=</ds:DigestValue>
</ds:Reference>
<ds:Reference URI="#_3">
    <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <exc14n:InclusiveNamespaces PrefixList="wsu wsse S"/>
        </ds:Transform>
    </ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlsig#sha1"/>
<ds:DigestValue>/bcGg9463R9cM17UAhaUgqCy7AM=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>wk0Ch/OI2bnQI9ykesv3ohJOE0c=</ds:SignatureValue>
<ds:KeyInfo>
    <wsse:SecurityTokenReference>
        <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-
message-security-1.1#EncryptedKeySHA1" EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary">Tyrr2//3sWXA1wGgXw5yTI6ZFqY=</wsse:KeyIdentifier>
    </wsse:SecurityTokenReference>
</ds:KeyInfo>
</ds:Signature>
</wsse:Security>
</S:Header>
<S:Body wsu:Id="_5006">

```

```

    <xenc:EncryptedData xmlns:ns10="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"
Type="http://www.w3.org/2001/04/xmlenc#Content" Id="_5007">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
    <ds:KeyInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="keyInfo">
        <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-
message-security-1.1#EncryptedKeySHA1" EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary">Tyrr2//3sWXAiwGgXw5yTI6ZFqY=</wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
    </ds:KeyInfo>
    <xenc:CipherData>
        <xenc:CipherValue>RBk7cq7pGbxfl6CumsF+NcYXxgP+wAY0TRho7GBtQ2QCoo7no/u4Lacs7u
2xMGATZI4GsSPZNHfPOKrzoBZZKl/mc5NsPL+W1Kx33zZf2Pdz5pOtxdcibKXcM1OQJ52qjiEUJ
NfFYlr31rCu6qHx+w==</xenc:CipherValue>
    </xenc:CipherData>
    </xenc:EncryptedData>
</S:Body>
</S:Envelope>

```

As indicated by the highlighted text it is observed that in the SOAP request there is an exchange of security information in the XML header after the verification of the certificate by the glass fish server then there is the flow of data from the client to the server in the encrypted format. Whereas in the SOAP response log there is no exchange of authentication information only the values computed by the web service are sent back to the client in encrypted form.

6.9 Conclusion: Because of the wide use of web services in security intensive operations like banking and business operations the security issues that arise in web services is of utmost significance. The security implementations in Java and .NET are studied. These are the two most widely used platforms for developing web services. In web services data is finally transmitted as XML documents, XML security as well as web service security is studied. The platform that has been developed allows web services from different platforms to exchange certificates with each other and then to transmit data in a secure fashion. This platform can be used by young researchers to test many more security specifications.

Conclusion and Future Scope of Work

Web services are a very powerful technology which enables machine to machine interaction across different platforms, this interoperable feature of web services is mainly due to the fact that web services are based on XML which is a platform neutral language. In order to understand the working of web services the building blocks of web services like UDDI, WSDL and SOAP are studied.

Web services find usage in a number of applications ranging from small applications for currency conversion to a big supply chain management or a travel website like Travelocity which automates the process of ticket purchase, hotel reservation and rent a car booking. Web services could be of great use when large volumes of data need to be analyzed.

Data Mining is a very important process in data analysis and forecasting. As a case study a desk top data mining application is considered and converted it to a web service. The output obtained by the standalone application is programatically verified to be the same as that of a web service hence confirming the belief that when an application is converted to a web service there is no loss of precision. The advantage of using a web service is that the data mining application need not be loaded on the user's system thus saving memory space and providing the benefit that the web service can be accessible to a number of users. Many organizations may not even require to perform data mining operations very frequently on their data in such cases it would not even be needed to load the data mining application on their system , in such cases it would be an ideal situation to use the web service which is hosted at a different location.

Future extension of this work could be to consider a distributed data mining application such that multiple processors work on the distributed data in parallel.

Another important extension of the thesis could be to secure the SOAP messages by encrypting the messages such that confidential data can be transmitted without loss of confidentiality.

It is noted that in order to harness the full potential of web services, web services should be accessible to mobile devices. In today's highly volatile global market it is essential for people to be always accessible to information. Mobile devices satisfy this requirement of anytime anywhere accessibility. Therefore tapping this potential market, the developed application Mobilink enables people to access web services on their mobile. Another important feature of this application is that it is accessible to people even on their desktops and they can create the web service as per their needs to interact with the back end database. Mobilink adds a level of abstraction to a user using this application. A further extension of this work could be a search for video files, audio files or image files. If video files are to be processed the application would have to interface with graphics files and high end image processing may be required to provide a real time experience. Mobiles are very commonly used with context based or location based services like Google maps, the application can be extended to work with GPS systems. People are increasingly using their mobiles for bank transactions and other secure transactions, it would be a good idea to add security to the application such that all transactions occur in a secure manner hence giving a feeling of security to the client to use a mobile for secure transactions.

Interoperability is the key word which led to the success of web services, but in spite of a lot of research and analysis done in this area it is seen that certain issues still exist when transmitting data between Java web service and a .Net client or vice versa, tests are conducted and it is shown that issues arise when rounding up decimals, when sending objects containing different data types or when sending across an array with null data elements among other issues. These issues would continue to exist because of the difference in interpretation of data in different ways by the different platforms, the clients using these data

types should take extra precaution when using these specific data types. Work could be extended to test more data types and interoperability across different platforms like Oracle could be tested.

For the success of any system, the system should be able to earn the confidence of its users. This confidence can be earned only when a system is secure. In the last phases of this thesis a secure platform has been created for a web service and a web service client to exchange security information by the exchange of certificates. Once certificates are exchanged and the web service and the client authenticate each other, secure data could be exchanged between the two. Work could be extended to include more forms of security exchange like SAML, SSL, public key encryption etc. A performance analysis test can be conducted taking different security implementations and considering different loads on the web service.

Specific Contributions

During the course of this work four specific tasks were accomplished.

- i) Conversion of a data mining desk top application to a web service and verifying programmatically that the output obtained on processes like clustering, classification and text classification when implemented as a web service is exactly the same as that of a desktop application. This test result is very significant because it proves that a number of everyday applications could be converted to web services thereby increasing the number of services that could be offered to users. There are a number of such services which are already available like file conversion from one format to another like word to pdf etc. Software like Microsoft Office may also be offered as a service. Our application can be offered as a service to a number of people who could invoke the web service when they require any kind of analysis on their data.
- ii) Mobilink the application developed gives users the flexibility of invoking a developed web service either from the desktop or from a mobile device. The beauty of this application is that it abstracts the user about the nitty gritty of developing web services. A front end GUI is provided which enables the developer to design the web service which performs a search in the back end data base as per the requirement. The mobile web service client is a great application for people who are constantly on the move.
- iii) Test cases are designed to check the various interoperability issues that arise when web services and clients are developed on different platforms. These issues are identified and programmatically verified.
- iv) A platform for exchange of certificates between interoperable web services is developed. The format in which certificates are represented is different in different platforms. It therefore becomes important to convert the certificates to a form which can be understood by both platforms, this is taken care in the developed platform. The exchange of authentication information can be tracked by the status of the SOAP request and response messages, this task is accomplished and security data exchange is verified.

List of References

1. New to SOA and Web Services, IBM Developer Works”, available at: <http://www.ibm.com/developerworks/webservices/newto/websvc.html>
2. Yuhui Chen and Alexander Romanovsky., ”Improving the Dependability of Web services Integration”, IEEE IT Pro, May/June 2008. pp.29-35.
3. Rohit Dhand., ”Web Services: A trend shift from conventional Distributed Computing Model”, IEEE Computer Society, 2009 ,pp.313-317.
4. Khalil A. Abuosba and Asim A. El-Sheikh. Formalizing Service Oriented Architectures, IEEE IT Pro, July/August 2008, pp.34- 38.
5. Jan Bosch. ”Service Orientation in the Enterprise”, IEEE Computer Society, November 2007. pp. 51- 56.
6. Neal Levitt. ”Are Web Services Finally Ready to Deliver?” IEEE Computer, November 2004,pp. 14-18.
7. Pro, March/April 2008, pp 24-30. Yuhong Yan, Yong Liang, Xinge Du, Hamadou Saliah- Hassane and Ali Ghorbani, IEEE Computer Society IT Pro, March/April 2006 , pp 27- 33.
8. Lihui Lei and Zhenhua Duan, ”Integrating AJAX and Web Services for Cooperative Image Editing”, IT Pro, May|June 2007, pp.25- 29.
9. R.Yuan, L. Zunchao and Feng Boqin, ”Research on Reused- Based Web Service Composition”, Academic J. Xiám Jiao Tong Univ.(English ed.), 2005,vol.17,no.1,pp.10-14,48.
10. Firat Kart, Louise E. Moser, and Michael Melliar-Smith, ”Building a Distributed E-Healthcare System using SOA”, IEEE computer Society IT.
11. Pilioura. T., Tsalgatidou. A., Hadjiefthymiades.S., Scenarios of using web services in m-commerce. SIGecom Exch. **3**(4), 28–36 (2003). DOI <http://doi.acm.org/10.1145/844351.844356>.
12. Yang. X, Bouguettaya. A., Adaptive data access in broadcast-based wireless environments. IEEE Trans. Knowl. Data Eng. 2005, **17**(3), 326–338.

13. Yang. X, Bouguettaya. A, Medjahed. B, Long. H, He. W., Organizing and accessing web services on air. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.*, 2003, **33**(6), 742–757.
14. Min Tian, Andreas Gramm, Hartmut Ritter, Jochen Schiller and Thiemo Voigt, “Adaptive QoS for Mobile Web Services through Cross-Layer Communication”, *IEEE Computer Society*, February 2007, pp 59-63.
15. Xu yang and Athman Bouguettaya, “Semantic Access to Multichannel M-Services”, *IEEE Transactions on Knowledge and Data Engineering*, Vol 21, No2, February 2009, pp.259-260.
16. Christian S. Jensen, Carmen Ruiz Vicente, and Rico Wind, “User Generated Content: The Case for Mobile Services”, *IEEE Computer*, December 2008, pp.116-118.
17. Ariel Pashtan, Andrea Heusser and Peter Scheurmann “Personal Service Areas for Mobile Web Applications”, *IEEE Internet Computing*, November-December 2004, pp. 34-39.
18. Claudia Canali, Michele Colajanni, and Riccardo Lancellotti, “Performance Evolution of Mobile Web-Based Services”, *IEEE Internet Computing*, March / April 2009, pp. 60-67.
19. Mustafa Adacal and Ayse B. Bener, “Mobile Web Services: A New Agent-Based Framework”, *IEEE Internet Computing*, May-June 2006, pp. 58-65.
20. Hong Tina Gao, Jane Huffman Hayes and Henry Cai, “Integrating Biological Research through Web Services”, *IEEE Computer*, March 2005, pp. 26-31.
21. Chunying Chen, Xiongwei Zhou, Jianzhong Zhang, “Web Data Mining System Based on Web Services”, *IEEE Computer Society*, 2009 Ninth International Conference on Hybrid Intelligent Systems, pp. 216- 220.
22. George Zheng and Athman Bouguettaya, “Service Mining on the Web”, *IEEE Transactions on Service Computing*, Vol.2, NO1, January-March 2009.
23. Dorgival Guedes, Wagner Meira Jr and Renato Ferreira, “Anteater: A Service-Oriented Architecture for High Performance Data Mining”, July / August 2006 *IEEE Internet Computing*, pp. 36-43.

24. Yalin yarimagan and Asuman Dogac, "A semantic based solution for UBL Schema Interoperability", IEEE Internet Computing, May June 2009.
25. Narayana Rao Surapaneni, Dhanjay Katre, "Java and .NET, A Developer's guide to Interoperability and Migration", Prentice Hall of India, 2004.
26. George Feurlicht, Sooksathit Meesanthit, "Design method for Interoperable Web service", ICSSOC 04 November, ACM 2004.
27. Hamid R. Motachari Nezhad and Boualaem Benatallah, Fabio Casati, Farouk Toumani, "Web Services Interoperability Specifications", IEEE Computer Society, 2006, pp. 24-32.
28. Wangming Ye, "Web services programming tips and tricks: Improve interoperability between J2EE technology and .NET", IBM Developer Works Part 1, available at [:http://www.ibm.com/developerworks/webservices/library/ws-tip-j2eenet1/](http://www.ibm.com/developerworks/webservices/library/ws-tip-j2eenet1/)
29. Wangming Ye, "Web services programming tips and tricks: Improve interoperability between J2EE technology and .NET", IBM Developer Works Part 2 available at: <http://www.ibm.com/developerworks/webservices/library/ws-tip-j2eenet2.html>
30. Wangming Ye, "Web services programming tips and tricks: Improve interoperability between J2EE technology and .NET", IBM Developer Works Part 3 available at <http://www.ibm.com/developerworks/webservices/library/ws-tip-j2eenet3/>
31. Eric Jaen Villoldo, Joan Serrat-Fernandez, Emilio Luque, "Improving Web Service Interoperability with Binding Extensions", 2007 ICWS, pp.1-7.
32. Hamid R.Motahari Nezhad and Boualem Benatallah, Fabio Casati, Farouk Toumani, "Web Services Interoperability Specifications", IEEE Computer, pp.24-32.
33. Marjin Janssen and Hans J. Scholl, "Interoperability for Electronic Governance", ICEGOV2007, pp.45-48.
34. Thinagaran Perumal, Abd Rahman Ramli and Chui Yew Leong, Shattri Mansor and Khairulmizam Samsudin, "Interoperability among

- Heterogenous Systems in a Smart Home Environment”, 2008 IEEE Conference on Signal Image technology and Internet Based Systems”, pp.177-186.
35. Eyhab Al-Masri and Qusay H.Mahmoud, “Interoperability among Service Registry Standards”, IEEE Internet Computing, May-June 2007,pp. 74-77.
 36. IBM Developer Works available at: www.ibm.com/developerworks
 37. Yumi Yamaguchi, Hyen-Vui Chung, Masayoshi Teraguchi and Naohiko Uramoto, “Easy to use programming model for web service security”, 2007 IEEE Asia-Pacific Services Computing Conference, pp.275-282.
 38. Kezhe Tang, Shiping Chen, David Levy, John Zie and Bo Yan, “A Performance Evaluation of Web Services Security”, Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC’06).
 39. Lenin Singaravelu, Calton Pu,”Fine Grain, End-to-End Security for Web Service Compositions”, 2007 IEEE International Conference on services Computing(SCC 2007).
 40. Michiaki Tatsubori, Takeshi Imamura, Yuhichi Nakamura,”Best practice Patterns and Tool Support for Configuring Secure Web services Messaging”, IEEE International Conference on Web Services (ICWS’04).
 41. Selma Elsheikh,”Access Control Scheme for Web Services”, 2008 IEEE International Conference on Computer and Communication Engineering.
 42. Nils Agne Nordbotton,”XML and Web Services Security Standards”, IEEE Communications Survey and Tutorials, Vol. 11, No.3 Third quarter 2009.
 43. Steve Graham, Doug Davis et al, “Building Web Services with Java”, Pearson Education, Second Edition,2008.
 44. Queue, March 2003, A conversation with Adam Bosworth.
 45. David A Chappell and Tyler Jewell, “Java Web services”, O’Reilly Publication,2007.
 46. Mike P. Papazoglou, Willem-Jan van den Heuvel, “Service oriented architecture: approaches, technologies and research issues, The VLDB Journal (2007). pp. 389-415.

47. Ramesh Nagappan, Robert Skoczylas, Rima Patel Sriganesh, "Developing Java Web Services", Wiley Publication, 2007.
48. Bilal Siddiqui, "Exploring XML Encryption Part 1", IBM Developer Works, <http://www.ibm.com/developerworks/xml/library/x-encrypt/index.html>.
49. C.Low, J Randall, M Wrey, Self describing Data Representation, Hewlitt Packard Labs, 1997.
50. Geert Jan Bex, Frank Neven, Jan Van den Bussche, "DTD Vs XML schema : A practical study", Seventh International Workshop on Web and Databases(WebDB2004), June 17-18, 2004, Paris, France.
51. Fabio Simeoni, David Lievens and Richard Connor, "Language Bindings to XML", IEEE Internet Computing, January – February 2003.
52. B.V.Kumar, S.V. Subramanya, " Web Services an Introduction" , Tata McGraw Hill, 2007.
53. Luigi Lo Iacono and Hariharan Rajasekaran, " Secure Browser based access to web services" , IEEE Communication Society proceedings of IEEE ICC 2009.
54. Liang-Jie-Zhang, Jia Zhang, "An integrated service Model Approach for enabling SOA", I.T. Pro September/ October 2009.
55. Lihui Lei and Zhenhua Duan, "Integrating AJAX and Web Services for Cooperative Image Editing", May|June 2007, IT Pro, pp 25- 29.
56. Sylvain Hallé, Tevfik Bultan, Graham Hughes, and Muath Alkhalaf, Roger Villemaire, "Runtime verification of Web Service Interface Contracts", IEEE Computer Society, March 2010.
57. Hong Tina Gao, Jane Huffman Hayes , Henry Cai, "Integrating Biological Research through Web Services", IEEE Internet Computing , March 2005.
58. Juan Jose, Garcia Adeva and Rafael Calvo, "Mining Text with Pimiento", IEEE Internet Computing, July August 2006, pp 27-35.
59. Paul Heymann, Georgia Koutrika and Hector Molina, "Fighting spam on social websites, A survey of approaches and future challenges", IEEE Internet Computing ,Nov- Dec 2007.

60. Souptik Datta, Kanishka Bhaduri, Chris Gianella and Hillol Kargupta, "Distributed Data Mining in Peer to Peer Networks", IEEE Internet Computing, July August 2006, pp 20- 21.
61. Anoop Kumar, Metimed Kantardzic and Samuel Madden, "Distributed Data Mining Framework and Implementations", IEEE Internet Computing, July August 2006, pp 15-19.
62. Ian H Witten, Eibe Frank, "Data Mining Practical Machine Learning Tools and Techniques", Morgan Kaufmann Publications, pp 14-395.
63. Markov Zdravko, and Russell Ingrid, An Introduction to the WEKA Data Mining System, available at: <http://www.cs.ccsu.edu/~markov/weka-tutorial.pdf>
64. Ian H Witten, Eibe Frank, WEKA Machine Learning Algorithms in Java, available at: <http://www.inf.ed.ac.uk/teaching/courses/dme/html/Tutorial.pdf>
65. Haridas Mandar, CIS764-Step By Step Tutorial for Weka. <http://www.docstoc.com/docs/2582601/CIS764---Step-By-Step-Tutorial-for-Weka-By-Mandar-Haridas->
66. Frank Eibe, et al., Data Mining in Bioinformatics using Weka, pp 1-2. <http://bioinformatics.oxfordjournals.org/cgi/reprint/20/15/2479>
67. Dimov Rossen, WEKA: Practical Machine Learning Tools and Techniques in Java. http://www.dfki.de/~kipp/seminar_ws0607/reports/RossenDimov.pdf
68. Mark F. Hornick, Eric Marcade, Sunil Venkayala, "Java Data Mining Strategy, Standard, and Practice", Morgan Kaufmann publication pp 3-116.
69. Web Services on Wikipedia available at: <http://en.wikipedia.org/wiki/WebServices/>
70. Data Mining: What is Data Mining? Available at: <http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/datamining.htm>
71. Weka 3: Data Mining software in Java, available at: <http://www.cs.waikato.ac.nz/~ml/weka/>

72. Classification methods available at:
<http://www.d.umn.edu/~padhy005/Chapter5.html>
73. StatSoft, Electronics Statistics Textbook available at:
<http://www.statsoft.com/textbook/cluster-analysis/>
74. K-nearest algorithm from Wikipedia available at :
http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm
75. Frank Sommers, "Leading Edge Java, Mine your own data with JDM API, Exploring the Java Data Mining API", July 2005 available at:
http://www.artima.com/lejava/articles/data_mining.html
76. Qi Yu, Xumin Liu, Athman Bouguettaya, Brahim Medjahed, "Deploying and Managing Web service issues, solutions and directions", Springer Verlag,2006.
77. Marijin Janssem, Hans J Scholl, "Interoperability for Electronic Governance", ICEGOV 2007, 2007 ACM.
78. D. Talia, P. Trunfio and O. Verta, "Weka4WS: A WSRF-Enabled Weka Toolkit for Distributed Data Mining on Grids", Springer-Verlag, 2006, pp. 309-320.
79. Andre Tost, "Web services interoperability, Part I", available at:
<http://www.ibm.com/developerworks/library/ws-bpinter/>
80. Dereck Ashmore, ' The J2EE Architects Handbook", 2006, available at:
<http://www.theserverside.com/news/1369773/Free-Book-The-J2EE-Architects-Handbook>
81. TTI. Taylor, A. S. Ali and O. Rana, "Web services composition for distributed data mining", IEEE International Conference on Parallel Processing Workshops (ICPPW'05), 2005, pp. 11-18.
82. Chad Vawter and Ed Roman, "J2EE VS Microsoft.NET A comparison for building XML based Web Services", June 2001, available at:
<http://www.theserverside.com/tt/articles/article.tss?l=J2EE-vs-DOTNET>
83. Keith Ballinger, ".NET Web Services Architecture and Implementation ", Addison Wesley.

84. Internet Information services (IIS) 7.5, available at:<http://www.microsoft.com/windowsserver2003/evaluation/overview/dotnet/uddifaq.mspx>
85. Alex Ferrara ,Mathew MacDonald, “Programming .NET Web services”, O’Reily Publication.
86. XML-RPC.Net available at:<http://www.xml-rpc.net/>
87. Advanced Web Service Interoperability available at:
<http://www.netbeans.org/kb/60/websvc/wsit.html>
88. Web services Enhancement (WSE) 3.0 for Microsoft.Net available at:
<http://www.microsoft.com/downloads/details.aspx?familyid=018A09FD-3A74-43C5-8EC1-8D789091255D&displaylang=en>
89. Moez Ben Haj Hmida, Yahya Slimani, “ WSRF Services for Learning Classifiers from Data Grid”, 978-1-4244-3806-8/09,2009, IEEE.
90. Roger Riggs, Antero Taivalsaari and others , “Programming Wireless Devices with Java 2 Platform, 2006.
91. Kim Topley, “J2ME in a Nutshell”, O’Reilly Publication,2005.
92. Sing Li and Jonathan Knudsen, “Beginning J2ME from Novice to Professional”, Apress Publication.
93. J.J.Garret, “AJAX A New Approach to Web Applications”, 18 Feb 2005, available
at:<http://www.adaptivepath.com/ideas/essays/archives/000385.php>
94. Lihui Lei and Zhenhua Duan, “Integrating AJAX and Web Services for Cooperative Image Editing”, May|June 2007, IT Pro, pp 25- 29.
95. Steve Vinoski, “Scripting JAX-WS”, IEEE Internet Computing May-June 2006.
96. About MySQL available at: <http://www.mysql.com/about/>
97. Mustafa Adacal and Ayse B Bener, “Mobile Web Services : A new agent based framework, IEEE Internet computing ,May June 2006.
98. Bilal Siddiqui, “Building a secure SOAP client for J2ME, part 1: Exploring web services API (WSA) for J2ME, IBM Developer Works .available at:

<https://www6.software.ibm.com/developerworks/education/ws-soa-securesoap1/ws-soa-securesoap1-pdf.pdf>

99. Claudia Canali, Michele Colajanni and Riccardo Lancellotti, "Performance Evolution of Mobile Based Web-Based Services", IEEE Internet Computing, March/April 2009, pp.60-67.
100. Nils Agne Nordbotten, "XML and Web Services Security Standards", IEEE Communications Survey and Tutorials, Vol 11, No3, Third Quarter 2009.
101. Netbeans tutorials available at: www.netbeans.org
102. Nitin Gupta, "Secure your Web Services", Intel whitepaper available at: http://cache-www.intel.com/cd/00/00/32/07/320703_320703.pdf
103. Lili Sun, Yan Li, "XML and Web Services Security", 978-1-4244-1651-6/08, IEEE 2008, pp. 765-770.
104. Diego Zuquim Guimarães Garcia and Maria Beatriz Felgar de Toledo, "Web Service Security Management Using Semantic Web Techniques", ACMSAC, March 2008, pp. 2256-2260.
105. Building secure ASP.NET applications: Authentication authorization and secure communication. available at: <http://msdn.microsoft.com/en-us/library/aa302378.aspx>
106. SSL Shopper available at: <http://www.sslshopper.com/article-most-common-java-keytool-keystore-commands.html>
107. Lori Delooze, "Providing Web Service Security in a Federated Environment", IEEE Security and Privacy, January February 2007, pp.73 - 75.
108. John Veiga, Jeremy Epstein, "Why applying Standards to Web Services is not enough", IEEE Security and Privacy, July August 2006, pp. 25-31.
109. Richard Ford, Michael Howard, "Introduction to Microsoft .NET Security, November/ December 2008 IEEE Security and Privacy, pp. 73-78.

APPENDIX

Appendix A

Code for J48 Classifier converted to a web service

```
/*
 * To change this template, choose Tools / Templates
 * and open the template in the editor.
 */
package demol;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.core.Instances;
import weka.core.OptionHandler;
import weka.core.Utils;
import weka.filters.Filter;
import java.io.FileReader;
import java.io.BufferedReader;
import java.util.Vector;

@WebService()
public class NewWebService12 {
    /**
     * Web service operation
     */
    @WebMethod(operationName = "execute")
    public String execute(@WebParam(name = "input")
        String input)throws Exception {
        Classifier m_Classifier = null;
    /** the filter to use */
    Filter m_Filter = null;

    /** the training file */
    String m_TrainingFile = null;

    /** the training instances */
    Instances m_Training = null;

    /** for evaluating the classifier */
    Evaluation m_Evaluation = null;

    String classifier = "";
    String filter = "";
    String dataset = "";
    Vector classifierOptions = new Vector();
    Vector filterOptions = new Vector();

    System.out.println(input);
    /**splitting the input into six parts */
    String args[] = input.split("\\s");
    for ( int i=0;i < args.length;i++)
    System.out.println ( args [ i ] );
    int i = 0;
    String current = "";
    boolean newPart = false;
    do {
        // determine part of command line
        if (args[i].equals("CLASSIFIER")) {
            System.out.println(args[i]);
            current = args[i];
            i++;
            newPart = true;
        }
        else if (args[i].equals("FILTER")) {
```

```

    current = args[i];
    i++;
    newPart = true;
}
else if (args[i].equals("DATASET")) {
    current = args[i];
    i++;
    newPart = true;
}
if (current.equals("CLASSIFIER")) {
    if (newPart)
        classifier = args[i];
    else
        classifierOptions.add(args[i]);
}
else if (current.equals("FILTER")) {
    if (newPart)
        filter = args[i];
    else
        filterOptions.add(args[i]);
}
else if (current.equals("DATASET")) {
    if (newPart)
        dataset = args[i];
        //dataset="C:/Users/s1/Documents/NetBeansProjects/JavaApp/build/classes/iris.arff ";
}
// next parameter
i++;
newPart = false;
}
while (i < args.length);
    m_Classifier = Classifier.forName(classifier,
        (String[]) classifierOptions.toArray(new String[classifierOptions.size()]));
    m_Filter = (Filter) Class.forName(filter).newInstance();
if (m_Filter instanceof OptionHandler)
    ((OptionHandler) m_Filter).setOptions((String[]) filterOptions.toArray(new String[filterOptions.size()]));

m_TrainingFile = dataset;
m_Training = new Instances(
    new BufferedReader(new FileReader(m_TrainingFile)));
m_Training.setClassIndex(m_Training.numAttributes() - 1);

// run filter
m_Filter.setInputFormat(m_Training);
Instances filtered = Filter.useFilter(m_Training, m_Filter);

// train classifier on complete file for tree
m_Classifier.buildClassifier(filtered);

// 10fold CV with seed=1
m_Evaluation = new Evaluation(filtered);
m_Evaluation.crossValidateModel(
    m_Classifier, filtered, 10, m_Training.getRandomNumberGenerator(1));

StringBuffer result;

result = new StringBuffer();
result.append("Weka - Demo\n=====\\n\\n");

result.append("Classifier...: "
    + m_Classifier.getClass().getName() + " "
    + Utils.joinOptions(m_Classifier.getOptions()) + "\\n");
if (m_Filter instanceof OptionHandler)
    result.append("Filter.....: "
        + m_Filter.getClass().getName() + " "
        + Utils.joinOptions(((OptionHandler) m_Filter).getOptions()) + "\\n");
else
    result.append("Filter.....: "
        + m_Filter.getClass().getName() + "\\n");

```

```

result.append("Training file: "
    + m_TrainingFile + "\n");
result.append("\n");

result.append(m_Classifier.toString() + "\n");
result.append(m_Evaluation.toSummaryString() + "\n");
try {
    result.append(m_Evaluation.toMatrixString() + "\n");
} catch (Exception e) {
    e.printStackTrace();
}
try {
    result.append(m_Evaluation.toClassDetailsString() + "\n");
}
catch (Exception e) {
    e.printStackTrace();
}

return result.toString();
}
}

```

Appendix B

Code for EM Clusterer converted to a web service

```

package demo;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import weka.core.Instances;
import weka.clusterers.DensityBasedClusterer;
import weka.clusterers.EM;
import weka.clusterers.ClusterEvaluation;
import java.io.FileReader;
import java.io.BufferedReader;

@WebService()
public class NewWebService2 {
    /**
     * Web service operation
     */
    @WebMethod(operationName = "execute")
    public String execute(@WebParam(name = "input")
        final String input)throws Exception{

        ClusterEvaluation eval;
        Instances data;
        String[] options;
        DensityBasedClusterer cl;
        data = new Instances(new BufferedReader(new FileReader(input)));
        StringBuffer result;
        result = new StringBuffer();

        result.append("Weka - Demo\n=====\n\n" + "\n\n--> normal\n");
        // normal
        //System.out.println("\n--> normal");
        options = new String[2];
        options[0] = "-t";
        options[1] = input;
        //System.out.println(ClusterEvaluation.evaluateClusterer(new EM(), options));

        result.append(ClusterEvaluation.evaluateClusterer(new EM(), options));

        // manual call
        //System.out.println("\n--> manual");
        cl = new EM();
        cl.buildClusterer(data);
        eval = new ClusterEvaluation();
        eval.setClusterer(cl);
    }
}

```

```

eval.evaluateClusterer(new Instances(data));
//System.out.println("# of clusters: " + eval.getNumClusters());
try {
result.append("\n-> manual" + "\n\n# of clusters: " + eval.getNumClusters() + "\n");
}
catch (Exception e) {
e.printStackTrace();
}
// density based
//System.out.println("\n-> density (CV)");
cl = new EM();
eval = new ClusterEvaluation();
eval.setClusterer(cl);
eval.crossValidateModel(
cl, data, 10, data.getRandomNumberGenerator(1));
//System.out.println("# of clusters: " + eval.getNumClusters());

try {
result.append("\n-> density (CV)" + "\n\n# of clusters: " + eval.getNumClusters() + "\n\n");
}
catch (Exception e) {
e.printStackTrace();
}
return result.toString();
}
}
}

```

Appendix C

Code to convert Text Classifier into a web service

```

package demo;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import java.util.*;
import weka.core.*;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.Attribute;
import weka.classifiers.*;
import weka.classifiers.Classifier;
import weka.filters.unsupervised.attribute.StringToWordVector;

```

```

@WebService()
public class TextClassifierService {
/**
* Web service operation
*/
@WebMethod(operationName = "execute")
public String execute(@WebParam(name = "input")
String input) {
//TODO write your implementation code here:

Instances instances = null;
Classifier classifier = null;
Instances filteredData = null;
Evaluation evaluation = null;
Set modelWords = null;
// maybe this should be settable?
String delimitersStringToWordVector = "\\s.:\"'()?!";

// String classString = "weka.classifiers.bayes.NaiveBayes";
String classString = "weka.classifiers.lazy.IBk";
// String classString = input;

```

```

String[] inputText = {"hey, buy this from me!", "do you want to buy?", "I have a party tonight!", "today it is a nice weather", "you are best", "I have a horse", "you are my friend", "buy, buy, buy!", "it is spring in the air", "do you want to come?"};

```



```

String[] inputClasses = {"spam","spam","no spam","no spam","spam","no spam","no spam","spam","no spam","no spam"};

String[] testText = {"you want to buy from me?","usually I run in stairs", "buy it now!","buy, buy, buy!","you are the best,
buy!","it is spring in the air"};

if (inputText.length != inputClasses.length) {
    System.err.println("The length of text and classes must be the same!");
    System.exit(1);
}
// calculate the classValues
HashSet classSet = new HashSet(Arrays.asList(inputClasses));
classSet.add("?");
String[] classValues = (String[])classSet.toArray(new String[0]);
//
// create class attribute
//
FastVector classAttributeVector = new FastVector();
for (int i = 0; i < classValues.length; i++) {
    classAttributeVector.addElement(classValues[i]);
}
Attribute ClassAttribute = new Attribute("class", classAttributeVector);
//
// create text attribute
//
FastVector inputTextVector = null; // null -> String type
Attribute TextAttribute = new Attribute("text", inputTextVector);
for (int i = 0; i < inputText.length; i++) {
    TextAttribute.addStringValue(inputText[i]);
}

// add the text of test cases
for (int i = 0; i < testText.length; i++) {
    TextAttribute.addStringValue(testText[i]);
}
//
// create the attribute information
//
FastVector AttributeInfo = new FastVector(2);
AttributeInfo.addElement(TextAttribute);
AttributeInfo.addElement(ClassAttribute);

/*this.inputText = inputText;
this.inputClasses = inputClasses;
this.classString = classString;
this.attributeInfo = attributeInfo;
this.textAttribute = textAttribute;
this.classAttribute = classAttribute;
*/

StringBuffer result = new StringBuffer();
result.append("dataset:\n\n");
// creates an empty instances set
instances = new Instances("data set", AttributeInfo, 100);
// set which attribute is the class attribute
instances.setClass(ClassAttribute);
try {

    for (int i = 0; i < inputText.length; i++) {
        Instance inst = new Instance(2);
        inst.setValue(TextAttribute,inputText[i]);
        if (inputClasses != null && inputClasses.length > 0) {
            inst.setValue(ClassAttribute, inputClasses[i]);
        }
        instances.add(inst);
    }
    result.append("DATA SET:\n" + instances + "\n");

    StringToWordVector filter = null;
// default values according to Java Doc:
int wordsToKeep = 1000;

```

```

Instances filtered = null;
try {
    filter = new StringToWordVector(wordsToKeep);
    filter.setOutputWordCounts(true);
    filter.setSelectedRange("1");

    filter.setInputFormat(instances);

    filtered = weka.filters.Filter.useFilter(instances,filter);
    // System.out.println("filtered:\n" + filtered);

} catch (Exception e) {
    e.printStackTrace();
}
filteredData = filtered;

// create Set of modelWords
modelWords = new HashSet();
Enumeration enumx = filteredData.enumerateAttributes();
while (enumx.hasMoreElements()) {
    Attribute att = (Attribute)enumx.nextElement();
    String attName = att.name().toLowerCase();
    modelWords.add(attName);
}

classifier = Classifier.forName(classString,null);
classifier.buildClassifier(filteredData);
evaluation = new Evaluation(filteredData);
evaluation.evaluateModel(classifier, filteredData);

try {
    result.append("\n\nINFORMATION ABOUT THE CLASSIFIER AND EVALUATION:\n");
    result.append("\nclassifier.toString():\n" + classifier.toString() + "\n");
    result.append("\nevaluation.toSummaryString(title, false):\n" + evaluation.toSummaryString("Summary",false) + "\n");
    result.append("\nevaluation.toMatrixString():\n" + evaluation.toMatrixString() + "\n");
    result.append("\nevaluation.toClassDetailsString():\n" + evaluation.toClassDetailsString("Details") + "\n");
    result.append("\nevaluation.toCumulativeMarginDistribution:\n" + evaluation.toCumulativeMarginDistributionString() +
"\n");
} catch (Exception e) {
    e.printStackTrace();
    result.append("\nException (sorry!):\n" + e.toString());
}
result.append("\n\n");

// check instances
int startIx = 0;
String testType = "not test";

try {
    result.append("\nCHECKING ALL THE INSTANCES:\n");
    Enumeration enumClasses = ClassAttribute.enumerateValues();
    result.append("Class values (in order): ");
    while (enumClasses.hasMoreElements()) {
        String classStr = (String)enumClasses.nextElement();
        result.append("'" + classStr + "' ");
    }
    result.append("\n");
    // startIx is a fix for handling text cases
    for (int i = startIx; i < filteredData.numInstances(); i++) {
        SparseInstance sparseInst = new SparseInstance(filteredData.instance(i));
        sparseInst.setDataset(filteredData);
        result.append("\nTesting: " + inputText[i-startIx] + "\n");
        // result.append("SparseInst: " + sparseInst + "\n");
        double correctValue = (double)sparseInst.classValue();
        double predictedValue = classifier.classifyInstance(sparseInst);
        String predictString = ClassAttribute.value((int)predictedValue) + " (" + predictedValue + ")";
        result.append("predicted: " + predictString);
        // print comparison if not new case
        if (!"newcase".equals(testType)) {
            String correctString = ClassAttribute.value((int)correctValue) + " (" + correctValue + ")";

```

```

        String testString = ((predictedValue == correctValue) ? "OK!" : "NOT OK!") + "!";
        result.append(" real class: " + correctString + " ==> " + testString);
    }
    result.append("\n");
result.append("\n");
    // result.append(thisClassifier.dumpDistribution());
    // result.append("\n");
}
} catch (Exception e) {
    e.printStackTrace();
    result.append("\nException (sorry!):\n" + e.toString());
}

result.append("\n");

} catch (Exception e) {
    e.printStackTrace();
    result.append("\nException (sorry!):\n" + e.toString());
}

result.append("\n\nNEW CASES\n");

Instances testCases = new Instances(instances);
testCases.setClass(ClassAttribute);
//
// since some classifiers cannot handle unknown words (i.e. words not
// a 'model word'), we filter these unknowns out.
// Maybe this should be done only for those classifiers?
// E.g. Naive Bayes have prior probabilities which may be used?
//
// Here we split each test line and check each word
//
String[] testsWithModelWords = new String[testText.length];
int gotModelWords = 0; // how many words will we use?
for (int i = 0; i < testText.length; i++) {
    // the test string to use
    StringBuffer acceptedWordsThisLine = new StringBuffer();
    // split each line in the test array
    String[] splittedText = testText[i].split("[ "+delimitersStringToWordVector+" ]");
    // check if word is a model word
    for (int wordIx = 0; wordIx < splittedText.length; wordIx++) {
        String sWord = splittedText[wordIx];
        if (modelWords.contains((String)sWord)) {
            gotModelWords++;
            acceptedWordsThisLine.append(sWord + " ");
        }
    }
    testsWithModelWords[i] = acceptedWordsThisLine.toString();
}
// should we do something if there is no modelWords?
if (gotModelWords == 0) {
    result.append("\nWarning!\n\nThe text to classify didn't contain a single\n\nword from the modelled words. This makes it hard for
the classifier to\n\nndo something usefull.\n\nThe result may be weird.\n\n");
}
try {
    // add the ? class for all test cases
    String[] tmpClassValues = new String[testText.length];
    for (int i = 0; i < tmpClassValues.length; i++) {
        tmpClassValues[i] = "?";
    }

    for (int i = 0; i < testsWithModelWords.length; i++) {
        Instance inst = new Instance(2);
        inst.setValue(TextAttribute, testsWithModelWords[i]);
        if (tmpClassValues != null && tmpClassValues.length > 0) {
            inst.setValue(ClassAttribute, tmpClassValues[i]);
        }
        testCases.add(inst);
    }
}

```

```

StringToWordVector filter = null;
// default values according to Java Doc:
int wordsToKeep = 1000;
Instances filtered = null;
try {
    filter = new StringToWordVector(wordsToKeep);
    filter.setOutputWordCounts(true);
    filter.setSelectedRange("1");
    filter.setInputFormat(testCases);
    filtered = weka.filters.Filter.useFilter(testCases,filter);
} catch (Exception e) {
    e.printStackTrace();
}
Instances filteredTests = filtered;
int startIx = instances.numInstances();
String testType = "new case";
try {

    result.append("\nCHECKING ALL THE INSTANCES:\n");
    Enumeration enumClasses = ClassAttribute.enumerateValues();
    result.append("Class values (in order): ");
    while (enumClasses.hasMoreElements()) {
        String classStr = (String)enumClasses.nextElement();
        result.append("'" + classStr + "' ");
    }
    result.append("\n");

    // startIx is a fix for handling text cases
    for (int i = startIx; i < filteredTests.numInstances(); i++) {
        SparseInstance sparseInst = new SparseInstance(filteredTests.instance(i));
        sparseInst.setDataset(filteredTests);
        result.append("\nTesting: " + testText[i-startIx] + "\n");
        double correctValue = (double)sparseInst.classValue();
        double predictedValue = classifier.classifyInstance(sparseInst);
        String predictString = ClassAttribute.value((int)predictedValue) + " (" + predictedValue + ")";
        result.append("predicted: " + predictString);

        // print comparison if not new case
        if (!"newcase".equals(testType)) {
            String correctString = ClassAttribute.value((int)correctValue) + " (" + correctValue + ")";
            String testString = ((predictedValue == correctValue) ? "OK!" : "NOT OK!") + "!";
            result.append(" real class: " + correctString + " ==> " + testString);
        }
        result.append("\n");
        result.append("\n");
    }
} catch (Exception e) {
    e.printStackTrace();
    result.append("\nException (sorry!):\n" + e.toString());
}
result.append("\n");
} catch (Exception e) {
    e.printStackTrace();
    result.append("\nException (sorry!):\n" + e.toString());
}
return result.toString();
}
}

```

Appendix D

Client code for J48 Classifier as a web service:

```

package org.weka;

import demo1.NewWebService12Service;
import java.io.*;
import java.net.*;

```

```

import javax.servlet.*;
import javax.servlet.http.*;
import javax.xml.ws.WebServiceRef;

public class NewServlet extends HttpServlet {
    @WebServiceRef(wsdlLocation = "http://localhost:13699/WebApplication12/NewWebService12Service?wsdl")
    private NewWebService12Service service;

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {

            // TODO output your page here

            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet NewServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet NewServlet at " + request.getContextPath () + "</h1>");

        }

        // Call Web Service Operation
        demo1.NewWebService12 port = service.getNewWebService12Port();
        // TODO initialize WS operation arguments here

        java.lang.String input = "CLASSIFIER weka.classifiers.trees.J48 -U FILTER
weka.filters.unsupervised.instance.Randomize DATASET C:/Users/s1/Documents/NetBeansProjects/JavaApp/build/classes/iris.arff";

        // TODO process result here

        java.lang.String result = port.execute(input);
        out.println("Result = "+result);
    } catch (Exception ex) {

        // TODO handle custom exceptions here

    }

    out.println("</body>");
    out.println("</html>");

    } finally {
        out.close();
    }
}

// <code>editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code."</code>
/**
 * Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 */

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP <code>POST</code> method.
 * @param request servlet request

```

```

    * @param response servlet response
    */

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
    * Returns a short description of the servlet.
    */

    public String getServletInfo() {
        return "Short description";
    }

    // </editor-fold>
}

```

Appendix E

Client code for EM Clusterer as web service

```

package org.web;
demo.NewWebService2Service;
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.xml.ws.WebServiceRef;

public class NewServlet extends HttpServlet {
    @WebServiceRef(wsdlLocation = "http://localhost:13699/WebApplication4/NewWebService2Service?wsdl")
    private NewWebService2Service service;

    /**
    * Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
    * @param request servlet request
    * @param response servlet response
    */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            //TODO output your page here
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet NewServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet NewServlet at " + request.getContextPath () + "</h1>");

        } catch (Exception ex) {
            out.println(ex.getMessage());
        }
    }

    // Call Web Service Operation

    demo.NewWebService2 port = service.getNewWebService2Port();

    // TODO initialize WS operation arguments here

    java.lang.String input = "C:/Users/s1/ Documents/NetBeansProjects/ClusteringDemo/build/classes/weather.arff";

    // TODO process result here

    java.lang.String result = port.execute(input);
    out.println("Result = "+result);
} catch (Exception ex) {

```

```

// TODO handle custom exceptions here
}
    out.println("</body>");
    out.println("</html>");

    } finally {
        out.close();
    }
}

// <editor-fold defaultstate="collapsed" desc="HttpServletRequest methods. Click on the + sign on the left to edit the code.">
/**
 * Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 */

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP <code>POST</code> method.
 * @param request servlet request
 * @param response servlet response
 */

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Returns a short description of the servlet.
 */

public String getServletInfo() {
    return "Short description";
}
// </editor-fold>
}

```

Appendix F

Client code for text classifier as web service:

```

package org.web;

import demo.TextClassifierServiceService;
import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.xml.ws.WebServiceRef;

public class NewServlet extends HttpServlet {
    @WebServiceRef(wsdlLocation = "http://localhost:13699/TextClassifierService/TextClassifierServiceService?wsdl")
    private TextClassifierServiceService service;

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     */

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

```

```

response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
try {
    // TODO output your page here

    out.println("<html>");
    out.println("<head>");
    out.println("<title>Servlet NewServlet</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Servlet NewServlet at " + request.getContextPath () + "</h1>");

try {

// Call Web Service Operation

demo.TextClassifierService port = service.getTextClassifierServicePort();

    // TODO initialize WS operation arguments here

    java.lang.String input = "weka.classifiers.lazy.IBk";

// TODO process result here

    java.lang.String result = port.execute(input);
    out.println("Result = "+result);
} catch (Exception ex) {

// TODO handle custom exceptions here
}

    out.println("</body>");
    out.println("</html>");

    } finally {
        out.close();
    }
}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
/**
 * Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}
/**
 * Handles the HTTP <code>POST</code> method.
 * @param request servlet request
 * @param response servlet response
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}
/**
 * Returns a short description of the servlet.
 */
public String getServletInfo() {
    return "Short description";
}
// </editor-fold>
}

```


List of Publications

Conferences

1. "Midlet Security : An Overview", proceedings of National Conference on Knowledge Based Computing Systems & Frontier Technologies NCKBFT-07, February 19 & 20, 2007, held at Manipal Institute of Technology, Manipal, India.
2. "Comparison of Web Service Development Platforms", Proceedings of the 2008 International Conference on Semantic Web and Web Services", WORLDCOMP'08, July 14-17, 2008, Las Vegas Nevada, USA.
3. "J2ME Based Tool for interaction With Web Services", Proceedings of the 2008 International Conference on Semantic Web and Web Services", WORLDCOMP'08, July 14-17, 2008, Las Vegas Nevada, USA.
4. "WEKA Based Desktop Data Mining as Web Service", Proceeding of WASET, ICCESSE'10, April 28-30, 2010, Rome, Italy.

Journals

1. "Interoperability Issues seen in Web Services", IJCSNS International Journal of Computer Science and Network Security, VOL.9 No 8, August 2009.
2. "Design and Security Analysis of web application based and web services based Patient Management System (PMS)", IJCSNS International Journal of Computer Science and Network Security, VOL.10 No. 3, March 2010

Brief Biography of Sujala D. Shetty

Sujala Shetty has finished her MTech in Computer Science and Engineering from MIT, Manipal in 2002. She has over 15 years of teaching experience. She has worked as lecturer in the Computer Science Department of MIT, Manipal from 1997 to 2002. She is currently working as Senior Lecturer in the Computer Science Dept of BITS, PILANI-Dubai from 2002. She has six publications in various international journals and conferences. Her current areas of interest are web services and security and databases and data mining.

Brief Biography of Prof. Dr S.Vadivel

Prof.Dr.S.Vadivel has got his PhD in Computer Science and Engineering from I.I.T Madras, India in 1989. After getting his degree he worked in Crompton Greaves in Bombay as research executive for 3 years. After which he worked as Assistant Professor in Government Engineering College in Tamil Nadu, India for 4 years. After which he joined Think Business Networks a multinational software company in Tamil Nadu as Research Lead.

He has joined BITS, Pilani - Dubai as faculty in Computer Science in January 2003 and is currently working as Professor and Head of Department of the Computer Science Department in the same institute .He has 10 publications in various international journals and conferences. His current research interests are in web services and security, embedded controllers, data mining, and architecture of enterprise software applications.

