A Framework for Mobile SOA using Compression

By

Evan Saunders

Submitted in partial fulfilment of the requirements for the

degree of Magister Scientiae in the Faculty of Science

at the Nelson Mandela Metropolitan University

December 2010

Supervisor: Prof. Jean Greyling

Co-Supervisor: Dr. Lester Cowley

Acknowledgements

I would like to thank and acknowledge the following persons for their efforts, encouragement, and support:

- My supervisors, Prof. Jean Greyling and Dr. Lester Cowley, for their support, encouragement and guidance throughout this research. I wish to express my gratitude to them for their many hours they have spent reading the various drafts of this dissertation and for the valuable feedback, which they provided to improve its structure and content.
- The staff in the Department of Computing Sciences at NMMU for their support, use of facilities and equipment.
- Mr. Danie Venter, statistical data analysis consultant at NMMU, for his assistance in designing a suitable analysis outline for the evaluations conducted in the course of this research.
- Telkom SA, as my financial sponsor for my Masters degree.
- The NMMU/Telkom Centre of Excellence for providing me with equipment to complete this research.

Dedications

Firstly, I would like to give thanks to my Father in heaven who has seen me through this journey.

I would like to dedicate this dissertation to Crucinda, as well as my family.

To Crucinda, thank you for caring and supporting me from the start. Thank you for the inspiration, calming words and motivation you have given me.

To my family, thank you for the encouragements and asking about progress.

Summary

The widely accepted standards of Service-Oriented Architecture (SOA) have changed the way many organisations conduct their everyday business. The significant popularity of mobile devices has seen a rapid increase in the rate of mobile technology enhancements, which have become widely used for communication, as well as conducting everyday tasks. An increased requirement in many businesses is for staff not to be tied down to the office. Consequently, mobile devices play an important role in achieving the mobility and information access that people desire. Due to the popularity and increasing use of SOA and mobile devices, Mobile Service-Oriented Architecture (Mobile SOA) has become a new industry catch-phrase. Many challenges, however, exist within the Mobile SOA environment. These issues include limitations on mobile devices, such as a reduced screen size, lack of processing power, insufficient processing memory, limited battery life, poor storage capacity, unreliable network connections, limited bandwidth available and high transfer costs. This research aimed to provide an elegant solution to the issues of a mobile device, which hinders the performance of Mobile SOA.

The main objective of this research was to improve the effectiveness and efficiency of Mobile SOA. In order to achieve this goal, a framework was proposed, which supported intelligent compression of files used within a Web Service. The proposed framework provided a set of guidelines that facilitate the quick development of a system. A proof-of-concept prototype was developed, based on these guidelines and the framework design principles. The prototype provided practical evidence of the effectiveness of implementing a system based on the proposed framework. An analytical evaluation was conducted to determine the effectiveness of the prototype within the Mobile SOA environment. A performance evaluation was conducted to determine efficiency it provides. Additionally, the performance evaluation highlighted the decrease in file transfer time, as well as the significant reduction in transfer costs. The analytical and performance evaluations demonstrated that the prototype optimises the effectiveness and efficiency of Mobile SOA. The framework could, thus, be used to facilitate efficient file transfer between a Server and (Mobile) Client.

Keywords: Service-Oriented Architecture, Mobile Service-Oriented Architecture, File Compression, Compression Framework.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background

Spurred on by the advent of the Internet, there is a growing need for information access. Living in an "information society", we are inundated with information (Edmunds & Morris 2000). People are increasingly requiring information access at any time, from any location, and need more information, better technologies and easier access to this information.

The use of personal computing for Internet access continues to grow and new applications and technologies continue to emerge. Mobile devices are starting to replace traditional computing devices (personal computers and laptops) as client devices, and must support growing, changing and emerging applications and technologies (Natchetoi, Kaufman & Shapiro 2008).

There are approximately 4.6 billion mobile devices worldwide (CBS News 2010), which was predicted to reach five billion by the end of 2010. It is therefore not surprising that mobile phones are becoming a popular new platform for business applications. Mobile devices that were initially intended as peripheral voice call devices have become small computers that are required to do virtually anything, anywhere (Johnsrud, Hadzic, Hafsøe, Johnsen & Lund 2008).

Although the theoretical background of the Internet dates back as early as the 1960s, it was not until the early 1990s that it became a popular technology. By the mid to late 1990s the Internet had become a common household technology, made accessible by the development of the World Wide Web (Lowe, Lomax & Polonkey 1996). Initially, the Web provided businesses with a means of advertising their product on a global scale from a small set of Web pages. The enhancement of Internet and Web technologies over time has enabled business to do more than simply advertise.

New opportunities present themselves on a daily basis, and businesses of today are expected to respond effectively and in a timely way to these opportunities (Papazoglou & van dem Heuvel 2007). However, most businesses still utilise a legacy approach to their systems, which may obstruct the response to these new opportunities (Sanders, Hamilton &

MacDonald 2008). Additionally, the development of these legacy systems is costly, and provides limited integration between systems within the business and external systems, as required.

Service-Oriented Architecture (SOA) has provided a means to support the integration of legacy systems, and reduce the time and cost of implementing these systems. SOA and Service-Oriented Computing (SOC) are the most recent approaches aimed at facilitating the design and development of applications in distributed systems (Jørstad, Dustdar & van Thanh 2005). Simply put, SOA is a collection of Services, which communicate with each other to achieve a goal.

The emergence and acceptance of SOA has shown that it is as revolutionary and important to businesses as the Internet was 10 to 15 years ago (Hurwitz, Bloor, Baroudi & Kaufman 2007). Mobile Services is an extension of SOA aimed at utilisation on mobile devices. The Web has presented a means for standardisation of information Services on the Internet, whereas mobile devices allow mobile, wireless access to these Services (Holonen & Ojala 2006). Mobile Services hold a promise of utilising the phone also for purposes other than voice and SMS communication, (van Gurp, Karhinen & Bosch 2005).

It was estimated that 80 percent of all mission critical applications, including business processes, will be based on SOA by the end of 2010 (Chattpar 2008). This indicates a definite drive toward SOA-based applications. Furthermore, mobile devices are becoming a common platform for business applications (Natchetoi et al. 2008). Technologies for managing information are continually changing, and although technologies for mobile devices and networks are becoming more advanced, it is not enough to deal with a number of issues that plague mobile computing. These issues include insufficient memory, processing capabilities and storage on a mobile device, low reliability of connections, limited available bandwidth, poor security for wireless and mobile networks, and high cost (Barton, Zhai & Cousins 2010; Huang 2009; Natchetoi et al. 2008; Ojala 2005). Additionally, the use of mobile devices is made more challenging by the complexity and high expenses required to implement mobile applications (Duda, Alesky & Schader 2008). These issues suggest that the resources available to mobile devices and the challenges in implementing mobile applications inhibit the evolution of SOA-based application, such as Mobile Services.

In the field of SOA, Web Services and many Web technologies, XML and associated technologies play a key role (Ojala 2005). The manner in which XML files are created often leads to bloated files, which contain redundant information. Compression of XML files is essential as it reduces the required bandwidth, storage, and processing (Natchetoi et al. 2008). Media files are an important aspect of visual and auditory aid to enhance user experience. Media files are typically classified as image, sound or video files. Because the size of media files is often too large for efficient use on mobile devices, compression is essential for quick file transfers to a mobile device, as well as the effective management of these files for presentation on the device.

This section briefly discussed the growth of mobile computing and introduced the concept of SOA and related technologies. The relevance of the research, the subsequent problem statement, the research questions and the scope of the study are discussed in the following sections.

1.2 Relevance of Research

Mobile devices do not offer the same resources as desktops. An example of this is the limited bandwidth of cellular networks, which may lead to a delay factor in transmission of files to and from the device. A solution to this problem is to reduce the size of the transmitted data by employing compression, (Savini, Ionas, Meier, Pop & Stormer 2007). There has been limited research into SOA in terms of mobile devices and standards. Research into SOA and Web Services for mobile devices has focused primarily on the end devices in processing Web Services. The effect of external influences, such as a slow wireless network to deliver these Web Services, is frequently overlooked. It is essential that any solution related to issues regarding mobile SOA is simple, intelligent and efficient.

The aim of this project is, therefore, to determine how an intelligent framework, supporting compression, can be designed to support Mobile SOA and to implement a prototype to evaluate the effectiveness and efficiency of the framework.

1.3 Research Outline

This section discusses the research outline by means of the problem statement, the thesis statement, the research questions, the research objectives, the goals, scope and constraints, and the methodology.

1.3.1 Problem Statement

Traditional use of SOA within business organisations has been limited to desktop computers. However, businessmen spend approximately a third of their time conducting business outside the office environment (Johnson, Manyika & Yee 2005), highlighting the need for mobile intervention. Current resources available to and on mobile devices are insufficient to cope with large amounts of data and processing. Coupled with this, the unreliability of connections and limiting bandwidth offered to mobile devices make it difficult to support Mobile SOA, which incurs large costs for business and government organisations.

1.3.2 Thesis Statement

The aim of this research is to illustrate that Mobile SOA can be optimised using compression. A framework will be designed and implemented for use within the Mobile SOA environment. The framework proposed is centred on file compression and demonstrates that improvement of file transfers to and from mobile devices is feasible. The thesis statement for the study is therefore:

"The efficiency and effectiveness of Mobile SOA can be improved by the implementation of a framework supporting intelligent[1] compression."

1.3.3 Research Objectives

With the purpose of researching the thesis statement, the primary objective of this research is to "Improve efficiency and effectiveness of Mobile SOA". This study also intends to achieve the following secondary objectives:

- To gain a comprehensive understanding of an SOA and its enabling technologies (Chapter 2);
- To acquire thorough knowledge of Mobile SOA and its enabling technologies (Chapter 3);
- To investigate compression and categorisation techniques of files used within Mobile SOA (Chapter 4);
    - To establish a set of criteria for measuring efficiency of compression;

[1] Intelligent Compression, in this research, is the ability to choose different compression techniques based on the categorisation of types of files, and the further categorisation within each type of file.

- To determine how a framework supporting compression can be designed for Mobile SOA (Chapter 5);
- To determine how a prototype can be implemented based on the design of a framework (Chapter 6); and
- To evaluate the design of a framework and implementation of a prototype for Mobile SOA (Chapter 7).

1.3.4 Research Questions

The main research question for this project is "how can the efficiency and effectiveness of Mobile SOA be optimised using a framework that supports intelligent compression?" In answering the main research question, a number of secondary questions must also be answered. These questions are listed in Table 1.1, along with the method by which these questions will be answered.

Table 1.1: Research Questions and Associated Methodology

#	Research Question	Research Method	Chapter
Main:	How can the efficiency and effectiveness of Mobile SOA be optimised using a framework that supports intelligent compression?		
R1	What is SOA and what are its components?	Literature Study	Chapter 2
R2	What is Mobile SOA and what are the relevant issues and constraints?	Literature Study	Chapter 3
R3	What are the issues related to file compression?	Literature Study	Chapter 4
R3.1	How can files be categorised in relation to intelligent compression?	Literature Study	
R3.2	What are the different file compression techniques?	Literature Study	
R3.3	What are the criteria for measuring efficiency of file compression?	Literature Study	
R4	How can a framework, supporting intelligent compression, be designed for Mobile SOA?	Service-Oriented Analysis and Design	Chapter 5
R5	How can a prototype, based on the proposed framework, be implemented?	Developmental/ Proof-of-Concept	Chapter 6
R6	Does the prototype adhere to the framework design principles?	Evaluation	Chapter 7
R7	Does the prototype show that the efficiency of Mobile SOA can be improved through intelligent compression?	Evaluation	

In order to answer these research questions, a research methodology was designed. This methodology is discussed next.

1.3.5 Research Methodology

Each of the research questions, listed in Table 1.1 above, will be answered by means of a research method outlined in the following sub-sections to achieve the objective defined in Section 1.3.3.

1.3.5.1 Literature Review

The goal of this research is to implement a compression framework for Mobile SOA. In order to achieve this goal, an understanding of both mobile and SOA technologies is required. A literature review on SOA is conducted to understand what SOA is and to establish what the components of SOA are (R1). A literature review is conducted to understand what Mobile SOA is, to establish what the components of Mobile SOA are, and to uncover the issues and constraints related to Mobile SOA (R2). The literature review on compression is conducted to understand what compression is and how it can be optimised (R3), how files may be categorised in relation to intelligent compression (R3.1), what are the different compression techniques available (R3.2), and what are the criteria for measuring the performance of these compression techniques (R4).

1.3.5.2 Framework Design

Previous research is examined to discover existing frameworks within the same, or similar, environment. A framework will be designed for this project as it provides the basis on which a prototype can be implemented (R5).

1.3.5.3 Proof-of-Concept

In order to evaluate the framework design and propose improvements to the framework, a proof-of-concept prototype has to be developed. This prototype will be implemented as a proof-of-concept to establish whether the framework adequately supports intelligent compression by means of file categorisation. This compression is performed on data and media files associated with the requested Service. The prototype forms the basis for answering research question R6.

1.3.5.4 Evaluation

Although the prototype provides a means for testing the proof-of-concept by answering research question R5, an evaluation of this prototype is required. The evaluation provides a means of determining whether the framework achieved the objectives that were outlined in Section 1.3.3. As shown in Table 1.1, the prototype must confirm the following:

- Whether the prototype adheres to the framework design principles (R6);
- Whether the prototype demonstrates that intelligent file compression improves the efficiency of Mobile SOA (R7).

In order to answer Research Questions R6 and R7, the evaluation process consists of two categories of evaluation, an analytical evaluation and an evaluation of performance metrics. The analytical evaluation demonstrates that the prototype is designed according to framework design principles, which are discussed in Section 5.3.1. The evaluation of performance metrics provides a means to determine the efficiency of the developed prototype.

1.3.6 Goals, Scope and Constraints

The goal of this research is to "improve the efficiency and effectiveness of Mobile SOA" by means of achieving each of the objectives listed in Section 1.3.3. The scope of this research is, therefore, limited to investigating how a framework supporting intelligent compression can be implemented within a Mobile SOA environment. The area in which the framework will be used is Mobile Web Services.

SOA is an architecture that presents standards for describing and interacting between components, and can be employed across multiple platforms. Web Services are the most common technology for implementing an SOA and offer numerous types of Services that provide different functions. This research focuses on Service-Orientation via Web Services, which can be adapted to suit the mobile environment. Discussions of these Web Services are restricted to the context of SOA. The domain constraints are outlined below:

- The domain is divided into two types of Services, namely Content Services and Added Value Services. These Services, discussed in Section 2.3, are limited to business queries that could be used within mobile environment. Three different examples of scenarios, based on commonly used Mobile Services, will be examined for evaluation purposes. The first example is based on a Service for delivering students marks. The second Service resembles a blog in which photos can be downloaded and uploaded. Finally, the third Service is based on a combination of Content and Added Value Service in which data and media files are sent via the Service;

- Data will be represented in the form of XML files due to the widespread utilisation of XML-based Web Services. Media will be represented in the form of image and audio files; and
- The requirements of the implemented prototype are limited to performing the following functions using Web Services:
    - Registering a first time user by means of capturing user and mobile device data;
    - Compressing files of the user-requested Web Services to be transferred to a mobile device; and
    - Decompressing these files on a mobile device for use with the requested Web Services.

File compression is restricted to XML, image and audio files. Video files are somewhat large, in terms of storage space required, even after performing compression. As with most Web technologies, there are time constraints when delivering content to a Client within the Mobile SOA environment. Providing on-the-fly video compression is a very time-intensive procedure, and, therefore, cannot meet the desired time specifications. Consequently, it was decided that video compression is not within the scope of this research.

The evaluation of the prototype is limited to testing on two mobile devices, namely HTC s710 and HTC TyTN. The specifications of these phones are discussed in Chapter 6.

1.4 Dissertation Structure

This section gives a brief outline of each of the eight chapters. Each chapter aims to achieve a research objective as outlined in Section 1.3.3. Figure 1.1 provides a summary of the structure of this dissertation.

Chapter 2: – Service-Oriented Architecture: This chapter provides a literature background on Service-Oriented Architecture and its related technologies and formally defines SOA. A number of projects related to SOA are discussed.

Chapter 3: – Mobile SOA: This chapter presents a discussion on mobile devices, and their capabilities, uses and limitations. Furthermore, a discussion on Mobile SOA is also conducted as a specialised domain within SOA. The benefits of Mobile SOA are discussed, as well as the drawbacks faced within Mobile SOA. The supporting technologies and components are also discussed in this chapter in conjunction with existing frameworks that support Mobile SOA.

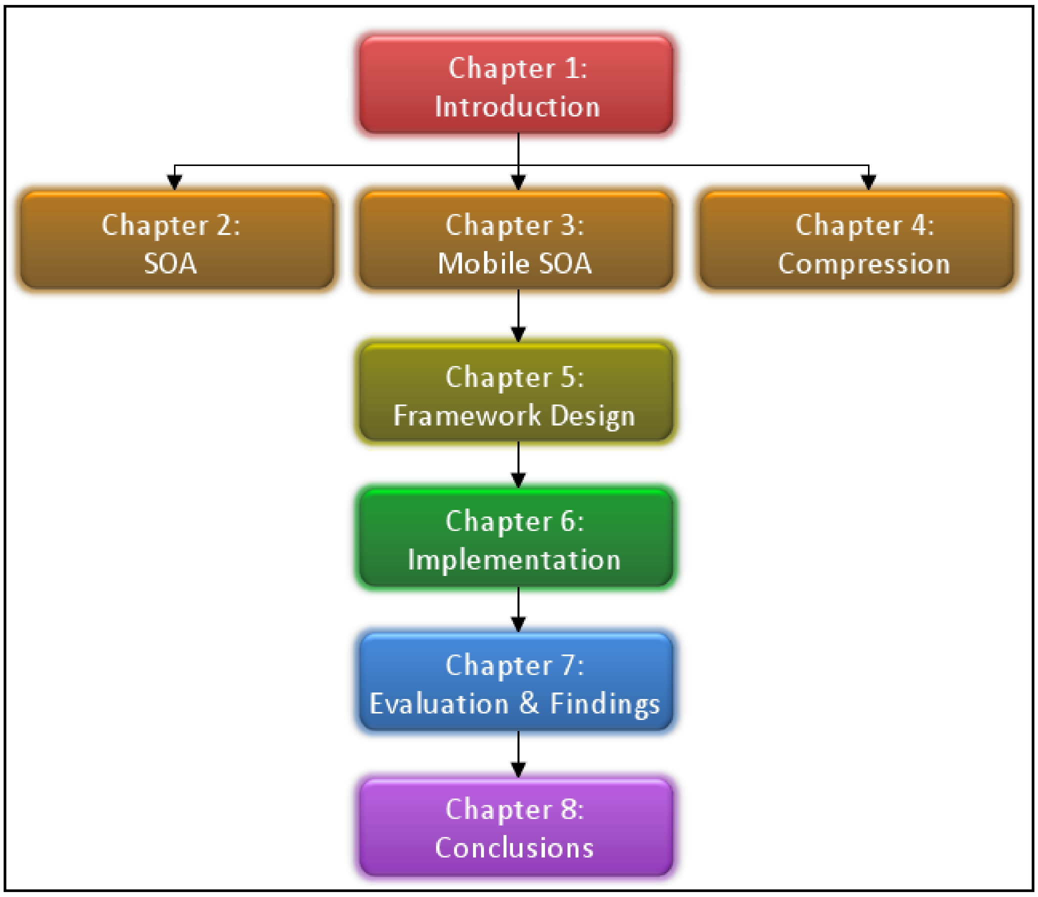


Figure 1.1: Dissertation Structure

Chapter 4: – File Compression: Research on file compression is discussed in this chapter. A sample of common files used within Mobile SOA is briefly discussed. Numerous compression techniques are investigated. For data, an initial five compression techniques are compared, and a further three for audio and image compression, respectively. The comparison of these compression techniques are discussed in Chapter 6.

Chapter 5: – Framework Design: This chapter discusses the design processes and methods of the proposed framework. In this chapter, the framework is designed, based on an existing framework, following specific design principles.

Chapter 6: – Implementation: A proof-of-concept prototype was developed based on the proposed framework. This chapter describes the design and implementation of this prototype. The implementation of the architecture, individual components and data design is described.

Chapter 7: – Evaluation and Findings: This chapter discusses the evaluation of the final prototype based on the specific performance metrics identified. The presentation and analysis of the results were gathered from testing the performance of the prototype. A test for efficiency and effectiveness of the prototype is the primary focus of this chapter.

Chapter 8: – Conclusions: Conclusions derived from this research are reviewed in this chapter. The chapter validates that the outlined objectives were achieved and presents ideas for future research.

1.5 Conclusions

As mobile devices are becoming increasingly popular year by year, there is an increasing need for an optimised compression to improve the effectiveness and efficiency of providing SOA to mobile devices. A standardised framework, supporting intelligent compression, is an essential research area in Mobile Service-Oriented Architecture (Mobile SOA).

This research proposes a framework, which supports intelligent compression, and determines how a prototype, based on the proposed framework, can be implemented using standardised design principles. This research will also entail a literature review (Chapters 2, 3 and 4), framework design, prototyping and experiment methods, and an evaluation of the framework and prototype. The prototype will be evaluated by means of specific software engineering metrics to test the performance in terms of effectiveness and efficiency it provides compared to normal service delivery. The research methods will address the identified objectives.

The next chapter is the first section of the literature review, and provides an overview of SOA and its underlying technologies.

Chapter 2: Service-Oriented Architecture

2.1 Introduction

Service-Oriented Architecture (SOA) is not a new concept or approach to software and system design. SOA is an evolution of the component style of distributed system development, where systems are designed and built in components that interact and exchange data (OASIS 2006) and these underlying concepts to SOA have been around since the 1970s (Datz 2004). In the past, operations have been carried out by multiple tiers of systems and now there is a shift towards making these systems work together in an SOA (Sanders et al. 2008).

This chapter answers research question R1, “What is SOA and what are its components?”, and is the first of three literature study chapters. A definition of SOA is provided in Section 2.2. Section 2.3 provides background knowledge of what SOA is and why it is so widely accepted. SOA design principles are discussed in Section 2.4. This is followed by a discussion of the enabling components of SOA in Section 2.5. Finally, the applications of SOA are discussed in Section 2.6.

Before delving into the driving forces, challenges, components, and applications of SOA, it is important to have a formal and concrete definition of SOA for this research.

2.2 Definition

Due to its wide acceptance and utilisation, SOA has become a popular term in the IT community. The IT community has seen SOA as a solution to a wide variety of architectural challenges (Senga, 2010). However, there is no one leading definition of SOA. This has meant that many businesses, institutions, groups, and vendors, within the IT community, have proposed varying definitions of what SOA is.

Erl (2005) defines SOA as an architectural model that proposes to improve the efficiency, agility, and productivity of an organisation by positioning Services as the primary method through which solution logic is represented in support of the fulfilment of strategic goals related to Service-Oriented Computing. Shen (2007) also provides a definition of SOA as an architectural style that aims to accomplish loose coupling between interacting and contracted

Services by way of communication protocols. Hurwitz et al. (2007) defines SOA as an architecture for developing business applications as a collection of loosely coupled components, organised to produce a well-defined level of service by connecting business processes.

These three definitions are consistent in that they define SOA as an architecture for combining previously developed business applications. This is achieved by means of integrating loosely coupled components and presenting them as Services. This research formally defines SOA as: *"an architectural approach to building loosely coupled business systems by integrating different components by means of Services. The integration of these components is independent of programming language and platform."* This definition will be used throughout the remainder of this dissertation.

2.3 Service-Oriented Architecture

The legacy systems, or existing working systems, of enterprise contain a mixture of different programming and execution environments, techniques and protocols that are difficult to maintain and update. SOA has become the new hype in both government operations and industry as it promises to increase business agility and ease costs through improved interoperability and reuse of shared business Services (Erradi, Kulkarni, Anand & Kulkarni 2006). An SOA is intended to allow developers to solve many distributed enterprise computing challenges, which includes application integration, transaction management, and security policies, whilst allowing multiple platforms and protocols, and influencing many access devices and legacy systems (Alonso, Casati, Kuno, Machiraju, 2004).

According to Gartner, the use of SOA is still growing (Abrams & Schulte 2008). The Gartner hype cycle (Carpenter 2009) (Figure 2.1) shows that SOA is still a rising technology, as it highlights that SOA has passed the peak of inflated expectations and the trough of disillusionment, and is well established on the slope of enlightenment. This is an indication that organisations have a better understanding of SOA, along with its benefits and capabilities.

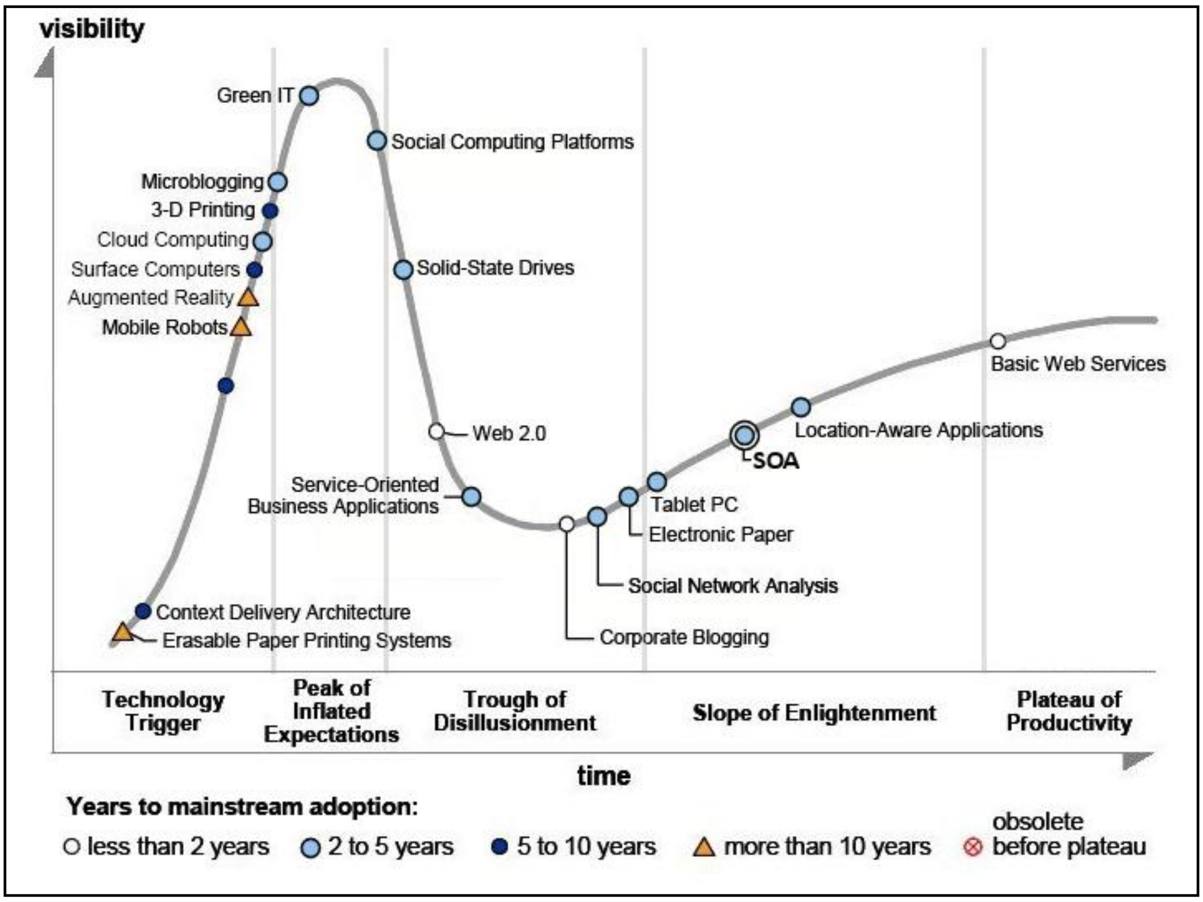


Figure 2.1: Gartner Hype Cycle for Emerging Technologies (Carpenter, 2009)

In an SOA, software resources are wrapped as Services that are well defined, self-contained modules that supply standard business functionality and are independent of the state or context of other Services (Papazoglou & van den Heuvel 2007). Services are defined in a standard definition language, have a published interface, and interact with each other, requesting execution of their operation in order to collectively support a central business task or process (Fremantle, Weerawarana & Khalaf 2002).

SOA is an architectural style that is based on loosely coupled system components. Loose coupling refers to the idea of allowing a Service or application using the Service to be agnostic to the underlying technical details of partner Services in order to use them to their fullest functionality and is important for maintainability, scalability, and future upgrade paths of applications using the Services (Sanders et al. 2008). This means that Services are not dependent on each other and can be mixed and matched with other component Services as needed (Hurwitz et al. 2007). Loose coupling has advantages over tight coupling in that it provides asynchronous communication, dynamic binding, enables composition of Services at runtime, is platform independent and any changes in one Service will not have additional impact on any associated Services (Liegl 2007).

Services within SOA are divided into two categories, Content Services and Added Value Services (van Gurp et al., 2005). Content Services are based on the delivery of media files, such as audio and images. Added Value Services are based on the delivery of data files, such as those used within the business environment.

2.4 Design Principles of SOA

A design principle is an accepted industry practice that provides a guideline, or set of guidelines, to realise a specific technology implementation (Erl 2005). SOA is a type of architecture that adheres to the principles of Service-Orientation, and when implemented by means of Web Services, SOA supports these principles during the business process and automation domains of an organisation (Erl, 2005). SOA design principles present guidelines by which best results are achieved when utilising SOA (Senga, 2010).

Within the IT industry, there are a number of SOA design principles. Of these, there are seven common design principles (Erl 2008; OASIS 2006; SOABooks 2010; IBM 2010). The seven design principles of SOA are listed as (Erl 2008):

1. **Service Composability: "**Services are effective composition participants, regardless of the size and complexity of the composition";
2. **Service Coupling:** "Service contracts impose low consumer coupling requirements and are themselves decoupled from their surrounding environment";
3. **Service Abstraction:** "Service contracts only contain essential information and information about services is limited to what is published in Service contracts";
4. **Service Statelessness:** "Services minimize resource consumption by deferring the management of state information when necessary";
5. **Service Reusability:** "Services contain and express agnostic logic and can be positioned as reusable enterprise resources";
6. **Service Autonomy:** "Services exercise a high level of control over their underlying runtime execution environment"; and
7. **Service Discoverability:** "Services are supplemented with communicative meta data by which they can be effectively discovered and interpreted".

These principles are applied together to the solution logic to structure it in a way that promotes important design characteristics that support the specific goals linked to SOC (SOABooks 2010). SOA, however, does not define a specific implementation technology, but instead defines the architectural components and concepts to describe a particular Service in a specific context (Sanders et al. 2008). Designing systems for SOA requires an understanding of the components in a system and how they interact.

2.5 Components

There are numerous variations of the component model of SOA. However, the basic components of SOA are standard. Figure 2.2 (Sanchez-Nielsen, Martin-Ruiz & Rodriguez-Pedrianes 2006) describes the SOA paradigm and the relationship between the basic components of SOA implemented by means of standard XML-based initiatives. These basic components are the Service Consumer (or Service Requestor/Service Client/Service User), the Service Provider and the Service Manager (or Service Registry/Service Directory). Service Consumers may be different Services, applications or end-users (Erl 2005). A Service Registry allows Service Providers to register and publish Services in a registry. The Service Providers are the owners that offer Services. The Service Provider registers and publishes a Service description to the Service Registry in which the Service Consumer can find any Service with specific request parameters. The registry returns the description of each relevant Service. The Service description contains sufficient information about the Service to enable the Service Consumer to bind to the Service Provider and use the Service. A Service Consumer can be in the form of other applications, Services or end-users (Erl, 2005).

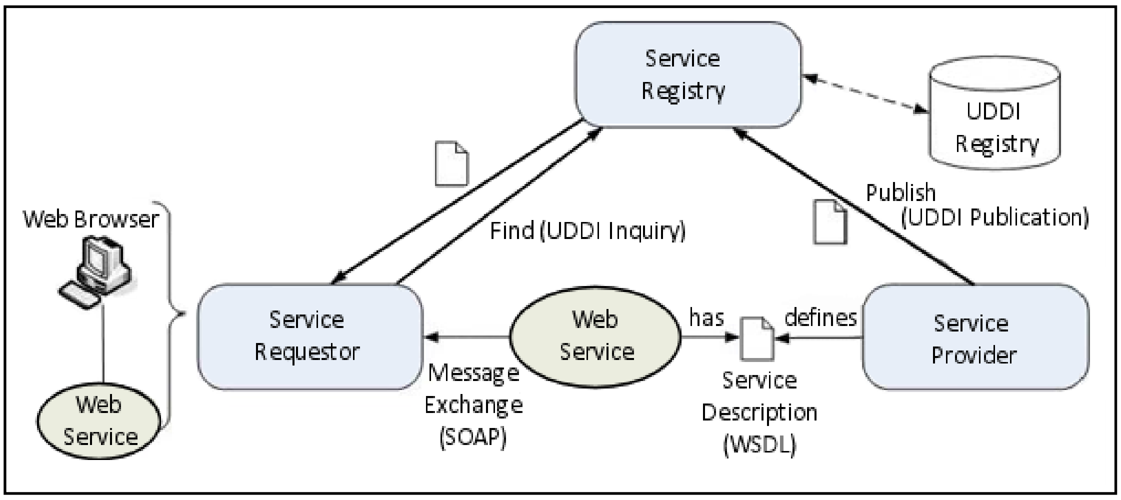


Figure 2.2: Service-Oriented Architecture (Sanchez-Nielsen et al., 2006)

Using a Web Service concept, a Client application (Service Consumer) makes a procedure call of a Web Service (Service Provider) in the same manner it invokes a local call, in which there are numerous types of communication between a Client application and a Web Service (Sanchez-Nielsen et. al 2006).

Although the Service Client, Service Provider and Service Registry are the basic components of an SOA, there are a number of additional components that may be used, depending on the domain within which it is used. For example, this research focuses on Mobile SOA, which employs the Mobility Controller as an additional component. This is discussed in more detail in Chapter 3 and illustrated in Figure 3.2.

2.5.1 Web Services

The term *Service* is often used to refer to a Web Service. This research focuses on SOA by means of Web Services and, thus, any mention of Services in this research refers to Web Services and vice-versa.

The increasing acceptance of SOA has been due to the popular use of Web Services, and the variety of today's systems (Sanders et al. 2008). Web Services are the most popular type of Services available today (Papazoglou & van den Heuvel 2007).

A Web Service is an integration technology and has become an obvious solution to (or at least as a simplification of) the integration challenge within SOA (Alonso et al. 2004). The main advantage it yields is that of standardisation, in terms of data format (XML), interface definition language (WSDL), transportation mechanism (SOAP), discovery description (UDDI), and many other interoperability facets (Nezhad 2006).

There are three major bodies which aim to define Web Standards such as SOA. These are the World Wide Web Consortium (W3C), WS-I, and the Organisation for the Advancement of Structures Information Standards (OASIS, 2006). W3C (2004b) defines a Web Service as a software system designed to enable and sustain interaction between interoperable machines via a network by means of an interface described in a machine-readable format. The OASIS group defines a Service as: "a mechanism to enable access to a set of one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies, as specified by the Service description" (OASIS 2006).

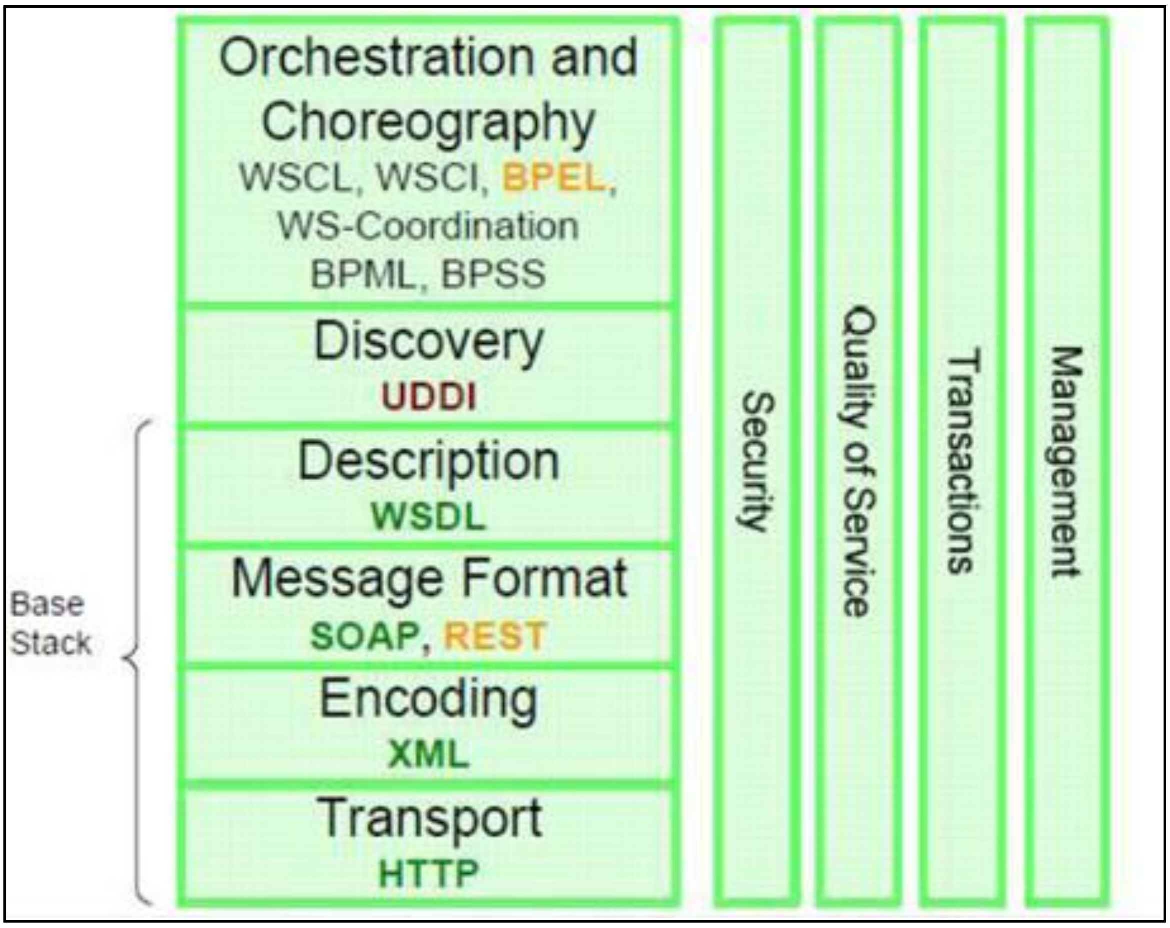


Figure 2.3: Web Services Protocol Stack (Lewis & Wrage 2006)

The main goal of Web Services is to make building blocks (initiatives) accessible over standard Internet protocols that are independent from platforms and programming languages (WebSphere, 2004). Web Services are based on a core set of open communication standards and protocols (Senga, 2010). These standards are the building blocks of Web Service and are shown graphically in Figure 2.3.

A discussion of the components within the Base Stack is presented later in this section. The security, QoS, transactions, and management components are optional, as they provide additional attributes to the implementation of a Web Service.

Figure 2.4 illustrates how these building blocks interact in a Web Service Architecture along with the basic components of SOA. The components of SOA interact with one another by transmitting requests and responses in the form of XML-based messages.

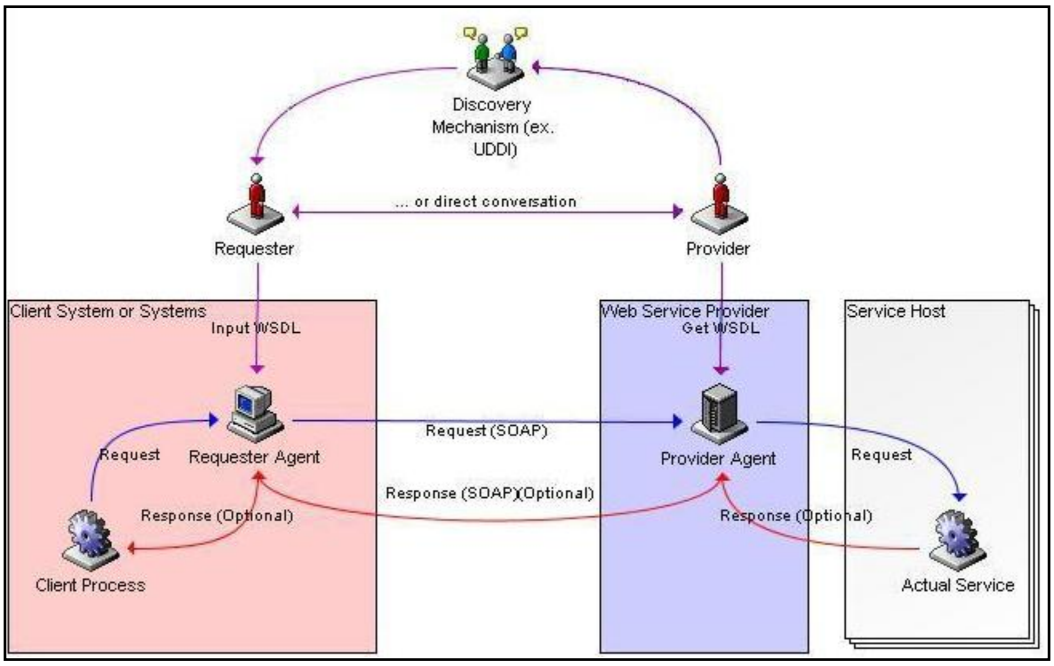


Figure 2.4: Web Service Interaction (W3C 2004b)

Web Services core standards have developed over the past five to ten years and are defined, in detail in the following subsections.

2.5.1.1 HTTP

Hypertext Transfer Protocol (HTTP) is a widely accepted Internet protocol for the transmission of data between interconnected machines; and applications that make use of this protocol can communicate with minimum effort, in spite being installed and operated on different platforms and potentially in different geographical locations (Senga 2010). HTTP is the standard for addressing Web pages, e.g. http://www.google.com/alerts, and in addition to defining addresses, HTTP can identify a Web Service, such as the news alerts Service that Google provides (Hurwitz et al. 2007). Web Services allow machines to communicate using standard communication protocols transmitted over HTTP using SOAP messages (discussed in Section 2.5.1.5).

2.5.1.2 XML

Extensible Mark-up Language (XML) is a simple, text-based configuration for representing structured information (W3C 2010). This information is saved in XML files, which have their internal structures available to users of the file through a process of metadata publishing. All definitions, descriptions and messages are based on XML. Since data is stored and transmitted in the form of XML files within the proposed framework, it is discussed in more detail in Chapter 4.

2.5.1.3 WSDL

A Service description is a collection of documents that describe the interface to and semantics of a Service. W3C (2004a) defines Web Services Description Language (WSDL) as a language for describing Web Services. This is done by means of messages, which have a specific network protocol and message format, exchanged between the Service Consumer and Service Provider. Simple changes to existing Internet infrastructure can consume Web Services for utilisation through Web browsers or directly within an application, and provided that both the sender and receiver agree on the Service description, the implementation behind the Web Services can be anything (W3C 2004a). WSDL is a standard based on the XML format. Through WSDL, a designer specifies the programming interface of a Web Service. This interface is specified in terms of methods supported by the Web Service, where each method is able to take one message as an input and return another as output.

2.5.1.4 UDDI

The Universal Description Discovery and Integration (UDDI) is a framework for describing, discovering and integrating business Services through the Internet (Hurwitz et al. 2007). It provides definitions for registries of business Services (Kanneganti & Chodavarapu 2008), which can be made accessible to the general public, or they can be private – that is, used within the confines of an organisation – or they could be between associate organisations (Senga 2010). The UDDI framework makes use of SOAP messages to communicate with applications that access it.

2.5.1.5 SOAP

Simple Object Access Protocol (SOAP) provides a standard, extensible, organised framework for packaging and exchanging XML messages (W3C 2004a). SOAP is lightweight and geared for the exchange of information in a decentralised environment in which messages are exchanged over standard HTTP/HTTPS. Using SOAP, Services are used to exchange messages by means of standardised conventions to convert a Service invocation into an XML message, to exchange the message, and to convert the XML message back into an actual Service invocation (Sanchez-Nielsen et al. 2006). The use of XML to define the messaging framework allows SOAP to be independent of platform and programming language (Gudgin, Hadley, Mendelsohn, Moreau, Nielsen, Karmarkar & Lafon 2007). Figure 2.5 shows an example of a SOAP request message, which Web Services utilise to exchange messages with other Web Services (Senga, 2010).

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap=http://www.w3.org/2001/12/soap-envelope
soap:encodingStyle="http://www.w3.org/2001/12/soap encoding">
            <soap:Body xmlns:m="http://www.example.org/stock">
                        <m:GetAStockPrice>
                              <m:StockName>MSFT</m:StockName>
                        </m:GetAStockPrice>
            </soap:Body>
</soap:Envelope>
```

Figure 2.5: Example of a SOAP Request Message (Senga 2010)

Different systems interact with a Web Service in a way specified by its description, in the form of WSDL, using SOAP messages, generally transferred using HTTP with an XML serialization in conjunction with other Web-related standards (W3C 2004b). The following describes the complete sequence of events that transpires when an XML-based Web Service is called (Quynh & Thang 2009):

1. The Client creates an instance of an XML-based Web Service proxy class, residing on the same computer or device as the Client.
2. The Client then invokes a method on the proxy class.
3. The infrastructure on the Client computer/device serialises the arguments of the Web Service method into a SOAP message and transmits it over the network to the Web Service.
4. The infrastructure accepts the SOAP message and deserialises the XML, after which it creates an instance of the class implementing the Web Service and invokes the Web Service method, feeding in the deserialised XML as parameters.
5. The Web Service method executes its code, finally setting the return value and any out parameters.
6. The infrastructure on the Web Server serialises the return value and out parameters into a SOAP message and transfers it over the network to the Client.
7. The Web Service infrastructure, on the Client computer/device, obtains the SOAP message, deserialises the XML into the return value and any out parameters, and passes them to the instance of the proxy class.
8. The Client receives the return value and any out parameters.

A Web Service, if implemented correctly, may provide a number of benefits, which has led to its wide acceptance and use for implementing applications within an SOA. These benefits are discussed in the next section.

2.5.2 Web Service Benefits

Web Service technology is widely accepted as a critical aspect of Service-oriented architecture, more specifically Service-Oriented Integration Architectures (SOIA). The standards provided within a Web Service provide an opportunity for seamless integration[1]. The benefits of using Web Services, within SOA, include the following (Cavanaugh 2006):

- Application and data integration;
- Versatility;
- Code reuse; and
- Cost saving.

The main advantage of Web Services, due to the platform independence of Web Services and standards used to implement Web Services, is that any application can invoke a Web Service, as long as the Service is discoverable. Web Services provide a standardisation in terms of data format (XML), interface description language (WSDL), transport protocol (SOAP), and Service discovery (UDDI) (Nezhad 2006). Although there are other standards, they have become the leading standards in industry for an SOA (Liegl 2007), hence its popularity. Additionally, the technology used to implement Web Services makes it possible for an interacting collection of simple Web Services to operate as one complex process (Sanders et al. 2008).

2.6 Service-Oriented Applications

The SOA design paradigm has allowed a new class of applications that allow scalable infrastructure and software to be delivered over the Internet as a Service (Senga 2010). An example of such an SOA application is cloud-computing. Essentially, computing is conducted in a cloud, in which data storage and information processing are performed in data centres. These data centres often utilise the grid-computing method, in which parallel

[1] Seamless integration is defined as two (or more) systems, which may be implemented on different platforms and in different programming languages, accessing core functionalities from the other system without any additional implementation.

algorithms and applications are executed to manage processing on a large scale. The infrastructure can be reused to provide on-demand Services to consumers by making them available as Web Services (Atkins, Droegemeier, Feldman, Garcia-Molina, Klein, Messerschmidtt, Messina, Ostriker & Wright 2003).

Loose coupling has enabled a new sector of the software industry known as "software as a Service" (SaaS), which allows a provider to host software for a Client so that the Client does not require to manage the software or purchase any hardware for it (Hurwitz et al. 2007). All that would be required is a connection to that Service. The immediate scalability of the infrastructure allows Service providers to benefit it, while consumers can use software on an as-needed basis (Senga 2010). Ironically, the computer industry began with a similar model until the mid-1970s, as the majority of companies that computerised their businesses subscribed to time-sharing Services, as computers were too costly for one company to own. Although hardware costs have dropped considerably, running software applications and maintaining data centres still incur large costs for businesses (Hurwitz et al. 2007).

There are numerous examples of other types of applications that can be delivered via the Internet as Services. These applications include, for example, Microsoft‘s Windows Azure Platform that provides a Windows operating system, EyeOS (EyeOS 2009) provides desktop functionality through a browser, while Google Docs (Google 2009), and Buzzword (Buzzword 2009) provide word-processing functionality from a cloud.

2.7 Conclusions

SOA is a technology that continues to grow year on year, and is not just used by large organisations and businesses, but by ordinary people to perform daily tasks, such as banking. However, there has been an excessive amount of differing definitions across vendors and businesses. Due to the vagueness and ambiguity in the number of definitions provided for SOA, this chapter has provided a formal definition of SOA. SOA is: *"an architectural approach to building loosely coupled business systems by integrating different components by means of Services. The integration of these components is independent of programming language and platform."*

SOA provides a platform independent means to deliver an application's functionality by invoking the functionality as Services. The most popular form of Services to enable SOA has been through the invocation of Web Services. A Web Service is a technology that is well suited to implementing SOA, which are software systems to facilitate machine communication over a network. The components of a Web Service are expressed using HTTP, XML, WSDL, UDDI, and SOAP, each of which has a specific role for Web Services. The advantages of Web Services include application and data integration, versatility, code reuse and cost saving. Web Services are used within applications to perform tasks based on the user request.

SOA continues to grow due to its rapid acceptance and wide use for both businesses and everyday people. Mobile technology has also seen continued growth during the past decade. The increase in the amount of connected mobile devices has meant that the technology has moved from simply being calling devices to acting as small computers. Due to the expansion of both these technologies, there has been an increasing interest in the domain of Mobile SOA. The standardised implementation criteria for SOA have meant that the technology may be adapted to operate within a mobile environment. However, many challenges still exist for both mobile technology and Mobile SOA.

The next chapter provides a literature review of mobile devices and the increasing invocation of Mobile SOA through Mobile Web Services.

Chapter 3: Mobile Service-Oriented Architecture

3.1 Introduction

According to research (CBS News, 2010) there are approximately 4.7 billion connected mobile devices worldwide. This number is increasing daily and was expected to reach 5 billion by the end of 2010, of which two billion of these devices will be cell phones. There are approximately 6.9 billion people in the world (U.S. Census Bureau 2010). Given that a number of people might have more than one mobile device, it is estimated that more than 60 percent of the population have some form of a mobile device. Approximately 1.6 billion people have access to online information (distinct from the ability of these devices to make and receive phone calls) from around the world via their cell phones. Gartner (2010) predicts that the number of users accessing the Internet via their mobile phones will overtake that of desktop computers by the end of 2013, with approximately 1.83 billion Internet-enabled mobile phones having been sold as opposed to the 1.78 billion computers sold at that time. This is confirmed by Meeker (2010), who predicts that the same event will occur by mid 2013 (Figure 3.1).

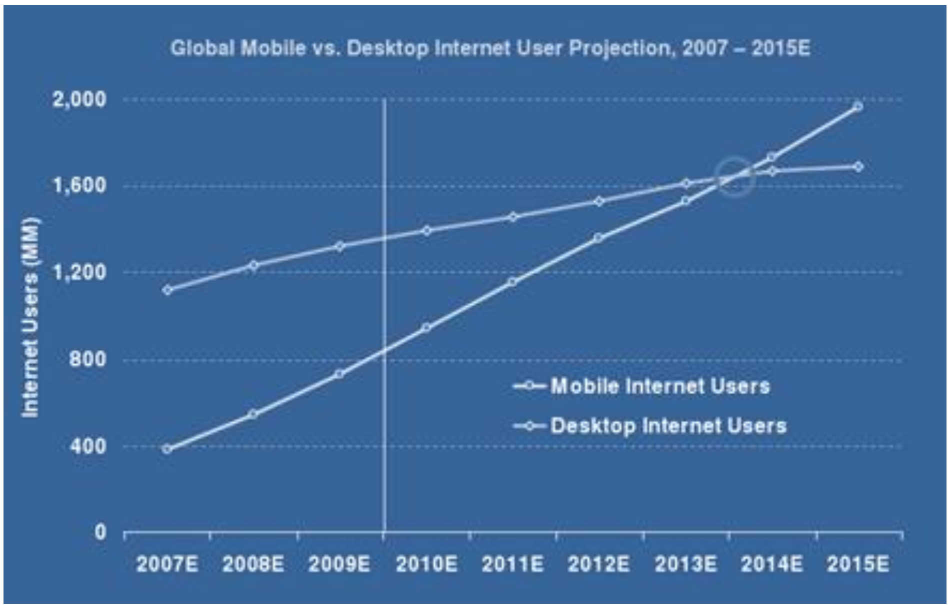


Figure 3.1: Global Mobile vs. Desktop Internet User Projection (Meeker 2010)

In addition, the requirement of the corporate world is that they do not want to be tied to their workstations in order to conduct their business. On average, mobile employees spend a third of their time at work out of the office, and approximately half of their time in the office away from their desks (Johnson et al., 2005). A growing number of employees want to make business decisions using their cell phones and as a result, cell phones will play a critical role as the devices used for accessing enterprise systems, such as ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), and BI (Business Intelligence) (Natchetoi et al., 2008). From financial to media Services, few markets will be untouched by mobility in the years to come with the impact on some sectors, such as banking, having already been profound.

There is a definite requirement for providing better solutions in providing SOA to mobile devices. In order to discuss Mobile SOA, its requirements, its benefits and drawbacks, a discussion on the capabilities of mobile devices, as well as the uses and challenges, is required. This discussion presents motivation for the necessity of using Mobile SOA.

This chapter provides a literature study on mobile devices and their technology. Section 3.2 provides a brief introduction to mobile technology, and also briefly introduces the need for Mobile SOA, a technology enabled by modern mobile devices. This is followed by a discussion on mobile devices, their capabilities, uses, and constraints (Sections 3.3-3.5). A specialised domain of SOA, Mobile SOA, is discussed in Section 3.6, as well as underlying technologies, drawbacks and existing systems within Mobile SOA.

3.2 Mobile Devices

Mobile technologies are revolutionising the manner in which people interact with their daily lives, work, and business (Sanchez-Nielsen et al., 2006). The last ten years have been witness to a telecommunications world that differs vastly from even the recent past, with developments in the mobile sector having significantly changed the Information and Communications Technology (ICT) landscape. This is particularly evident in Africa, as the mobile cellular growth rate there has been the highest of any region over the past five years, averaging close to 60% per annum (Botha, Makitla, Ford, Fogwill, Seetharam, Abouchabki, Tolmay & Oguneye 2010).

Many computer vendors, such as Apple and Microsoft, have moved into the smartphone market, while mobile phone vendors, such as Nokia, are starting to release devices that compete with netbooks and notebooks from computer manufacturers. Mobile phones started as a method for communicating via simple voice transmission over wireless networks, revised soon after by the Simple Message Service (SMS). These devices, however, have since grown into a multi-billion dollar industry where almost any form of computing is capable of being achieved using these phones. This includes these devices being used to access the Internet. Mobile devices are, in effect, becoming the new Client platforms (Natchetoi et al. 2008) required to support most of the new computing paradigms, including transmission of large files.

In order to discuss mobile device capabilities and uses, along with technologies enabled by mobile devices, a formal definition is required.

3.2.1 Definition

Mobile devices have multiple definitions, which include numerous devices such as mobile phones, PDAs and laptops, usually referring to communicators, multimedia entertainment and business processing devices to support portability for their users. In this research, a mobile device is defined as "any handheld, pocket-sized, mobile device that is connected to a network operator and is capable of connecting to the Internet." These devices include cell phones, smart phones, and PDAs. This definition will be used for the remainder of this research when referring to mobile devices, cell phones, smart phones or PDAs. Although laptops and other such devices can technically be defined as being mobile, they lack what is defined as true mobility, as they often require placement on a flat surface, and operation with both hands. Additionally, the resources available to these devices are not always a limiting factor, such as the screen size, input and processing power.

3.2.2 Mobility

Mobile devices, as described in its name, are defined by its capacity to be mobile and the level of mobility it provides. There is a range of different types of mobility (Poslad 2009):

- **Accompanied** - these devices are not worn or implanted, and can be either portable or hand-held, unattached, but transported in clothes or accessories;
- **Portable** – these are usually laptops, which are operated using two hands whilst being seated and are high resource devices;

- **Hand-held** – these devices can often be operated using one hand and, on occasion, hands-free, combining multiple applications such as communication, media recording and playback, as well as mobile office and are usually low resource devices;
- **Wearable** – these are devices, such as jewellery and accessories usually operated hands-free, sometimes autonomously, and are usually low resource devices, which can include watches, earpieces and glasses; and
- **Implanted or embedded** – these devices are often used for medical reasons, such as a heart pacemaker, usually enhancing the abilities of physical and mentally able humans.

In accordance with the definition of mobile devices, as defined in the previous section, this research focuses on hand-held mobility, in which users may carry their devices on their person and able to operate with one hand.

3.3 Mobile Device Capabilities

In addition to the increase in the amount of mobile devices, there has also been an increase in the capabilities of these devices (Natchetoi et al. 2007a). Mobile technologies have increased more rapidly than any previous technology and are, at present, the most ubiquitous technology worldwide. The influence of mobile technology and the implications on the lives of everyday people are extensive, as it empowers users with new abilities and capabilities, which allows them to connect to the information society as both contributors and as users (Botha et al. 2010).

Mobile phones dominate the international telecommunications market and exponential growth in the usage of these devices is recorded globally. Due to their massive popularity and flexibility, mobile phones have become equipped with hardware and software technologies such as Bluetooth, Global Positioning Systems, and digital cameras at greatly reduced prices (Natchetoi et al., 2008). It is this necessity that fuels the need for improved mobile device capabilities.

3.3.1 Hardware Capabilities

The hardware capabilities are categorised by the functions they perform or facilitate. On average, each mobile device is equipped with 30-40 hardware components grouped in the following categories:

- **Input** – The standard components for input include buttons and microphone (for communication using voice transfer). Other input components may include a touch screen, camera/s, microphone via headphone, light sensor, accelerometer, and proximity sensor;
- **Output** – These include the screen, receiver, loud-speaker, headphone audio jack, vibration, and flash for the camera;
- **Storage** – Storage on a mobile device is either internal, as flash memory, or external, as a storage card;
- **Processing** – All processing is carried out by the main processor;
- **Power** – Power is provided to the device by means of a lithium-ion battery; and
- **Memory** – memory is provided in the form of low-power memory modules.

3.3.2 Software Capabilities

The software capabilities available to mobile devices are limited only by the device itself and the functionality it can provide. The only required software on a mobile device is the operating system and the functionalities it can perform. All other software on a mobile device can be downloaded from third party software developers. Mobile software includes applications such as Internet browsers, email applications, games, and media players.

3.3.3 Network Capabilities

The primary contributor to the networking capabilities of mobile devices is the use of wireless connectivity. Wireless connectivity has provided a means where the primary advantage is the ability to connect mobile devices in locations where other infrastructure has either failed or proven too difficult and costly to implement. The most essential wireless connectivity feature, provided for all mobile devices, is the ability to communicate via voice transmission. Other wireless technologies are defined by specific architecture to meet a specific purpose, and include the following:

- **Bluetooth** – Bluetooth is a short distance, wireless protocol, which is free of any licence requirements (Madhavapeddy & Tse 2005) and is available from any Bluetooth-enabled device. Bluetooth was originally envisaged as a means for connecting mobile devices without the need for cables.
- **WAP** – Wireless Application Protocol (WAP) is a standard for network communications in a wireless network. It was proposed as a means to access Internet on mobile devices while providing for the special features of such devices (Juul & Jørgensen 2002). Although WAP connection speeds were fairly limited, typical usage includes sending and receiving of emails, and viewing news articles.
- **GPRS** – General Packet Radio Service (GPRS) provides packet-switched data transmission over the GSM network with efficient wireless protocols to cater for any erroneous data packets (Othman, Zakaria & Hamid 2008). GPRS is the most popular data communication system, utilised in more than 200 countries. GPRS provides transfer rates of between 56-114 Kbit/sec and is charged based on the amount of data downloaded to the mobile device.
- **EDGE** – Enhanced Data rates for GSM Evolution (EDGE) is a radio-based technology that allows improved data transmission over that offered by GPRS (Wandre 2002). EDGE is a 3G technology and was first deployed in 2003. It is capable of reaching download speed of up to 236 Kbit/sec.
- **HSPA** – High Speed Packet Access (HSPA) is a 3G standard supporting download speeds of up to 14.4 Mbit/sec (Tso, Teng, Jia & Xuan 2010). HSPA is available in 80 countries, which support 3G networks.
- **HSDPA** – High Speed Downlink Packet Access (HSDPA) is a 3G standard also provided 14.4 Mbit/sec download speeds (Sedoyeka, Almasri, Rahman & Hunaiti 2008), although most networks capable of deploying HSDPA provide download speeds of up to 7.2 Mbit/sec due to the limitations on mobile devices.
- **HSUPA** – High Speed Uplink Packet Access (HSUPA) is a 3G standard supporting download speeds of up to 5.76 Mbit/sec (Li, Zaki, Weerawardane, Timm-Giel & Goerg 2008). HSUPA offers improvements over HSDPA as it supports enhanced uplink features, such as Voice Over IP (VoIP), and the sending of much larger emails.
- **HSPA+** - Evolved High Speed Packet Access (HSPA+) is a standard for 4G, Universal Mobile Telecommunications System (UMTS) enabled networks (Tokgoz,

Meshkati, Zhou, Yavuz & Nanda 2009). Released in July 2010, HSPA+ supports download speeds of up 42 Mbit/sec.

- **Wi-Fi** – Wireless Fidelity (Wi-Fi) is a technology based on IEEE 802.11 standards and is a wireless local area network that allows mobile devices to connect to the Internet (Flickenger, Okay, Pietrosemoli, Zennaro & Fonda 2008). It has become a feature of mobile business.

3.4 Everyday Uses of Mobile Devices

The growth in usage suggests a change in the way these devices are being used by people. Mobile devices are being used to a greater extent by different people in different types of environments. In addition to its wide use as standalone personal organisers and voice communication devices, mobile devices are used for many daily tasks, examples of which include the following:

- **Entertainment** – Except for conventional voice and SMS communication, the most widespread use for mobile devices is that of entertainment. Entertainment on a mobile device includes playing music, watching movies, and playing games (Huang 2009). Personalisation has become a key component of mobile devices, which includes changing the user interface of a mobile device by way of downloading ringtones, wallpapers, screensavers, themes, and widgets. Entertainment forms one of the largest market areas for mobile sales, so much so that mobile devices are seldom sold by manufacturers without some form of entertainment such as preloaded games, music and video (Huang 2009).
- **Medicine** – In medicine and eHealth fields, mobile devices are being used to help diagnosis of patients, add to the decision-making process and enhance the effectiveness of communication between the patient and the hospital throughout follow-up treatments (Velde, Atsma, Hoekema, Luijten, Buddelmeijer, Spruijt, & Putten 2004). Patients are also able to install applications on their mobile devices to facilitate or guide them in specific situations.
- **Education** – Mobile devices are being used in different schools and universities for certain applications to assist conventional teaching techniques and increase teacher-student collaboration. Applications such as electronic dictionaries, translators, scientific calculators and remote presentation controls are common educational software for mobile devices (Huang 2009). In recent years, educational institutes

have gone beyond supplying assistance via mobile devices to essentially teaching using mobile learning. Although many challenges, such as psychological, pedagogical and technical limitations exist, m-learning can be conducted using mobile devices (Shudong & Higgins 2006).

- **GPS** – In recent years, Global Positioning Systems (GPS) have become a valuable and influential functionality asset when purchasing a mobile device. The GPS satellite navigation system is owned and operated by the U.S. Department of Defence and internationally accessible for everyday use. This system uses a receiver to obtain a signal from orbital satellites to triangulate the geographical location of the device (Huang 2009).
- **Business** – As stated earlier, mobile employees spend a third of their time away from the office and half the time away from their desks (Johnson et al. 2005). This has prompted the need for business solutions available from mobile applications. These applications provide functionality, such as mobile office, reading and sending emails, calculators and converters, organisation through calendars, and Internet browsing.
- **Daily Living** – Mobile users have, in recent years, utilised their phones to perform everyday tasks. These tasks include online booking, such as tickets for movies, concerts and flights, and online banking, such as account payments and money transfers. In more recent times, mobile users have started using their devices for online chatting and updating blogs, which just five years ago were not considered an everyday task.

The continuous growth and usage of mobile devices has resulted in an increase in the amount of information transferred to and from these mobile devices. It is, thus, important that information be managed more efficiently, not only to reduce cost, but also to reduce the overhead required to transfer this information. This can be achieved by means of file compression, in which information is manipulated in such a way as to reduce the amount of space required for storage or transfer over a network. File Compression is discussed in Chapter 4.

3.5 Matters to Consider in Mobile Computing

The large volumes of information available both online and offline continue to grow rapidly. Living in an “information society”, people are showered with information whether they actively seek it or not (Edmunds & Morris 2000). Mobile devices, which were initially proposed as secondary voice call devices have become small computers required to do almost anything, anywhere (Johnsrud et al. 2008). Additionally, people require more in terms of the information access, better technologies, and easier access to this information.

Mobile devices and standards continue to improve, and are frequently becoming enterprise application delivery platforms (Natchetoi et al. 2008). Most business applications require significant amounts of data processing, either locally or through high-speed networks. Most of existing cell phones, however, cannot fulfil these requirements, as mobile devices are characterised by their limited resources. This presents many new challenges as mobile devices are now required to support most computing paradigms while different brands, run-time platforms, screen sizes, networks and many other aspects all add to the fact that there is a large variety of mobile devices on the market (Venu 2008).

As with all new technologies mobile devices have many challenges, which might always be a restrictive factor. Due to the fact that mobile phones were initially designed as simple voice transmission devices, many components of the device as an everyday computing platform were overlooked. These issues are major factors in new technological developments and mobile enhancements. Usability, ease of access, and efficiency are the main design requirements frequently not effectively addressed, as tradeoffs need to be made on mobile devices to improve portability, where users are restricted by small screen dimensions, high network costs and varying support for fonts and colours (Wang & Sajeev 2007). Usability and accessibility are critical on mobile devices due to limited screen size or difficult navigation, which have a negative influence on a variety of users. Mobility issues are discussed in the following subsections and classified as either hardware, software, network or human issues.

Hardware Issues

Because of the restrictions in its design, mobile devices have a number of hardware issues, arising from a necessity for portability, as well as size and weight reduction. These issues include the following:

- Reduced Screen Size – Limited screen size may lead to difficult viewing of text or images (Huang 2009), which were initially designed for viewing on standard desktop computer monitors, and not on small screens designed for the mobile environment. Mobile device screen resolutions typically vary between 96x65 and 800x480 pixels, and are not envisaged to reach the resolutions of desktop displays. These screens will remain small, as mobile devices are required to be portable.

- Insufficient Processing Memory – As mobile devices are expected to have greater capabilities, so have the demands for high-speed, low-power RAM increased. An average mobile device has 64MB of RAM, too small to perform very complex computational tasks. In recent years, a demand for larger RAM for more demanding mobile applications has propelled the development of 512MB low-power memory modules. The majority of consumer mobile devices, however, still have limited memory capacities.
- Lack of Processing Power – The most recent processors for mobile devices have reached clock speeds of 1 GHz. Although this provides better computational capability, phone processors will be restricted to around this speed, as higher clock speeds lead to higher power consumption on the battery, which reduces mobility (Barton, Zhai & Cousins 2010). A number of approaches have been designed to combat this issue. SOA, more specifically Web Services, is one solution to reducing computational strain on a device.
- Low Battery Life – Due to the fact that mobile devices are required to operate with a limited battery charge, power consumption has become an important issue in design. Elevated processing capability does not lead to enhanced performance of a mobile device, as battery capacities are not undergoing the same rapid growth rates (Ravi, Scott, Han & Iftode 2008). Mobile devices have, however, advanced to more than just communication devices and because of this the battery life expectancy has diminished to between one and three days. Clever, uncomplicated solutions are needed to deal with the challenge of increasing processing capacity while increasing battery life and preserving pocket-sized portability for mobile devices.
- Poor Storage Capacity – When using mobile phones for functions other than just voice or SMS communication, limited onboard storage capacity hampers the usage of the device and requires some form of external storage, such as a storage card. Onboard storage capacity ranges anywhere from an average of 10 MB to 100 MB in basic mobile devices, to an average of 1 GB in the more advanced mobile devices. Even though storage cards can provide storage of up to 32GB, this still trails what is considered to be sufficient storage for a desktop computer.

- Input Limitations – Word input speed is less than ten % when using mobile devices compared to that of a computer keyboard (Shudong & Higgins 2006). This is a direct result of the input methods used on mobile devices where buttons are often smaller than the tips of a human finger. These input methods are slow and inconvenient and frequently lead to repetitive strain injury.

Software Issues

Software issues are sometimes a result of the hardware for the mobile devices. Software issues on mobile devices include the following:

- Navigational and Browsing Difficulties – Undersized screens of mobile devices is the major cause for most navigation and browsing difficulties. Wang & Sanjeev (2007) demonstrated that in order to facilitate information display suited for larger screens the information is required to be split into smaller presentation units that is able to fit into the small screens of mobile devices. This directly affects the difficulty of organising information effectively to aid navigation to and from the desired information (Huang 2009). Most common Internet Web pages are distorted on mobile phone screens and a great deal of multimedia elements are lost (Shudong & Higgins 2006). It is therefore important when designing Web pages to consider mobile devices, either by designing for both desktop and mobile devices or adding special Web pages specific for use on mobile devices.
- Visual Constraints – Even though images and icons are typically considered as important types of data and information visualisation in desktop computers, it is, however, quite restricted in mobile devices with regard to the display of graphical depictions, such as images, drawings, diagrams, maps, and logos (Huang 2009). Therefore, the downscaling of an image from its initial size, used for viewing on larger desktop screens, to a suitable size, suited for a mobile device screens, is an important research requirement that should not be overlooked (Rist & Brandmeier 2002).
- Content Layout – On a desktop we often contend with varying screen sizes, but standards and HCI principles have been developed as to which sizes should be accommodated. For mobile devices there are numerous amounts of varying screen sizes and resolutions, making it difficult to predict the sizes for which content needs to be designed. Lengthy text, incorrectly spaced layout, variable navigation,

inadequately formed link text, and longwinded headings can all be a factor in making a site less legible for many users. Intelligent use of colour, images, and layout can assist the ease of readability on desktop screens and is equally, if not more relevant when it comes to mobile device screens.

<u>Network Issues</u>

Mobile devices have a base requirement to be connected to a wireless network. Common functions of mobile devices, such as voice communication, often have adequate infrastructure to support these functions. However, wireless data connectivity is often troublesome and, at times, non-existing. These issues include the following:

- Unreliable Connections – Due to the innate characteristics of wireless networks, communication is often inferior when compared to fixed networks, mainly due to substantial transmitted packet loss, and the inherent complications in sustaining fast wireless connections over considerable distances and longer intervals (Ojala 2005). Reliability could also be as a direct result of ineffective and inadequate infrastructure provided by mobile network operators.
- Limited Communication Bandwidth – There are numerous causes for the inadequate bandwidth provided by networks, most of which result in quite poor latency and radically fluctuating connection quality, leading to inadequate bandwidth (Ojala 2005). In certain cases, infrastructure used by mobile networks is insufficient to deal with the amount of connected mobile devices in specific locations, and only offers limited communication bandwidth to each user.
- High Costs – Irrespective of the issues regarding reliability of connections and limited communication bandwidths, it is common practice for mobile network companies to attach a cost to any communication provided to the mobile device, even to that of receiving data. Over and above this, higher bandwidth, which is not always guaranteed equates to higher costs, in certain instances. Because of the mediocre connections provided, loss of packets transferred is a frequent occurrence. This frequently results in those packets having to be retransmitted, which, consequently, relates to higher costs to the user. In South Africa, an average fee of R2 per MB is observed for wireless downloads. In comparison, fixed land line charges average R29 per GB, or approximately 3 cents per MB, downloaded and offer superior transfer

speeds. This further highlights the need for smart file transfers, which reduces the amount of overhead required for transfer to and from a mobile device.

<u>Human Issues</u>

Regardless of the common issues, human acceptance still plays an important role in the design of mobile devices. These issues include the following:

- Psychological – In some instances, continually developing new technologies is not always ideal, as humans take time to adapt to change. An example of this is that email and digital signature technology have been available for everyday purpose for many years, yet people still tend to use telephone or fax as forms of confirmations and, therefore, in these cases, the issues for mobile devices are not network technologies or limited bandwidth, but rather of habit (Shudong & Higgins 2006).
- Hype – Although there has been no direct verification that continuous use of a wireless device, such as a mobile phone, causes brain or aural damage, many individuals still distrust the frequent use of these devices for these reasons and in some cases considerations, such as these, have kept people from using mobile phones altogether (Shudong & Higgins 2006).

These four categories, hardware, software, network, and human issues, are critical in the design of mobile devices. The lack of resources available on mobile devices, such as processing power, memory, storage capacity and communication bandwidth, are inadequate to deal with the large amounts of information transferred to the devices or perform computationally expensive business tasks (Natchetoi et al. 2007b). It is, thus, important that any solution related to these issues is simple, smart and efficient.

Mobile SOA may provide an answer to several of these concerns. The lack of processing power, limited processing memory and storage capacity, poor battery life, unreliable connections, low bandwidth, and high cost may be eased with intelligent use of a framework within Mobile SOA. Mobile SOA is discussed in more detail in the following section.

3.6 Mobile Service-Oriented Architecture

Mobile Service-Oriented Architecture (Mobile SOA) is the ability to employ SOA within a mobile environment. In theory, providing a Service to a mobile application is no different to that of other computing platforms. In essence, any SOA can be implemented on any mobile infrastructure, as long as the device supports it. However, the limiting resources of mobile devices and its inherent implementation difficulties mean that supporting SOA within a mobile environment is not easily realised.

Mobile Services are divided into two categories, Content Services and Added Value Services. Content Services provide delivery of content, such as ringtones, greeting cards, wallpapers, and music. Services that provide end users with added service functionality are more challenging to invoke (van Gurp et al. 2005). These include access to information resources, such as searching, language translation, newspaper reports and weather forecasting, telemetry, mobile shopping, reserving rental cars, restaurants, mobile banking, and m-government (Sanchez-Nielsen et al. 2006). Mobile SOA is realised by using Web Services to leverage the standardised Web technologies for a mobile device. As defined in Section 2.5.1., Web Services are independent of platform and programming language. This allows Web Services to be designed for mobile environments.

Mobile Services and solutions hold the promise of transforming African businesses in the same manner in which the Internet transformed organisations in the more affluent countries (Botha et al. 2010). In the following subsections, Mobile SOA is formally defined, technologies, components and drawbacks discussed, along with existing systems and projects.

3.6.1 Definition

Mobile Service-Oriented Architecture has no formal definition as it is simply an extension of SOA on a mobile platform. Mobile SOA is the by-product of the convergence between the fields of mobile computing and SOA. For the purposes of this research Mobile SOA is defined, similarly to SOA, as: *"An architectural style for the mobile environment whose goal is to achieve loose coupling by positioning Services as the primary means through which solution logic is represented."*

3.6.2 Component Model

An SOA-based mobile Service architecture extends the basic SOA with slight adjustments to support the mobile environment. Jørstad et al. (2005) illustrates a new outlook of an SOA that supports mobile Services, Figure 3.2. Due to the fact the key components of a mobile Service are equivalent to existing SOA framework, and are exposed as Services of their own, the only additional component is that of the Mobility Controller. The Mobility Controller handles the state transfer and coordinates the overall mobile Service. A Mobility Controller stub is required in each composite Service that can coordinate actions towards the mobility controller.

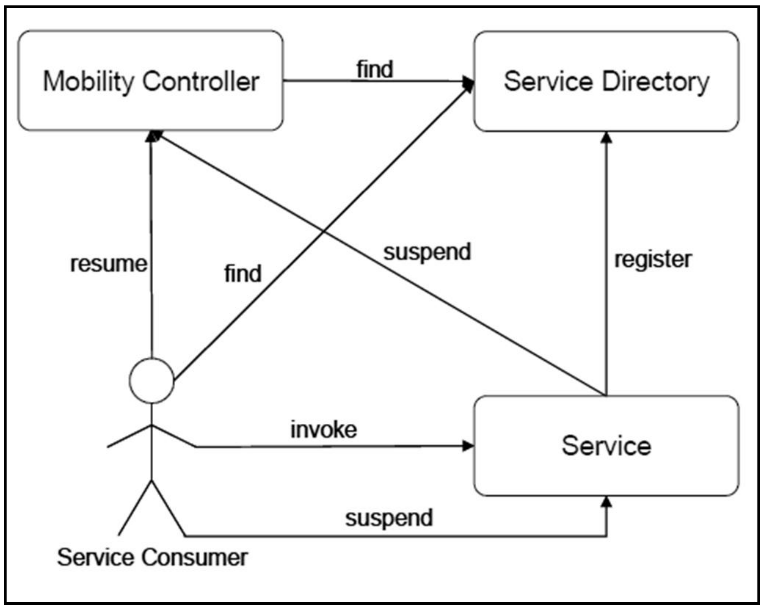


Figure 3.2: Simplified View of SOA with Support for Mobile Services (Jørstad et al. 2005)

The framework proposed in this research forms a specific section within the Mobility Controller, which manages information transfer to mobile devices. The requested Service is sent to the Client via the Mobility Controller, which manages the files more efficiently before the files are sent to the mobile device. The mobile application receives these files and then builds the requested Service. The proposed framework is discussed in more detail in Chapter 5.

3.6.3 Mobile SOA Benefits

Mobile SOA and Mobile Web Services have a number of benefits above that of the mobility it provides. These Services should, by definition, be available at any time and any place using any device with wireless capabilities (Jørstad et al. 2005). Mobile SOA plays an important role in delivering Services, such as education, health, business operations, banking procedures, weather updates, and other similarly required information. These Services are of particular significance to rural communities where access to services, such as banks and clinics, are not as easily accessible as in residential and urban areas.

Businesses require that their employees make business decisions with their mobile devices (Natchetoi et al. 2008). In addition to providing Services to the rural community, Mobile

SOA also offers additional capabilities to businesses by way of providing their employees with information access even when not in their office environment.

The most valuable aspect of Web Services lies in its interoperability, which also applies to Mobile Web Services within Mobile SOA. This interoperability allows different systems to interact with one another, regardless of any platform or programming language it was implemented in. Information exchange is an important factor when designing any Web Service, including that of Mobile Web Services. The information is thus easier to deliver to the mobile device, without having to consider its make and model.

3.6.4 Mobile SOA Drawbacks

In addition to the issues of mobile devices, as discussed in Section 3.5, Mobile SOA suffers from drawbacks of its own. Because of the varying numbers of mobile devices, there are numerous mobility deployment obstacles. In order to provide an application that consumes Web Services, or alternate equivalents, it must allow every different mobile network configuration to connect wirelessly, and because of this, implementation of such applications is often complicated and time consuming. In most cases, this swells additional costs, which is not ideal, as mobile applications are often required to be low cost equivalents of desktop applications.

Existing SOAs are commonly used to offer Web Services on wired networks (Sanchez-Nielsen et al. 2006), and not much attention has been devoted to the development of Service protocol for mobile devices despite the obvious trends in the market. Web Services pose additional drawbacks as the XML files used to transfer information between the Client and Server are often verbose their structure (Natchetoi et al. 2008). This highlights the need for intelligent compression of these XML files.

3.6.5 Extant Systems and Related Work

Before addressing the issues of Mobile SOA or proposing any solution to these issues, a literature review on extant systems is required. This section discusses two existing systems, which relate to this research.

3.6.5.1 M-Service Framework (Sanchez-Nielsen et al. 2006)

The framework used consists of four components, illustrated in Figure 3.3. These are the Service providers, Service managers, Service Clients, and UDDI registry, all of which are based on an XML infrastructure to provide uniformity. The framework's components are defined and described as follows:

- **Service Providers** – are the owners that implement and offer different Services and define descriptions of their Services using WSDL specifications;
- **Service Clients** – are wireless device-oriented users interested in standard Services and searching of facilities when it is needed, without prior knowledge of available Services and bringing these facilities to their wireless devices in a transparent way;
- **Service Managers** – act as a mediator layer between the Service providers and mobile devices and are responsible for information flow between both components. A Service manager is a Web Service entity that uses dynamic invocation interface (DII) as a communication mechanism between the different Service providers. With DII, a Service manager can invoke Web Services without knowing their communication interface at compile time;
- **UDDI Registry** – can be used by mobile users in order to locate new Services. Service discovery is computed at runtime by the Service manager, once the user has sent their request of new Services in the UDDI registry; and
- **XML-based Infrastructure** - a uniform infrastructure using XML-encoded data exchange is used with two purposes: to define the Service registry structure and establish the communication between the mobile device and the Service manager.

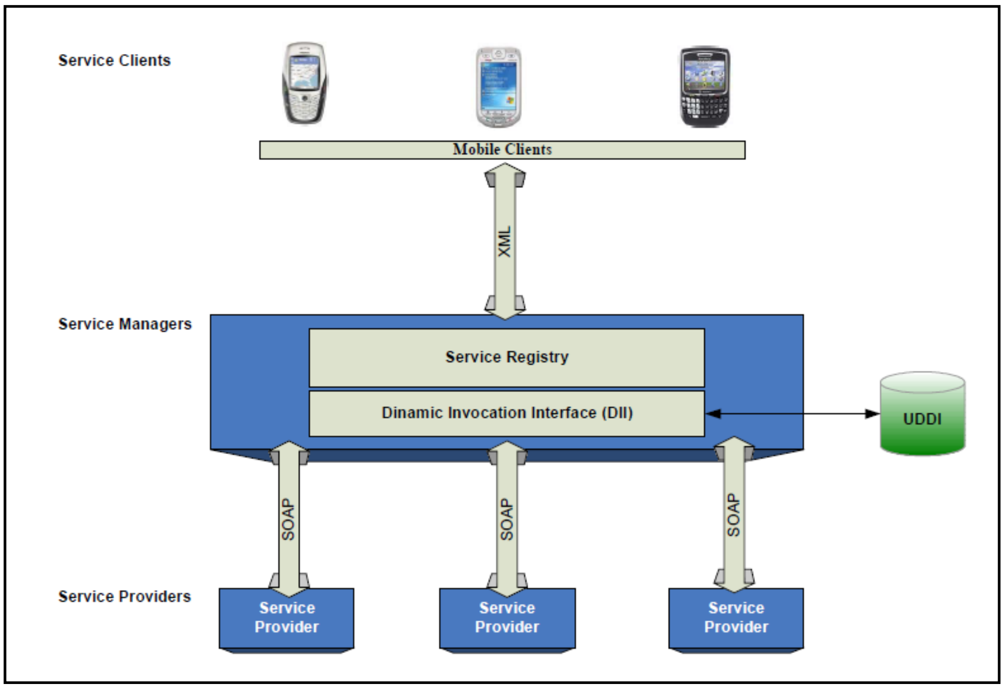


Figure 3.3: Dynamical M-Services Architecture (Sanchez-Nielsen et al. 2006)

In the approach used above, the business logic was assigned to Service managers, resolving the issues faced by direct access from mobile devices to the Web Services. Additionally, this approach reduces the computational cost of the mobile devices and, therefore, optimises the response times to mobile users and memory resources.

In their research, Sanchez-Nielsen et al. (2006) proposed mobile devices as being complete contributors in networked SOAs to increase the variety of accessible Services and innovative business opportunities in the mobile space. A framework was proposed that allowed Service Providers to create, update and change Services at any time and mobile users may locate new Services without knowing the accessible Services. Their prototype was implemented using open source tools, and found that the implementation of the Web Service technology using these tools was suitable.

Sanchez-Nielsen et al. (2006) established that there were some drawbacks within their prototype, as the UDDI registry is not directly accessible by mobile devices. Furthermore, the implementation specifications were required in order to support complex functions when their *dynamic invocation interface* was used.

3.6.5.2 EXEM (Natchetoi, Wu, Babin & Dagtas 2007)

In their research paper, Natchetoi et al. (2007b), introduced a compression approach to the domain of Mobile SOA. This approach, EXEM, manages XML documents exchanged between a client and server in the context of mobile applications. It was intended to address the issue of limited resources of a mobile device, such as the lack of bandwidth available, limited storage capacity and computational capability. EXEM is a framework based on compression of XML documents.

The experiments conducted produced significantly improved compression ratios compared to that of other compression techniques, indicating the potential for more efficient file transfer to and from mobile devices. The EXEM framework is discussed in more detail in Chapter 4, as part of the design process for the framework proposed in this research.

3.7 Conclusions

As the number of mobile devices increases and more people require information access via these devices, so the need for computing paradigms for the mobile environment increases. Web Services is a key standardised technology used to provide Services for mobile devices in the field Mobile SOA. There are numerous benefits of providing Web Services to mobile devices, and although there are issues, there is no reason why Mobile Web Services cannot provide the level of information access required by businesses. In recent years, the general public have also developed a need for information access. Thus, Mobile SOA has become an increasing aspect of people's daily lives, both for work and personal use.

Sanchez-Nielsen et al. (2006) proposed a framework, which allowed Service Providers to create, update and change Services at any time while mobile users may locate new Services without knowing the accessible Services. Their proposed framework consisted of four components namely the Service Providers, Service Clients, Service Managers and UDDI registry, which interacted via XML-based messages. Natchetoi et al. (2007b) proposed a compression framework known as EXEM. Their proposed framework supports efficient data exchange, using compressed XML, between the Server and Mobile Client.

The files used within Mobile SOA, both data and media files, play an important role. User requests are performed by way of Web Services invoked within a mobile application or browser. Smart management of resource-heavy files is essential when being transferred to mobile devices to successfully complete user requests in less time and at reduced cost. A popular management technique for these files makes use of compression, discussed in the following chapter.

Chapter 4: File Compression

4.1 Introduction

In the computing domain, compression is widely used (Blelloch 2001). The images received via the Internet and the video transmitted via satellite are examples of everyday media that make use of compression. The application of specially designed algorithms, suited for specific types of files, allows the process of compression to transpire. These algorithms use data structures, such as hash tables, trees, and dictionaries to compress files. The main aim of this research is to optimise file transfer within Mobile SOA by means of compression, as the resources available on a mobile device are limited.

This chapter provides answers to research question R3, *"What are the issues related to compression?"*, and all the sub-questions of R3. In answering research question 3 and all its sub-questions, this chapter is divided into nine sections. Section 4.2 provides a brief discussion of compression. Section 4.3 and 4.4 discuss the two types of compression, lossless compression and lossy compression, respectively. Textual data compression, image compression and audio compression are discussed in Section 4.5, Section 4.6 and Section 4.7, respectively. This is followed by a brief discussion on video compression in Section 4.8, and why it is not suited for the proposed framework. The advantages and disadvantages of compression are discussed in Section 4.9. This is followed by a discussion on how to choose the best compression algorithm in Section 4.10.

Before the different types of compression are discussed, a formal definition of compression is required.

4.2 What is Compression?

When referring to compression in terms of any Computer Science it often refers to data compression. Data compression is the technique of removing redundant information from a message to minimise the number of bits that need to be stored on a disk or transferred over a network. Another definition is that compression is the process of reducing the size of a file by encoding its data information more economically (Investintech 2006). Simply stated, data compression is the process of encoding information using less bits.

For this research, compression is defined as: *“the procedure in which information is reduced for storage or transmission, using either a lossless or lossy compression technique.”* This definition will be used for the remainder of this research when referring to compression. Furthermore, the terms *“coding”* and *“encoding”* will be encountered later, and are often used when referring to compression.

Most compression algorithms take advantage of the fact that data contains a lot of redundancy. The compression algorithm requires a two-step procedure in order to be effective. Firstly, information is said to be compressed after an encoding algorithm is applied to it. This encoding algorithm, if successfully completed, reduces the amount of storage space required to save the information on a computer or mobile device. However, in order to use this information again, the information has to be decompressed by means of a decoding algorithm. The decoding algorithm recreates the original information if a lossless compression technique was used. If a lossy compression technique was used, a reduced, but usable, version of the information is created. Data compression only works when both the sender and receiver of the information share knowledge of the encoding algorithm.

All compression algorithms consist of two distinct components known as the model and coder. Figure 4.1 shows the general framework of a model and coder. Using different techniques for different compression standards, the model component encapsulates the probability distribution of the information by knowing or revealing something about the structure of the input (Blelloch 2001). The decision to produce a specific code for a specific character, or set of characters, is based on the model component (Nelson & Gailly 1995).

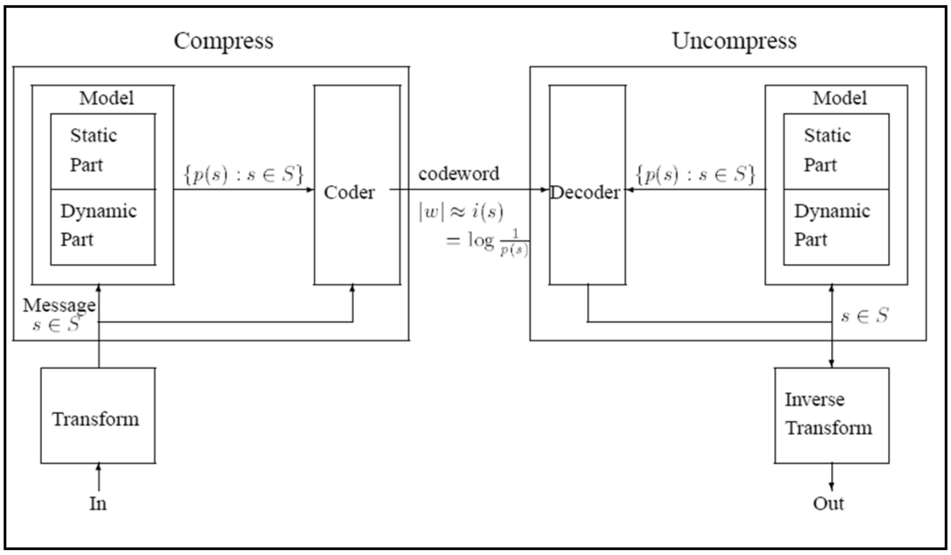


Figure 4.1: General Framework of a Model and Coder (Blelloch 2001)

The coder component, also known as the encoder, utilises the probability biases generated by the model component to generate codes. The process undertaken to generate codes lengthens low probability messages and shortens high probability messages. Figure 4.2 shows a more specific model and coder (Huffman Encoder), which makes use of statistical probabilities.

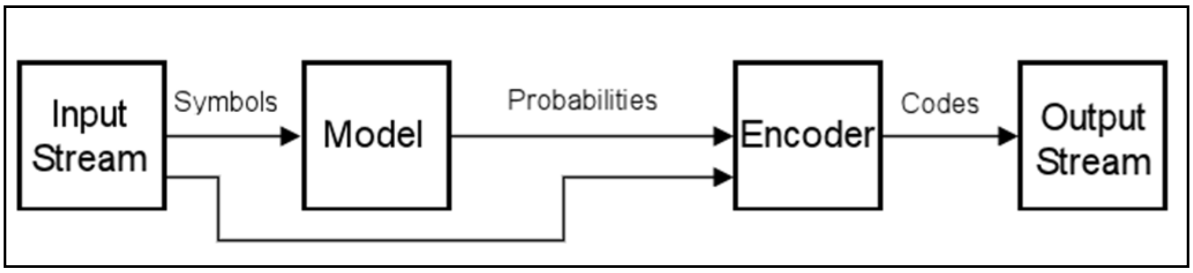


Figure 4.2: A Statistical Model with Huffman Encoder (Nelson et al. 1995)

When discussing compression, it is important to understand what entropy is. Entropy is the measure of disorder of a system – more entropy implies more disorder. Negative entropy is added to a system to provide more order to that system. When compressing information, the entropy of that information is reduced, i.e. there is less disorder. The original level of disorder, entropy, is restored during decompression, but only when lossless compression was used.

Compression is categorised into two groups, namely, lossless and lossy compression. Lossless compression is the process by which information can be compressed for storage or transmission, and then decompressed to the original information without any loss. Lossy compression, however, allows some loss of information, which is either not noticeable or is of an acceptable lower quality.

4.3 Lossless Compression

Lossless data compression is so named for what its algorithm achieves. A lossless data compression file can be restored to its exact state, as it was before compression. The process involves finding repeated patterns in the input information and encoding those patterns more efficiently. Using this logic, a lossless data compression algorithm reduces the redundancy of information. Consequently, lossless compression is not ideal for compressing files consisting of random information, such as with images and audio, as the algorithm is dependent on discovering patterns. This makes the lossless compression algorithm ideal for textual data and software. Examples of coding (encoding) methods used within a lossless compression algorithm will now be discussed.

4.3.1 Run-Length Coding

Run-length coding (RLE) is a compression technique that compresses any sequence of at least four repeating characters. RLE reduces the size of a repeating string of characters, called a run. RLE can compress any type of data regardless of its information content, but the content of data to be compressed affects the compression ratio. Consider a character run of 10 'A' characters, which usually require 10 bytes of storage space. The characters are replaced with a compression code '10A', one of the characters and a value that represents the number of characters to repeat, which only requires two bytes to store the same character run. The number '10' is referred to as the count and the character 'A' referred to as the character. A more complex example is the string "AAAAAbbbbCCCDD", which is stored as 5A4b3C2D, reducing the original 15 byte string to 8 bytes.

RLE is the simplest of compression techniques, which is easy to implement and quick to execute. This makes it a popular basis for data compression algorithms. RLE can, however, not achieve high compression ratios compared to other, more advanced compression methods.

4.3.2 Variable-Length Coding

Variable-length coding (VLC) uses the fact that certain characters in a string are more common than others. VLC essentially maps input characters to a variable number of bits. The advantage of VLC over RLE is that the more repeated characters are coded with fewer bits, while the less occurring characters are coded with more bits, resulting in fewer bits being used to represent the entire set of characters (Li & Drew 2004). This is advantageous when transferring information across a network, where the length of time needed for transmission is proportional to the amount of characters needed to encode it. VLC enables lossless compression with zero error, and can be read back character by character. The Shannon-Fano, Huffman Coding, and Adaptive Huffman Coding algorithms are popular compression algorithms utilising the principles of variable-length coding.

4.3.3 Dictionary-Based Coding

Dictionary-based coding (DBC) is a lossless compression algorithm that reads in input and searches for groups of characters that appear in a stored dictionary. If a string match is located, an index, pointer or codeword linked to the dictionary, which contains the pattern, can be used as an output instead of the code for that character (Seong 2006). This results in an efficient use of space as information is stored only once, in a dictionary, and the compressed file consists mainly of pointers to that dictionary. The longer the resulting matches between the string and dictionary entry, the better the compression percentage.

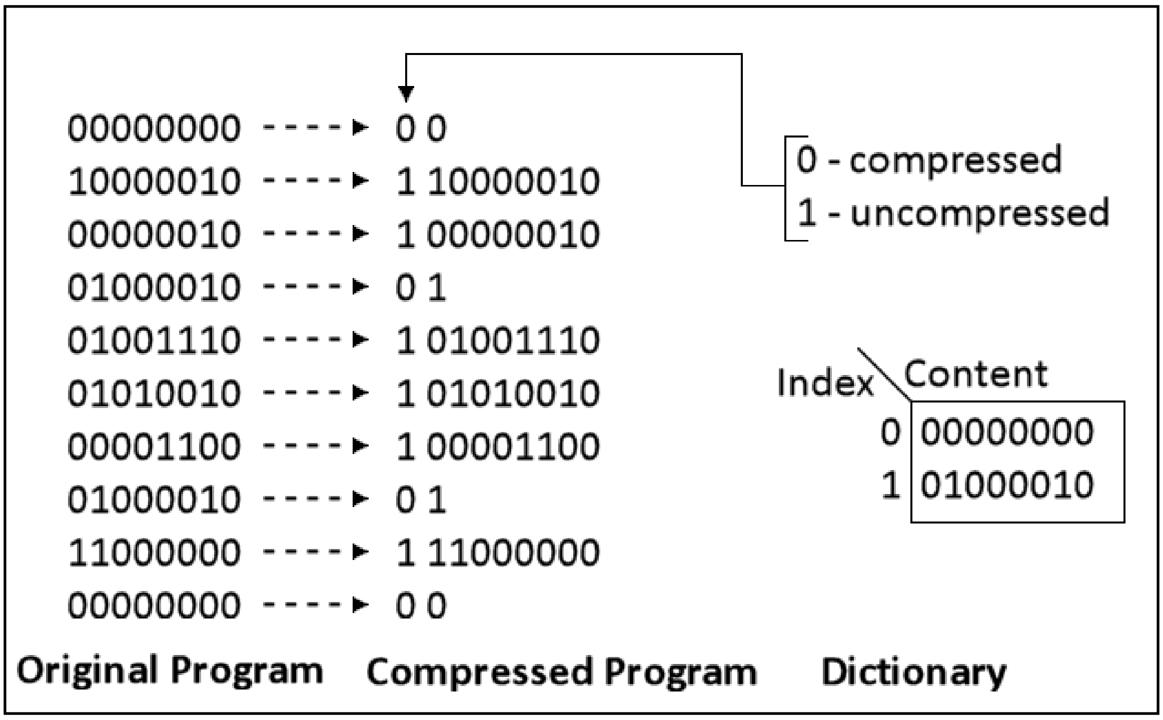


Figure 4.3: Example of Dictionary-Based Code Compression (Seong 2006)

DBC provides efficient compression, in addition to fast decompression. The dictionary also contains decompression instructions used to decode the compressed information. Figure 4.3 shows an example of a dictionary-based coding compression using a simple program binary, where the binary consists of ten 8-bit patterns, a total of 80 bits and the dictionary contains two 8-bit entries. The output compressed program needs only 62 bits and the dictionary needs 16 bits. The more matches the dictionary contains, the better the compression yielded.

The static dictionary poses problems matching that of the problem the user of a statistical model faces: the dictionary is required to be transmitted along with the compressed information, resulting in a certain amount of overhead added to the compressed information (Nelson et al. 1995). The use of an adaptive dictionary method eases this problem. The Lempel-Ziv-Welch (LZW) algorithm makes use of an adaptive, dictionary-based compression technique, which, unlike variable-length coding, uses fixed-length codewords to represent variable-length strings of characters that frequently appear together, such as the words in English text (Li & Drew 2004).

4.3.4 Arithmetic Coding

Arithmetic coding (AC) is a more recent coding method, which is based on the initial idea originally established by Shannon (1948). The logic behind arithmetic coding is to assign to each character an interval. Starting with the interval [0, 1), each interval is divided in a number of subintervals. The sizes of each interval are proportional to the existing probability of the equivalent character of the alphabet. The resulting subinterval from the coded character is then allocated as the interval for the succeeding character. The generated output is the interval of the last character.

Unlike Huffman Coding, AC does not use a discrete number of bits for each character to compress. Arithmetic codes almost always yield better compression than prefix codes, but lack the direct correlation between events in the input data set and bits in the coded output file (Howard & Vitter 1994). However, despite this, the main benefits of AC are its optimal and inherent separation of coding and modelling.

Lossless compression algorithms provide efficient information transfer and storage without affecting quality. Although there are continued improvements being made to lossless compression algorithms, the level of compression does not match that of lossy compression algorithms. Due to the characteristics of lossy compression, however, it is not suited for compression of textual data (e.g. information stored within a database), and other stored files in which information retention is important. Lossy compression is discussed next.

4.4 Lossy Compression

Lossy compression is a type of compression, which results in a certain level of information lost from the original information sequence. The resulting compressed information can, therefore, not be restored to the original information (Blelloch 2001). A lossy compression algorithm is typically used for images, audio and video compression, where a decrease in quality is indiscernible, or is of an acceptable lower quality.

Despite the fact that information is lost, it does not mean that the quality of the compressed output information is always reduced. An example of this would be the occurrence of random noise, which has high information content, and when noise is prevalent within an image or audio file, the reduction thereof would be acceptable. Due to the limitation nature of the human ear and eye, certain information would be unnoticeable, as certain frequencies of sound are inaudible and certain levels of detail are indiscernible. These reasons contribute to lossy compression algorithms on images and audio often resulting in better compression by a factor of two over their lossless compression counterparts providing flawless compression (Blelloch 2001).

It is important to ensure that the degradation in the quality of information provided is minimally objectionable to the intended viewer/hearer. However, there is a limit to the amount of quality loss permissible. In conjunction with this, continually compressing and decompressing information within an image or audio file will result in it gradually losing quality. With lossless compression algorithms, this is not the case, as quality is never lost even over numerous repeated compression and decompression cycles.

Four lossy compression algorithms are discussed in the following subsections.

4.4.1 Quantisation

Quantisation is the process of reducing the number of distinct output values to a much smaller set. In some form or another, quantisation is the backbone of any lossy compression technique (Li & Drew 2004). The input and output of a given algorithm (quantiser) can be either scalar or vector values. There are three main types of quantisation:

- **Uniform Scalar Quantisation** – A uniform scalar quantisation process partitions the domain of input values into uniformly spaced intervals, with exception possibly to the two boundary intervals. The output or reconstruction value associated to each interval is taken to be the midpoint of the interval. There are two types of uniform scalar quantisers, namely midrise and midtread quantisation. Midrise quantisers have an even number of output levels, while midtread quantisers have an odd number of output levels. Figure 4.4 shows the difference between the two types of uniform scalar quantisation.
- **Non-uniform Scalar Quantisation** – If the source input is not uniformly distributed, a uniform quantiser may be inefficient. Increasing the number of decision levels within the region where the source is densely distributed can effectively lower granular distortion.
- **Vector Quantisation** – According to Shannon's information theory on data compression, any compression system performs better if it operates on vectors or groups of samples rather than individual characters or samples. Instead of using single reconstruction values as in scalar quantisation, vector quantisation makes use of

code vectors with n components. A collection of these code vectors form the codebook. Figure 4.5 shows an example of the basic vector quantisation procedure.

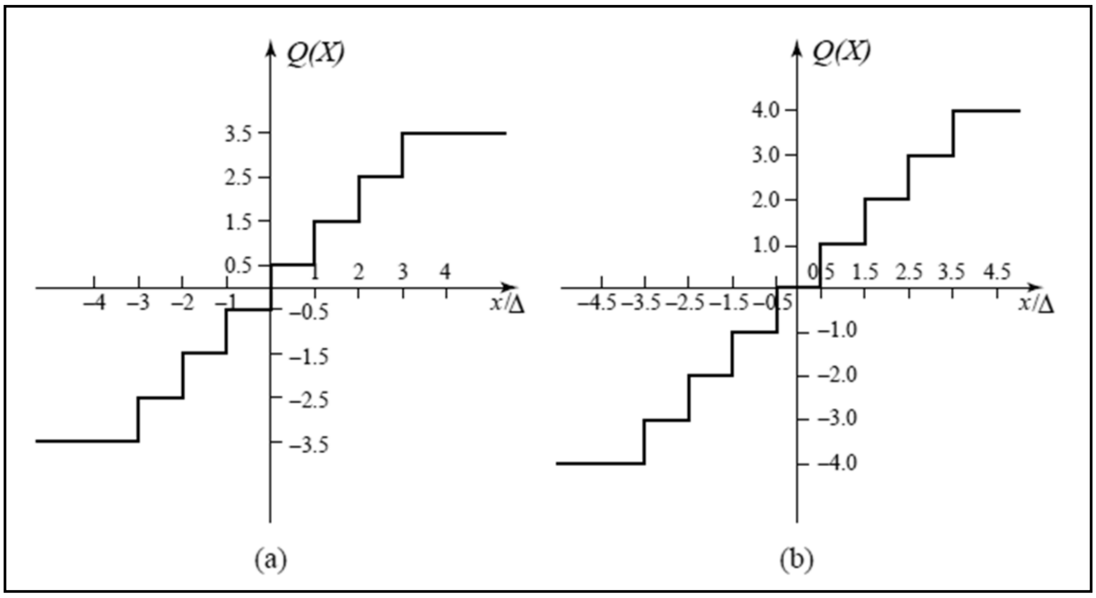


Figure 4.4: Uniform Scalar Quantisation: (a) Midrise, (b) Midtread (Li & Drew 2004)

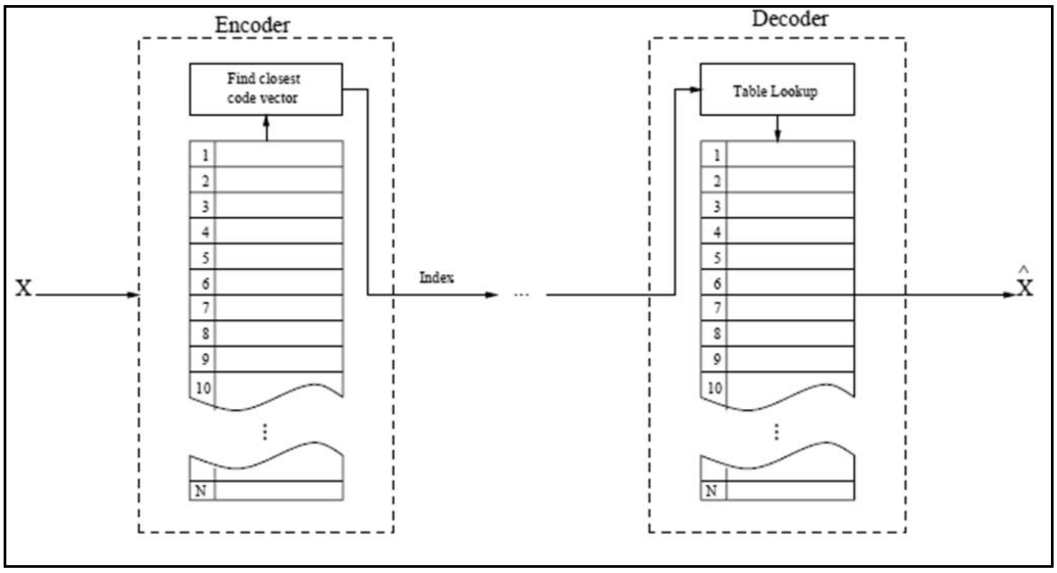


Figure 4.5: Basic Vector Quantisation Procedure (Li & Drew 2004)

4.4.2 Transform Coding

Transform Coding (TC) is a lossy compression technique in which the idea is to transform input information into a different form, which can then improve compression, or allow the algorithm to more easily release certain information without as much qualitative loss in the output (Blelloch 2001). This transformation is based on utilising interpixel correlation. The transformation component chooses which format of input source should be quantised and encoded. If the bulk information is correctly described by the first few components of a transformed vector, the remaining components can be approximately quantised.

TC is typically used to compress natural data, such as photos or audio, using specific knowledge to select which information to throw away. Once this lossy compression is applied to the information, the remaining information is compressed using a variety of other compression methods. Due to the ease of hardware computation and good energy concentration for a broad range of natural images, the Discrete Cosine Transform (DCT) has become the transform of choice for numerous image coding techniques. The DCT is spatially variant, but is sensitive to phase.

4.4.3 Wavelet-Based Coding

Wavelets are defined as small waves. The aim of the wavelet-based coding is to split the input signal, for compression purposes, into components that are easier to manage, have distinctive interpretations, or have certain components that can be discarded by adjusting the threshold. This works on the principle of being capable of at least approximately rebuilding the initial signal given these components (Li & Drew 2004). Wavelet-based coding explores tree-based structures in wavelet coefficients, which produce higher levels of compression compared to DCT-based coding, while maintaining better quality.

The theory behind wavelet-based coding was developed many years before it was ever useful. It became popular when Ingrid Daubechies used special filters in the filterbank for subband coding that was generated from the wavelet by using the perfect reconstruction relation of the filter bank (Blelloch, 2001). Higher compression rates are observed due to the possibility of elimination of the information contents of the filter components. The compression ratio of any wavelet-based coding scheme is not only dependent on the efficiency of the coding scheme, but also dependent on the choice of appropriate wavelet filters.

The JPEG compression standard makes use of the Embedded Block Coding, which uses an optimal truncation (EBCOT) compression scheme to upgrade to the JPEG2000 compression standard. This truncation compression scheme is wavelet-based. Figure 4.6 shows four popular wavelets used for compression. Simply defined, wavelet compression works by distinguishing a signal in terms of a specific underlying generator (Blelloch 2001). Wavelet transformation is often utilised in domains other than for compression. An example of such a domain includes its use to filter noisy data or detect similarity across varying time scales.

The study of wavelet coding has also been used for medical imaging, computer vision, and investigation of celestial X-ray sources (Blelloch 2001).

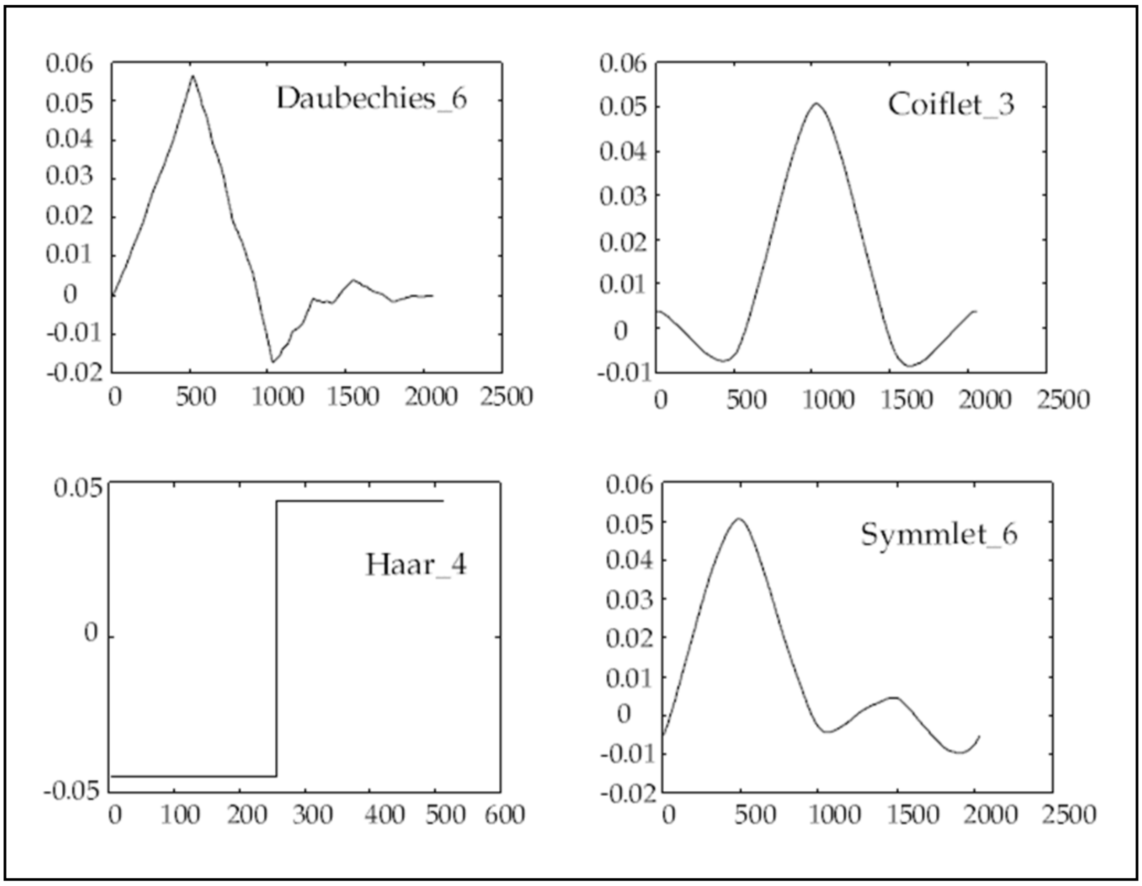


Figure 4.6: Sample of Popular Wavelets (Blelloch 2001)

Section 4.3 discussed lossless compression algorithms and Section 4.4 discussed lossy compression algorithms. Lossy compression algorithms provide superior compression level than that of most lossless compression algorithms. With the increased quality retention, lossy compression algorithms have seen widespread acceptance. Certain categories of files, however, require specific compression algorithms due to the structure, content, or importance of the information contained within these files.

A discussion on specific types of compression will be given next.

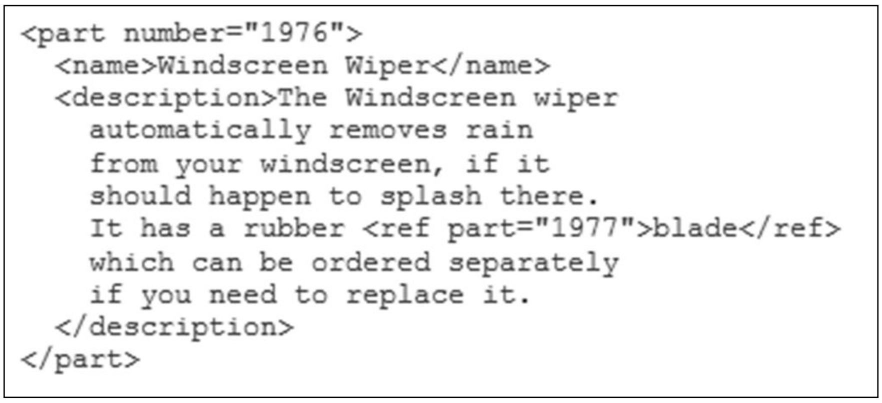
4.5 Textual Data Compression

This research focuses on improving efficiency within Mobile SOA. Due to the overwhelming use of XML-based messages and files within the Web Service environment, textual data in this research is limited to information stored and transferred within XML files.

Extensible Mark-up Language (XML) is a simple, text-based configuration for representing structured information (W3C 2010). XML describes a class of data objects known as XML documents, or XML files, and partly describes the behaviour of computer applications that process them (W3C 2004c). XML has gained unsurpassed acceptance since being developed by the XML Working Group in 1998 with the backing of the World Wide Web Consortium (W3C). XML is recognised as a standard data representation for interoperability on the Web. These XML files are composed of storage units known as entities, which contain either parsed or unparsed data (W3C 2006).

The information stored within an XML file includes documents, data, configuration, books, transactions, invoices, and any other textual information. XML was derived from a previous standard format called SGML in order to be more suitable for Web utilisation (W3C 2010).

The design goals for XML, together with associated standards, provide all the critical information required to understand XML Version 1.1 and create computer applications to process it. These design goals include principles, such as XML should be easily usable over the Internet, must support a wide variety of applications, the XML language must be formal and to the point, and the XML documents must be easy to create (W3C 2006). Figure 4.7 illustrates the typical structure of an XML document, designed to meet the design goals for XML.



```
<part number="1976">
  <name>Windscreen Wiper</name>
  <description>The Windscreen wiper
    automatically removes rain
    from your windscreen, if it
    should happen to splash there.
    It has a rubber <ref part="1977">blade</ref>
    which can be ordered separately
    if you need to replace it.
  </description>
</part>
```

Figure 4.7: Example of XML Structure (W3C 2010)

XML plays a key role in the delivery of standardised messages between interoperable systems, more specifically in SOA and Web Services. However, due to the manner in which XML files are created, they often contain verbose information. These bloated XML files are one of the main reasons why Mobile SOA has not been as successful as it should be, as they require more resources to manage, something not easily available on a mobile device

(Augeri, Mullins, Baird, Bulutoglu & Baldwin 2007). It is, therefore, essential that compression be employed to reduce the amount of resources required to successfully manage XML files on a mobile device.

Due to the inherent issues regarding the verbosity of XML files, numerous XML-specific compression techniques have been developed. In this section, three XML-specific compression techniques are studied, along with three multi-purpose compression techniques.

There are two main compressors used to compress XML files: arithmetic and dictionary compressors. Arithmetic compressors are useful to determine entropy and as control algorithms, but often have larger memory usage along with longer compression and decompression time requirements. Dictionary compressors store dictionaries for each message sent in order to produce better compression, and enjoy widespread use; most having open source formats (Augeri et al. 2007).

The following subsections discuss six different XML compression techniques, which will be tested and compared in Section 6.2.

4.5.1 bZip2

bZip2 is a patent free, lossless data compressor, which is available free of charge (Seward 2007). bZip2 employs RLE encoding to compresses information in blocks, sized from 100 to 900 Kilobytes, which makes use of the Burrows–Wheeler transform to convert a frequently-recurring character series into strings of matching letters. Once this transform is successfully completed, the algorithm then applies move-to-front transform and Huffman coding, as opposed to arithmetic coding used by the previous gZip compression algorithm. It offers compression of files to within 10-15% of best compression techniques available. bZip2 offers more efficient data compression of files than other leading compressors, but the compression is slower.

4.5.2 gZip

GNU Zip (gZip) is a compression technique that aims to be an alternative for Compress technique with benefits over Compress in terms of better compression, independent of patented algorithms (Gailly & Adler 2003). The gZip compression algorithm was designed by Gailly and the decompression algorithm by Adler, which utilises a combination of the

LZ77 algorithm and Huffman coding. gZip is the most widely used commercial compressor available for general use and on the Internet.

4.5.3 DotNetZip

The DotNetZip compression technique is available within the DotNetZip library. This library is a comprehensible, straightforward, and free class library and toolset for manipulating zip files or folders for all .NET applications, including mobile applications that use the .NET Compact Framework, written in any .NET language (Microsoft 2010a). The library can be easily imported to any .NET application, which exposes a number of compression procedures, such as create, read, extract and update. The file format used for the DotNetZip library is the zip file. This allows the compressed data to be stored in archives, able to contain more than a single compressed file.

4.5.4 XMill

XMill was developed by Hartmut Liefke and Dan Suciu in 1999 at AT&T Labs Research in New Jersey (Liefke 2004). XMill is a compression utility for compressing XML data efficiently and is based on a reordering strategy that leverages the effect of highly-efficient compression techniques in other compression utilities such as gZip (Liefke 2004). The reordering strategy is designed to take advantage of the XML elements. XMill groups XML text strings with respect to their meaning and exploits similarities between those text strings for compression. Therefore, XMill generally attains significantly improved compression rates than standard compressors such as gZip, without sacrificing much speed. The developers have since released the project and the source code, which is now a SourceForge project.

4.5.5 XMLppm

XMLppm is a data compression utility, which is a combination of the recognised Prediction by Partial Match (PPM) algorithm for text compression, and an approach to modelling tree-structured data termed Multiplexed Hierarchical Modelling (MHM) (Cheney 2009). XMLppm increases compression of XML files between 5 and 30 percent compared to current text or XML-specific compressors. An XML file is initally parsed by means of a SAX parser to produce a stream of SAX events. Each event is then encoded using a bytecode representation called Encoded SAX (ESAX). These ESAX bytecodes are encoded using one of many multiplexed PPM models based on its syntactic structure. The XMLPPM source code is also a SourceForge project and forms part of an XML compression project.

4.5.6 XWRT

A XML-WRT (XWRT) is a high-performance XML compressor that transforms XML to more compressible form, which utilises zlib, LZMA, PPMVC, or lpaq6 as back-end compressors (Skibinski 2009). XWRT produces a partially dynamic dictionary and substitutes regularly occurring words with shorter codes. The Xwrt compression algorithms are used in a number of environments, which includes data logging, Wi-Fi messages, along with several text editors.

Intelligent compression of XML files is important, as it will decrease the necessary bandwidth, storage, and processing on a mobile device (Natchetoi et al. 2008). These compression techniques play an important role when used in the XML-based SOA environment, more specifically that of mobile Web Services that use XML-based SOAP, WSDL and UDDI messages. These compression techniques can be incorporated into Mobile SOA. However, before choosing which technique to use within the proposed framework, a comparison test will be conducted. The results of this comparison test are discussed in Chapter 7.

4.6 Image Compression

Computer graphics is a standardised means of organising and storing graphics created using computers and digital representations of images, such as photos. Image files are made up of either pixel (raster graphics) or vector (vector graphics) data. The pixels that represent an image are ordered as a grid of columns and rows with each pixel consisting of numbers representing magnitudes of brightness and colour. A raster graphic is a data structure representing an array of pixels, or points of colour. Raster images are stored in image files with varying formats. Unlike raster graphics, where the data describes the parameters of each individual pixel, a vector graphic contains a geometric, or mathematical, description that can be rendered smoothly at any display size. Raster graphics are used for this research, as vector graphics are unable to be compressed.

The amount of file formats in multimedia continues to increase (Miano 1999). File formats for graphics and images are no different with a large number of different formats, most of which have developed their own compression technique. The most popular of these formats are BMP, GIF, JPEG and, recently, PNG. The BMP file format stores images without compression, while GIF, JPEG and PNG all use some form of compression and their common use is attributed to the fact that most Web Browsers can decompress and display them accordingly. Image compression reduces the need for sophisticated hardware in order to manage graphical images. This is of particular importance for digital photography where cameras, and other such devices capable of capturing photos, do not have the resources available to them as with desktop and mainframe computers. The need to efficiently access and store images in digital form has resulted in the development of many image compression standards for different applications and requirements (Li & Drew 2004). The developed standards have a superior longevity than specific programs or devices, and, thus, merit vigilant research and continuous improvement (Li & Drew 2004).

In addition to compression, images can be resampled. Image resampling is based on the changing in the number of pixels in an image. This is useful in order to correctly display an image of high resolution on a low resolution screen. This is not to be confused with image resizing, which reduces the size an image will print without changes to the number of pixels. Image resampling, in particular downsampling, produces a smaller sized image, both in resolution and storage size. Added to the high storage capacity raw images require, computer and mobile screens vary in both size and resolution. This further emphasizes the need for smarter image compression able to adapt to the daily changing market.

Research into image compression standards started as early as the late 1970s, and was particularly prolific in the 1990s when the Internet and Web browsers first became a household technology. Image compression techniques accomplish compression between 10 and 90 percent, depending on the quality required. Although there have been many compression standards and file formats developed over the years, this research focuses on three current standards, which are the most popular in use for Web browsers, both on desktop computers and mobile devices.

4.6.1 GIF

A Graphics Interchange Format (GIF) was developed by UNISYS Corporation and Compuserve, initially for transmitting graphical images over phone lines via modems (Li & Drew 2004). GIF files use a variant of the dictionary-based LZW compression format to compress a repeated series in screen images (Nelson et al. 1995). The GIF compression standard is used to compress non-photographic images, usually textual images or images with few distinctive colours.

When the Internet first became popular in the 1990s, 2400 baud modems were in use. These modems were limited to the transfer speeds they could achieve. Image compression became crucial. Despite transfer rates in excess of 14.4 Kbps, the extensive use of the World Wide Web on the Internet meant an increase in demand for high-speed transfer of graphics, and an increased dependence on the GIF compression standard for non-photographic images (Nelson et al. 1995).

The GIF file format is made up of the following components:

- The GIF signature;
- The screen descriptor to specify the width, height and pixel colours of the image;
- A local colour map (when the file contains more than one image definition) to map the 8 bit images to 24 bit screens;
- A global colour map if a local colour map is not used; and
- An image descriptor, which describes each image within the file, and raster area, which is compressed before being stored, that can be repeated any amount of times when several image are required in animations for example.

The original GIF standard was the GIF87a specification. The later, updated specification, GIF89a, supports simple animation, which offers control over functionalities such as time delay, a transparency index, and image count. However, the standard was limited to 8 bit (256) colour images only, which lacked acceptance as the necessity for higher quality images increased.

4.6.2 PNG

Portable Network Graphics (PNG) is platform-independent format that was created as a direct result of the patent issue of the LZW compression method used in the GIF standard (Li & Drew 2004). Industry viewed the patent of the GIF standard as unreasonable, as GIF had already gained wide acceptance for use on most Web browsers by the time the patent was requested (Nelson et al. 1995). Image developer groups joined together in an attempt to replace the GIF standard. These efforts lead to the establishment of the PNG standard. The

PNG standard is license- and patent-free, and contains numerous improvements over the existing GIF format. These improvements include (Li & Drew 2004):

- 48 bits of colour information, as opposed to 8 bits (256 colours) offered in the GIF standard;
- Gamma-correction information for correct display of colour images; and
- Alpha-channel information to enable control over levels of transparency.

However, the standard does not provide any animation support, but offers superior compression along with higher quality images.

4.6.3 JPEG

The Joint Photographic Experts Group (JPEG) format is the most important existing standard for image compression (Pennebaker & Mitchell 1993). The JPEG compression standards employs a lossy compression algorithm based on the limitations of the human vision system. The eye is unable to distinguish exceptionally fine levels of detail, even more so when viewing colour images. The compression algorithm, thus, decreases the quality of the image by reducing the colour information, throwing away a large amount of colour information when the division and truncation step is executed (Li & Drew 2004). This compression technique compresses images from 1 to 10 percent of its original size.

Figure 4.8 shows the process undertaken by the JPEG compression standard to compress an image file. The first step transforms the image's RGB values to YIQ (or YUV) values. A discrete cosine transform (DCT), a type of transform coding, is performed on the blocks of each plane. Quantisation is then applied to the image, which is where most of the loss occurs. Zigzag scans (type of run-length coding) convert the 8x8 matrix of the image into a 64-vector. The final step is to perform entropy coding, which is a lossless procedure that converts a series of input characters into a sequence of bits, such that the average amount of bits per character comes close to the entropy of the input characters (ITU, 1992).

JPEG-2000, developed as a replacement of the JPEG standard, provides superior compression over that of the JPEG standard. The JPEG-2000 standard uses the same compression steps of the JPEG standard, as shown in Figure 4.8, but uses a wavelet-based coding method instead of the DCT coding method. This standard, however, is not extensively supported by Web browsers, and, thus, has not yet replaced JPEG as an alternative image compression standard.

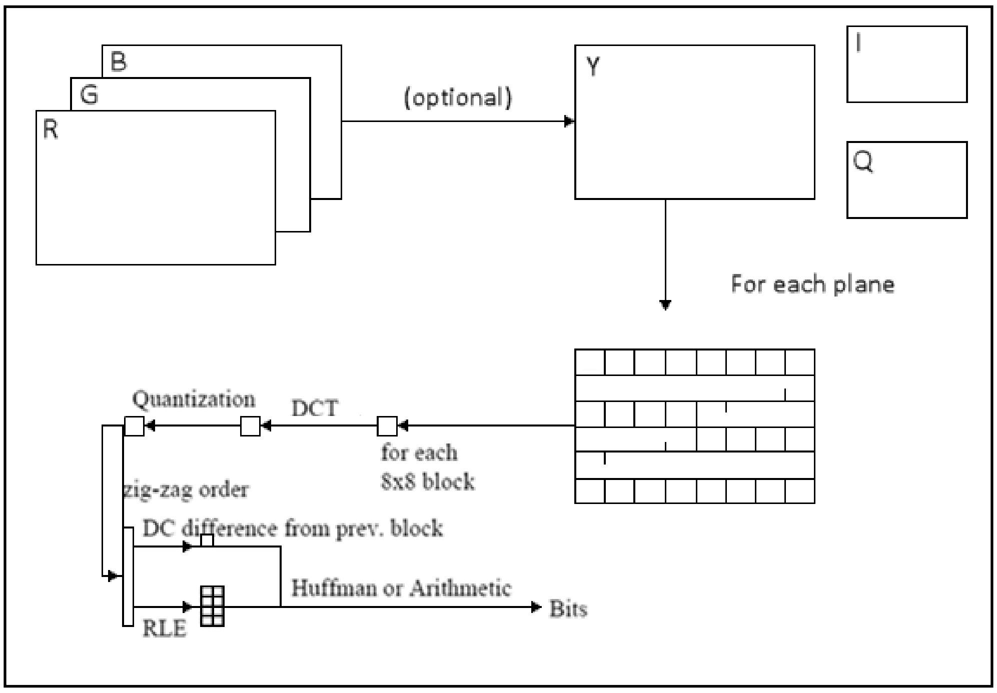


Figure 4.8: Steps in JPEG Compression (Blelloch 2001)

4.7 Audio Compression

An audio file is a file for storing digitised audio data on a computer. These files can be categorised as uncompressed and compressed audio formats. The audio file contains waveform data that is capable of being played on audio playback software. There is one major uncompressed audio format known as Pulse Code Modulation (PCM). PCM files are stored in a WAV file format for Windows operating systems and an AIF file format for Macintosh operating systems. This uncompressed audio format is a direct representation of the digital sample values of an audio recording, and is stored as 1s and 0s.

Audio compression refers to the encoding of digital audio. Sound is a wave consisting of amplitude values changing over time. This amplitude value is continuous. However, due to the high amount of storage required to save this continuous data, audio compression is required. Additionally, using stereo information as opposed to mono doubles the required storage space. As with image compression, generic data compression algorithms do not perform well with audio data. Audio compression takes into account the human perception of sound, which statistically removes bits of audio information humans are unlikely to hear. Each audio compression technique has three steps to perform: transformation, loss, and coding. The transformation step converts audio information, which is easier and more efficient to compress. In the next step, loss is introduced using a specific level of quantisation known as Pulse Code Modulation (PCM). Because sound is a continuous wave, quantisation is used to set the amount of reconstruction levels of the sound wave, which enables the digitisation of audio. Depending on the compression, further compression is used using either a lossless or lossy compression algorithm. The final step is to assign a codeword to each output level, which could either use a fixed-length or variable-length coding system (Li & Drew 2004).

Due to the increased necessity for multimedia, many different audio compression file formats exist. Three file formats are reviewed, two of which use lossless compression techniques, and one uses a lossy compression technique.

4.7.1 MAC

Monkey's Audio Compression (Ashland 2009) is an audio file format, which compresses information using a lossless compression technique, and uses the .APE file extension for file storage. MAC uses an open source algorithm, which allows better research into optimising the algorithm.

MAC is designed to be integrated into other software with minimum difficulty. The SDK used for MAC allows easy management of compression, decompression, verification, and converting of audio information. The SDK also includes more complex features, such as on-the-fly encoding and decoding, analysis, and tagging of APE files. The SDK provides patent-free algorithms for the above-mentioned features.

Apart from the original three-step compression to digitise audio, MAC makes use of its own three-step lossless compression algorithm. The first step requires that the left (L) and right (R) channel values be converted into X and Y values, where X is the midpoint between the L and R channel, and Y is the difference between the two. The next step requires a predictor, in which the X and Y values are passed through to remove any redundancy. The final step is to encode the audio information. MAC compression algorithm uses rice coding to achieve this. Rice coding is a way of using less bits to represent small numbers, while still maintaining the ability to tell one from the next.

Unfortunately, the decompression speeds of MAC are unfavourable compared to other lossless compression file formats, which is a telling factor as to why it lacks popularity. Another reason this file format is not popular is due to the developments of other file formats, which have standards that are widely used on mobile audio players.

4.7.2 FLAC

Free Lossless Audio Codec (FLAC) (Coalson 2008) is an audio file format similar to MP3, but uses a lossless compression algorithm. Unlike Monkey's Audio, the FLAC file format allows playback on a number of different devices, but is not as popular as the MP3 file format. The decompression of FLAC is the fastest of all file formats using a lossless audio compression algorithm.

FLAC is built on the need to avail as much support as possible to end users, which include access to the source code, multiple device support, hardware, software and operating systems support, is licence-free, and is able to stream over a network in minimum time, with minimum error, independent of the level of compression.

The FLAC compression algorithm uses linear prediction to convert audio information to a sequence of small, disassociated numbers known as the residual. These numbers are then stored more efficiently using Golomb-Rice coding. For blocks of the same recurring information, such as silent gaps, a run-length encoding process is utilised.

FLAC is a popular archive format for owners of CDs, and other media where audio quality is of high standard, where the preservation quality is essential. The lossless compression algorithm employed by the FLAC standard allows the owners of such media to replace lost, damaged, or worn out audio with exact duplicates of the original data at any point in time. This restoration is not possible in lossy compression standards, such as with MP3 and WMA. However, the FLAC compression algorithm compresses audio information to within 10 percent of most lossy compression algorithms.

In addition to these advantages, FLAC files are also suitable for transcoding without the usually associated transcoding quality loss. When "ripping" an audio CD into FLAC files, the information of that CD is still maintained. This includes the original track order, timing gaps, and CD-Text, but does not, however, store additional information, such as lyrics.

4.7.3 MP3

Moving Picture Experts Group Audio Layer-3 (MP3) is an audio file format, which, unlike FLAC and Monkey's Audio, uses a lossy compression algorithm. The audio compression algorithm used in the MP3 standard was developed by the Fraunhofer Institute in 1991 (Fraunhofer 2010). Although the MP3 format uses patented encoding, it is the most popular standard used to store compressed digital audio. Its popularity is achieved due to its compression ratio, which is generally around 10 percent of uncompressed digital audio stored as wave files. The MP3 file format has become the main standard used to transfer and playback digital audio on both the desktop computer, and other portable devices.

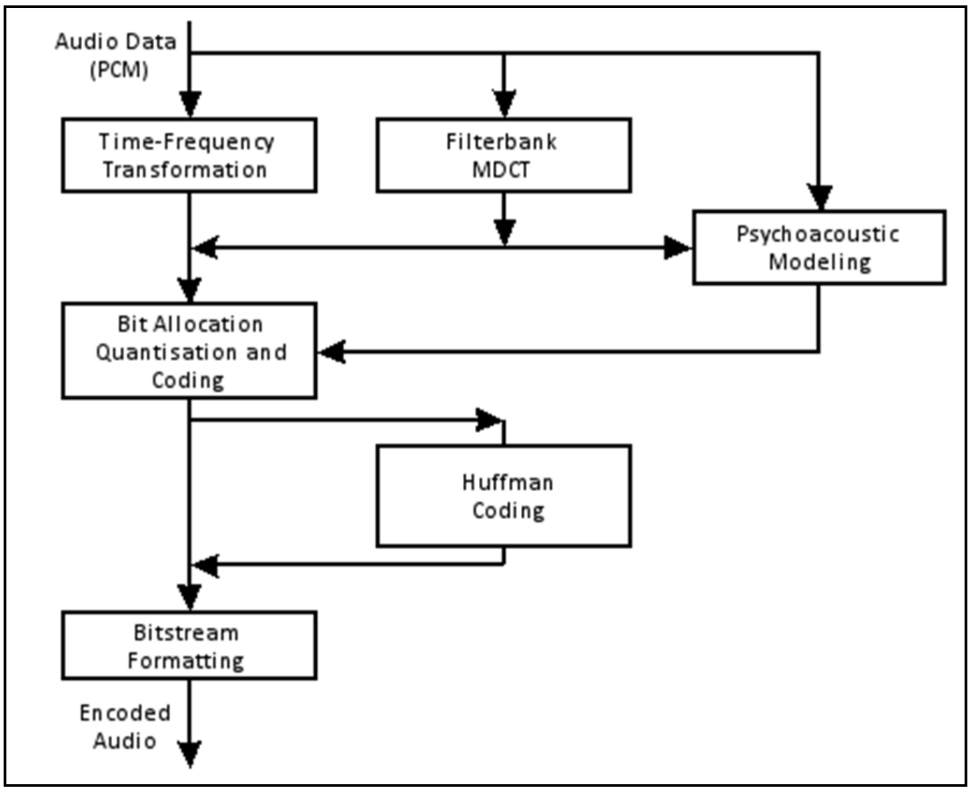


Figure 4.9: MPEG Audio Layer 3 Encoder

Figure 4.9 is based on the MPEG encoder as described by Li & Drew (2004) and Seeck (2005). The first step converts the input audio information from time into frequency values. This information is then divided into 32 frequency subbands using the filterbank. Using psychoacoustic modelling, the data is evaluated to take advantage of different facets of the human hearing perception (Seeck 2005). The psychoacoustic model includes masking effects (loud sounds cover quiet sounds), successive tone masking (when two different tones succeed one another), and joint stereo effect (higher frequencies are easier to locate). Simply stated, the psychoacoustic modelling removes all sounds that are not audible by humans, either because they are masked by other sounds or not within the frequency range audible to the human ear.

Bit allocation does not form part of the standard, and can, thus, be carried out in a number of different ways, where the aim is to ensure all the quantisation noise is below the masking thresholds (Li & Drew 2004). The information is then compressed using entropy coding, more specifically, Huffman coding. This coding removes redundancy, reducing the amount of information required for storage, and is a lossless compression technique.

4.8 Video Compression

Similarly to image compression, video compression was developed by the need to record video on devices, such as video cameras and mobile phones. These devices do not have ample storage space, advanced processing power or adequate memory to process raw video files after it has been captured.

A video is made up of a sequence of frames over time, along with optional audio. An apparent compression approach would be to use image and audio compression together to achieve acceptable compression rates and quality. This is, however, not the case, as video usually contains multiple frames per second, which would still result in a large sized storage space. The difference between two successive frames is vital to the compression of video. Typically, objects and camera movements do not change significantly over short time periods, which mean that the video has temporal redundancy (Li & Drew 2004).

Video compression standards provide significant reduction of the size and storage space required of video information. As discussed in Section 1.3.6, time is of great importance to the process of delivering content to mobile device by means of Web Services. Even the shortest of video lengths can take a significant amount of time to compress to an acceptable size. This size, however, is still deemed too large for transmission over a wireless network. This transmission may takes several minutes, if not hours, to transfer to a mobile device, which may also incur significant costs. Thus, video compression is not considered within the scope of this research.

4.9 Advantages and Disadvantages of Compression

To determine whether compression is indeed a solution to the issues regarding mobile device resources, a comparison between the advantages and disadvantages of compression is conducted. Based on research (Blelloch 2001; Li & Drew 2004; Natchetoi et al. 2007a; Nelson et al. 1995), the advantages of using data compression are as follows:

- Compressed files or information requires less disk space;
- Provides faster file transfers for both upload and download

- Uses less bandwidth;
- Saves costs on file transfers;
- Faster reading and writing from and to memory;
- Is independent of byte order;
- Reduces power consumption; and
- Enables faster printing speeds for compressed images.

Although some of the issues of compression can be overcome by means of using different methods and techniques, the disadvantages still exist. The disadvantages of using data compression are as follows:

- Adds some complication to file storage;
- Although uncommon, it can cause errors during transmission (if an error occurs during compression);
- Increases processing for complex compression and decompression algorithms;
- Requires decompression, which can often take some time to complete; and
- There is an unknown byte to pixel relationship (for images).

Compression plays an important role in content delivery, while reducing the storage space of information. Compression is more significant for devices with fewer resources available to it, such as with mobile devices.

4.10 Choosing the Best Algorithm

In order to measure which lossless compression algorithm performs best, several performance metrics are used. These metrics include the time to compress, the time to reconstruct, the size of the compressed file, and the generality (i.e. if it works for a certain file, will it work for another) (Blelloch 2001). Because the criteria for measuring the performance of lossless compression are based on objective values, the use of weighting is not often used. However, in cases where one criterion is more of a necessity than another, a weighted average is used. For example, if the time taken to compress information is of more importance than the other criteria, then the comparison between two or more lossless compression techniques will be weighted more toward the compression time.

In addition to these performance metrics, the amount of resources consumed during the compression and decompression processes plays an important role in the choice of compression algorithm to use. Compression techniques that operate more efficiently, in terms of resource usage, may be favoured over resource extensive techniques, regardless of the difference in compression levels they provide. This is of particular interest in the mobile environment, where limited resources are available. The metrics recorded to measure resource usage include the processing power and the memory footprint[1]. Additionally, battery usage may also be recorded when measuring resource usage on a mobile device.

Unlike lossless compression, measuring the performance of each lossy algorithm is not a matter of simply comparing metrics. An added complication when measuring a lossy compression algorithm is how good the quality of the compressed file is. There are, generally, tradeoffs between the amount of compression, the runtime, and the quality of the decompression (Blelloch 2001). A weighted performance metric is required in choosing the best algorithm, as some metrics may be more important than others depending on the specific requirement. For example, if storage space is of main importance, then the quality of the information and the time it takes to decompress this information is of less importance.

Due to the amount of data and media files available today, a compression technique performed on one type of file is, typically, not suited for all other types of files. For this reason, the manner in which the majority of lossy compression techniques are utilised are highly dependent on the media that is to be compressed. Lossy compression techniques for sound are vastly different to lossy compression techniques for images (Blelloch 2001).

The file compression techniques discussed in this chapter are evaluated and compared, in Section 6.2, using the performance metrics and approaches discussed in this section. An initial performance test is conducted in Section 6.2, and a more extensive evaluation is conducted in Chapter 7.

4.11 Conclusions

Compression is one of the most fundamental aspects of providing mobile devices with the necessary information people require. In a world consisting of unlimited, free bandwidth, data compression would not be a necessity. Bandwidth is limited, however, and usage thereof has an attached cost. Basic compression, therefore, not only reduces the need for higher bandwidth and reduces cost, but also places less pressure on the limited resources

[1] Memory footprint is the amount of processing memory used when executing the compression technique subtracted from the amount of processing memory allocated for that particular instance of a running application.

available to the mobile device, such as the processing power, memory, storage capacity, and battery life. Other aspects, such as the limiting screen size, poor navigation and browsing are eased with image compression where images are typically larger the screen can display correctly. Numerous amounts of data exists, in archives and elsewhere, and it has become crucial to compress this information (Li & Drew 2004). The need for compression has risen as a direct result from the need for multimedia presentations.

The goal of this chapter was to answer research question R3: "What are the issues related to compression?" This chapter briefly discussed the different types of compression available, and the principles each type employs to achieve high levels of compression. Using a lossless compression for data and a lossy compression for media, compression has optimised information management for use within a mobile environment. This optimisation has significant benefits and it reduces the amount of resources required on the mobile device, as well as reduce the cost of file transfer to and from the mobile device.

Several performance metrics were identified in order to compare the different compression techniques. Table 4.1 shows the list of each category of file, the compression techniques for these categories and the metrics used to compare these compression techniques.

Table 4.1: File Type, Compression Techniques and Metrics

Category	Compression Techniques	Performance Metrics
XML	*bZip2, gZip, DotNetZip, XMill, XMLppm, XWRT*	*Compression Ratio, Compression Time, Decompression Time, Memory Footprint*
Image	*GIF, JPEG, PNG*	*Compression Ratio, Image Quality*
Audio	*FLAC, MAC, MP3*	*Compression Ratio, Compression Time*

In addition to these performance metrics, the resource usage is also measured. Resource usage is measured by means of recording the processing power and memory footprint of each compression technique. Additionally, when measuring resource usage on a mobile device, the battery usage is also recorded.

Results from (Natchetoi et al. 2007b) suggest that compression of XML files significantly decreases the amount of overhead required for transfer to a mobile device, which does not entail utilising excessive resources from the mobile device to accomplish this. With XML being a key aspect of specific XML-based SOA components, such as WSDL, UDDI, and

SOAP, compression of these XML messages is important for delivering Web Services to a mobile device quicker and at a significantly reduced cost. Each type of file, data and media, has a preferred standard used for compression.

The management of compression techniques for each given file is important to the optimisation of file delivery within Mobile SOA. Hence, a framework designed to support intelligent compression of files used within Mobile SOA is beneficial. A proposed framework, supporting intelligent compression, is discussed in Chapter 5.

Chapter 5: Framework Design

5.1 Introduction

Mobile Web Services, a specialised domain within SOA, is a relatively new field, but has rapidly seen wide acceptance, from both businesses and the general population, due to the rise of SOA and mobile technology (Hurwitz et al. 2007; Natchetoi et al. 2008). There are still, however, a number of issues that exist when implementing a Mobile Web Service. These issues are based on the limited resources available to a mobile device, which include a reduced screen size, lack of processing power, insufficient processing memory, poor storage capacity, unreliable connections, limited bandwidth and high costs incurred. Chapter 4 highlighted compression as a solution to the most significant issues restricting the sustained use of a mobile Web Service.

Integration of mobile Web Services and the use of compression are required. A way to integrate these two aspects lies in the design of a framework to support intelligent compression for SOA within a mobile environment. The proposed framework should meet the requirements of SOA design principles, as well as provide minimum complexity to support sustained use on a mobile device where resources are limited. The design process is critical to ensure that the proposed framework is designed according to these SOA principles. The design process also provides a set of principles that the proposed framework should adhere to.

An architecture is specific to one particular domain and provides a logical overview of a technology. SOA, however, does not provide a total system view and, by itself, is insufficient to describe an entire system’s architecture (Sanders et al. 2008). A framework describes and guides the architecture approach. A framework is a library that provides tools and implementation guidelines for an application and can support multiple architectures. The possible behaviour and flow of control is also determined by the framework. When designing and developing applications within an SOA, it is important to do so according to design principles and guidelines (Kajko-Mattsson & Chapin 2010). This reduces the risk of failed or inefficient applications. This chapter describes the design of a framework, which supports intelligent compression.

The previous three chapters provided a literature review and have illustrated the wide acceptance and necessity of SOA, Mobile SOA and Compression. This chapter answers research question R4: *"How can a framework, supporting intelligent compression, be designed for Mobile SOA?"* Section 5.2 provides a literature study of an existing framework that supports compression, which this research extends. The extended framework is then discussed in Section 5.3, which discusses issues, such as the design, components, relevance, and benefits of the framework.

5.2 Extant System: EXEM Framework

Natchetoi, Wu, Babin and Dagtas (2007) highlight the need for efficient management of XML documents used for data exchange. This is done by means of compressing the XML documents using Efficient XML data Exchange Management (EXEM) framework. EXEM is a component-based framework that was designed to manage large data sets for mobile applications where XML is central to data exchange. The framework supports the efficient transfer of XML documents between a Client and a Server. This is of particular importance to mobile applications that use limited resources.

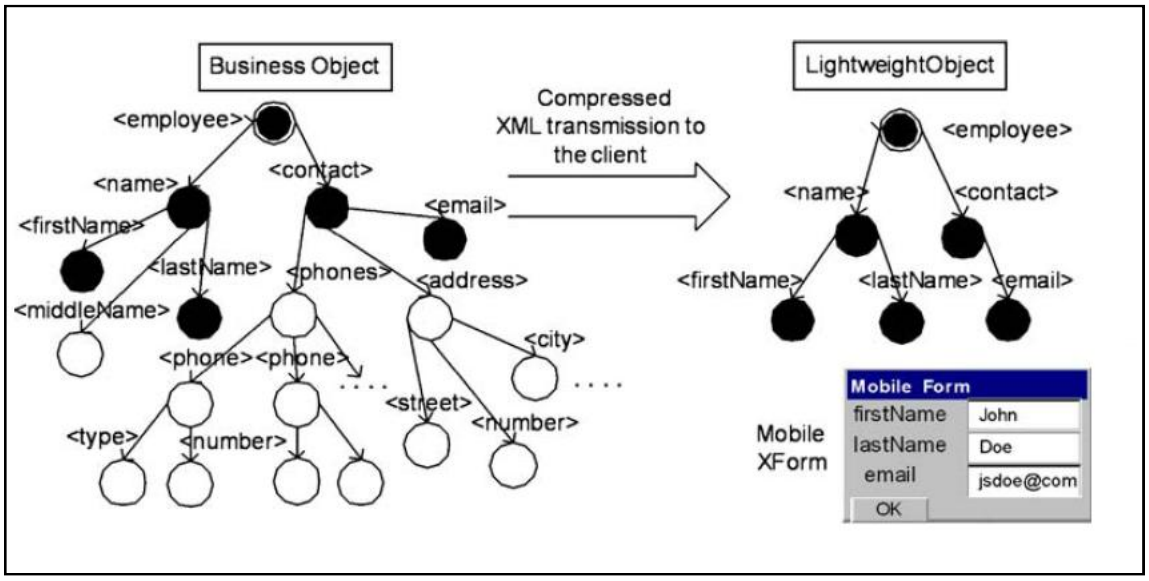


Figure 5.1: Lossy Compression (Pruning) of an XML Document (Natchetoi et al. 2007b)

The EXEM framework combines the use of lossy and lossless compression (see Sections 4.3 and 4.4). Lossy compression is used to prune data that is not required from the business object. This lossy compression is feasible only if there are no dependencies between the business object attributes and if there is knowledge of the information required by the Client. Figure 5.1 shows the process of pruning a business object within the EXEM framework. The pattern tree (left) is pruned by means of extracting only the information required (right). This means that only a subset of the XML document is transferred. However, pruning can only be performed within applications that are owners of the Business Object. An example of this would be the use of a Client/Server application within an organisation wishing to facilitate data exchange between its own applications. For instances where information is requested from an external source, such as a Service Directory for Services, it is improbable to predict the Business Object for all, or any, Services.

A lossless compression technique, implemented by Natchetoi et al. (2007b), was used after pruning was performed. Compression of the XML document is achieved by creating a Huffman code for all the elements, attributes and tags of the XML documents. A dictionary set is constructed to match each symbol with an appropriate code. The dictionary is managed over time, which collects statistics on item usage and is also updated on the Client side, and is dynamically adapted to the current applications available on the mobile device. To manage these dictionary updates a synchronisation method is used, which is invoked only at the start of each communication session. These dictionary sets are used to implement the context-dependent lossy compression and Huffman-based lossless compression algorithms.

Figure 5.2 illustrates the steps undertaken to provide efficient compression of the XML documents. The framework is divided into a Client and a Server (on which all computationally expensive tasks are performed). On the Server-side the business objects[1] are requested from the repository that is managed by the Back-end Server engine. These business objects are then sent to the EXEM Server where the data is converted and stored in an XML document before being pruned and compressed. The EXEM Server also provides updates for the dictionary based on statistics collected after each compression of a business object. The compressed XML message is then sent to the Client via a wireless communication network. This message is decompressed by the EXEM Client before it can be used by the Client application. Concurrently, the dictionary is updated on the Client side. The decompressed message is then used in the Business Query Engine, which extracts the required information for the Client Business Scenarios.

[1] Business Objects, in this context of use, relates to a set of variables, properties and associations, which represent the data that is stored.

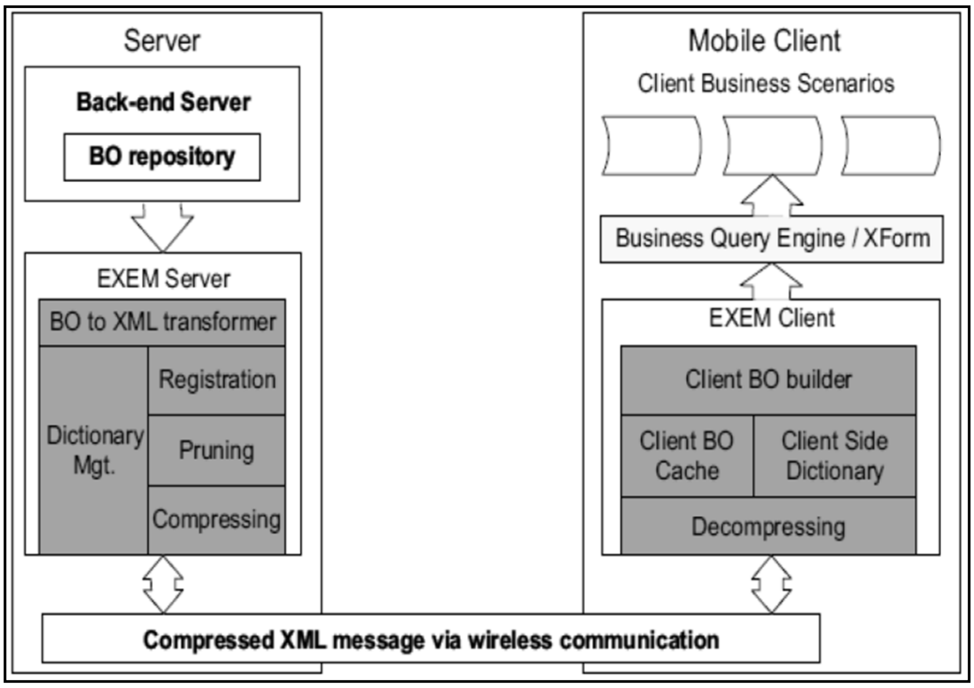


Figure 5.2: EXEM Framework (Natchetoi et al. 2007b)

Natchetoi et al. (2007b) conducted a comparative study of four compression techniques, including the use of an EXEM compression technique. The EXEM compression technique was utilised within the EXEM framework. The study used a sample of 65 randomly selected XML files categorised by file type. The EXEM compression technique significantly reduced the XML document size, and also observed better compression ratios than the other compression techniques. This allows for faster transmission of XML documents at a lower cost to the Client. Additionally, a compressed file is easier and quicker to process on the client-side, more so when it is transferred over a network. The EXEM compression technique, however, requires periodic dictionary updates over time to produce significant compression ratios. This leads to additional overhead transferred and a possibility of a bloated dictionary over time.

Although the EXEM framework provides a template for efficient file transfer between a Client and Server, the domain in which it was tested was limited to data within an organisation. As previously discussed, the need for information access has increased beyond that of a single business. Confirmation of this need lies in the growth of SOA (and Mobile SOA). In order to provide practical significance for a framework, it is important that the design of a framework addresses current needs using modern technologies within a realistic environment. A framework is proposed in the next section, which extends the existing EXEM framework.

5.3 Proposed Framework

Mobile SOA, generally, extends the basic SOA with adjustments made to support the mobile environment. Jørstad et al. (2005) provided an illustration in Figure 5.3[1] of what a Mobile Service-Oriented Architecture might resemble. The key components shown correspond to that of a standard SOA, with the addition of the Mobility Controller component, which manages the use of Services for the mobile environment. The framework proposed in this research forms part of the Mobility Controller component, as it forms the basic concept for file transfer within Mobile SOA. The framework provides guidelines to the realisation of more efficient delivery of a Service to a mobile application.

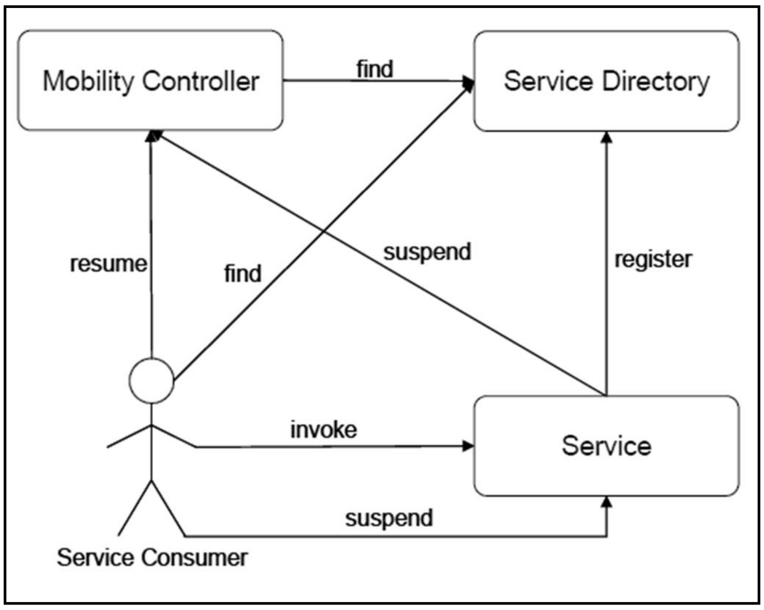


Figure 5.3: Simplified View of SOA with Support for Mobile Services (Jørstad et al. 2005)

While SOA provides the architecture for efficient interoperability between different systems, there are many additional difficulties that should be addressed (Phu & Yi 2005). The proposed framework addresses the issue of bloated XML documents and large media files used within the SOA environment, more specifically that of Mobile SOA. XML documents form the backbone for information exchange in an SOA between heterogeneous platforms. Thus, the compression of these XML documents is fundamental, as it reduces the required bandwidth for data transfer and also minimises the memory needed to process the data on a mobile device. Compression of images and audio are crucial for Content Services in which media files are the primary deliverable.

[1] Figure 3.2 is repeated as Figure 5.3 for ease of reference.

In order to successfully implement a prototype, a framework is required. The framework provides a set of guidelines on which the prototype is implemented. Without a thorough design process, however, the framework may not provide a beneficial set of guidelines. The next section discusses the design process of the proposed framework and the principles on which it was designed.

5.3.1 Design Process

For most organisations and vendors, the implementation of any software related to SOA can be daunting (Hurwitz et al. 2007). The design process gives clear understanding of how the framework should be structured and the prototype implemented. This not only aids similar design processes, but also facilitates the implementation of applications and Services within an SOA.

Consistent with Dey and Sohn (2003), a framework satisfies the two key objectives, which are that a framework:

- supports a faster iterative development process, in which applications are simpler to design, prototype and test; and
- provides designers and end-users with guidelines to build their own applications.

Additionally, the framework should minimise the complexities of building these applications. The proposed framework is designed in consideration of the SOA design principles discussed in Section 2.4. Although the framework itself is not required to adhere to these SOA principles, it should not cause the components of Mobile SOA to infringe on these principles. However, it is important that the proposed framework be designed to meet specific principles in order to successfully achieve efficiency within Mobile SOA. These principles are as follows:

- **Easy and Lightweight:** The framework should be lightweight and easy to execute in practice, be robust, and provide easy access to Services (Shretha 2010).
- **Compatibility:** The framework must work for different Services and be compatible for different devices, operating systems and programming languages. This is of particular importance to the mobile environment in which numerous mobile devices exist.

- **Robustness and Concurrency:** The framework must be robust, accurate, and error-free. Additionally, the framework should support multiple, concurrent access to requested Services. This feature should not, however, be detrimental to the performance of the framework.
- **Interchangeable Communication Medium:** Although not limited to the mobile environment, the framework must support different communication mediums for mobile devices, as listed in Section 3.3.3. The framework should perform equally in differing bandwidths and communication protocols from which the Service was requested.
- **Service Discovery:** The framework should support discovery of new Services via a specific Service Discovery method defined by the organisation or vendor that implements a working prototype of the framework.
- **Efficiency:** The framework must provide a means for efficiently managing the information produced by requested Services. The Service itself is external and autonomous. The implementation and operation of these Services is irrelevant to the performance the framework yields.
- **Flexibility:** The framework should fit into the existing SOA environment of an organisation. This removes the need to redesign or redevelop any existing applications used within the organisation.

An SOA does not exist in a vacuum and must be designed to support the ever changing technologies and protocols. These principles are essential to successful execution of the proposed framework itself and the services it provides within an SOA (more specifically Mobile SOA). The evaluation of whether the proposed framework adheres to the design principles mentioned in this section is discussed in Chapter 7.

The EXEM framework, supporting compression of XML files, provided a template on which the proposed framework extends. The EXEM framework, however, was implemented and evaluated within a closed environment of a single organisation. There is a continued growth of SOA (and Mobile SOA) and other applications where information is securely accessed from an external source. It is, thus, important that the proposed framework support efficient transfer of this information provided by a Service, whether it is composed within an organisation or invoked via an external source, such as a Service Directory.

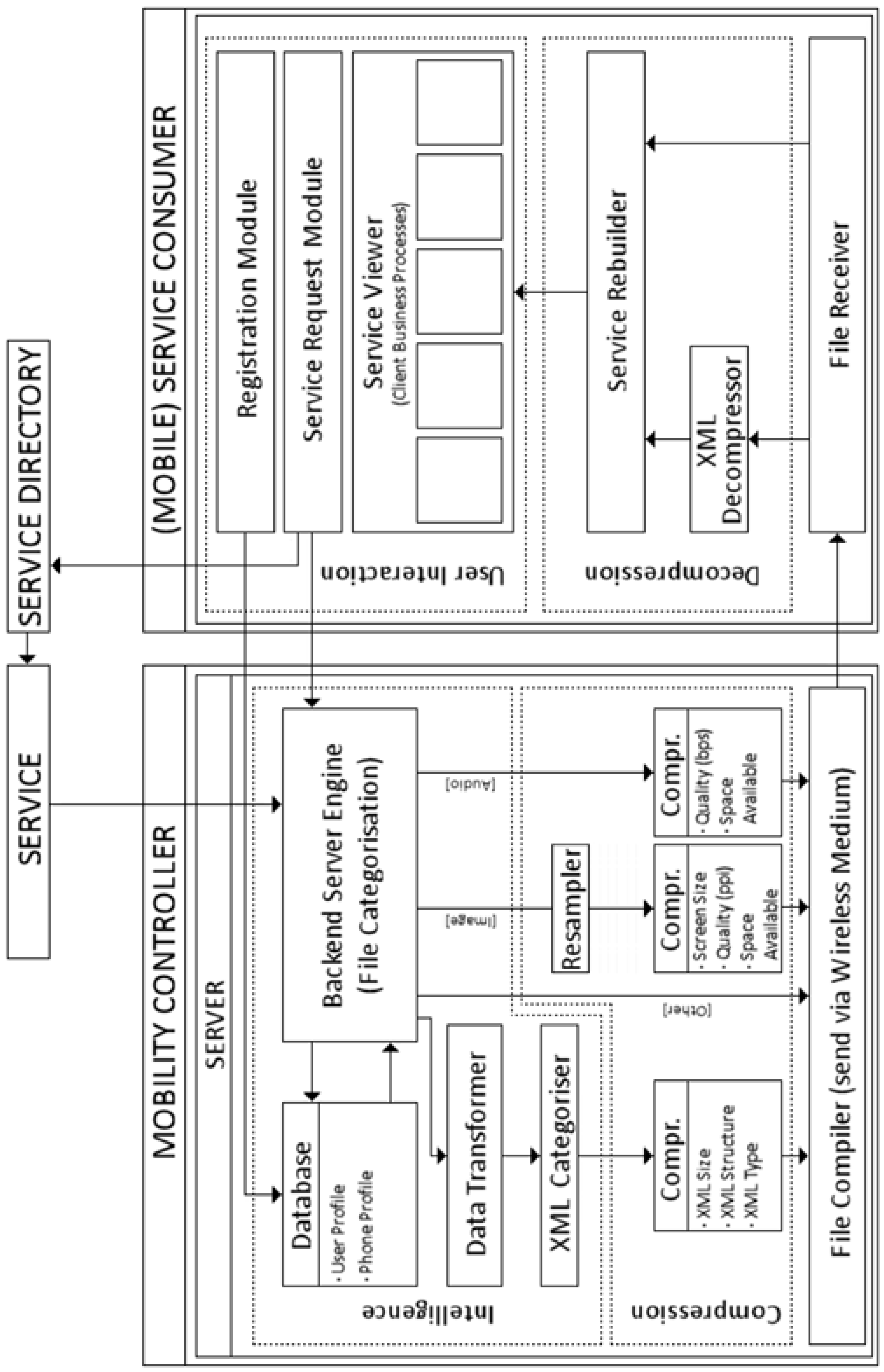


Figure 5.4: Proposed Compression Framework for Mobile SOA

The proposed framework supports intelligent compression to provide efficient file transfer of the files within a Mobile Web Service to a mobile device. Figure 5.4 illustrates how the different components of the proposed framework are combined. The proposed framework is divided into four components, namely: the User Interaction Component, the Intelligence Component the Compression Component, and the Decompression Component.

5.3.2 Components of Proposed Framework

Standard XML-based initiatives form the basis on which Web Service technology implements SOA (Sanchez-Nielsen et al. 2006). Similarly, these standards form the basis on which Mobile Web Services implements Mobile SOA. As discussed previously, the verbosity of the XML messages used within SOA, moreover Mobile SOA, has resulted in many XML-specific compression techniques. The proposed framework provides an approach to efficiently manage these XML messages by means of compression. Additionally, the framework uses compression for efficiently managing media content, in the form of image and audio files. The efficiency provided by the proposed framework lies within the functionality of its four main components, and the interaction between these components.

User Interaction Component

The User Interaction Component provides the basic functionality for registration, Service request and Service display. However, only the Registration Module provides distinctive functionality within the proposed framework. The Registration Module provides methods for collecting the information required to build the User Profile and Phone Profiles, which forms part of the Intelligence Component. The Service Request Module and Service Viewer provide generic functionality to request and invoke a Service. The registration, Service request and Service invocation processes are discussed in more detail in Section 5.3.3.

Intelligence Component

The intelligence of the proposed framework is managed within the Intelligence Component, and is derived from the information provided by the Database and Backend Server Engine. Information within the database is divided into two categories, the User Profile and the Phone Profile. The User Profile stores information about the Client, such as the unique IMEI number for the mobile device. Each User Profile is mapped onto a specific Device Profile.

The Device Profile stores information about the mobile device, such as the screen dimensions and storage capacity.

The database provides information to the Backend Server Engine. Information from the initial Service request is also fed into the Backend Server Engine. This information is utilised to categorise the files from the requested Service. These files are grouped according to their respective types. All data, image and audio files used by the requested Server are compressed individually within the Compression Component. Additional files, which do not fall within the three categories of files compressed by the proposed framework, are grouped by the Server, and are sent to the mobile device without compression. In the event that a compression technique exists to compress a specific file, within a specified time frame and compression ratio, the compression technique may be added to the framework.

Additionally, the Backend Server Engine assists in the compression process by providing the compression techniques with specific parameters based on the information collected. These parameters are requested from the database, which stores information collected during the registration process. However, before any data file, received from the requested Service, is compressed, it is transformed into an XML document, if not already in this form. It is essential that all data be transformed to a uniform standard. Since XML forms the backbone of data exchange in Web Services, it is, thus, selected as the standard for data exchange within the proposed framework.

Different XML compression techniques may perform better for different categories of XML files. XML files can be categorised on criteria, such as file type (APP, SOAP, WSDL, and RSS), number of repeated elements and attribute values, and the complexity of the XML structure (Natchetoi et al. 2007b). The XML categoriser categorises the XML depending on the XML compression technique preference. That is to say, if an XML compression technique outperforms all other techniques for a specific category, then that compression technique is used for that category.

Compression Component

The proposed framework promotes efficient file transfers of Mobile SOA to mobile devices using intelligent compression. As previously described, intelligence is derived from the ability to select the best compression technique for each of the categorised files. However, the compression of the files is still required. The aim of compression is to reduce the amount of resources and time required to manage and transfer the files used within Mobile SOA. Due to the limited resources available on mobile devices, there is a definite need for compression to reduce the effects of resource-intensive Web Services. The Compression Component controls all aspects of file compression within the proposed framework. The compression of these files is categorised into that of the data and media files.

The information stored within data files is often highly critical to the success of business processes. For this reason only lossless compression techniques are used to compress data files within the framework. Section 4.5 discusses the different XML compression techniques that were considered for use within the implemented prototype. A comparison of these compression techniques is discussed in the next chapter. In the case of media files, different standardised compression formats were considered for image and audio compression. Image and audio compression standards were discussed in Sections 4.6 and 4.7, respectively. The comparison of these standards is also discussed in the next chapter.

Decompression Component

The Decompression Component simply reverses the process of lossless compression. For data files, a decompression algorithm is applied based on the compression technique used. The contents of the data file are accessible only once a decompression algorithm is applied to the compressed data. For media files, however, the process is much simpler. Due to the widely accepted standards for many image and audio compression formats, most operating systems, programming environments, and applications accommodate the decompression algorithms of these standards. For example, when using the JPEG format for an image, it can be viewed on most mobile device operating systems, and it is not required that the decompression algorithm be used within the framework to decompress the image. Any lossy compression that had been applied to any of the files cannot be recovered.

The effectiveness of the proposed framework is dependent on the combined utilisation of these four components. The (Mobile) Service Consumer sends a request to the Service Directory, which in turn produces Services matching the request. The Service Consumer then selects the required Service, in which the files linked to the Service are managed by the Mobility Controller and transferred to the Service Consumer. The procedure for efficiently delivering a requested Service to a mobile device is discussed in the following section.

5.3.3 Process Steps of Framework

The proposed framework provides an invocation mechanism to support Web Service requests to the Service Registry. Business data and media content are exchanged between the Service Provider and Service Client, which communicate via SOAP messages. The proposed SOA-based mobile framework consists of four components, which provide efficient file transfer between the two parties. The proposed framework provides a guideline to the processes that should be performed in order to facilitate efficient file transfer to a mobile application, Figure 5.4. The process steps provide an important overview of the task flow of the proposed framework, and illustrate how intelligent compression of the requested Service files is achieved. Detailed explanations of these processes are now discussed.

Initial Registration

Before any Service requests, a Service Consumer (Mobile Client) is required to register via the mobile application. Registration is handled by the Registration Module. The User is only required to input specific information, such as their name, address, occupation and employer. The Registration Module automatically collects all the information about the mobile device by invoking background data collection methods. The User is prompted for any information that is not successfully obtained by the Registration Module. This registration process captures information about the Client and stores it within a database located on the Server. The information captured is categorised as either User Profile data or Device Profile data. This information is fed into the Backend Server Engine. More detail about the database structure and the stored information is provided in Chapter 6.

Service Request

Once the registration process is complete, the Client may request Services to perform a specific task or provide specific content. This request is sent via the Service Request Module to the Service Directory, and the Service is located by means of a search method. The Service Directory returns a list of Services available based on the request. Once the Client selects the desired Service, the information is sent to the Service Directory and the Backend Server Engine. Before the Service may be used, however, the data from the initiated Service is passed through the Mobility Controller (refer to Figure 5.4) to provide efficient file transfer to the Client.

File Categorisation

The intelligence of the framework lies in its ability to categorise files accordingly and select which compression technique to use. Initially, files are categorised into data and media files, as each require compression techniques specific to those category of files. The categorisation of these files is performed in the Intelligence Component. The XML Transformer converts all data, which is then stored within an XML file. These XML files are subsequently categorised using the XML Categoriser by criteria discussed in Section 5.3.2. Media files are further categorised into two categories, images and audio, within the Backend Server Engine. These media files are the main deliverable for Content Services.

Compression

Files are compressed using different compression techniques for different categories of files. The compression of these files is managed by the Compression Component. Parameters from the Intelligence Component are passed through to the Compression Component. These parameters are used, for example, to decide which compression techniques to use or to determine the dimension of the compressed image. The parameters fed into the compression techniques are the size, structure and file type of XML files, screen size, quality (pixels per inch) and space available for image files, and quality (bits per second) and space available for audio files. Before compression is performed on an image, it is required that the resolution be reduced to match the mobile device screen. This is achieved by resampling an image, based on the dimensions provided by the Phone Profile for that Client. The comparison and selection of the compression techniques are discussed in the Chapter 6.

File Transfer

The compressed files are transferred to the mobile device via a wireless connection, discussed in Section 3.3.3, which is established by the mobile application. One of the primary factors for proposing a more efficient framework within Mobile SOA is attributed to the high cost attached to file transfer when using these wireless connections. All files are assembled into one package and transferred to the mobile device. The mobile application then separates the package into files that do and files that do not require decompression. The compression technique, which also performs the decompression, is sent to the mobile device, along with the other compressed files, in order to perform decompression.

Decompression

Before the data files can be utilised by the mobile application, they must be decompressed. The files are decompressed using the reverse of the technique used for compression. This simply means that data files are restored to their original state using a decompression algorithm, which applies the same steps used to compress the file, but in the reverse order. Depending on whether a widely accepted compression standard was used, media files often do not require a decompression step, as it is supported by most mobile devices and their operating systems. XML files, however, need to be decompressed using the same technique that was used during compression.

Build Requested Service

When all files are in their required formats, the mobile application is able to build the requested Service, or utilise the downloaded content on the mobile device. Once the Service has been rebuilt, it is invoked within the Service Viewer. Data is the most important deliverable for Added Value Services. Because a lossless compression technique is used to compress the XML files, the data within these XML files are exact copies of the original. Media files used within Content Services have a set amount of loss, based on the parameters provided by the Intelligence Component. However, due to the limited screen size and low quality speakers, the loss within these media files is negligible.

Summary of Process Steps

These processes have explained how efficiency may be achieved for the transfer of Service files to a mobile device. An important aspect of this framework is to provide cost-effective and affordable access to content, in the form of Web Services, through an ever-growing technology most people already have access to, their mobile devices. Regardless of the core technology used, the only manner in which to make a framework viable for mobile applications is by providing the Client with as much applicable data as possible (Natchetoi et al. 2008). This chapter has provided a discussion on an approach to providing this data, which relies on the execution of a framework that supports intelligent compression. The proposed framework forms part of the Mobility Controller component, one of the four main components within Mobile SOA. Therefore, the implementation of a prototype based on the framework design does not require any adjustments to previously existing infrastructure supporting SOA. Due to this design principle, the proposed framework has a number of perceived benefits.

5.3.4 Perceived Benefits of the Proposed Framework

There are a number of issues and challenges that exist in the mobile environment and Mobile SOA. This has led to the proposal and development of numerous solutions for individual issues. The framework proposed within this research aims to provide a smart solution to these issues. The proposed framework has perceived benefits intended for Mobile SOA, although the principles of the framework may be adapted to support other mobile and desktop Client/Server applications. These benefits are listed as the following:

1. There is a projected reduction in the overall development cost in an organisation to develop mobile applications supporting SOA.
2. The mobile applications themselves may perform faster and better when developed according to a specific guideline, such as that described by the proposed framework.
3. The viewing of an image is enhanced due to image resampling to match the screen size for each individual mobile device. The content layout and navigation is also improved as a result.
4. Compression reduces the amount of processing power and memory required to manage larger files, in terms of transferring, accessing and storing of files.
5. The reduction in processing power and memory usage has a direct impact on the battery life of a mobile device. The less processing and memory required, the longer the battery may last between charges.
6. The framework may also facilitate improved file transfer where there may be unreliable connections and limited communication bandwidth.
7. The reduction in the amount of overhead required for transfer over these connections may result in the reduction of transmission costs. This is of significant importance for businesses, where there may be an extensive amount of information that is transmitted to and from a mobile device.

Benefits 1 to 3 list the advantages of developing an application based on the guidelines outlined in Section 5.3.1. These benefits are of great significance to businesses having to spend large amounts of money on the analysis and development of a mobile application. Benefits to the end-user, which may be a business, business employee or regular person, are listed from 4 to 7. These benefits highlight the significant reduction in cost and time to receive files from a Server even on poorly performing connections.

Although there are a number of benefits perceived for the proposed framework, there may always be limiting factors when developing for the mobile environment. Factors, such as human acceptance, visual constraints and input limitations, add to the challenges faced when implementing mobile applications. The evaluation performed in Chapter 7 provides a discussion on whether these perceived benefits are realised.

5.4 Conclusions

Research question R1, R2 and R3 are answered in Chapter 2, 3 and 4, respectively. The goal of this chapter was to provide an answer to research question R4: "How can a framework, supporting intelligent compression, be designed for Mobile SOA?" In answering this question, this chapter first provided a discussion on the framework proposed by Natchetoi et al. (2007b). The EXEM framework reviewed in this chapter presents a template for compression of data exchanged within a Client/Server mobile application. Natchetoi et al. (2007b) demonstrated that a framework, which supports compression, provides efficient transfer of files to a mobile device. The framework, however, does not provide a solution to the other challenges faced within a Mobile SOA environment. This template, therefore, provides a basis for the design of framework proposed in this research, and also provides motivation for the use of XML data exchange.

A discussion on the proposed framework was divided into four subsections. The design process, discussed in Section 5.3.1, illustrated the principles on which the proposed framework was designed, and upon which the prototype should be implemented. In Section 5.3.2, a discussion on the components of the framework provided a view on how these components interact. The three components, Intelligence Component, Compression Component, and Decompression Component, provide the backbone for efficient file transfer to a mobile device. Section 5.3.3 provided a discussion on the process steps of the proposed framework, which showed how efficiency may be achieved by means of intelligent compression. A list of perceived benefits was discussed in Section 5.3.4, which provided evidence of the predicted effectiveness and efficiency of the proposed framework. Furthermore, these benefits provided sufficient reasoning to the importance of the proposed framework within the Mobile SOA environment.

Table 5.1: Perceived Benefits Mapped onto Research Questions

Perceived Benefit	Research Question
There is a projected reduction in the overall development cost in an organisation to develop mobile applications supporting SOA.	R 5
The mobile applications themselves may perform faster and better when developed according to a specific guideline, such as the proposed framework.	R 6
The viewing of an image is enhanced due to image resampling to match the screen size for each individual mobile device. The content layout and navigation is also improved as a result.	R 7
Compression reduces the amount of processing power and memory required to manage larger files, in terms of transferring, accessing and storing of files.	R 7
The reduction in processing power and memory usage has a direct impact on the battery life of a mobile device. The less processing and memory required, the longer the battery may last between charges.	R 7
The framework may also facilitate improved file transfer where there may be unreliable connections and limited communication bandwidth.	R 7
The reduction in the amount overhead required for transfer over these connections may result in the reduction of transmission costs. This is of significant importance in for businesses, where there may be an extensive amount of information that is transmitted to and from a mobile device.	R 7

Each of the perceived benefits (Section 5.3.4) is mapped onto a specific research question, as illustrated in Table 5.1. The proposed framework provides guidelines to the implementation of a prototype. Chapter 6 provides a discussion on the implementation tools and methods used when developing such a prototype to test efficiency and effectiveness of the proposed framework. An evaluation of a prototype is conducted in Chapter 7 to determine whether the proposed framework is implemented according to the design principles of Section 5.3.1 and whether the perceived benefits of Section 5.3.4 are realised. The evaluation will provide evidence of whether the perceived benefits of the proposed framework realised.

The implementation of the proof-of-concept prototype is discussed next.

Chapter 6: Implementation

6.1 Introduction

In the literature review chapters (Chapters 2, 3, and 4), Service-Oriented Architectures (SOA), Mobile SOA, and File Compression were discussed. Chapter 2: highlighted the growth of SOA, and Chapter 3 discussed the recent development of Mobile SOA, as a specialised domain within SOA, along with the constraints encountered when implementing for this environment. A discussion was conducted in Chapter 4 on a possible solution to these issues using compression. Chapter 5 proposed a framework for Mobile SOA, which supports intelligent compression, and also discussed the design process, process steps and perceived benefits of the proposed framework.

The development methodology chosen for this research is based on the proof-of-concept methodology, for which a prototype was implemented. The framework proposed in Chapter 5 provided a set of implementation guidelines to support efficient file transfer of Web Services within Mobile SOA. A prototype was, therefore, developed according to these guidelines. A prototype, in terms of implementation of software, is an incomplete, but practical, form of an application used for testing (Preece, Rogers & Sharp 2007). In terms of this research, a prototype was used to test the feasibility and performance of the proposed framework.

The purpose of this chapter is to provide an insight on how a prototype of the proposed framework was implemented, in order to answer research question R5: *"How can a prototype, based on the proposed framework, be implemented?"* This chapter begins with a discussion on the initial testing of the XML, Image, and Audio compression techniques (Section 6.2), which compares results for each of the compression techniques, respectively. Section 6.3 provides a discussion on the implementation criteria of the prototype. The implementation tools that were used for the development of the prototype are discussed in Section 6.4. This is followed by a discussion on how the four components of the proposed framework were implemented for the prototype in Section 6.5.

6.2 Initial Testing

In order to discover the feasibility and usefulness of using compression within the proposed framework, it was essential to conduct several compression tests for the categories of files used within Mobile SOA, as discussed in Chapter 4. These categories of files are data (limited to textual data stored in XML files), images and audio. In addition to testing the practicality of compression within the proposed framework, the initial testing was used to evaluate the factors of compression and the effects thereof. These factors revealed effects within the proposed framework, when the prototype is implemented.

Comprehensive video compression tests were not conducted, as they do not fall within the scope of this research, due to its lengthy compression times and possible large file sizes.

6.2.1 Compression Testing Procedure

All tests were conducted on an Intel Core 2 desktop computer, which included the following specifications: 2.13 GHz Dual Core Processor, 320GB hard-drive and 2GB RAM. The list of files used in these tests, as well as additional details about each of the files, is available in Appendix A, Appendix B and Appendix C for XML, image and audio files, respectively. The following subsections provide detail on how the compression tests were conducted.

Although lossy compression techniques provide superior compression levels, the quality of the lossless compression techniques are better. Thus, both lossy and lossless compression techniques were tested for the image and audio compression tests. Each of the compression techniques tested for XML compression were lossless, as the data contained within the XML files are critical and cannot be discarded.

6.2.1.1 XML Compression Tests

Six XML compression techniques were tested; namely bZip2, gZip, DotNetZip, XMill, XMLppm and XWRT, which were discussed in Section 4.5. Each of the compression techniques could be executed via a command line interface (CLI). However, in order to accurately measure the compression metrics, a wrapper application was developed. The wrapper can execute each of the compression techniques by means of a Windows batch file, as well as record data, which was used to compare the compression techniques.

The XML files used for testing were divided into three categories based on their sizes, Small (0-1MB), Medium (1-10MB) and Large (10MB+). The compression metrics recorded in the tests included the original size of the XML file, compressed XML size, the compression ratio[1], compression time, decompression time, and the memory footprint. These metrics were chosen based on research (Palmer 2002; Canfora & Troiano 2002; Mirchandani 2001; Spanos 2009; Natchetoi et al. 2007b).

6.2.1.2 Image Compression Tests

Only two metrics were recorded when testing the different compression techniques for images, which were the compression ratio and image quality (after compression) (Miano 1999; Li & Drew 2004). Three image compression techniques were considered, namely GIF, JPEG and PNG. A total of 30 uncompressed image files (BMP format) were tested. These files were of varying sizes ranging from 17KB to 20MB, and contained different content, such as photos and graphics.

6.2.1.3 Audio Compression Tests

In testing the best audio compression technique three compression techniques were compared. Two lossless compression techniques and one lossy compression technique were used. The lossless compression techniques included FLAC and MAC, and the lossy compression technique used was MP3, all of which were discussed in Section 4.7. A total of 30 uncompressed audio files (WAV format) were used in the testing process. These audio files ranged from 11KB to 170MB. Based on research (Li & Drew 2004), the metrics recorded during testing were the compression ratio and compression time.

The descriptive and inferential statistics recorded for each of the three compression categories are discussion next.

6.2.2 Descriptive Statistics

The descriptive statistics provide a quantitative analysis of the results recorded for each of the XML, image and audio compression techniques tested. This analysis is discussed in the following subsections.

6.2.2.1 Descriptive Statistics for XML Compression

The compression ratio achieved for each XML file was recorded for each of the six compression techniques. These results are shown in Figure 6.1. These results indicate a

[1] Compression ratio is the compressed XML size divided by the original XML size.

better compression ratio for Medium sized XML files. The compression techniques do not perform as well for XML files less than 500KB and larger than 80MB.

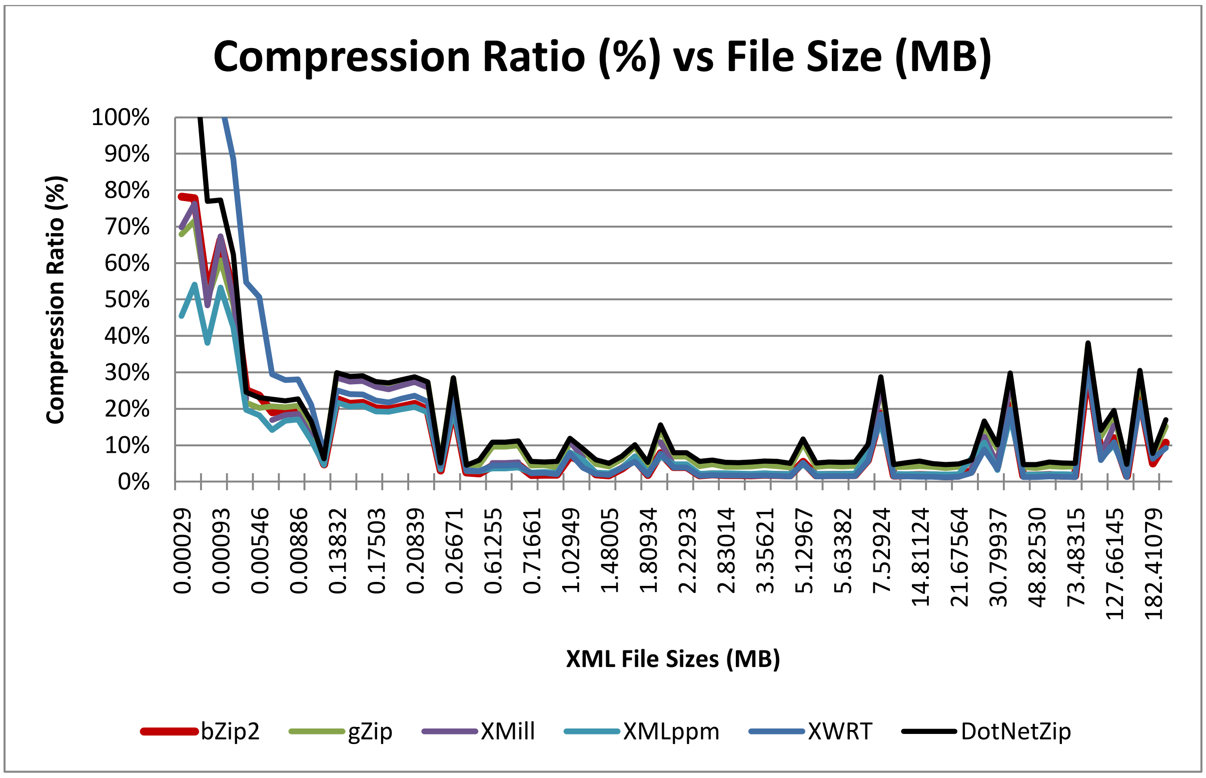


Figure 6.1: Compression Ratio (%) for each Compression Technique

Confirmation of these results are shown by the statistical data recorded in Table 6.1, with the mean and standard deviation of compression ratio for Medium size XML files significantly less than the Small and Large sized XML files.

Table 6.1: Compression Ratio Statistics

		bZip2	gZip	XMill	XMLppm	XWRT	DotNetZip
Mean	Small XML	22.2082%	24.4404%	29.9039%	18.1388%	35.9186%	30.3569%
	Medium XML	3.8213%	6.9610%	9.0736%	4.1353%	3.6290%	8.0463%
	Large XML	6.7879%	10.2473%	15.5199%	6.6445%	6.3049%	11.3469%
Median	Small XML	19.9614%	21.2547%	26.2313%	18.6153%	22.7936%	23.8396%
	Medium XML	1.9207%	4.5049%	6.8228%	2.3115%	1.8331%	5.6169%
	Large XML	1.8183%	4.2421%	12.3035%	2.0820%	1.4164%	5.5193%
Std.Dev.	Small XML	21.7310%	18.8712%	20.2526%	15.0685%	43.9663%	30.3958%
	Medium XML	3.6500%	5.0584%	6.5818%	3.2106%	3.6699%	5.1224%
	Large XML	8.4086%	9.8733%	10.5870%	8.1410%	8.6326%	9.9874%

Figure 6.2 shows the time taken to compress each of the XML files using the different compression techniques[1]. From this illustration, it is shown that for XML files less than 7MB the mean compression time (for all compression techniques recorded) is less than five seconds.

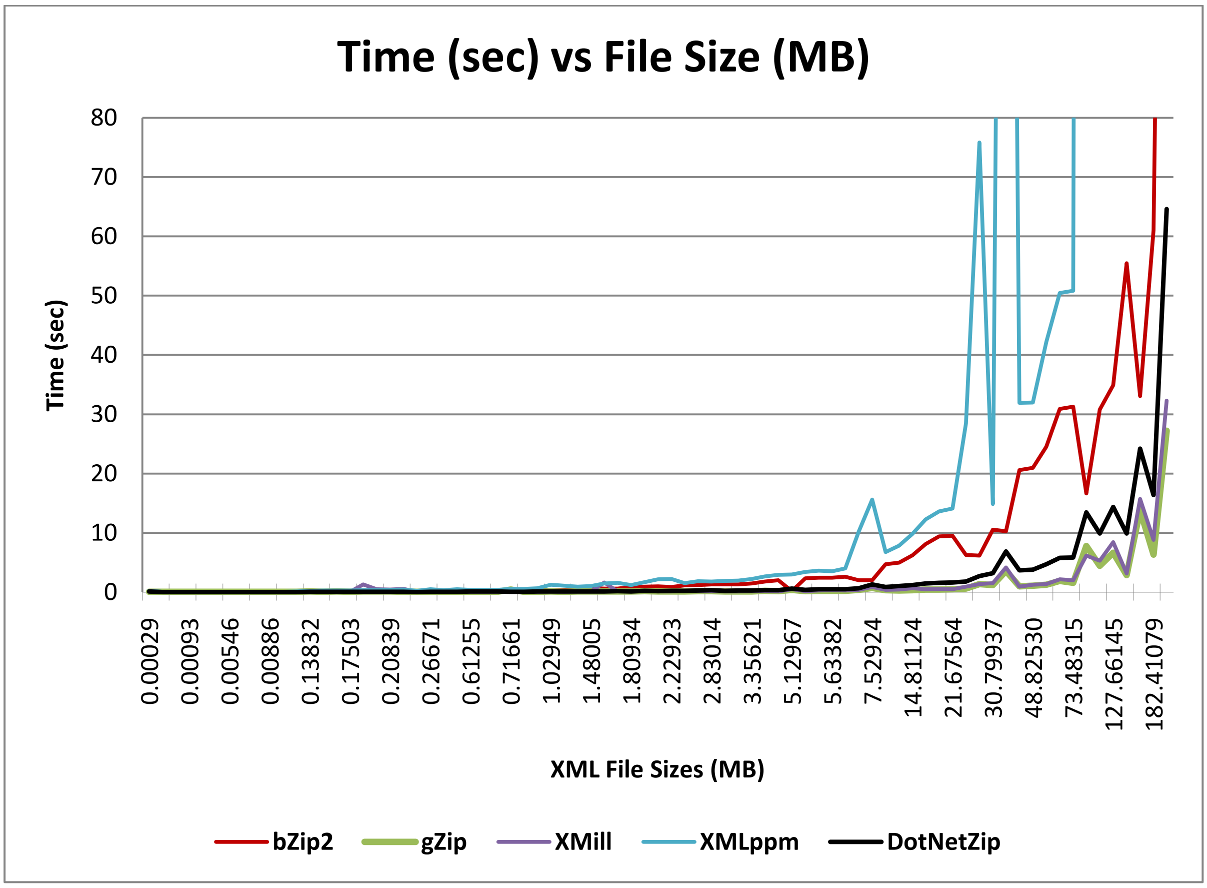


Figure 6.2: Compression Time (sec)

For XML files larger than 7MB, the mean compression time increases to approximately 60 seconds. For businessmen needing to access data via their mobile devices, time is an important factor. Data files larger than 7MB, however, generally store substantial amount of information potentially containing thousands of records not viewable on a mobile device. Additionally, Web Services offer functionality that do not require an entire data file to be transferred to the Client, but rather smaller chunks of information containing only necessary data. Thus, the lengthy compression time of large XML files pertain to situations often not encountered when using Web Services on a mobile device.

[1] Execution errors meant that the time taken to compress the XML files for the XWRT compression technique was not recorded.

Table 6.2: Compression Time Statistics

		bZip2	gZip	XMill	XMLppm	DotNetZip
Mean	Small XML	0.14330	0.07993	0.15683	0.26303	**0.04515**
	Medium XML	1.30122	**0.18560**	0.27676	2.98476	0.32988
	Large XML	30.24009	**3.85936**	4.48036	287.87893	9.03369
Median	Small XML	0.07800	0.06300	0.06300	0.27300	0.04736
	Medium XML	1.18800	0.12500	0.16700	1.95313	0.25977
	Large XML	18.61750	1.32050	1.43700	42.18750	4.22900
Std.Dev.	Small XML	0.13272	0.06643	0.26133	0.18177	0.03538
	Medium XML	0.71796	0.13772	0.32430	3.20537	0.25901
	Large XML	47.16192	6.18859	7.26906	554.88253	13.81142

The statistical data tabulated in Table 6.2 shows that the DotNetZip compression technique performs best for Small XML sized files, in terms of time taken to compress, with a mean of 0.045 seconds and standard deviation of 0.035 seconds. For the Medium and Large sized XML files, gZip performed best with a mean of 0.185 and 3.859 seconds, and a standard deviation of 0.066 and 6.188 seconds, respectively. Statistical evidence is provided and discussed later in this section as part of the analysis of the compression tests.

The decompression time was recorded for four of the six compression techniques[1]. Figure 6.3 illustrates that the bZip2, gZip and DotNetZip compression techniques take less than two seconds to decompress XML files up to approximately 25MB. In contrast, the decompression performance of the XMLppm compression technique was poor, as shown by its erratic decompression times.

Based on the recorded times plotted in Figure 6.3, it is evident that gZip generally outperforms the other compression techniques (for which a decompression time was recorded). With decompression times averaging less than 10 seconds for XML files up to 80MB (bZip2, gZip and DotNetZip), the perceived efficiency of using compression before transferring data from a Server to a Client is plausible.

[1] Execution errors meant that the time taken to decompress the XML files for the XMill and XWRT compression techniques were not recorded

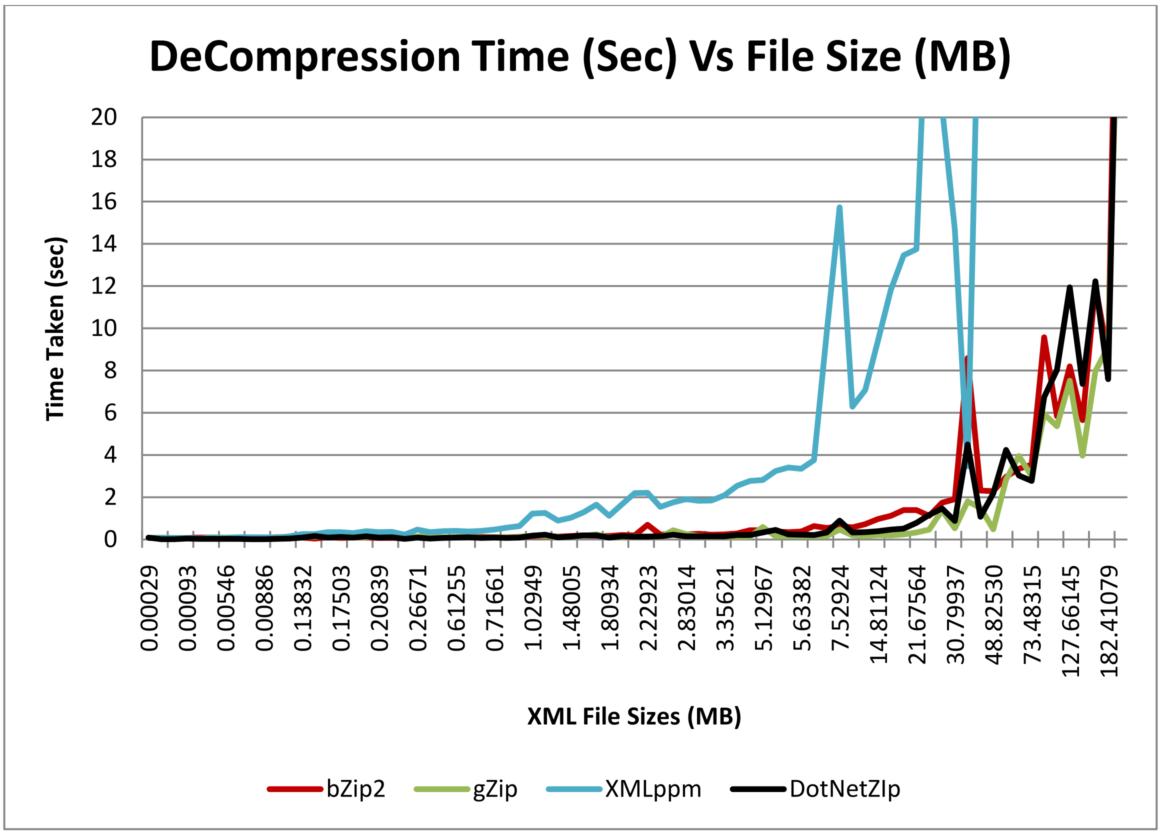


Figure 6.3: Decompression Time (sec)

Table 6.3 provides statistical information recorded for the decompression times. The DotNetZip compression technique once more performs best for Small sized XML files, with a mean of 0.063 seconds and a standard deviation of 0.039 seconds. Similarly to the compression times recorded in Table 6.2, gZip performs best, in terms of mean decompression times recorded, for Medium and Large sized XML files.

Table 6.3: Decompression Time Statistics

		bZip2	gZip	XMLppm	DotNetZip
Mean	Small XML	0.07604	0.09844	0.25938	**0.06276**
	Medium XML	0.30625	**0.18500**	2.91438	0.22211
	Large XML	5.63849	**4.00639**	94.76382	5.13361
Median	Small XML	0.07813	0.09375	0.27344	0.07031
	Medium XML	0.23438	0.14063	1.90625	0.19531
	Large XML	2.64063	1.64844	29.45313	2.47266
Std.Dev.	Small XML	0.01822	0.03981	0.17089	0.03873
	Medium XML	0.16707	0.12415	3.19009	0.16045
	Large XML	8.42464	6.66841	181.30503	7.62320

Although memory footprint data was recorded for the compression techniques[1], illustrated in Figure 6.4, the data collected indicated that a memory footprint above 0.70MB was recorded only once, and a mean of 0.380MB was observed. Additionally, the mean memory footprint for the different compression techniques was, statistically, similar. These results demonstrate that the memory footprint produced is insignificant, and, thus, will not be used as a comparison metric between the different compression techniques.

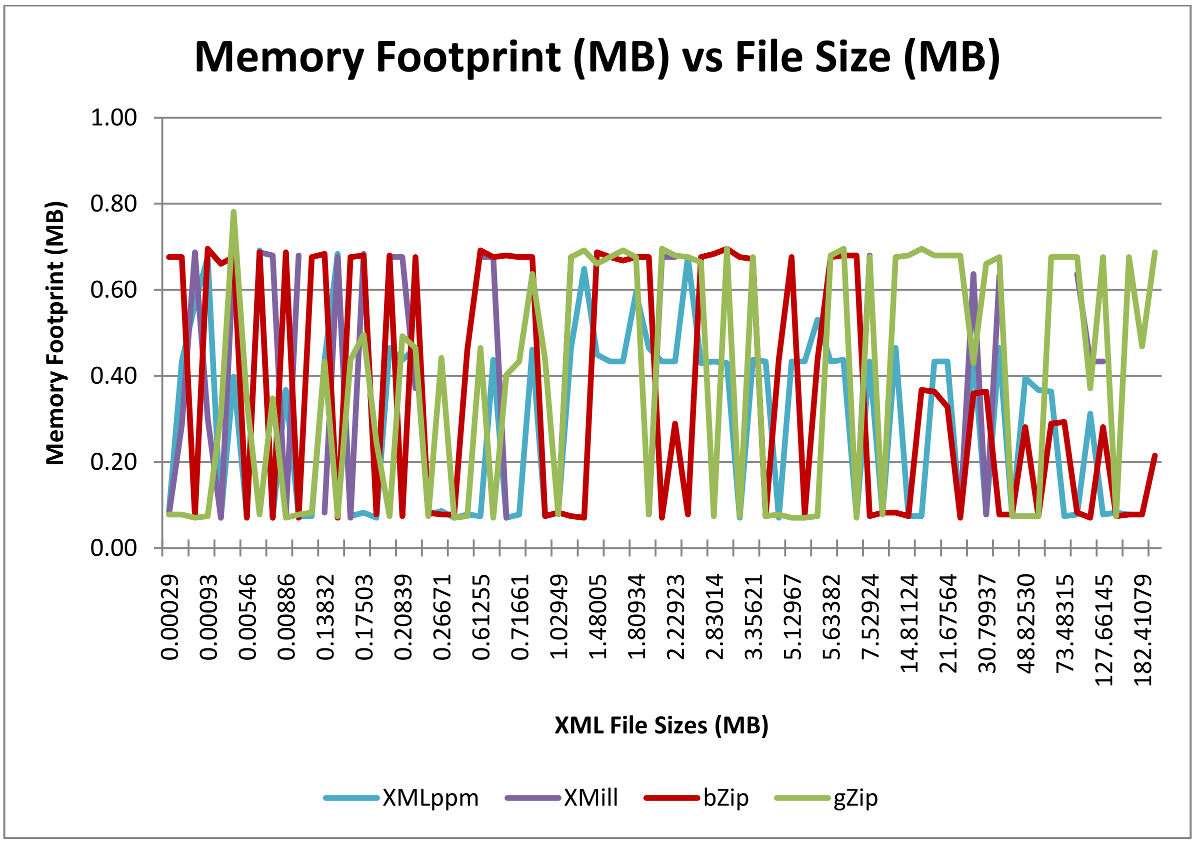


Figure 6.4: Memory Footprint (MB)

6.2.2.2 Descriptive Statistics for Image Compression

Table 6.4 shows a summary of the compression results recorded, in terms of compressed size, for each of the three compression techniques. The full table is shown in Appendix D. From these results, it is clear that the JPEG compression technique performs best, with a mean of 7.599% and standard deviation of 9.376%. A statistical analysis is conducted and discussed at the end of this section to provide a statistical motivation for the best compression technique.

[1] An execution error meant that the memory footprint was not recorded for the DotNetZip compression technique.

Table 6.4: Summary of Results for Image Compression

		GIF		JPEG		PNG	
Image	**Original Size (KB)**	**Compr. Size**	**Compr. Ratio**	**Compr. Size**	**Compr. Ratio**	**Compr. Size**	**Compr. Ratio**
Test1	17	4	23.52941%	3	17.64706%	4	23.52941%
Test3	106	13	12.26415%	12	11.32075%	15	14.15094%
Test9	770	9	1.16883%	39	5.06494%	4	0.51948%
Test13	2305	90	3.90456%	76	3.29718%	107	4.64208%
Test15	3165	302	9.54186%	48	1.51659%	159	5.02370%
Test20	4922	70	1.42219%	245	4.97765%	43	0.87363%
Test21	5499	212	3.85525%	104	1.89125%	361	6.56483%
Test25	10355	1310	12.65089%	384	3.70835%	2942	28.41140%
Test26	11075	694	6.26637%	380	3.43115%	2031	18.33860%
Test29	12830	2645	20.61574%	881	6.86672%	6041	47.08496%
Test30	21241	595	2.80119%	551	2.59404%	1162	5.47055%
Mean			10.69007%		**7.59941%**		19.29410%
Median			7.21936%		4.28371%		11.49662%
Std. Dev.			11.21166%		9.37601%		20.85987%

In order to compare the quality for the compression techniques, a rating system was used. A rating was given to each compressed image based on the quality of the image. The image quality rating is based on the visual similarities between the original, uncompressed image and that of the compressed image. A rating of 1 was given to poor quality images, 2 for acceptable quality images and 3 for good quality images.

Table 6.5: Quality Ratings for Image Compression

	GIF	**JPEG**	**PNG**
Image	**Rating**		
Test1	3	3	3
Test3	3	3	3
Test9	3	3	3
Test13	3	3	3
Test15	3	3	3
Test20	2	3	3
Test21	2	3	3
Test25	1	3	2
Test26	3	3	3
Test29	2	2	2
Test30	2	3	3
Mean	2.633333	**2.9**	2.833333
Median	3	3	3
Std. Dev.	0.614948	0.305129	0.461133

A summary of quality ratings, achieved for each image compression technique, are shown in Table 6.5. The full table is presented in Appendix E. Although the overall results were close, the table shows that the JPEG compression technique achieved better image quality, with a mean of 2.9 and standard deviation of 0.305.

6.2.2.3 Descriptive Statistics for Audio Compression

A summary of the compression results is shown in Table 6.6. The complete table of the results is shown in Appendix F. Table 6.6 provides the mean, median and standard deviation for the compression size, compression ratio and time taken during the compression process.

Table 6.6: Summary of the Results for Audio Compression

	MP3			**FLAC**			**MAC**		
	Compr. Size (KB)	**Ratio (%)**	**Time (sec)**	**Compr. Size (KB)**	**Ratio (%)**	**Time (sec)**	**Compr. Size (KB)**	**Ratio (%)**	**Time (sec)**
Mean	**2288.200**	**31.346%**	8.331	12303.333	44.738%	**2.735**	11525.967	40.367%	5.829
Median	553.500	9.089%	2.675	2132.000	41.570%	1.110	194750.500	38.068%	1.475
Std. Dev.	3579.689	87.414%	11.611	18504.906	24.411%	3.675	17428.186	19.837%	8.866

Based on the results from the table above it is clear that the MP3 compression technique performs the best in terms of file compression. The FLAC compression technique, however, performs best in terms of time taken to compress the audio files, although it yields the worst file compression of the three techniques. An ANOVA test was used to determine if there is a significant difference in terms of compression ratio and time between the three audio compression techniques.

The inferential statistics recorded for each of the three categories of compression techniques is discussed next.

6.2.3 Inferential Statistics

In order to select which compression technique performed best, for each of the three categories of compression techniques tested, an analysis was conducted for each of the performance metrics recorded. An Analysis of Variance (ANOVA) was used to provide statistical analysis on the performance data. ANOVA is used to test for significant differences between the means (StatSoft 2010).

The tests, for which performance data was recorded, were conducted to compare the mean performance of the compression techniques. An ANOVA test provides statistical analysis on whether there is a significant difference in performance between each of the compression techniques for each of the specific performance metrics.

Additionally, if a significant difference between the compression techniques was observed, Tukey's HSD (Honestly Significant Difference) Test was used to determine if there was a significant difference between any two compression techniques.

6.2.3.1 Inferential Statistics for XML Compression

Comparisons between the means of the different XML compression metrics are discussed in this subsection.

Compression Ratio

At a significance level of 95%, the null hypothesis that the mean for compression ratio are equal is rejected. Thus, there is a significant difference in the mean for the six compression techniques. Results show that for Small XML files the XMLppm compression technique outperforms all other compression techniques. For the Medium and Large sized XML files, the bZip2 and XWRT compression techniques performed best. Using Tukey's HSD, there is insufficient statistical evidence to support the conclusion that there is a difference in compression ratios between these two compression techniques.

Compression Time

There is a significant difference in the mean compression times, at a significance level of 95%, for the five compression techniques for which compression times were recorded. Thus, the null hypothesis that all mean compression times are equal is rejected. Using Tukey's HSD, however, there is insufficient evidence to conclude that there is a significant difference in compression times between the gZip, XMill, and DotNetZip compression techniques, which performed best overall.

For Small sized XML files, the DotNetZip compression technique performed best. For Medium and Large sized XML files both gZip and XMill performed best, yielding the fastest compression times. Although there is insufficient statistical evidence to suggest that there is a considerable difference in compression times recorded for the Small and Medium sized XML files, there is sufficient statistical evidence to conclude that there is a significant difference in compression times for Large sized XML files. The gZip compression technique performed best for Large sized XML files.

Decompression Time

At a significance level of 95%, there is a significant difference in the mean decompression time for the four compression techniques for which decompression times were recorded. There is, however, insufficient evidence, based on Tukey's HSD, to conclude that there is significant statistical difference in decompression times between the gZip and DotNetZip compression techniques, which performed best overall.

Memory Footprint

There is insufficient evidence to suggest, at a significance level of 95%, that there is a difference in mean memory used for each of the compression techniques for which memory usage was recorded. Additionally, due to the minimal amount of memory used, the memory footprint metric is not considered when comparing the compression techniques.

Ease of Implementation

In addition to performance metrics, the ease of integrating the compression technique (when implementing for the mobile environment) was also investigated. During the testing phase, a number of implementation and execution errors were observed for certain compression techniques.

When conducting the compression tests, the XWRT and XMill compression techniques encountered numerous execution errors. Thus, the compression time was not recorded for XWRT, and the decompression time was not recorded for XMill. For all of the metrics recorded, XMill performed admirably only in terms of compression time. For these reasons, XWRT and XMill were not considered for the implementation of the prototype.

Overall Analysis

When comparing the performance of the different compression techniques, compression ratio, compression time, and decompression time was examined. Overall, XMLppm performed the best in terms of compression ratio. In terms of compression and decompression times, gZip performed best for Medium and Large sized XML files, and DotNetZip performed best for small sized XML files.

When testing the feasibility of the compression techniques within the mobile environment, only two compression techniques were able to be executed without having to recode and republish the source code. However, the DotNetZip compression technique was significantly less complex to execute than the gZip compression technique.

Overall, DotNetZip was considered the best compression technique. It performed statistically similar to the best compression techniques for each of the respective metrics. Additionally, it was significantly easier to execute for both the desktop and mobile environments. Thus, DotNetZip will be utilised for the development of the prototype.

6.2.3.2 Inferential Statistics for Image Compression

Comparisons of the mean compression ratio and mean image quality were conducted to determine whether there was significant difference between the three image compression techniques.

<u>Compression Ratio</u>

At a significance level of 95%, the null hypothesis that the mean for compression ratio are equal is rejected. There is a significant difference in the mean for the three image compression techniques. Additionally, there is sufficient statistical evidence, based on Tukey's HSD, to support the conclusion that there is a difference in compression ratios between the GIF and JPEG compression techniques, which performed the best. Thus, JPEG statistically performed the best in terms of compression ratio achieved.

<u>Image Quality</u>

Lossy image compression uses psychovisual modelling, which removes colours and artefacts that human beings cannot see with the naked eye. Lossless image compression, however, uses compression algorithms, which retain the original image quality. The image quality rating was based on the perceived difference between the original, uncompressed image and the compressed image.

There is insufficient statistical evidence to suggest, at a significance level of 95%, that there is a significant difference in image quality for each of the image compression techniques. Due to the standardisation of these compression techniques in industry and its improvements over a number of years, the image quality is of high standard. This is evident even at a mean compression level of less than 20% of the original uncompressed size.

Overall Analysis

The JPEG compression technique performed best in terms of compression ratios yielded. Additionally, although there is no statistical difference between the compression techniques, the JPEG compression technique performed best in terms of image quality. It is this superior compression ratio and image quality that has led JPEG to become the most popular compression technique in industry and every day use. Thus, the JPEG compression technique will be utilised for the development of the prototype.

6.2.3.3 Inferential Statistics for Audio Compression

A comparison of the mean compression ratio and mean compression time were conducted to determine whether there was significant difference between the three audio compression techniques tested.

Compression Ratio

The null hypothesis that the mean for compression ratio for the three audio compression techniques are equal is rejected. At a significance level of 95% there is a significant difference in the mean for the three compression techniques. Further statistical analysis, using Tukey's HSD, supports the claim that there is a significant difference in compression ratios between the two best compression techniques in terms of compression ratio. Thus, based on the statistical evidence, the MP3 compression technique performs the best in terms of compression ratio yielded.

Compression Time

At a significance level of 95%, there is a significant difference in the mean compression times for the three audio compression techniques. The null hypothesis that all mean compression times are equal is, therefore, rejected. Using Tukey's HSD, however, there is insufficient statistical evidence to conclude that there is a significant difference in compression times between the FLAC and MAC compression techniques, which performed best overall.

Overall Analysis

The MP3 compression technique performed considerably better in terms of compression of the audio files. The FLAC and MAC compression techniques performed best in terms of the amount of time taken to compression the audio files. Although the MP3 compression technique did not perform as well in terms of time taken for compression, the mean time taken was only six seconds slower than the mean recorded for FLAC.

Additionally, the MP3 format is well supported by most phones. In contrast, the FLAC and MAC compression techniques are not readily supported in the mobile environment, and require the download of specific codecs or music players in order to playback these formats. Therefore, based on the overall performance and widespread use, the MP3 compression technique is selected for use in the implementation of the prototype.

6.2.4 Initial Testing Summary

The tests conducted confirmed the possible usefulness and feasibility to use compression within the prototype. Additionally, the tests illustrated that compression may provide efficiency within the prototype. Although the best compression techniques were selected for use within the prototype, based on the comparisons conducted, the intelligence provided by the prototype should allow different compression techniques to be used simultaneously, as well as plug in new and improved techniques.

Further evaluation of the prototype is discussed in Chapter 7. Before the implementation process is discussed, the criteria for which the prototype was developed is discussed in the next section.

6.3 Implementation Criteria

A Mobile SOA was defined in Section 3.6.1 as: "*An architectural style for the mobile environment whose goal is to achieve loose coupling by positioning Services as the primary means through which solution logic is represented.*" This section provides a brief discussion on the criteria for the prototype, which was developed as a proof-of-concept of the proposed framework.

When selecting a domain for the proof-of-concept prototype, certain criteria were considered. The criteria were, thus, as follows:

- The prototype had to be developed for the mobile environment;
- A topical and interesting application domain was required;
- The prototype is required to be feasible for the current mobile technology;

- The basis of the prototype can be adapted to run on different mobile devices available on today's market; and
- The prototype should be representative of the proposed framework and support all underlying functionalities proposed.

The implementation domain was divided into two categories of Service delivery. It was decided to implement a mobile application, which provides access to both types of Web Services, Content Services and Added Value Services. The primary deliverable of Content Services is media files, such as images and audio. Added Value Services are based on the exchange of data between the Client and Server in order to accomplish a specific task.

Three Web Services were developed for the prototype, a Content Service, an Added Value Service, and a combination of both types of Service. In terms of the Content Service, a Web Service was developed based on the delivery of images and audio. This Web Service is similar to Content Delivery Services made popular by media companies, such as 35050[1], 31314[2] and ExactMobile[3], whereby a User sends an SMS with a specific code to a specific number, with a fixed charge, and receives an image, audio or video file.

For the Added Value Service, a Web Service was implemented that performs a specific task for the User. The data used for this Service was based on the marks of students for a certain module. The data included information about the student, the marks of their semester tests, and their exam mark. The tasks to be performed on this data included the calculation of the students' average for their semester tests, whether they have received a DP[4] (Duly Performed) for this module, the weighted average of their semester mark and exam mark, and whether they have successfully completed and passed this module.

A Web Service resembling that of a social network or blog, such as Facebook[5], MySpace[6] and Twitter[7], was developed as a combination of Content Services and Added Value

[1] http://www.mobilemagic.co.za

[2] http://31314.mobi

[3] http://www.exactmobile.co.za

[4] A DP is the minimum mark required to be allowed to write an exam for a particular module. The mark is calculated as the (weighted) average of semester tests and any additional tasks for which marks are recorded.

[5] http://www.facebook.com

[6] http://www.myspace.com

[7] http://twitter.com

Services. The Web Service provided information about a particular subject, and was accompanied by an image, which would assist in the explanation of that subject. The choice of the domain was, thus, feasible for current mobile technology, as confirmed by the execution of similar systems within the particular domains.

The prototype, implemented as a proof-of-concept in this chapter, was developed according to the design principles (Section 5.3.1) and within the specified domain. The three Web Services developed for the prototype are representative of the two common types of Services available within the Mobile SOA environment.

The prototype allows mobile Users to perform tasks and request media content in an efficient manner via a Mobile Web Service. The implementation tools used to develop the proof-of-concept prototype are discussed in the following section.

6.4 Implementation Tools

The successful implementation of the proof-of-concept prototype is determined by the implementation tools. This section discusses the implementation tools used to develop the prototype, which is based on the proposed framework, as identified in Chapter 5. Each of the different elements of the proposed framework requires different implementation methods and tools.

6.4.1 Database Server

Microsoft SQL Server 2008 (Microsoft 2010b) was used for the database server. MS SQL Server 2008 is a database server used for the development and management of a database, and includes features, such as business intelligence and data warehousing. The database used within the proof-of-concept prototype stores information required to build the User and Phone profiles. This information assists the categorisation and compression process, as discussed in Chapter 5. The structure and arrangement of this database is discussed in Section 6.5.2.2.

6.4.2 Development Languages and Programming Environment

One of the principles of an SOA is that a Web Service may be accessed from any platform, regardless of the development language. For this reason, there is little significance in the selection of any one development language in which the prototype is implemented.

However, due to the scope constraints of the two mobile devices utilised for testing, the .Net Framework was used. The .Net Framework is an important Windows component, which supports the development and execution of Windows applications and XML-based Web Services (Microsoft 2010c). For this purpose, C# was selected as the standard programming language in which to implement the XML-based Web Services and proof-of-concept prototype.

The prototype was implemented using the Microsoft Visual Studio IDE (Microsoft 2010d). This IDE supports implementation of C# applications in conjunction with the .Net Framework. More significantly, Visual Studio supports the implementation of XML-based Web Services. Visual Studio 2010 has development support only for Windows Mobile 7, which is available only on more recent and up-market mobile phones. Thus, Visual Studio 2008 was selected as it provides support for all Windows Mobile platforms prior to Windows Mobile 7.

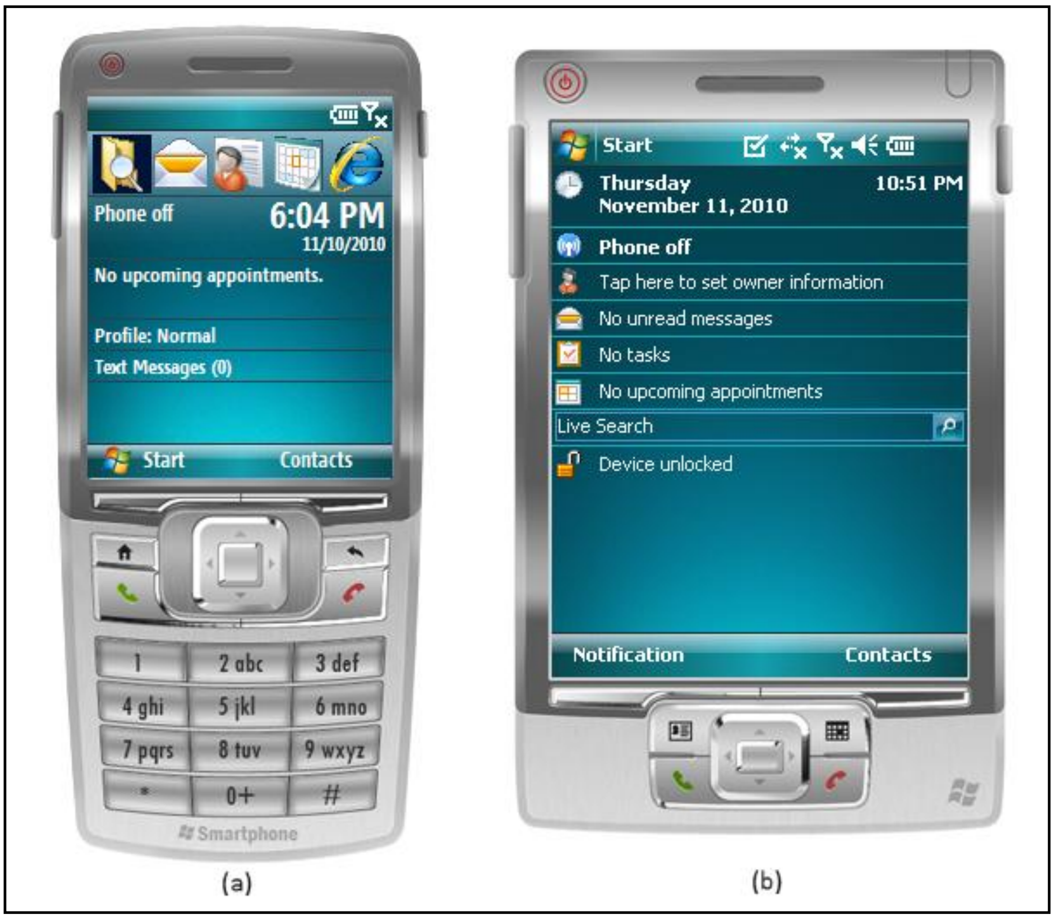
6.4.3 Mobile Platform

The proof-of-concept prototype was developed for use in a mobile environment. The mobile devices used for testing the prototype both run the Windows Mobile operating system. As a result of its backward compatibility to previous Windows Mobile versions, the Windows Mobile 6 (Microsoft 2010e) developer platform was used during implementation. As discussed previously, Visual Studio 2008 includes development support for all versions of Windows Mobile prior to Windows Mobile 7.

In order to provide greater development support and additional tools for Visual Studio 2008, Windows Mobile 6 SDK (Microsoft, 2010f) was imported. Windows Mobile 6 SDK provides additional documentation, sample code, header and library files and emulator images[1].

Figure 6.5 illustrates the emulator for the (a) Windows Mobile 6 Standard SDK and (b) Windows Mobile 6 Professional SDK packages. This emulator simulates the mobile environment, making it easier to implement and test mobile applications for the Windows Mobile operating system.

[1] Emulator Images facilitate the emulation of a specific mobile environment on the developer's computer to simulate how a mobile application may perform on an actual mobile device.


Phone off
6:04 PM
11/10/2010
No upcoming appointments.
Profile: Normal
Text Messages (0)

Start
Thursday
November 11, 2010
10:51 PM
Phone off
Tap here to set owner information
No unread messages
No tasks
No upcoming appointments
Live Search
Device unlocked

Figure 6.5: Windows Mobile SDK Emulator (a) Standard (b) Professional

Two different mobile phones were used for testing the feasibility and performance of the prototype. These were the HTC s710 and the HTC TyTN II. The specifications for both phones are shown in the Table 6.7.

Table 6.7: Test Phones' Specifications

	HTC S710	**HTC TyTN II**
Operating System	Windows Mobile 6 (Standard)	Windows Mobile 6 (Professional)
Processor	TI's OMAP 850: 201MHz	Qualcomm MSM 7200: 400MHz
Processing Memory	64MB SDRAM	128MB SDRAM
Onboard Storage	128MB	256MB
Max. External Storage	MicroSD	MicroSD (SD 2.0 Compatible)
Screen Dimensions	240x320 pixels	240x320 pixels
Network Capability	Bluetooth, WiFi (IEEE 802.11), GSM/GPRS/EDGE Quad-band	Bluetooth, WiFi (IEEE 802.11), HSDPA/UMTS Tri-band GSM/GPRS/EDGE Quad-band
Battery	Rechargeable Lithium-Io Polymer Battery 175 Hours Standby 7 Hours Talk Time	Rechargeable Li-Polymer Battery 350 Hours Standby 7 Hours (GSM), 4.5 Hours (UMTS)

6.4.4 Service Deployment

The developed Web Services are published on Internet Information Services (IIS) 7 (Microsoft 2010g) running on Microsoft Windows Server 2008 (Microsoft 2010h). IIS is a Web Server software package, which provides tools and features to facilitate secure and easily managed hosting on the Web. Additionally, IIS provides easy integration for applications and Web Services developed using the .Net Framework. Thus, IIS was selected as the application server to host the implemented Web Services. Microsoft Windows Server 2008 is server operating system, which provides secure management of a server. Windows Server 2008 runs on a Sun Fire V40z server with specifications shown in Table 6.8.

Table 6.8: Sun Fire V40z Server Specification

Processors	4 x AMD Opteron Model 848 2.19 GHz
Processing Memory	6 GB RAM (4 x 1GB, 4 x 512MB)
Storage Disks	3 x 80 GB Ultra320 SCSI (10 000 rpm)
Network Adapter	2 x Gigabit Ethernet
Operating System	Windows Server 2008 (64-Bit) Service Pack 2

The most commonly used approach for implementing for SOA is the XML-based Web Service. This approach utilises popular standards, such as WSDL, SOAP and UDDI. Although an SOA is not required to be constructed using these standards, the industry standards, the availability of implementation tools (as discussed above) and numerous interoperability benefits strongly recommend their use (Liegl 2007).

6.5 Implementing Prototype for Framework

The design of the proposed framework was based on the EXEM framework (Natchetoi et al. 2007b). The prototype was developed as a proof-of-concept of the proposed framework, which was designed and delineated in Chapter 5.

The implementation of this prototype provided a practical example of how the various components of the proposed framework, Figure 6.6, may be implemented and how these components may interact with one another in an effort to achieve effective and efficient file transfer of a Web Service to a mobile device.

Additionally, the prototype was developed to provide a means for evaluating effectiveness and efficiency, and the perceived benefits of the proposed framework. When developing prototypes for software applications, it provides limited functionality as it simulates only a few features of the fully developed product. Typically, prototypes are developed to allow users to test the graphical interface of the application in question.

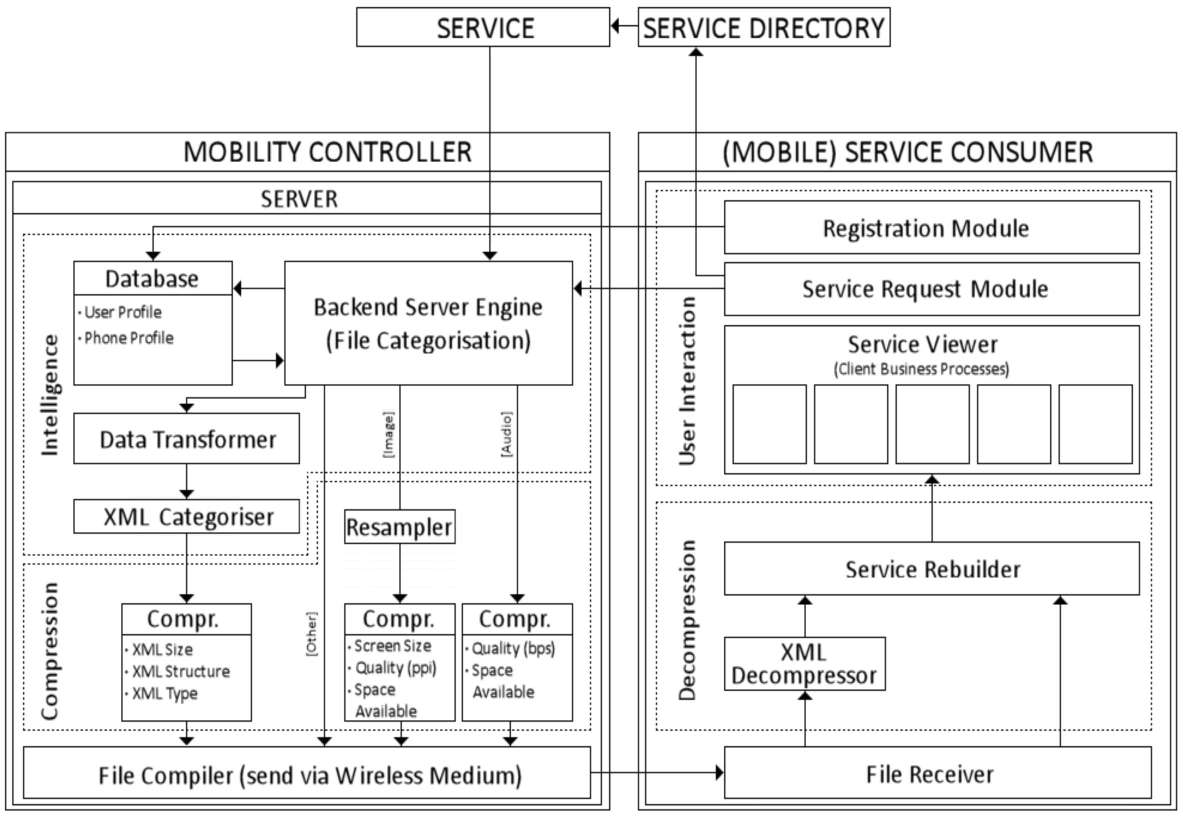


Figure 6.6: Proposed Framework[1]

When developing a framework, however, where a graphical interface is of negligible importance, users are not required when testing the prototype. The feasibility and performance of the proposed framework is of higher importance. Thus, a performance and capacity prototyping method was used. This type of prototyping defines, demonstrates and predicts how the proposed framework will perform, as well as to demonstrate and evaluate non-functional aspects of the framework. The following subsection discusses the implementation of the framework components.

6.5.1 User Interaction Component

The User Interaction Component was the first of the four framework components involved in the facilitation of intelligent compression and efficient file transfer of Web Services. This

[1] Figure 5.4 is repeated as Figure 6.6 for ease of reference.

component consists of three subcomponents, namely the Registration Module, Service Request Module and Service Viewer.

6.5.1.1 Registration Module

The Registration Module performs the task that its name suggests. Figure 6.7 illustrates the *default registration screen* and the *incomplete registration screen*, developed for the prototype. The incomplete registration screen provides an alternative process if the default registration process is not completed successfully. The registration process is conducted by means of simply typing in a name, surname, address, occupation and employer.

All other information is gathered automatically by the mobile application, such as the IMEI number, phone manufacturer, phone model, operating system and screen dimensions. In the case when the mobile application fails to extract information that is critical to the performance of the framework, the mobile application displays the full list of information that it requires. The information already gathered is automatically filled in, as shown in Figure 6.7 (b), which requires that only the missing information be filled in.

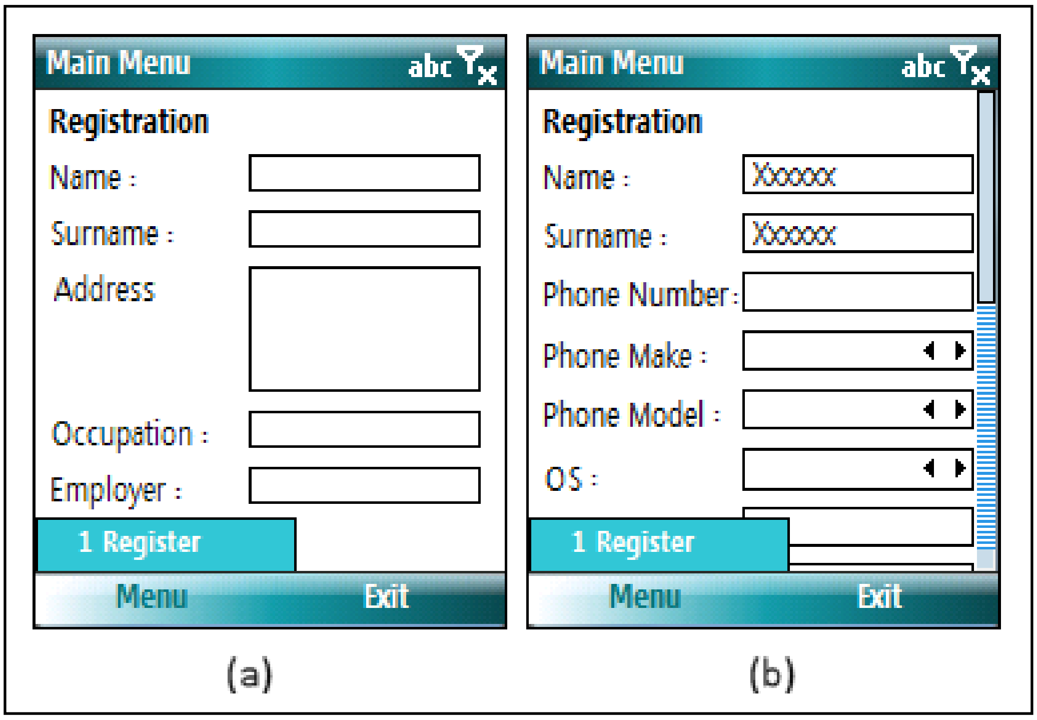


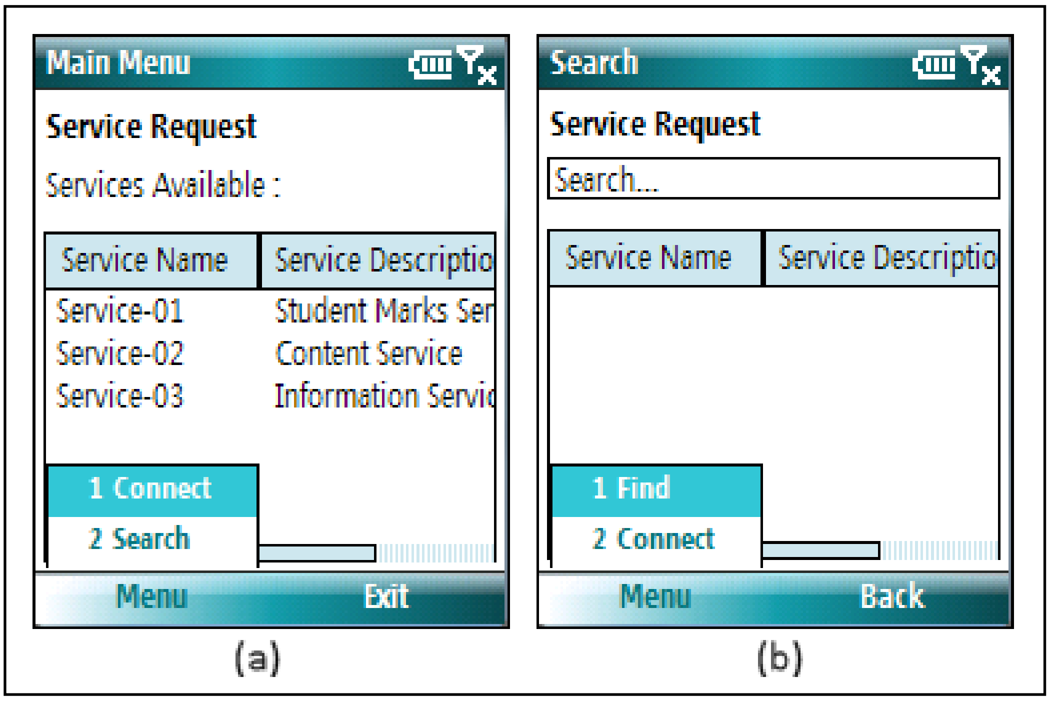
Figure 6.7: Registration Screens (a) Default Screen (b) Incomplete Registration Screen

The registration process is only conducted once. The information collected is stored within a database, discussed in Section 6.5.2.2. When running the mobile application after the registration process is successfully completed, the User and Phone Profiles are automatically loaded upon start. Additionally, after successful registration, the User may request or search for Services using the Service Request Module.

6.5.1.2 Service Request Module

Three Web Services were implemented to test the prototype, as discussed in Section 6.3. The mobile application automatically lists the Services available on the Service Directory. For this research, the Service Directory is simulated by a local Server using IIS 7 (Microsoft 2010g).

The three Services are listed in conjunction with their individual Service descriptions. To select a Service, the User simply scrolls to the Service they wish to connect to and selects the Connect option using the Menu button, as shown in Figure 6.8 (a).



Service Request

Services Available :

Service Name	Service Descriptio
Service-01	Student Marks Ser
Service-02	Content Service
Service-03	Information Servic

Service Request

Search...

Service Name	Service Descriptio

Figure 6.8: Service Request Forms (a) Default Screen (b) Search Screen

Additionally, if there are too many Services listed, or the desired Service does not appear on the list, the User may search for additional Services by selecting the Search option using the Menu button. The User is then prompted to type in specific criteria on which to search, and then selects the Find option using the Menu button, as illustrated in Figure 6.8 (b). If any Services were found matching the search criteria, the Services would be listed as in the default screen, Figure 6.8 (a).

6.5.1.3 Service Viewer

When the User connects to a specific Service, that Service's operational tasks are listed. The User may then invoke these tasks by selecting them, which then generates a specific result. Figure 6.9 illustrates an image request invocation by the User via the Content Service.

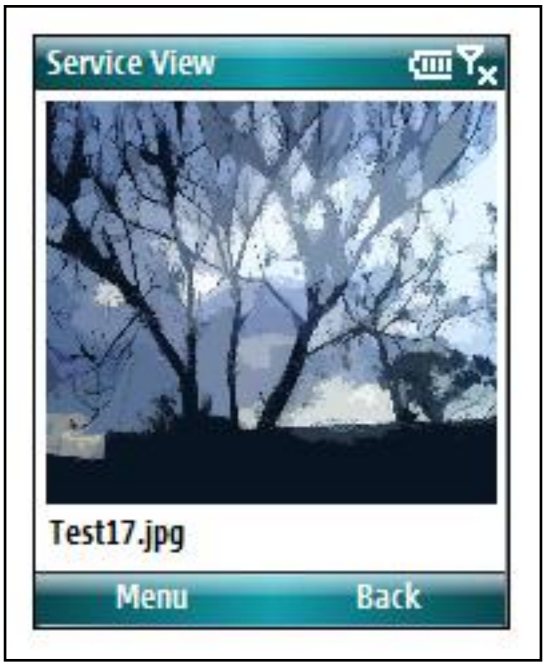


Figure 6.9: Service Viewer

Before the content is delivered to the Mobile Client a sequence of tasks is performed. Information from the Mobile Client is passed to the Intelligence Component, the files from the requested Service are compressed using the Compression Component, sent via a wireless medium, received and decompressed on the mobile device, and then displayed using the Service Viewer.

The following section describes the implementation of the Intelligence Component.

6.5.2 Intelligence Component

The intelligence of the prototype is managed within the Intelligence Component. Information gathered within this component provides the Compression Component with parameters to improve the efficiency of file transfer by achieving better compression performance. The subcomponents of the Intelligence Component are the Database, Backend Server Engine, Data Transformer and XML Categoriser.

6.5.2.1 Backend Server Engine

The Backend Server Engine manages all information and data from the Service Provider and (Mobile) Service Client. Information retrieved during registration is processed and stored within the database. This information is grouped as the User Profile or Phone Profile data. Due to numerous amounts of new mobile devices being introduced on a regular basis, the Phone Profile stores its information using three categories. These categories group the mobile devices according to their performance, and are categorised as Low-end, Medium-end and High-end mobile devices.

In order to group the mobile devices accordingly, the Backend Server Engine uses the information gathered during the registration process. This information is analysed and given a specific rating according to the mobile device's performance. The performance of the mobile device is compared against specific criteria for each specification. This rating is used to categorise a mobile device as being a Low-end, Medium-end or High-end device. The formula to calculate this rating is given in Figure 6.10, designed specifically for the prototype.

$$Rating\ Parameter\ x = \begin{cases} 1, & P_x < \frac{P_m}{3} \\ 2, & \frac{P_m}{3} < P_x \leq \frac{2P_m}{3} \\ 3, & P_x \geq \frac{2P_m}{3} \end{cases}$$

Where:
$x\ is\ the\ given\ paramer,\ P_x\ is\ the\ performance\ of\ parameter\ x,$ and
$P_m\ is\ the\ maximum\ performance\ for\ the\ given\ parameter$

$$Total\ Rating = \begin{cases} 1, & P_T < n \\ 2, & n < P_T \leq 2n \\ 3, & P_T \geq 2n \end{cases}$$

Where:
$P_T = \sum_{x=1}^{n} R_x$, where $R_x\ is\ the\ rating\ of\ parameter\ x$ and
$n\ is\ the\ number\ of\ parameters\ for\ that\ group$

Figure 6.10: Ratings Formula

For example, when categorising the general specifications of the mobile, which includes the CPU clock speed, the processing RAM and operating system, each parameter is given a rating from one to three. This rating is dependent on how the mobile devices compares to the best available performance for that specific parameter. If the best mobile device CPU clock speed is 1GHz, this value is separated into three groups of 0-333MHz, 334-666MHz and 667-1000MHz and given ratings of 1, 2 and 3, respectively. Similarly, ratings are assigned to all other parameters. The ratings for each general specification parameter are summed to give a total rating. Once again this rating is separated into three, this time yielding an overall rating for general specifications for the mobile device.

The rating system was developed for future use to improve intelligence and assist the compression process. It allows the Server to determine specific levels of compression to use, based on the performance of the mobile device. For example, a mobile device with a low rating will not perform high level compression, as it would take much longer, use more processing power and memory, and may lead to battery power depletion.

Additionally, the Backend Server Engine collects the required files from the requested Web Services, as well all the necessary information used to facilitate improved compression of these files. The information collected from the database and Mobile Client is passed through as parameters to the respective compression techniques. The Backend Server Engine also decides which compression technique to use to compress a given file. In addition to this, the Backend Server Engine facilitates the transfer of the compression technique, used during compression, to the mobile device, which enables it to perform the necessary decompression. The compression technique, is, however, only transferred once to the mobile device, which stores it on the device for future use.

6.5.2.2 Database

The database serves the purpose of storing important information to be used within the Intelligence Component in order to achieve better compression results. This information is divided into two groups, the User Profile and Phone Profile, as shown in Figure 6.11.

The User Profile contains one table, namely the UserInformation table. The UserInformation table stores the Name, Surname, PhoneNo, and PhoneTypeID records. The UniqueID record generally stores the mobile device's IMEI number, unique to every mobile device. In the event that the IMEI number is not accessible, the phone number may be used as the

UniqueID for that Client. This table also stores the Name and Surname records. This table is linked to the PhoneType table using the PhoneTypeID foreign key, which links specifications about the Client’s mobile device to their User Profile.

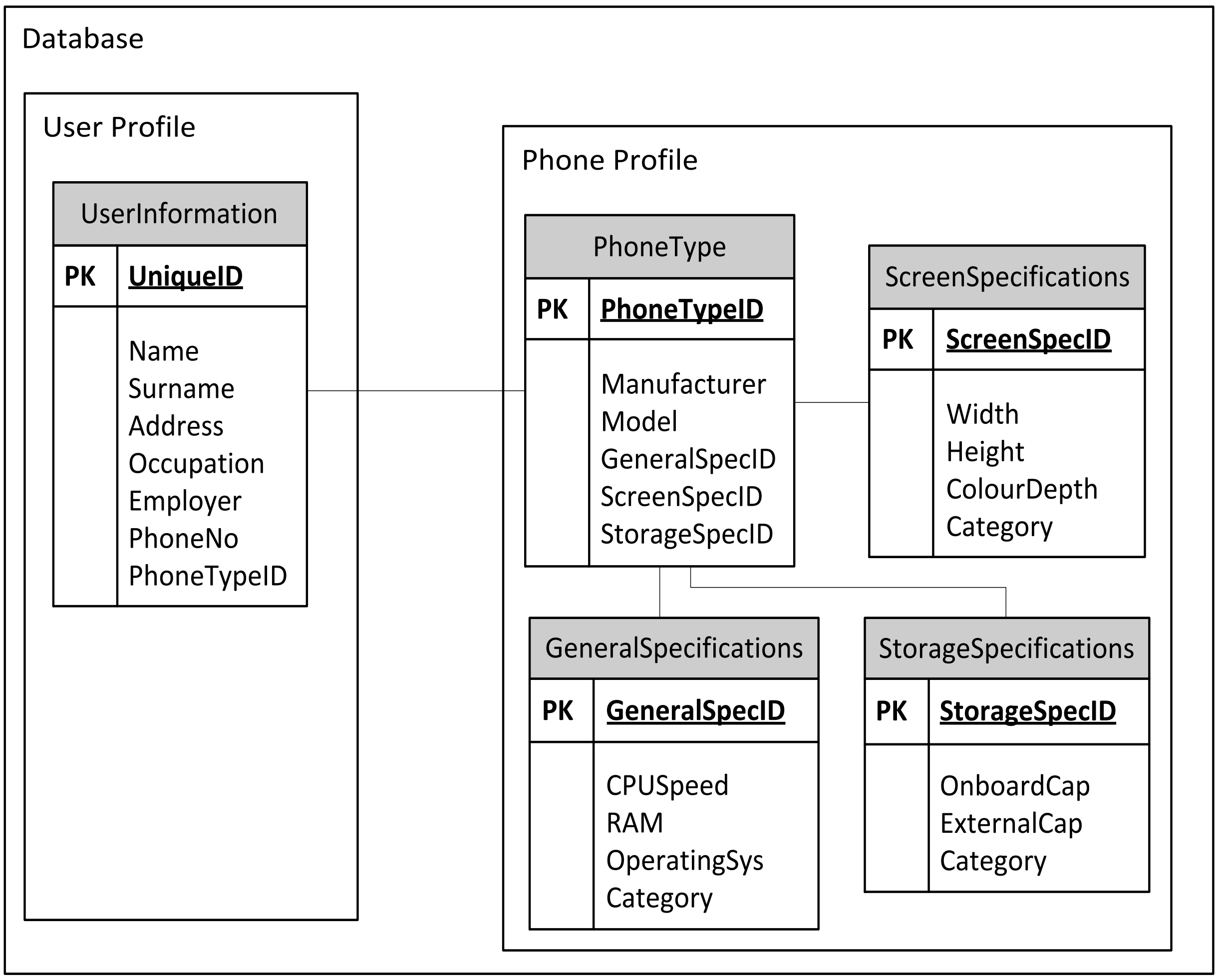


Figure 6.11: System Database

The Phone Profile stores the most important specifications about the mobile devices used by the Client. In the *PhoneType* table, *thePhoneTypeID* is stored, which stores information about each *Manufacturer* and *Model* for each new mobile device that is registered. This table is linked to the specifications tables, namely the *GeneralSpecifications*, *ScreenSpecifications* and *StorageSpecifications* tables.

The *GeneralSpecifications* table stores information about the performance of the mobile device, which includes the CPU clock speed, available processing RAM and operating system. Each of these fields are categorised accordingly. For example, a mobile device with a high CPU clock speed and large amount of processing RAM available will be categorised under the High-end category. This categorisation process is conducted by the Backend Server Engine.

For the *ScreenSpecifications* table, information about the width and height of the viewing screen is recorded. Additionally, the colour depth is recorded in order to select a specific colour depth for images when being compressed. The *StorageSpecifications* table stores information about the storage space available on the mobile device. Although some phones may have poor onboard storage, the use of an external storage, such as an SD card, may give the mobile device a higher storage rating, and, thus, a better categorisation.

6.5.2.3 XML Categoriser

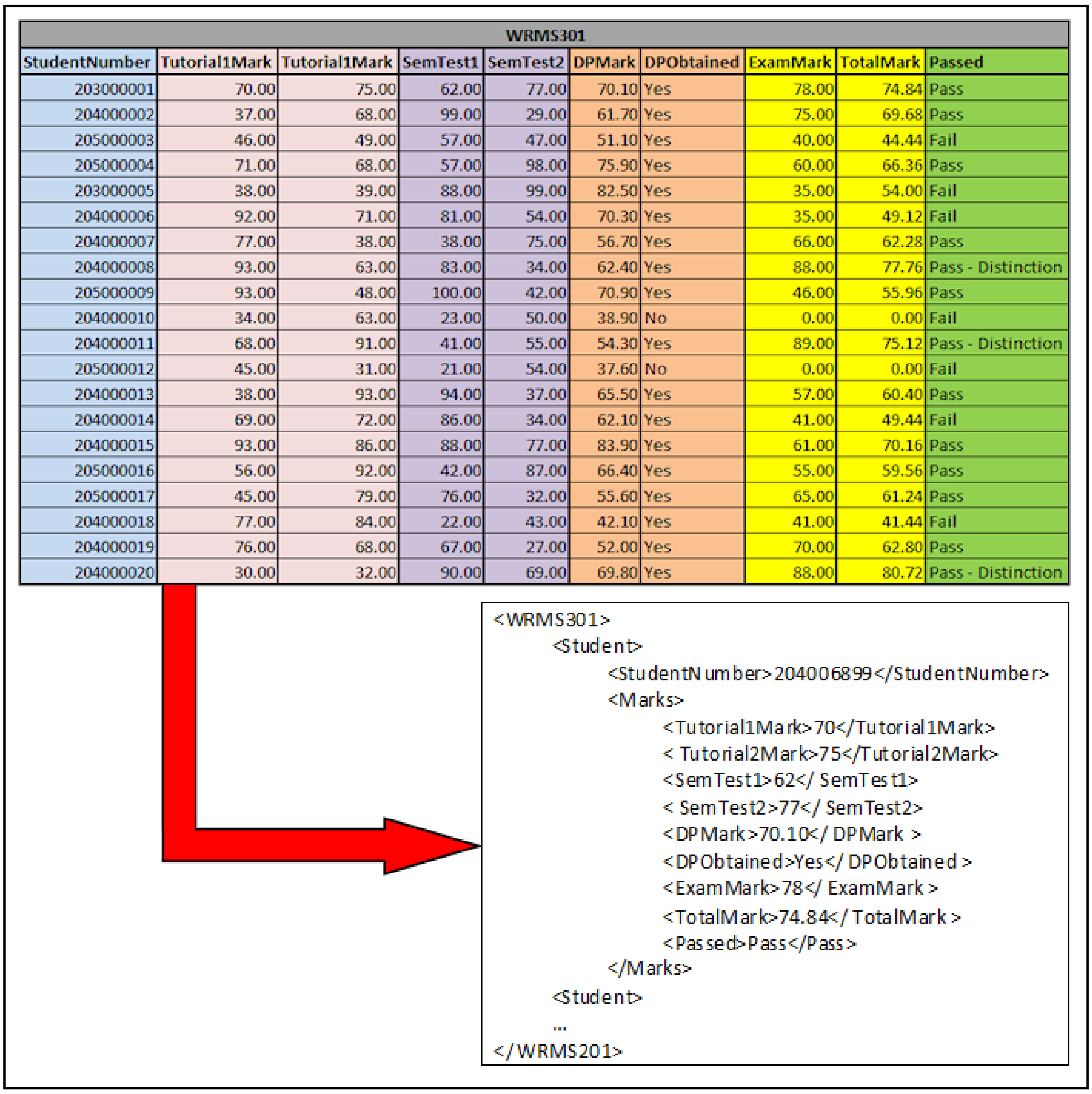
The purpose of the XML Categoriser is to group XML files according to their internal structures, content, file type and file size. Based on research (Lian, Cheung, Mamoulis & Yui 2004; Yongming, Dehua & JIajin 2008; Levene & Wood 2002) the categorisation and grouping of XML files are important in providing efficient data mining, searching and, fundamentally, compression of these files.

For the implementation of the prototype, the categorisation of XML files was considered out of scope. The categorisation of XML files is beneficial only if different compression techniques perform better for the different categories of XML files. Additionally, testing would require testing of each compression technique for each category of XML file.

The aim of developing the prototype for this research is to provide information about the feasibility and efficiency of the proposed framework within the Mobile SOA environment. The proposed framework, however, does include the use of this feature, but no categorisation algorithms were evaluated for use within the prototype.

6.5.2.4 Data Transformer

Before data files are compressed, the information stored within these files are converted to an XML format and stored as an XML file. This transformation process is managed by the Data Transformer subcomponent. For example, data stored within an EXCEL file is extracted, the information transformed using the XML format, and then stored as an XML file. Figure 6.12 illustrates this example by transforming student marks from an EXCEL file into its equivalent XML format.



WRMS301									
StudentNumber	Tutorial1Mark	Tutorial1Mark	SemTest1	SemTest2	DPMark	DPObtained	ExamMark	TotalMark	Passed
203000001	70.00	75.00	62.00	77.00	70.10	Yes	78.00	74.84	Pass
204000002	37.00	68.00	99.00	29.00	61.70	Yes	75.00	69.68	Pass
205000003	46.00	49.00	57.00	47.00	51.10	Yes	40.00	44.44	Fail
205000004	71.00	68.00	57.00	98.00	75.90	Yes	60.00	66.36	Pass
203000005	38.00	39.00	88.00	99.00	82.50	Yes	35.00	54.00	Fail
204000006	92.00	71.00	81.00	54.00	70.30	Yes	35.00	49.12	Fail
204000007	77.00	38.00	38.00	75.00	56.70	Yes	66.00	62.28	Pass
204000008	93.00	63.00	83.00	34.00	62.40	Yes	88.00	77.76	Pass - Distinction
205000009	93.00	48.00	100.00	42.00	70.90	Yes	46.00	55.96	Pass
204000010	34.00	63.00	23.00	50.00	38.90	No	0.00	0.00	Fail
204000011	68.00	91.00	41.00	55.00	54.30	Yes	89.00	75.12	Pass - Distinction
205000012	45.00	31.00	21.00	54.00	37.60	No	0.00	0.00	Fail
204000013	38.00	93.00	94.00	37.00	65.50	Yes	57.00	60.40	Pass
204000014	69.00	72.00	86.00	34.00	62.10	Yes	41.00	49.44	Fail
204000015	93.00	86.00	88.00	77.00	83.90	Yes	61.00	70.16	Pass
205000016	56.00	92.00	42.00	87.00	66.40	Yes	55.00	59.56	Pass
205000017	45.00	79.00	76.00	32.00	55.60	Yes	65.00	61.24	Pass
204000018	77.00	84.00	22.00	43.00	42.10	Yes	41.00	41.44	Fail
204000019	76.00	68.00	67.00	27.00	52.00	Yes	70.00	62.80	Pass
204000020	30.00	32.00	90.00	69.00	69.80	Yes	88.00	80.72	Pass - Distinction

```
<WRMS301>
        <Student>
                <StudentNumber>204006899</StudentNumber>
                <Marks>
                        <Tutorial1Mark>70</Tutorial1Mark>
                        < Tutorial2Mark>75</Tutorial2Mark>
                        <SemTest1>62</ SemTest1>
                        < SemTest2>77</ SemTest2>
                        <DPMark>70.10</ DPMark >
                        <DPObtained>Yes</ DPObtained >
                        <ExamMark>78</ ExamMark >
                        <TotalMark>74.84</ TotalMark >
                        <Passed>Pass</Pass>
                </Marks>
        <Student>
        ...
</WRMS201>
```

Figure 6.12: Conversion of Data from EXCEL Format to XML Format

The XML file contains only textual data, thus, requiring less storage space and less processing power when managing this file. The use of compression provides further increased efficiency when managing the transformed file.

6.5.3 Compression Component

Information gathered by the Intelligence Component is passed through as parameters for the Compression Component. The task of the Compression Component is simply to compress the data and media content provided by the Service Provider, and send the compressed content to the Service Client. In the initial tests conducted in Section 6.2, three compression techniques were selected for the compression of XML, image and audio files, respectively.

The prototype, however, is not limited to selecting these compression techniques. These compression techniques were chosen, based on the overall performance, to be implemented within the prototype as a proof-of-concept.

6.5.3.1 XML Compression

As discussed in Section 6.2.3.1, the DotNetZip compression technique was selected to be utilised for the development of the prototype. The DotNetZip compression technique is available as part of the DotNetZip library. In order to use the compression technique it is required to import the library to the application. DotNetZip allows the option of choosing the level of compression to use. However, better compression ratios require more compression time. It is, therefore, important to choose a compression level suited to the context of use, such as fast, low compression for smaller files, and slower compression for larger files yielded better compression ratios. For larger XML files, a slower compression was used, which achieved better compression ratios. The Backend Server Engine determines the level of compression to use based on the amount of information required for compression.

6.5.3.2 Image Compression

In Section 6.2.3.2, tests were conducted to decide which image compression technique performed the best. Based on the results of these tests, the JPEG compression technique was selected for use within the prototype. However, before the compression technique is used, the images are resampled, more specifically downsampled, based on the screen specifications of the mobile device. If the image is larger than the screen width or height, the resolution of that image would be reduced in order to fit within the screen dimensions. Downsampling of the image is conducted using a method so at to maintain the original aspect ratio[1].

Once the downsampling of an image is complete, the JPEG compression technique is invoked, using parameters from the Backend Server Engine. The parameters include the quality and colour depth to be used during compression. In addition, the quality and colour depth of an image will be lowered to improve compression levels for mobile devices with limited space available for storage.

[1] Maintaining the image aspect ratio during resampling involves the reducing the amount of pixels in an image using percentages, e.g. width x 10%, height x 10%.

6.5.3.3 Audio Compression

The MP3 compression technique was selecting in Section 6.2.3.3 to be utilised with the prototype. The parameters used to provide improved compression results include the quality (in bits/sec) and space available. For mobile devices with limited storage space, a lower bit-rate is used to further reduce the size of an audio file.

6.5.4 Decompression Component

The Decompression Component consists of two main subcomponents, the XML Decompressor, and Service Rebuilder. The goal of the Decompression Component is to return the compressed content to a format that is usable by the mobile application. Because of the popularity of the JPEG and MP3 compression techniques, most mobile devices provide built-in decompression mechanisms in order to process and read files compressed by these compression techniques. Thus, only compressed XML files were required to be decompressed in order to be utilised by the mobile application.

6.5.4.1 XML Decompressor

Due to the design of the proposed framework, different compression techniques may be used, and thus, require different decompression methods. In order to provide for this, the compression technique that was used during the compression process is loaded onto the mobile application (only once), in order to perform the correct decompression on the compressed files. This compression technique is transferred along with the compressed files.

For the prototype, the XML Decompressor utilises the DotNetZip compression technique to decompress the compressed XML files. Decompression is a simple task of reversing the compression procedure, which returns the XML files to their original size and state.

```
using (ZipFile zip = ZipFile.Read(ExistingZipFile))
{
  foreach (ZipEntry e in zip)
  {
    e.Extract(DestinationFolder, true);
  }
}
```

Figure 6.13: Decompression Method

Likewise to the import of the DotNetZip library to facilitate compression, it is required that the library be imported into the mobile application to initiate decompression. Figure 6.13 shows an example of using a foreach loop to traverse each entry within the compressed file in order to extract each entry to a specific destination folder.

6.5.4.2 Service Rebuilder

Files received by the mobile application are only usable if in the correct format. XML files are sent to the XML Decompressor, while all other files are sent directly to the Service Rebuilder subcomponent. After XML decompression is conducted, the XML files are sent to Service Rebuilder as well. Once all files are in a usable format, the Service Rebuilder prepares the files for use within the Service Viewer, i.e. the Service is recompiled to reflect the initial Service requested by the User.

6.6 Conclusions

This chapter provided a study to determine how a prototype, based on the proposed framework discussed in Chapter 5, may be implemented. This study provided an answer to research question R5: *"How can a prototype, based on the proposed framework, be implemented?"*

Before the prototype was implemented, a series of tests were conducted to determine the best compression techniques for each of the three types of files, namely XML, image and audio files. These tests concluded that the DotNetZip, JPEG and MP3 compression techniques were the best in terms of their overall performance and ease-of-use within the mobile environment.

A proof-of-concept prototype was developed once the initial tests were conducted. The implementation criteria were discussed in Section 6.3 to determine the environment in which the prototype was developed. Furthermore, the implementation domain discussed the scope of the research. The implementation tools were listed in Section 6.4, which provided a discussion on each of the tools used during the development of the prototype. The implementation of the four framework components were discussed in Section 6.5, which included a discussion about the interface, formulae, and database used for the prototype.

The development of the prototype demonstrated the successful implementation of the proposed framework and its individual components. The implemented prototype provides a means to evaluate the framework. In order to determine whether the prototype adheres to the design principles listed in Section 5.3.1, it essential to conduct a detailed evaluation. Additionally, an evaluation of the prototype will also determine if the perceived benefits of the proposed framework, discussed in Section 5.3.4, are realised. The evaluation of the prototype is discussed in Chapter 7.

Chapter 7: Evaluation and Findings

7.1 Introduction

Literature reviews were conducted in Chapters 2, 3 and 4. These chapters discussed Service-Oriented Architectures (SOA), Mobile SOA and File Compression, respectively. In Chapter 5, the proposed framework was presented. The chapter also discussed the design process, components, relevance and perceived benefits of the proposed framework. Chapter 6 provided a discussion of the implementation of a proof-of-concept prototype, which was based on the proposed framework, as well as a discussion on the implementation criteria and implementation tools for developing the prototype.

This chapter focuses on the evaluation of the proof-of-concept prototype described in Chapter 6, based on an evaluation strategy as presented in Section 7.2. Pilot studies, which were used to evaluate the prototype, in order to examine the feasibility and perceived overall efficiency, are discussed in Section 7.3. The analytical evaluation is then discussed in Section 7.4, which assesses whether the prototype was developed according the design principles described in Section 5.3.1. Finally, Section 7.4 discusses the performance evaluation of the prototype for both the Server and Client. The performance evaluation is the primary study within this chapter, which evaluates the efficiency of the prototype.

7.2 Evaluation Strategy

The Thesis Statement for this research, as introduced in Chapter 1, is as follows:

"The efficiency and effectiveness of Mobile SOA can be improved by the implementation of a framework supporting intelligent compression."

Consequently the main research objective is to *"improve efficiency and effectiveness of Mobile SOA."* In order to determine whether this objective was achieved, the following research questions need to be answered:

- **R5**: *"How can a prototype, based on the proposed framework, be implemented?"*
- **R6**: *"Does the prototype adhere to the framework design principles?"*

- **R7**: *"Does the prototype show that the efficiency of Mobile SOA can be improved through intelligent compression?"*

An evaluation strategy was designed in order to answer these questions. This strategy included four approaches, namely: pilot studies, proof-of-concept, analytical evaluation and performance metrics evaluation. This section describes the four evaluation phases, and the objectives of each phase.

7.2.1 Pilot Studies

Before the prototype was developed, three pilot studies were conducted to test the feasibility of implementing a prototype and efficiency of file transfer between the Server and Client for the proposed framework. These three pilot studies are described in Section 7.3.

7.2.2 Proof-of-Concept Prototype

The implementation of a prototype can be regarded as a proof-of-concept of the proposed framework (as discussed in Chapter 5) determining the effectiveness of the framework. The development of the proof-of-concept prototype answers research question R5: *"How can a prototype, based on the proposed framework, be implemented?"*

The proposed framework facilitates efficient file transfer of Web Services within the mobile environment. The prototype was implemented as a Client/Server application, which sends requests from a mobile device to a Service Directory. The requested Service then sends files to the mobile device via a Server, which intelligently compresses these files before sending the files to the mobile device, where the files are decompressed and used as a Service.

The implementation of the proof-of-concept prototype was described in Chapter 6.

7.2.3 Analytical Evaluation

The focus of the analytical evaluation is to determine whether the prototype is implemented according to the framework design principles discussed in Section 5.3.1. The analytical evaluation answers research questions R6: *"Does the prototype adhere to the framework design principles?"* By analysing the proof-of-concept prototype, it is possible to establish to what extent the prototype conforms to the design guidelines of the proposed framework. Section 5.3.1 highlights seven design principles to implement a prototype for the proposed framework. The prototype is tested against these principles in Section 7.4.

7.2.4 Performance Evaluation

The analytical evaluation was conducted to determine if the prototype was developed according to specific design principles. The performance evaluation was conducted to measure the overall performance of the prototype. This evaluation answers research question R7: *"Does the prototype show that the efficiency of Mobile SOA can be improved through intelligent compression?"*

The results from this evaluation is based on tests conducted using the proof-of-concept prototype. These results, and the analysis thereof, will be presented in Section 0. The evaluation of the prototype was divided into two parts: the testing of the Server performance and the testing of the Client performance. These tests were devised to test the efficiency of the prototype.

The next section discusses three pilot studies, which were conducted to determine the feasibility of developing a prototype based on the proposed framework.

7.3 Pilot Studies

The implementation of a prototype was based on an iterative process. This process included the analysis of the design principles of the proposed framework, the development of a number of prototypes, and the evaluation of these prototypes. The iterative process was used with the purpose of analysing the effectiveness of the proposed framework. This process involved developing a number of smaller prototypes, and conducting performance testing and evaluation in the form of pilot studies. Three pilot studies were conducted, which aimed to provide answers to the following questions:

- **PS1:** *"Is it feasible to use decompression within the mobile environment?"*
- **PS2:** *"Can a prototype provide efficient transfer of files for a Web Service?"*
- **PS3:** *"Can a prototype facilitate sustained usage on a mobile device?"*

These pilot studies are discussed in the following subsections.

7.3.1 Pilot Study 1: Evaluation of Compression used within Mobile Environment

Compression is the focus point of the development of a prototype. Thus, it is essential that the decompression procedures for the selected compression techniques can be executed within the mobile environment. Due to the standardisation and widespread use of the JPEG and MP3 compression techniques, most mobile devices provide features to utilise files stored as .jpg and .mp3 formats. In order to answer question PS1 a simple experiment was conducted. The goal of this pilot study is, therefore, to develop a prototype in order to determine whether XML decompression can be performed on a mobile device.

Three XML files were used of sizes 60KB, 1MB and 10MB. XML files were compressed on a desktop computer and copied into a folder used by the mobile emulator. A mobile application was implemented, which was used to execute the DotNetZip decompression process. All three files were successfully decompressed by the application. Thus, the test effectively verified the feasibility to use decompression within the mobile environment. Since this pilot study found that the decompression of XML files on a mobile device is feasible, a second prototype was implemented. The evaluation of this second prototype is discussed next.

7.3.2 Pilot Study 2: Evaluation of Framework using Mobile Emulator

Once it was established that compression is feasible for the development of a prototype, a second prototype was implemented. The second prototype was developed to evaluate whether the proposed framework, supporting intelligent compression, provides efficient file transfer to a mobile device. The goal of this pilot study is to determine the level of efficiency that compression yields when used within the proposed framework. This test was simulated using a mobile emulator provided by the Windows Mobile 6 SDK.

7.3.2.1 Procedure

An experiment was conducted in order to test the efficiency provided by the prototype. The experiment involved the requesting of data, audio and image files from a Server via a mobile application. These files were then compressed before being transferred to the mobile device. The experiment tested criteria, such as different screen sizes, image files of different sizes and types, different audio files, and XML files of different sizes and data stored. A total of 10 XML files were used, ranging from 10KB to 30MB in size, which were categorised according to file sizes. Each XML file was compressed on the Server using the DotNetZip compression technique. Six uncompressed audio files were used, from sizes 11KB to 9.5MB. Five images were used, varying from 24KB to 11MB in size. Three screen sizes were tested, which included a screen of size 128x128 pixels, 240x320 pixels and 480x800 pixels. These sizes were chosen to match typical screen sizes of Low-end, Medium-end and High-end mobile devices. These screen sizes were used as parameters for image resampling, before using the JPEG image compression technique.

To test every scenario, each compressed XML, audio and image file were linked. These compressed files were combined into a single file, called a package, resulting in a total of 300 different packages. These packages were then copied across to the mobile device emulator representing each screen size. Once these packages were transferred, the files within the packages were extracted. As a result of this extraction, the XML files were decompressed.

7.3.2.2 Results

A mean size of the uncompressed XML files was recorded, which includes the mean compression size that was recorded for each of the three categories. The sizes of the audio and image files were recorded. The mean compressed size of the XML files were summed with each of the five compressed images and six compressed audio files.

Table 7.1: Mobile Emulator Compression Results

		Screen Size (128x128)		**Screen Size (240x320)**		**Screen Size (480x800)**	
	Without Compression	With Compression		With Compression		With Compression	
	Size (KB)	**Size (KB)**	**Compression Time (sec)**	**Size (KB)**	**Compression Time (sec)**	**Size (KB)**	**Compression Time (sec)**
Category 1	283	42	0.652	43	0.652	49	0.653
	554	85	0.751	80	0.745	135	0.747
	1 979	66	1.251	71	1.341	90	1.346
	5 400	178	0.661	202	0.745	304	0.649
	17 288	559	2.793	575	2.844	653	2.851
Category 2	3 501 859	421	1.009	422	0.995	428	0.822
	3 502 416	408	1.547	403	1.544	458	1.544
	3 504 773	517	0.649	522	0.473	541	0.831
	3 511 214	902	2.986	926	2.987	1 028	2.984
	3 522 722	1 250	4.121	1 266	3.847	1 344	3.975
Category 3	20 589 454	20 118	2.388	20 110	2.345	20 125	2.385
	20 590 011	20 144	2.965	20 108	3.189	20 194	3.181
	20 592 368	20 253	1.497	20 116	1.934	20 277	1.985
	20 598 809	20 638	4.281	20 136	4.281	20 764	4.481
	20 610 317	20 986	4.958	20 125	5.152	21 080	5.181

Table 7.1 shows a summary of the compression results, which includes the mean size for the three XML categories summed with randomly selected compressed audio and image files. Overall, XML files were compressed to a mean of less than 12% of their original file size. Audio files were compressed to an overall compression ratio of less than 10%. Image

compression yielded the best results with a mean compression ratio of less than 6.5%. The combined packages resulted in a joined mean compression of less than 11%.

Table 7.2 tabulates the expected minimum transfer times of the three most popular wireless technologies used by mobile devices today. These transfer times were recorded by calculating the time it would take to transfer a file of a specific size at the maximum transfer rate[1], in Kilobits per second. For example, the maximum transfer rate for GPRS is 114 Kilobits per second, which equates to 14.25 Kilobytes per second. Thus, a file of 10 Kilobytes would be transferred in 0.702 seconds.

Table 7.2: Expected Minimum Transfer Times (sec)

	GPRS	**EDGE**	**HSDPA**
	114 Kbps	up to 236.8 Kbps	up to 14.4 Mbps
File Size	**File Transfer Speed (sec)**		
10KB	0.702	0.338	0.043
100KB	7.018	3.378	0.434
1MB	71.860	34.595	4.444
10MB	718.596	345.946	44.444
100MB	7185.965	3459.459	444.444
1GB	73584.281	35424.865	4551.111

Using the results from both Table 7.1 and Table 7.2, It is evident that the packages, containing compressed content, take significantly less time to transfer, even with the time taken to conduct compression and decompression. For example, using a GPRS connection, the most common wireless technology, a 5MB file would take approximately 360 seconds to be transferred to a mobile device. However, the same file would take a total of 43.68 seconds to compress, transfer via a GPRS connection, and decompress on a mobile device.

Table 7.3 is generated by calculating the transfer times of the compressed and uncompressed file sizes for each of the three wireless technologies. This table provides a comparison between uncompressed and compressed files transferred over the selected wireless technologies. Using the maximum transfer rates of these wireless technologies, the transfer time is recorded (in seconds). Three example packages, containing compressed content, were

[1] Note: Maximum transfer rates of any wireless technology are not often achieved by network providers due to a number of factors, such as the distance between the Mobile Device and Cell Tower, the number of erroneous packets during file transfer, and Cell Tower usage capacity.

chosen, which had file sizes of 620KB, 5MB, and 30MB, respectively. These packages were randomly selected.

It is evident that the packages, containing compressed content, take significantly less time to transfer, even with the time taken to conduct compression and decompression. For example, using a GPRS connection, the most common wireless technology, a 5MB file would take approximately 360 seconds to be transferred to a mobile device. However, the same file would take a total of 43.68 seconds to compress, transfer via a GPRS connection, and decompress on a mobile device.

Table 7.3: Time Taken to Transfer Original and Compressed Files

	Original File Transfer Speed (sec)			**Compressed File Transfer Speed (sec) [including compression and decompression time]**		
	GPRS	**EDGE**	**HSDPA**	**GPRS**	**EDGE**	**HSDPA**
File Size (KB)	56-114 Kbps	56-236.8 Kbps	up to 14.4 Mbps (avg 1.8Mbps)	56-114 Kbps	56-236.8 Kbps	up to 14.4 Mbps (avg 1.8Mbps)
620	44.070	21.216	2.726	6.347	3.508	1.212
5 120	359.298	172.973	22.222	43.680	21.772	4.027
31 744	2155.789	1037.838	133.333	221.161	111.884	23.407

A third pilot study was conducted to test the battery life of two mobile devices using the prototype. This study is discussed next.

7.3.3 Pilot Study 3: Evaluation of Battery Life of Mobile Device

Based on the positive results of the first and second pilot studies, a third pilot study was conducted. This pilot study was conducted to determine the effect of prolonged use of the prototype on a mobile device. The goal of this pilot study is, therefore, to establish whether the prototype can support sustained used on a mobile device.

7.3.3.1 Procedure

In order to achieve the goal of this pilot study an experiment was conducted. This experiment tested the battery life of a mobile device while performing decompression on a compressed file. A random file was selected from each of the three categories of XML files, as discussed in Section 6.2.1.1. The uncompressed files were of sizes 180KB, 3.3MB and 31MB. Each of these files was compressed individually to sizes 49KB, 987KB and 3.1MB, respectively.

Two mobile devices were used, the HTC s170 and HTC TyTN II, as described in Section 6.4.3. One thousand duplicates of each of the files were created and loaded onto the two mobile devices separately. After each of the mobile device's battery was fully charged, the files were then decompressed on the mobile device using the prototype. A log file was created, which recorded the time taken to decompress the given file, the percentage of battery life remaining after each file was decompressed, and the total number of files that were decompressed before the battery charge was exceeded.

7.3.3.2 Results

A log file was created after the three tests, for each group of file, were conducted for both mobile devices. Table 7.4 provides a summary of the results recorded from the log files. The table illustrates the mean decompression time, the number of files decompressed, and the battery charge remaining, either after the 1000 files were decompressed or the battery charge was depleted. For both the small and medium sized files, both mobile devices successfully completed the decompression of 1000 files. For large sized files, however, both mobile devices failed to decompress all the files.

Table 7.4: Mobile Decompression Results

	Small Sized File			**Medium Sized File**			**Large Sized File**		
Mobile Device	Mean Decompression Time (sec)	# files	% left	Mean Decompression Time (sec)	# files	% left	Mean Decompression Time (sec)	# files	% left
HTC S710	4.4	1000	32%	7.6	1000	25%	43.7	150	0%
HTC TyTN II	3.1	1000	86%	5.4	1000	81%	28.2	750	1%

Due to the faster CPU clock speeds and overall superior specifications, the HTC TyTN II outperformed the HTC s710 in all tests. The HTC s710 did manage to decompress the total 1000 files for small and medium sized files. It is unlikely that, within the course of a working week, one individual will require a total of 1000 file decompressions. This suggests that even the more basic mobile devices are capable of performing numerous file decompressions. Thus, the use of the prototype, which uses decompression on a mobile device, can be sustained over a period of time before the battery requires recharging.

Based on the results achieved in these three pilot studies, a final prototype was developed, as described in Chapter 6. The analytical and performance evaluations of this prototype are discussed in Section 7.4 and 0, respectively.

7.4 Analytical Evaluation

The design of the proposed framework was discussed in Chapter 5. The implementation of a prototype, described in Chapter 6, provides an answer to research question R5: *"How can a prototype, based on the proposed framework, be implemented?"* This section provides an analytical evaluation to determine whether the prototype adheres to the design guidelines, as discussed in Section 5.3.1. The analytical evaluation is presented to show how the prototype adheres to the following seven design principles:

- **Easy and Lightweight Execution:** The framework should be lightweight and easy to execute in practice, robust, and provide easy access to Services (Shretha 2010).
- **Compatibility:** The framework must work for different Services and be compatible for different devices, operating systems and programming languages. This is of particular importance to the mobile environment, in which numerous mobile devices exist.
- **Robustness and Concurrency:** The framework must be robust, accurate, and error-free. Additionally, the framework should support multiple, concurrent access to requested Services. This feature should not, however, be detrimental to the performance of the framework.
- **Interchangeable Communication Medium:** Although not limited to the mobile environment, the framework must support different communication mediums for mobile devices, as listed in Section 3.3.3. The framework should perform equally in differing bandwidths and communication protocols from which the Service was requested.
- **Service Discovery:** The framework should support discovery of new Services via a specific Service Discovery method defined by the organisation or vendor that implements a working prototype of the framework.
- **Efficiency:** The framework must provide a means for efficiently managing the information produced by requested Services. The Service itself is external and autonomous. The implementation and operation of these Services is irrelevant to the performance the framework yields.
- **Flexibility:** The framework should fit into the existing SOA environment of an organisation. This removes the need to redesign or redevelop any existing applications used within the organisation.

Existing technologies continue to improve, while newer technologies are discovered. It is, therefore, essential that the implemented prototype is able to provide support for these technologies. In order to provide this support the prototype must adhere to the design principles listed above. The following subsections provide discussions in order to illustrate how the prototype was implemented according to the seven design principles.

7.4.1 Easy and Lightweight Execution

The proof-of-concept prototype was implemented using the tools described in Section 6.4. The combined use of the selected tools allowed for the individual implementation of the four components of the proposed framework, which were integrated into the prototype. Additionally, a total of 80 hours was spent implementing the prototype. The implementation tools used, the integration of the four components, and the amount of time spent during implementation meant that the proof-of-concept prototype was *easy* to execute.

Lightweight execution, in this context, refers to the amount of resources used to execute the prototype. During the execution of the prototype, the amount of resources used was insignificant, on both the Server and Client. The resource usage is discussed as part of the performance evaluation in Section 0. The prototype may, thus, be considered *lightweight*.

7.4.2 Compatibility

The implementation of the prototype was divided into the four components of the proposed framework, namely the User Interaction Component, the Intelligence Component, the Compression Component and the Decompression Component. The Intelligence and Compression Components are Server Side implementations, whereas the User Interaction and Decompression Components are (Mobile) Client Side implementations.

The manner in which the Server Side components were implemented, the tools which were used during implementation, as well as the deployment of the prototype (Section 6.4) allows it to be accessed from any application, independent of its implementation tools. The mobile application for the Client Side was implemented for the Windows Mobile operating system. Therefore, the mobile application is not compatible for use on other mobile operating systems. However, the implementation of a mobile application for a particular operating system is, generally, not compatible with any other operating system.

7.4.3 Robustness and Concurrency

After the implementation of the proof-of-concept prototype, no errors or system crashes were observed. The results recorded during the performance evaluation (more specifically in Table 7.7 and Table 7.8) shows that a task completion rate of 100% was observed for each of the three Web Services tested. Thus, the prototype is robust, accurate and error-free.

Two mobile devices were utilised to determine whether the prototype can support concurrent use. These mobile devices used were the HTC s170 and HTC TyTN II. Although an ideal test would require numerous amounts of different mobile devices being used at the same time, the test conducted confirmed that the prototype does support concurrent use. The Server Side implementation of the prototype can support use of more than 1000 requests, which is supported by IIS 7, MS SQL Server 2008 and MS Windows Server 2008. Thus, concurrent use of the prototype is possible.

7.4.4 Interchangeable Communication Medium

The use of the prototype within the mobile environment means that the mobile device using it is required to connect via any wireless technology available to that mobile device. Three wireless technologies were tested, namely GPRS, EDGE and Wi-Fi. When using the prototype, each of the three wireless technologies were able to efficiently transfer files between the Server and Client. The only difference in performance was as a result of the transfer rates provided by the three wireless technologies.

7.4.5 Service Discovery

The use of a WSDL (Section 2.5.1.3) provides sufficient information to describe the interface and semantics of a Service. Additionally, the information provided by a WSDL is used to classify a Service, allowing Service Consumers to search for different Services. This information is known as meta-data, on which search criteria, such as keywords, are based.

Similarly, the Services, which were used to evaluate the proof-of-concept prototype, provide information in the form of meta-data, in which keywords were stored. The search functionality provided by the mobile application, as illustrated in Figure 6.8 (b), searches the meta-data for matching keywords. Services were accessed by means of a simple call request. It can, therefore, be concluded that the proof-of-concept prototype provides support for Service Discovery.

7.4.6 Efficiency

The implementation of a prototype used for Pilot Study 2 illustrated the level of efficiency provided. The Pilot Study demonstrated that the use of compression significantly reduced the storage and transfer sizes of XML, audio and image files, and decreased the time taken to transfer these file. The reduction of storage and transfer size, and the time taken to transfer the file resulted in less processing required to manage these files, which reduced the amount of battery consumed during these processes. XML files were compressed to a mean of less than 12% of their original file size, audio files to less than 10%, and image files to less than 6.5%. Overall, a combined compression ratio of less than 11% was yielded. Additionally, the Pilot Study illustrated the potential reduction in costs, which may be incurred during file transfer.

The efficiency of the proof-of-concept prototype is discussed in more detail in Section 0.

7.4.7 Flexibility

The flexibility of the prototype was not tested in an already existing SOA environment. However, the tools which were used to implement the prototype provide flexibility, in that the developed system may provide seamless integration. These tools included IIS 7, MS SQL Server 2008, MS Windows Server 2008, and MS Visual Studio 2008, in which the Windows Mobile SDK was used.

7.4.8 Summary of Analytical Evaluation

The analytical evaluation of the proof-of-concept prototype answered research question R6. In addition, the analytical evaluation provided evidence that the prototype was developed according to the design principles discussed in Section 5.3.1. For the cases in which tests were not extensively conducted, the implementation tools provided sufficient motivation to suggest that the specific design principle was adhered to. Based on the analytical evaluation conducted in this section, it can, therefore, be concluded that the proof-of-concept prototype does adhere to all the framework design principles.

Additionally, the successful implementation of the proof-of-concept prototype demonstrated the effectiveness of the proposed framework for the Mobile SOA environment. By implementing the prototype according to the framework design principles, it demonstrated that the prototype was easy and lightweight to execute, was compatible for the mobile environment, supported concurrent use and different communication mediums, was robust, provided Service Discovery functionality, and was flexible.

The performance of the proof-of-concept prototype is evaluated in the following section.

7.5 Performance Evaluation

The performance of a system is the degree to which it meets its goals (Browne 1976). Performance evaluation is the measurement and analysis of a system's performance. It is often used to analyse a working system, from which enhancements are made to the performance of the system, based on the results of the evaluation. The performance evaluation, however, is used as a means for determining the level of efficiency that the prototype provides. Efficiency, in terms of the prototype, is measured using specific performance metrics.

Many definitions of a performance metric exist. To avoid confusion in the meaning of a performance metric, a performance metric is defined as: a quantifiable measurement that denotes a specific performance value (Deru & Torcellini 2005). Performance metrics were used to measure and analyse the performance of the final prototype. Based on research (Palmer 2002; Canfora & Troiano 2002; Mirchandani 2001; Spanos 2009), the following categories of performance metrics were used:

- **Response Time:** In order to determine whether the prototype reduces the amount of time taken to request a Service, it is essential that the response time is measured. The response time is measured by recording the compression time of the Server Side, and the decompression time on the Client Side. The file transfer time is also recorded.
- **Resource Usage:** Although the prototype may reduce the time taken to request a Service, it is required that the amount of resources used during this process be calculated. The resource usage is measured by recording the memory and processing power used by both the Server and Client Side. The amount of space saved is determined by the recorded compression ratio, which also indicates the amount overhead saved during file transfer. Additionally, the battery usage is recorded on the Client Side.
- **Reliability:** As with all systems, reliability is of great importance. Reliability is measured by means of recording the task completion rating. The task completion rating is the number of successfully completed tasks divided by the total number of tasks.
- **Costs:** The cost incurred while using the prototype to request Services is compared against the cost incurred when not using the prototype. Data transfer rate per Megabyte is used to measure the cost.

The main objective of the performance evaluation is to determine the efficiency of the prototype. The efficiency of the prototype is determined by the overall time that has elapsed from the initial Service request to the delivery of those Services, the amount of resources saved, and the reduction of overall costs incurred.

7.5.1 Test Procedure

There are two types of Web Services available, Content Services and Added Value Services. Three Web Services were implemented in order to evaluate the performance of the prototype (Section 6.3). These Web Services were requested via the two testing mobile devices (Section 6.4.3). During this process, performance data was recorded on both the Server and Client Side of the prototype.

Three separate tests were conducted for each of the Web Services; that is to say that a total of nine Service requests were made to test the prototype. The results from these tests were collected, from which the mean was calculated for each metric per Web Service. The data that was captured during the tests conducted were stored as Comma Separated Values (CSV) files.

The resulting data was then analysed to determine the level of efficiency provided by the prototype. The compressed files were sent via Wi-Fi from the Server to the Client (mobile device). During the testing procedure, the initial registration by the Client was not evaluated. Table 7.5 illustrates the types of file used within each of the three Web Services tested during the performance evaluation.

Table 7.5: Web Service File Types

Web Service	XML File	Image File	Audio File
Web Service 1	✓	×	×
Web Service 2	×	✓	✓
Web Service 3	✓	✓	✓

Due to the implementation of the proof-of-concept prototype, there may be more than one Client, but only one Server. It is, therefore, important to individually measure the performance of the Client and Server. Thus, the performance evaluation of the prototype is divided into two categories, the Server performance and the Client performance.

7.5.1.1 Server Performance

The Intelligence and Compression components reside on the Server. These two components provide the primary factors, which determine the level of efficiency provided by the prototype, in terms of the compression techniques chosen and the level of compression yielded by these compression techniques. The performance metrics recorded to measure the efficiency of the Server are listed as follows:

- **Compression Time:** The compression time is the total time taken to compress all the files required by the requested Service. Compression time is measured in the amount of seconds that have elapsed during compression.
- **Compression Ratio:** Compression ratio is the compressed file size of a file divided by the original size of the file. In order to calculate the compression ratio, the original file size and compressed file size are also recorded. The compression ratio is measured as a percentage.
- **Processing Power:** This is the amount of processing power that is used during the entire compression process, i.e. the processing power used by both the Intelligence and Compression components. Processing power is measured as a percentage of CPU usage for a specific time slot. The processing power used by the prototype is calculated as the overall processing power in use subtracted by the mean idle processing power.

These results of the Server performance are used to determine the level of efficiency of the prototype on the Server Side.

7.5.1.2 Client Performance

Although the efficiency of the prototype is largely determined by the level of performance of the Server Side, the overall performance and efficiency also depends on the Client. The information provided during registration and the Decompression Component on the Client Side play a pivotal role in the overall efficiency of the prototype. Thus, the second performance evaluation was conducted to determine the performance of the prototype on the Client Side (mobile device).

- **Decompression Time:** The decompression time is based on the time taken to decompress all the compressed files transferred to the Client. Decompression time is measured in the amount of seconds that have elapsed during the decompression process.

- **Memory Footprint:** The memory footprint is determined by the amount of memory used by the application subtracted by the amount of memory that was allocated to the application for a specific process. Memory footprint is recorded in the performance evaluation in order to determine the amount of resources used by the prototype.
- **Battery Consumption:** During the process of requesting a Service, receiving the compressed files associated with that Service, decompressing these files and then displaying the requested Service all require some level of processing power. In order to use the processing power available on a mobile device, it is required that some battery power be utilised. The amount of battery life is recorded before the Service is requested, and again once the Service is displayed. The amount of battery power used is measured as a percentage.
- **File Transfer Time:** This is the amount of time that has elapsed while transferring a file from the Server to the Client. File transfer time is measured in seconds, for which a value recorded implies a better transmission time. In addition to the transfer time recorded for Wi-Fi, the minimum transfer time for GPRS, EDGE and HSDPA, using the theoretical maximum transfer rate, was recorded and tabulated.
- **Task Completion Rate:** In order to determine the reliability of the prototype, task completion rate is recorded. A task is successfully completed once the requested Service is accurately displayed on the Client Side. Task completion rate is measured as the percentage of the amount of tasks completed divided by the amount of tasks conducted.
- **Cost:** This is the potential cost that may be incurred during the file transfer from the Server to the Client and is measured in Rands. These costs are based on the rate charged, in cost per Megabyte, for GPRS, EDGE and HSDPA.
- **Overall Time:** The overall time is recorded, which is the time that has elapsed from when a Service is requested up to when it is displayed on the mobile device. This time was recorded for a Wi-Fi connection, and is measured in seconds.

These results of the Client performance are used to determine the level of efficiency of the prototype on the Client Side. The recorded results of all metrics and analysis thereof are discussed in the following subsections.

7.5.2 Results

The results of the evaluation are discussed in relation to the performance metrics measured during the conducted evaluation. These recorded results are presented in a tabular format. The results are categorised into the performance metrics for the Server and the (Mobile) Client. The full list of results, recorded for the performance evaluation, is shown in Appendix G.

7.5.2.1 Server Performance Results

The performance of the Server is dependent on its two components, the Intelligence Component and the Compression. Although the Intelligence Component played a critical role in deciding which compression parameters to use, it was the Compression Component that utilised most of the resources available on the Server.

Table 7.6 illustrates the performance results on the Server Side for each of the three requested Web Services. Although the original size and compressed size were recorded, they were used only to calculate the compression ratio yielded during the compression process. The compression ratio, compression time and processing power are the performance results obtained by the Compression Component. The overall server time is recorded throughout the entire process on the Server Side.

Table 7.6: Server Side Performance Results

Metric	Web Service 1	Web Service 2	Web Service 3
Original Size (KB)	*112*	*9609*	*7439*
Compressed Size (KB)	*7.950*	*1105.900*	*576.000*
Compression Ratio (%)	7.098%	11.509%	7.743%
Compression Time (sec)	0.360	3.403	2.853
Processing Power (%)	12.927	52.150	57.761
Overall Server Time (sec)	0.405	3.670	3.026

A significant compression ratio was yielded for each of the three Web Services used during the evaluation. Web Service 1 yielded a compression ratio of 7.098%, Web Service 2 a compression ratio of 11.509%, and Web Service 3 a compression ratio of 7.743%.

For Web Services 2 and 3, it was required that at least one audio file be compressed. As was indicated during the initial tests conducted in Section 6.2, the audio compression process yielded considerably higher compression times compared to that of XML and image compression. Therefore, a higher compression time, and, consequently, a higher overall time, was observed when requesting Web Service 2 and 3, as they both contained audio files.

During the compression, on the Server Side, the processing power was recorded. The processing power is recorded as the percentage of CPU usage for a specific time slot, in which the compression was conducted. Web Service 1 yielded processing power of 12.927%, 52.150% for Web Service 2, and 57.761% for Web Service 3. Based on the overall server time, the percentage of processing power was used for a maximum of 0.405, 3.670, and 3.026 seconds for Web Services 1, 2, 3, respectively.

7.5.2.2 Client Performance Results

The performance of the Client is dependent primarily on the Decompression Component. The User Interaction Component serves the purpose of registration, requesting a Service and displaying a Service. The initial registration was not evaluated.

Table 7.7 illustrates the performance results recorded for each of the three requested Web Services on the HTC s710 Client. It is important to note that no decompression time was recorded for Web Service 2, as only image and audio files were compressed on the Server Side for this request. Image and audio files did not require a separate decompression process, as the mobile devices utilised both support the JPEG and MP3 standards used for compression.

Table 7.7: HTC s710 Performance Results

Metric	Web Service 1	Web Service 2	Web Service 3
Decompression Time (sec)	6.985	NA	5.041
Memory Footprint (MB)	0.076	0.084	0.086
Battery Consumption (%)	0.093%	0.040%	0.066%
Task Completion Rate (%)	100%	100%	100%
Overall Time (sec)	10.561	7.334	10.957

Table 7.8 shows the performance results recorded for each of the three requested Web Services on the HTC TyTN II Client. Once again, no decompression time was observed for Web Service 2. A task completion rate of 100% was observed for both Clients over all Service requests. Due to the improved decompression time for the HTC TyTN II over the HTC s710, a better overall time and a lower battery consumption percentage was observed.

Table 7.8: HTC TyTN II Performance Results

Metric	Web Service 1	Web Service 2	Web Service 3
Decompression Time (sec)	4.614	NA	3.781
Memory Footprint (MB)	0.078	0.081	0.084
Battery Consumption (%)	0.031%	0.011%	0.037%
Task Completion Rate (%)	100%	100%	100%
Overall Time (sec)	8.846	6.977	9.562

Decompression times of 6.985 and 5.041 seconds were observed on the HTC s710 for Web Service 1 and 2, respectively. For the HTC TyTN II, decompression time of 4.614 and 3.781 seconds were observed for Web Service 1 and 2, respectively. These decompression times were used in conjunction with the compression and transfer times in order to calculate the overall time when using the prototype to request a Service, in comparison to requesting the Service using standard methods. These transfer times are discussed later in this section.

The memory footprint recorded during the processing of all three Web Services on both mobile devices was insignificant, with no decompression process requiring more than 0.086MB of processing memory. The results recorded for the memory footprint, for all three Web Services, verified the results obtained for the memory footprint in Section 6.2.2.1.

The battery power consumed, on both mobile devices, during the entire process was negligible. The percentage of battery power consumed for each test conducted was less than 0.1%. For example, the maximum battery power consumption yielded was during the decompression of Web Service 1 for the HTC s710. A percentage of 0.093% was observed. This means that the same procedure may be conducted approximately 1000 times before the battery requires recharging. This further demonstrates the insignificant amount resources used by the prototype on both the Server and Client.

Additionally, a task completion rate of 100% was observed for all tests conducted on both mobile devices, as illustrated in Tables Table 7.7 and Table 7.8. The task completion rate demonstrated the level of effectiveness of the proof-of-concept prototype for the Mobile SOA environment.

The transfer times for the uncompressed files for each of the three Service requests are tabulated in Table 7.9. This table is provided in order to compare the transfer times between the uncompressed files, and the compressed files for the HTC s710 and HTC TyTN II for three wireless technologies, GPRS, EDGE and HSDPA. In conjunction with Table 7.10, this table is used to compare the difference in transfer time for compressed and uncompressed files for each of the three Web Services.

Table 7.9: Uncompressed File Transfer Times

	File Transfer Rate (sec)			
	Original File Size (KB)	GPRS	EDGE	HSDPA
Web Service 1	112	7.860	3.784	0.486
Web Service 2	9609	674.316	324.628	41.706
Web Service 3	7439	522.035	251.318	32.287

The transfer times for the compressed files for both the HTC s710 and HTC TyTN II are tabulated in Table 7.10. The total transfer time includes the time taken to compress the files on the Server Side, the time taken to transfer the compressed files over the three wireless technologies and the time taken to decompress the requested Service files on the respective Clients.

Table 7.10: Mobile Device Transfer Times

		HTC s710 Transfer Rate (sec) [includes compression and decompression time]			HTC TyTN II Transfer Time (sec) [includes compression and decompression time]		
Web Service	**Compressed File Size (KB)**	**GPRS**	**EDGE**	**HSDPA**	**GPRS**	**EDGE**	**HSDPA**
Web Service 1	8	7.546	7.255	7.020	5.175	4.884	4.649
Web Service 2	1105	77.382	32.331	4.796	77.165	31.861	4.662
Web Service 3	576	45.462	24.500	7.541	44.202	23.240	6.281

In Table 7.10, a better transfer time was observed for Web Service 1, which compresses an XML file, only when files were transferred using GPRS. Based on this result, it would be more practical to transfer files of sufficiently small size for faster wireless technologies, such as EDGE and HSDPA, without having to compress the files first. A higher cost, however, would be incurred during the file transfer. The transfer times for both the compressed files of Web Service 2 and 3 are considerably quicker than when transferring uncompressed files for these Services.

Figure 7.1 illustrates the potential transfer costs[1] for each of the three Web Services transferred. The potential costs incurred when transferring the uncompressed Web Services are compared to the potential costs incurred when transferring the compressed Web Services. The results are based on the mean cost per megabyte of data downloaded across four network operators. The potential cost incurred per megabyte for each of the three wireless technologies (GPRS, EDGE and HSPDA) are, generally, the same. Based on the results provided in Figure 7.1, it is evident that compression significantly reduces the costs incurred during file transfer.

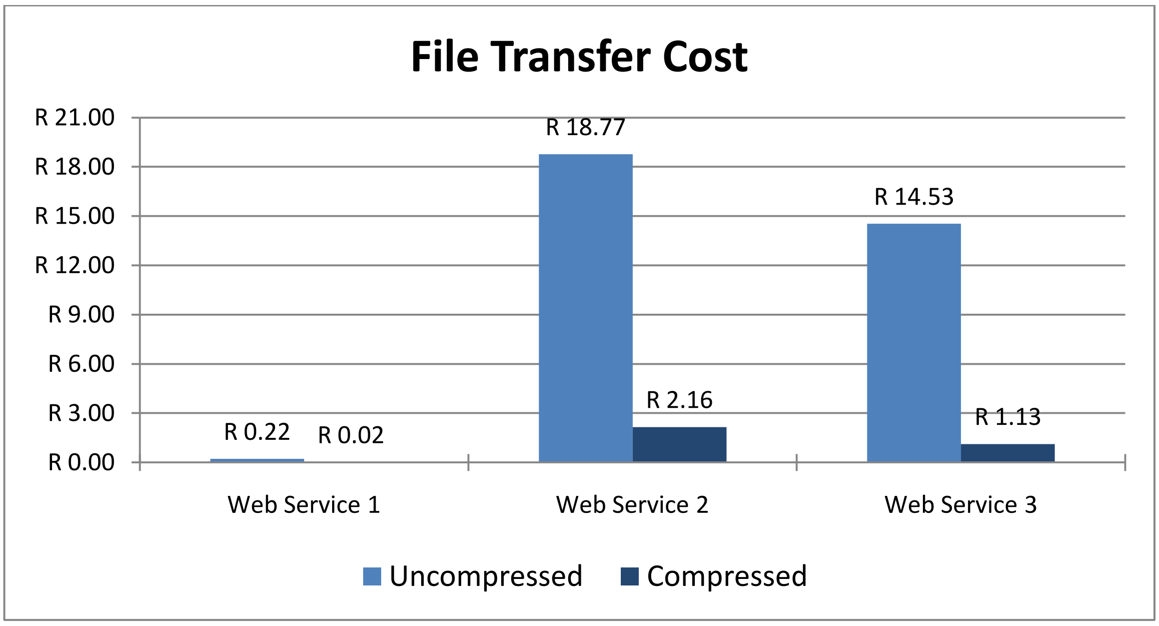


Figure 7.1: Comparison in Costs Incurred During File Transfer

For Web Service 1, the transfer of uncompressed files incurred a potential cost of 22c, whereas the transfer of compressed files would, potentially, cost only 2c. A potential cost of R18.77c was observed for uncompressed files for Web Service 2, compared to R2.16c for compressed files of the same Web Service. Similarly, the potential cost of transferring files for Web Service 3 was observed, with a potential cost of R14.53c observed for uncompressed files, and R1.13c for compressed files. The possible reduction in costs, when using the prototype to transfer files of a Service between the Server and (Mobile) Client, are significant, especially after numerous amounts of file transfers. This is of particular importance in the business environment, where large amounts of data are transferred and reducing costs is of great benefit.

[1] Cost is based on pre-paid, out-of-bundle usage. The costs for subscriptions and data bundles vary significantly and, thus, were not used.

The summary of the performance evaluation and the analysis of the results recorded for both the Server and Client are discussed next.

7.5.3 Summary of Performance Evaluation

Section 0, including its subsections, discussed the performance metrics used during the performance evaluation of the prototype. Although the evaluation process was not divided into two separate processes, the results from the evaluation were divided according to the performance of the Server Side and Client Side of the prototype.

The results from this evaluation process validated the preliminary results yielded by the pilot studies, which showed that the prototype provides efficient transfer of a Web Service to a mobile device. It was observed (from Tables Table 7.9 and Table 7.10) that uncompressed files of sufficiently small sizes are transferred quicker over a wireless network. This, however, is realised with a higher cost incurred during file transfer.

The compressed files produced by the prototype facilitate efficient transfer to a mobile device, not only in the reduction of cost and time, but also in the amount of resources expended on the mobile device. The decrease in file transfer time and reduction of storage space required implies that a reduction in processing power and time are required, which, in turn, entails that the consumption of battery power is minimised.

The performance evaluation demonstrated, by means of the results yielded, the level of efficiency provided by the prototype within the Mobile SOA environment.

7.6 Conclusions

The objective of this chapter was to answer research questions R6: *“Does the prototype adhere to the framework design principles?”* and R7: *“Does the prototype show that the efficiency of Mobile SOA can be improved through intelligent compression?”* These questions were answered by means of an evaluation strategy, which consists of three pilot studies, an analytical and performance evaluation.

The three pilot studies were conducted to determine the feasibility of developing a prototype. These tests were conducted on the Windows Mobile emulator. The results from the first pilot study verified the feasibility of using compression in the mobile environment, by showing that decompression on a mobile device is viable.

A second prototype was implemented as part of the second pilot study. This study was conducted to determine the level of efficiency of the prototype using compression. Results suggest a significant reduction in file size, thus, reducing the amount time taken to transfer files, and decreasing cost and storage space.

The third pilot study was conducted to determine the sustainability of using the prototype on a mobile device by testing the battery life. The tests were conducted on two mobile devices. These tests demonstrated that performing decompression on a mobile device by using the prototype has a minimal effect on the battery charge and can support multiple file decompressions before the battery requires recharging.

The analytical evaluation of the prototype was conducted to determine whether implementation of the proof-of-concept prototype adheres to the design principles discussed in Section 5.3.1. These design principles are listed as the following:

- Easy and Lightweight;
- Compatibility;
- Robustness and Concurrency;
- Interchangeable Communication Medium;
- Service Discovery;
- Efficiency; and
- Flexibility.

Each of the seven design principles were evaluated separately. The results from the analytical evaluation provided sufficient evidence to deduce that the prototype was implemented according to the design principles. Furthermore, the implementation of the proof-of-concept prototype verified the effectiveness of the proposed framework, which supports intelligent file compression, for the Mobile SOA environment.

The performance evaluation was conducted to determine the level of efficiency provided by the prototype. The results from the evaluation were divided into two categories, the Server Side performance and the Client Side (mobile device) performance. Performance metrics for both the Server and Client were recorded. The results illustrated a reduction in the overall file transfer time, processing power and time, and battery consumption on the mobile device, and costs incurred during file transfer.

The file transfer rate for uncompressed files of sufficiently small size is much better for the faster wireless technologies, such as EDGE and HSDPA, than when the same files are compressed on the Server, transferred via these wireless mediums and decompressed on the mobile device. Higher costs are incurred as a result of transferring the uncompressed files. It is, thus, important to determine the cut off point when uncompressed files are transferred more quickly across a wireless network, for which the cost difference is not significant. Additionally, the amount of resources consumed when transferring uncompressed files is also greater, which could result in the battery power being exhausted at a higher rate. The results recorded for the performance evaluation, however, suggest an overall improved efficiency for requesting, transferring and viewing Web Services on a mobile device.

The results of the analytical evaluation confirm the effectiveness of the prototype within Mobile SOA. The results of the performance evaluation demonstrated the efficiency of the prototype. Thus, based on the analytical and performance evaluation, it may be concluded that the proposed framework is effective in optimising the effectiveness and efficiency of Mobile SOA.

The dissertation is concluded in Chapter 8. The contributions, problems encountered and achievements are discussed, and further research and recommendations are also identified.

Chapter 8: Conclusions, Findings and Recommendations

8.1 Introduction

Due to the benefits of using Service-Oriented Architecture (SOA), such as application and data integration, versatility, code reuse, platform independency and cost saving, SOA has had rapid growth in recent years. The growth of SOA has been met with enormous acceptance. This is evident in the business environment, as it is calculated that 80 percent of mission critical applications, together with business processes, will be implemented around SOA by the end of 2010 (Chattpar 2008).

The rapid growth in the number of connected mobile devices has resulted in its inevitable acceptance as the standard platform for business applications. With increasing utilisation of both SOA and mobile devices, it is not surprising that Mobile SOA has become a new sought after technology. As with all new technologies, however, Mobile SOA has a number of limitations and challenges.

The aim of this research was to design a framework, which would support effective and efficient transfer of files commonly used within Services. A proof-of-concept prototype was implemented and evaluated in order to determine whether the proposed framework *"optimises the effectiveness and efficiency of Mobile SOA."*

Chapter 2 provided a discussion on Service-Oriented Architectures (SOA) and its enabling technologies. A discussion on Mobile SOA (a subset of SOA) and its enabling technologies was provided in Chapter 3. File Compression was described in Chapter 4, which entailed a discussion on different compression approaches for three categories of files commonly used within Mobile SOA. In Chapter 5, the design process, components, relevance and perceived benefits of the proposed framework were discussed. Chapter 6 described the implementation process of the proof-of-concept prototype, which was based on the proposed framework. The evaluation of this prototype was conducted in Chapter 7 in order to determine the effectiveness and efficiency provided by the prototype.

The objectives of this research, as listed in Section 1.3.3, are revisited in this chapter in order to determine whether they have been successfully realised. The research contributions, which provide a discussion on the theoretical and practical contributions of this research, are discussed in Section 8.2. The problems encountered within this research are discussed in Section 8.3, which includes the limitations of implementing a prototype for the proposed framework. Section 8.4 provides a discussion on the research benefits, based on the evaluations conducted. Finally, future research and recommendations are suggested for both theory and practice in Section 8.5.

8.2 Research Contributions

The primary objective of this research, as introduced in Section 1.3.3, was to *"improve efficiency and effectiveness of Mobile SOA"*. In order to achieve this goal, the following secondary objectives were identified:

- To gain comprehensive understanding of an SOA and its enabling technologies (Chapter 2);
- To acquire thorough knowledge of Mobile SOA and its enabling technologies (Chapter 3);
- To investigate compression and categorisation techniques of files used within Mobile SOA (Chapter 4);
    - Establish a set of criteria for measuring efficiency of compression;
- To determine how a framework supporting compression can be designed for Mobile SOA (Chapter 5);
- To determine how a prototype can be implemented based on the design of a framework (Chapter 6); and
- To evaluate the design of a framework and implementation of a prototype for Mobile SOA (Chapter 7).

These research objectives were analysed and discussed throughout the chapters of the dissertation. These research objectives were mapped onto a set of research questions, which were used to achieve a specific research objective. Table 8.1 illustrates the research questions from Section 1.3.4, and how each of the research questions was answered.

Table 8.1: Research Questions and Associated Methodology

#	Research Question	Research Method	Chapter
Main:	How can the efficiency and effectiveness of Mobile SOA be optimised using a framework that supports intelligent compression?		
R1	What is SOA and what are its components?	Literature Study	Chapter 2
R2	What is Mobile SOA and what are the relevant issues and constraints?	Literature Study	Chapter 3
R3	What are the issues related to file compression?	Literature Study	Chapter 4
R3.1	How can files be categorised in relation to intelligent compression?	Literature Study	
R3.2	What are the different file compression techniques?	Literature Study	
R3.3	What are the criteria for measuring efficiency of file compression?	Literature Study	
R4	How can a framework, supporting intelligent compression, be designed for Mobile SOA?	Service-Oriented Analysis and Design	Chapter 5
R5	How can a prototype, based on the proposed framework, be implemented?	Developmental/ Proof-of-Concept	Chapter 6
R6	Does the prototype adhere to the framework design principles?	Evaluation	Chapter 7
R7	Does the prototype show that the efficiency of Mobile SOA can be improved through intelligent compression?	Evaluation	

The research contributions are discussed in two categories, namely the theoretical and practical contributions. These are discussed in the following subsections.

8.2.1 Theoretical Contributions

The theoretical contributions can be broken down into the literature review and framework design. The achievements of the literature review are based on the research work done in Chapters 2, 3 and 4. The proposed framework, discussed in Chapter 5, highlights the achievements of the framework design.

8.2.1.1 Literature Review

The literature review achievements are emphasised in the research done into SOA, Mobile SOA and File Compression within Chapters 2, 3 and 4, respectively. These achievements include:

- Research into SOA, its enabling technologies and design principles;
- Research into Mobile SOA, its enabling technologies and extant systems; and
- Research into different types of compression and compression techniques.

<u>Service-Oriented Architecture (SOA)</u>

Research Question R1: "*What is SOA and what are its components?*"

A study was conducted in Chapter 2 to examine SOA, and its underlying components and enabling technologies. A formal definition was provided in order to clarify what SOA is in relation to this research. SOA was, therefore, defined as: *"an architectural approach to building loosely coupled business systems by integrating different components by means of Services. The integration of these components is independent of programming language and platform."* Gartner has found that the use of SOA is still increasing (Abrams et al. 2008), which indicates that businesses have a good understanding of SOA, and are able to access its benefits and capabilities.

SOA is based on an architectural style that integrates loosely coupled components across different platforms developed in different programming languages. The primary components of SOA are the Service Consumer (Service Requestor/Service Client/Service User), the Service Provider and the Service Manager (Service Registry/Service Directory). The components of SOA are developed as Services, which may be accessed by any application as long as it is discoverable. The implementation of a system based on SOA provides a number of benefits, which include application and data integration, versatility, code reuse and cost saving.

This research focused on Services in the form of XML-based Web Services. Although this is not the only manner in which to implement a Service, the industry standards and wide acceptance strongly suggest its use (Liegl 2007). An XML-based Web Service is developed using a number of essential standards, namely: HTTP, XML, WSDL, UDDI, and SOAP.

Hypertext Transfer Protocol (HTTP) is an internationally accepted Internet Protocol for the transmission of information between interconnected devices. HTTP is the standard for addressing Web pages, which can also identify Web Services.

Extensible Mark-up Language (XML) is a simple, text-based configuration, which represents structured information (W3C 2010). This information is stored within an XML file, which has its internal structures accessible by means of a process of metadata publishing.

A Web Service Description Language (WSDL) is a language for describing Web Services (W3C 2004a). Messages that have specific network protocols and message formats are exchanged between the Service Provider and the Service Consumer. A designer may specify the programming interface of a Web Service through WSDL. This interface is denoted in terms of procedures supported by the Web Service, where each procedure is capable of taking a message as an input and producing another message as an output.

The Universal Description Discovery and Integration (UDDI) is a framework for describing, discovering and integrating business Services through the Internet (Hurwitz et al. 2007). The UDDI provides definitions for registries, which can be made available publicly, privately, or between partner organisations (Senga 2010). The UDDI framework uses SOAP messages in order to exchange information with applications that access it.

Simple Object Access Protocol (SOAP) provides a standard, organised framework for wrapping and exchanging XML messages (W3C 2004a). SOAP is equipped for the exchange of information in a decentralised environment where messages are exchanged over HTTP/HTTPS. Using XML to describe the messaging framework means that SOAP is independent of platform and programming language (Gudgin et al. 2007).

<u>Mobile Service-Oriented Architecture (Mobile SOA)</u>

Research Question R2: "*What is Mobile SOA and what are the relevant issues and constraints?*"

Mobile SOA is a specific domain within SOA, and is focused on realising SOA in the mobile environment. Chapter 3 provided a formal definition for Mobile SOA and an in-depth discussion of what Mobile SOA is. The underlying technologies, component model, benefits and drawbacks of Mobile SOA were also presented. Two extant systems were reviewed, on which this research extends and solutions were proposed to the identified shortcomings.

The first extant system reviewed was the M-Service Framework developed by Sanchez-Nielsen et al. (2006). The approach used in this system reduced the computational cost of the mobile devices, which optimised the response times to mobile users and reduced memory usage. The EXEM framework, developed by Natchetoi et al. (2007b), was the second system reviewed. The framework provided a basis on which the proposed framework, described in Section 5.3.1, extends. The EXEM framework demonstrated an approach to efficient transfer of XML files using compression.

Mobile SOA was defined as: *"An architectural style for the mobile environment whose goal is to achieve loose coupling by positioning Services as the primary means through which solution logic is represented."* The necessity for SOA within the mobile environment is driven by the dramatic growth in the amount of connected mobile devices. This growth is evident in that there is an estimated 4.6 million connected mobile devices around the world (CBS News 2010). Mobile devices were developed as simple voice and SMS communication devices, but have since transformed into powerful handheld computers where almost any form of computing is capable of being achieved. However, there are a number of hardware, software, network and human limitations to consider when developing for this environment.

Mobile SOA has one additional primary component to that of the three primary components of SOA. This added component is the Mobility Controller. The Service Consumer, Service Provider and Service Registry are the other components found in Mobile SOA. The Mobility Controller controls the state transfer and manages the overall Mobile Service.

In addition to the mobility provided, there are a number of benefits when using Mobile SOA. These benefits extend to numerous branches of use, which include education, health, business, banking, weather forecast and providing much needed information to rural communities. Similarly to SOA, Mobile SOA offers interoperability with different systems on different platforms and allows interaction and information exchange between these systems. The challenges faced in developing for Mobile SOA are based on the mobile device constraints, the continually increasing amount of mobile devices available and the high costs of transferring data across wireless networks. The benefits of Mobile SOA, however, far outweigh the challenges, and can, thus, be considered an ideal candidate for future research and improvements.

File Compression

Research Question R3: "*What are the issues related to file compression?"*

Chapter 3 identified limitations when implemented for the Mobile SOA environment. File compression was proposed as a solution to a number of these limitations. Chapter 4 provided a discussion on the two types of data compression, the different algorithms used within these types of compression, and several compression techniques for each of the three categories of files used within this research, namely XML, image and audio files.

The two types of data compression identified were lossless and lossy compression. Lossless compression is the process of compressing information for storage or transmission, which can then be decompressed, producing a copy of the original information without any loss. In contrast, lossy compression allows a certain level of information loss, which is either not perceptible or is of a acceptably lower quality. Several algorithms were identified for both lossless and lossy compression. These algorithms illustrate the procedure of compressing information in order to achieve high levels of compression. Although the compression algorithms differ significantly, they are still based on early work done by Shannon (1948) and Huffman (1952).

The files used within this research were categorised as Data or Media files. For this research data files represent textual information and are stored within XML files. For media files, image and audio files were used. Six lossless compression techniques were identified for XML compression, namely: bZip2, gZip, DotNetZip, XMill, XMLppm and XWRT. Three compression techniques, two lossless and one lossy, were identified for image and audio files, respectively. The lossless image compression techniques were GIF and PNG, and the lossy image compression technique was JPEG. The lossless audio compression techniques were APE and FLAC, and the lossy audio compression technique was MP3. A comparison on the performance of each of these compression techniques was conducted in Chapter 6.

8.2.1.2 Framework Design

Research Question R4: *"How can a framework, supporting intelligent compression, be designed for Mobile SOA?"*

The achievements for the framework design are underlined in the design of the proposed framework, the implementation of a prototype based on this framework, and the evaluation of this prototype. Chapter 5 provided a discussion on the design process, the four underlying components, process steps and perceived benefits of the proposed framework.

Seven design principles were identified in the design process of the proposed framework. These principles emphasized the necessity for a simple, lightweight, easy to implement, cost effective, interchangeable, flexible and efficient framework, which supports the use of intelligent compression.

The four components of the proposed framework included the User Interaction Component, the Intelligence Component, the Compression Component and the Decompression

Component, as illustrated in Figure 8.1[1]. These components work together to achieve high levels of efficiency when managing files of a requested Web Service.

The process steps of the proposed framework provided a detailed discussion on how the four components interact. This discussion included a step-by-step walkthrough from when the mobile application was first used, to the delivery and display of a requested Web Service. The perceived benefits of the proposed framework were identified, which served as a benchmark for performance. A proof-of-concept prototype was evaluated on how it performed against these perceived benefits.

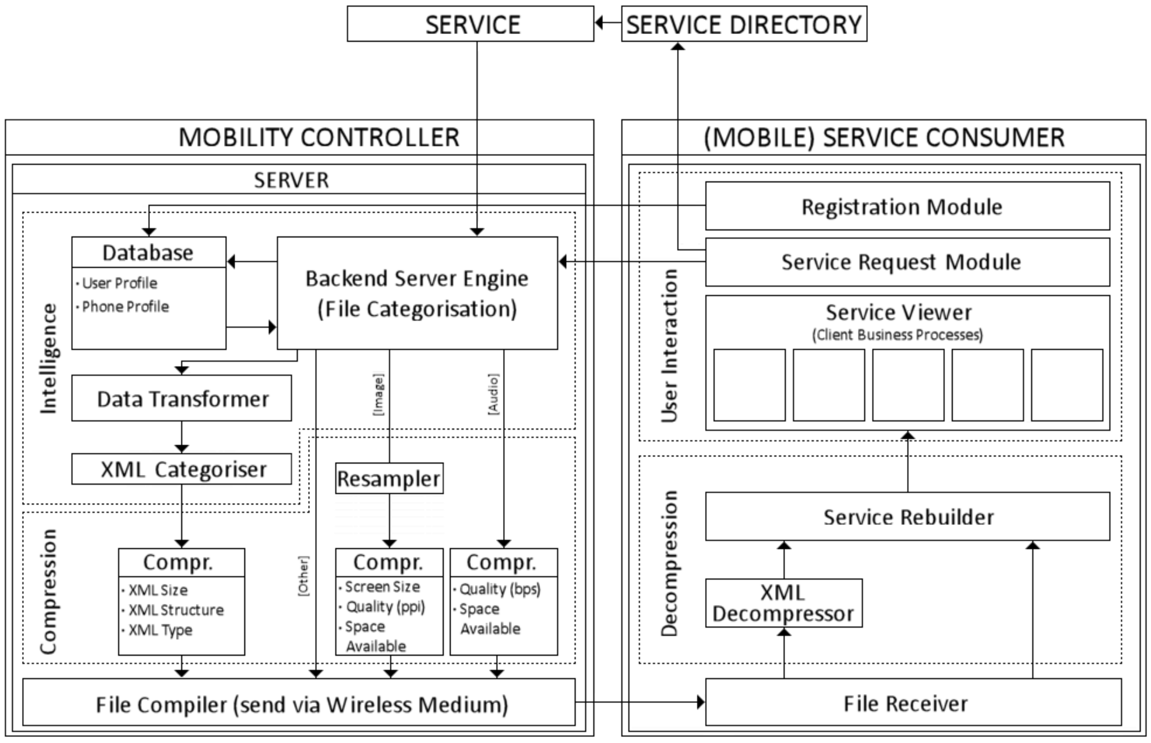


Figure 8.1: Proposed Framework

The next section provides a discussion on the practical contributions of this research, which includes the implementation of the proof-of-concept prototype and the evaluation thereof.

8.2.2 Practical Contributions

In terms of practical contributions, the primary contribution of this research is the implementation of a proof-of-concept prototype based on the proposed framework. The secondary practical contribution is the evaluation of this prototype. The practical

[1] Figure 5.4 is repeated as Figure 8.1 for ease of reference.

contributions, and the related research questions that were addressed, are discussed in the following subsections.

8.2.2.1 Development of Proof-of-Concept Prototype

Research Question R5: "*How can a prototype, based on the proposed framework, be implemented?*"

In order to conduct an evaluation to determine whether the proposed framework optimises the effectiveness and efficiency of Mobile SOA, a proof-of-concept prototype was implemented. The implementation of the proof-of-concept prototype, in itself, provided evidence that it was feasible to implement a prototype, based on the proposed framework, which supports intelligent compression.

Chapter 6 provided a discussion on the implementation criteria, the tools used during the implementation process, the implementation of the prototype and the four underlying components of the proposed framework. Additionally, Chapter 6 provided a discussion on the initial tests conducted before the proof-of-concept prototype was implemented.

An evaluation strategy was designed in order to comprehensively evaluate the proof-of-concept prototype. These evaluations are discussed next.

8.2.2.2 Evaluation of Proof-of-Concept Prototype

The evaluation of the prototype was divided into three categories of testing, namely: three pilot studies, an analytical evaluation and a performance evaluation.

<u>Pilot Studies</u>

Three pilot studies were conducted in order to determine whether compression is feasible within the mobile environment, whether the proposed framework supported efficient file transfer and whether the proposed framework supports sustained usage on a mobile device, respectively. A test was conducted for each of these pilot studies. The first test conducted simply tested whether decompression could be performed on a mobile device. The efficiency of a prototype, developed according to the proposed framework, was evaluated in the second test. The third test provided an evaluation on the battery life of two mobile devices in order to determine if the prototype supports continuous use. For this test, decompression was performed until the battery life reached its minimum.

The pilot studies provided positive results, from which it was concluded that it was feasible to implement a prototype, based on the proposed framework. This prototype could support decompression on a mobile device, provide significant efficiency and support extensive usage on a mobile device before the battery requires recharging.

Analytical Evaluation

Research Question R6: *"Does the prototype adhere to the framework design principles?"*

The analytical evaluation was conducted to determine whether the proof-of-concept prototype adheres to the seven design principles identified in the design process of the proposed framework in Section 5.3.1. These design principles are listed as the following:

- Easy and Lightweight;
- Compatibility;
- Robustness and Concurrency;
- Interchangeable Communication Medium;
- Service Discovery;
- Efficiency; and
- Flexibility.

Six of the seven design principles were directly tested in the analytical evaluation. Although the *flexibility* was not test specifically, the manner in which the prototype was implemented, and the tools that were used during implementation, provided sufficient motivation to conclude that the design principles were adhered to. Thus, based on the analytical evaluation, it was concluded that the proof-of-concept prototype was implemented according to the design principles.

Performance Evaluation

Research Question R7: *"Does the prototype show that the efficiency of Mobile SOA can be improved through intelligent compression?"*

The primary objective of this research was to provide evidence that the proposed framework optimises the effectiveness and efficiency of Mobile SOA. In order to determine whether this objective was realised an evaluation of the proof-of-concept prototype was conducted. The performance evaluation measured the overall performance of this prototype.

The performance metrics were identified and divided into four categories, which included response time, resource usage, reliability and cost. Furthermore, testing the performance of the prototype was divided into the Server performance and the (Mobile) Client performance. Server performance metrics included the compression time, compression, and processing power. The Client performance metrics included the decompression time, memory footprint, battery consumption, file transfer rate, task completion rate, cost and overall time elapsed.

The results observed during the evaluation of the prototype confirmed those yielded during the pilot studies. The prototype produced a significant performance improvement over the common transfer method of a Web Service. For smaller sized XML files, it was observed that the compression, transfer and decompression of these files took longer than transferring it without compression. A higher cost, however, was incurred when transferring without the use of compression.

The prototype provided efficient file transfer to a mobile device by reducing the cost and time. Furthermore, the prototype reduced the amount of resources expended on a mobile device. The decrease in file transfer time and reduction of storage space needed meant that a reduction in processing power and time was observed. This reduction implied that the battery power consumed by a mobile device was reduced.

In answering research questions R5, R6 and R7, the practical contributions highlighted the development of the proof-of-concept prototype and evaluation thereof. The limitations and problems encountered while conducting this research is discussed in the next section.

8.3 Problems Encountered

Some delineation decisions were taken at the outset of the research:

- **Video Compression:** Due to the timing and cost restraints of file transfer, video compression was not considered within this research.
- **Only Two Mobile Devices Used for Tests:** Developing for the mobile environment provides an additional challenge in trying to ensure that the mobile application works on more than just a few mobile devices. As a result of the vast amounts of mobile brands, and the large number of mobile devices developed for each brand, ensuring that a mobile application runs on these different devices is highly unlikely. Thus, for this research, the objective was to prove that the concept of implementing a

prototype, based on the proposed framework, is feasible. This application was not required to work on each mobile device available on the market.

Although above decisions resulted in less extensive implementation criteria, a number of problems were still encountered when implementing the proof-of-concept prototype. Implementing for the mobile environment provided significant challenges over that of implementing for the desktop environment. The following problems and challenges were identified during implementation of the prototype:

- **Decompression on Mobile Device:** For the original XML compression techniques tested within this research, the corresponding decompression techniques could not be performed on a mobile platform. As a result of this, a number of delays were caused, as the source code had to be recompiled, in some cases rewritten, for the mobile environment. DotNetZip was the only compression technique that offered support for the mobile environment.
- **Service Requests:** Due to network restrictions, external Service requests could not be conducted. This led to the development of three Services in order to adequately test the efficiency of the prototype.
- **Connection between Client and Server:** As a result of network restrictions and network limitations of the mobile devices, numerous connection issues were encountered between the Client and Server. A Virtual Private Network (VPN) had to be set up on the mobile devices, with the specific network settings, in order to provide a partial solution to this problem. Additionally, specific connection protocols were included within the implementation of the mobile application, a Client Side subset of the proof-of-concept prototype.

The problems encountered during the implementation process affected the evaluation of the proof-of-concept and also imposed a number of limitations within this research. The following problems were identified during evaluation:

- **Unable to Record Mobile Information:** The information recorded during the registration process and the performance evaluation were recorded and saved by the implemented system. However, specific information was not recorded. This information included the processing power and screen colour depth on a mobile device.

- **Cost Evaluation:** Due to the difference in cost of data transfer between networks, and the difference in cost of data transfer for each data package available, it was not possible to compare the cost saving of using the prototype for file transfer. The decision was made to use only the costing model of out-of-bundle data transfer in order to provide an adequate comparison between using the prototype to transfer files, and using the standard file transfer protocols.

Although there were numerous problems and limitations, this did not affect the implementation of the proof-of-concept prototype, and the conclusions derived from the evaluation thereof.

8.4 Benefits of Research

Significant benefits were identified within this research, as well as for the proposed framework. The results and findings of the research are divided into the benefits of the research, and the advantages and disadvantages of the proposed framework discussed in Chapter 5.

This research demonstrated that the proposed framework is able to optimise the effectiveness and efficiency of Mobile SOA.

- **Improved Implementation:** The proof-of-concept prototype was implemented according to the framework design principles listed in Section 5.3.1. Using an analytical evaluation, conducted in Section 7.4, it was determined that the prototype adheres to these design principles. Implementing a system based on these design principles may improve the overall implementation process, which includes a reduction in the implementation time and cost. The improved implementation demonstrated the effectiveness of the prototype.
- **Improved Efficiency:** This research provided evidence that using the framework, when implementing a Client/Server system, significantly improves the efficiency when transferring and managing files. The results recorded during the performance evaluation demonstrated a significant reduction in transfer time, the amount of resources used and potential costs incurred during file transfer.

Section 5.3.4 identified the perceived benefits of implementing a proof-of-concept prototype based on the proposed framework. The performance evaluation conducted in Section 0 verified specific perceived benefits of the proof-of-concept prototype. These benefits are listed as follows:

- The mobile applications may perform faster and better when developed according to a specific guideline, such as that described by the proposed framework.
- Compression of files reduced the processing power and memory footprint required to manage files, in terms of transferring, accessing and storing of these files.
- The reduced processing power and memory footprint had a direct influence on the battery life of the mobile devices.
- The viewing of an image was enhanced by means of image resampling, which matched the image resolution to the screen resolution for each individual mobile device. The content layout and navigation was also improved as a result.
- The framework provided improved file transfer between the Server and (Mobile) Client.
- Compression also reduced the amount of overhead required for transfer over a network. Although not tested on a commercial network, the results illustrated that compression would significantly reduce the cost of transferring files between a Client and Server.

This research, therefore, strongly recommends the use of the proposed framework, or variant thereof, when implementing a Client/Server information exchange system.

8.5 Future Research and Recommendations

The design of the proposed framework and the successful implementation of a proof-of-concept prototype provides both practical and theoretical evidence that the effectiveness and efficiency of Mobile SOA can be optimised.

During the initial testing of the XML compression techniques, a number of phenomena were observed. One significant observation is the ‘spikes’ in compression ratio for each of the compression techniques during compression of the same file, as illustrated in Figure 6.1. An XML file may be categorised by its structure (and structure complexity), type (APP, XSL-FO, SOAP, etc), number of repeated elements and attribute values within the XML file, and content. This categorisation is an important factor to consider when compressing XML files. Testing the different compression techniques against the different categories of XML files may provide additional information, which may improve the compression efficiency of XML files.

In the case of mobile devices, the Client has an additional risk of the mobile device being stolen or lost, which results in the potential exposure of sensitive data. All data transferred and stored on the mobile device should, therefore, be encrypted. Security and file encryption was not included within the scope of this research, and may be an important asset when providing efficient transfer of files and Services to and from a mobile device.

Due to the varying data transfer rates and high costs, the wireless connectivity offered by network operators were not utilised. In order to accurately measure the performance of the prototype, implemented according to the framework design principles, a more reliable Wi-Fi connection was used. However, to evaluate the prototype in a real-world situation may provide valuable information, which could improve the efficiency when transferring files between a Server and (Mobile) Client.

Theoretically, the framework, proposed in Section 5.3.1 may support the compression of all files. The framework, however, was designed in order to support the efficient transfer of a Service by means of compressing the XML, image, and audio files it commonly uses. Other file types were not considered within this research. Furthermore, the primary objective of the framework was limited to providing efficient transfer of files used by a Service between the (Mobile) Service Consumer and the Service Directory. The framework was not supported by, and did not support the efficiency of the Service Provider, the Service Directory, the Service Manager, or the Service itself.

There are many factors that determine the transfer rate of data over a wireless network. These factors include unreliable connections, limited bandwidth, poor latency, distance from the closest network cell tower, and the amount of users requesting data via the same cell tower at a given time. Because of the fluctuating transfer rates, Wi-Fi was used as the wireless connection between the Client and Server. In order to compare the transfer rates between the wireless technologies, the maximum theoretical transfer rates for a given wireless technology were used. Although a significant cost of transferring files may be incurred, the benefit of using real network operator costs may be beneficial in comparing the difference in costs incurred between compressed and uncompressed files.

Although a successful implementation and evaluation of a prototype (based on the proposed framework) was conducted, the prototype was implemented and evaluated within a closed and controlled setting. Ideally, testing the prototype within a real-world environment would provide more concrete, statistical evidence to support the conclusions on the efficiency of the prototype. Thus, it is recommended that future research incorporate this real-world testing environment within its scope. Additionally, it is recommended that sufficient testing be conducted when transferring compressed information from the (Mobile) Client to the Server. Although the functionality to compress Client Side data exists, no significant tests were conducted.

8.6 Summary

Chapter 1 provided an outline of this research study by means of the Thesis Statement, research problem, research objectives, research questions and research methodology. The Thesis Statement for this research study was:

"The efficiency and effectiveness of Mobile SOA can be improved by the implementation of a framework supporting intelligent compression."

The aim of this research study was to design a framework for the Mobile SOA environment. Based on the Thesis Statement, a framework was proposed in Chapter 5, based on specific design principles, which supports intelligent compression. A proof-of-concept prototype, based on the proposed framework, was implemented in Chapter 6. This prototype was evaluated in Chapter 7, which determined that it is feasible to optimise the effectiveness and efficiency of Mobile SOA. The evaluation process was divided into pilot studies (tested the feasibility of the framework), an analytical evaluation (determined the degree to which the prototype adheres to the design principles of Section 5.3.1) and a performance evaluation (determined whether the prototype provides sufficient efficiency, and what level of efficiency it provides). The results and statistical analysis from the evaluation illustrate that a successful implementation was achieved.

The aim of this research study was successfully realised theoretically, with the design of the proposed framework, and practically, with the implementation of a proof-of-concept prototype (based on this framework) and the evaluation thereof.

References

31314, Xcite – *Games, Ringtones, Wallpapers And More!*, available at http://31314.mobi/ [online], 2010, [accessed 1st November 2010].

35050, *35050*, available at http://www.mobilemagic.co.za/ [online], 2010, [accessed 1st November 2010].

Abrams, C. & Schulte, R.W., *Service-Oriented Architecture Overview and Guide to SOA Research*, Report Number G00154463, Gartner, Stamford, CT, USA, available at: http://www.gartner.com/DisplayDocument?doc_cd=154463 [online], 2008, [accessed 15th October 2009].

Alonso, G., Casati, F., Kuno, H. & Machiraju, V., *Web Services: Concepts, Architectures, and Applications*, Springer, 2004.

Ashland, M.T., *Monkey's Audio – A fast and powerful lossless audio compressor*, available at http://www.monkeysaudio.com/ [online], 2009, [accessed 30th July 2010].

Atkins, D.E., Droegemeier, K.K., Feldman, S.I., Garcia-Molina, H., Klein, M.L., Messerschmitt, D.G., Messina, P., Ostriker, J.P. &Wright, M.H., *Revolutionizing Science and Engineering Through Cyberinfrastructure:Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure*, National Science Foundation, available http://www.nsf.gov/od/oci/reports/atkins.pdf [online], 2003, [accessed 15th March 2010].

Augeri, C. J., Mullins, B. E., Baird III, L. C., Bulutoglu, D. A. & Baldwin, R. O., *An Analysis of XML Compression Efficiency*, U.S. Air Force, Department of Computer Science, ExpCS, San Diego, CA, 13–14 June 2007.

Barton, J., Zhai, S. & Cousins, S., *Mobile Phones Will Become The Primary Personal Computing Devices*, available at www.almaden.ibm.com/u/bartonjj/jbarton-PhoneBeatsPC.pdf [online], 2010, [accessed 2nd May 2010].

Blelloch, G.E., *Introduction to Data Compression*, Carnegie Mellon University, Computer Science Department, 2001.

Botha, A., Makitla, I., Ford, M., Fogwill, T., Seetharam, D., Abouchabki, C., Tolmay, J.P. & Oguneye, O., *The mobile phone in Africa: Providing Services to the masses*, Proceedings from CSIR Conference, Science real and relevant conference, 2010.

Brown, J.C., *A Critical Overview of Computer Performance Evaluation*, ICSE '76, Proceedings of the 2nd International Conference on Software Engineering, pp. 138-145, 1976.

Buzzword, *Buzzword*, available at: https://buzzword.acrobat.com/ [online], 2009, [accessed 31st May 2010].

Canfora, G. & Troiano, L., *The Importance of Dealing with Uncertainty in the Evaluation of Software Engineering Methods and Tools*, SEKE '02, Ischia, Italy, pp 691-698, 2002.

Carpenter, H., *Gartner Hype Cycle for Emerging Technologies 2009, What's Peaking, What's Troughing?*, available at: http://bhc3.wordpress.com/2009/07/27/gartner-hype-cycle-2009-whats-peaking-whatstroughing/ [online], 2009, [accessed 13th September 2010].

Cavanaugh, E., *Web Services: Benefits, Challenges, and a Unique, Visual Development Solution*, Altova WhitePaper, USA, available at http://www.altova.com/whitepapers/webservices.pdf, 2006, [accessed 22nd December 2010].

CBS News, *Number of Cell Phones Worldwide Hits 4.6B*, CBSNews.com, available at http://www.cbsnews.com/stories/2010/02/15/business/main6209772.shtml [online], 2010, [accessed 1st May 2010].

Chattpar, A., *Increased business agility through BRM systems and SOA*, available at http://www.ibm.com/developerworks/architecture/library/ar-brmssoa/ [online], 2008, [accessed 24th April 2010].

Cheney, J., *XMLPPM: XML-Conscious PPM Compression*, available at http://xmlppm.sourceforge.net/ [online], 2009, [accessed 12th July 2009].

Coalson, J., *FLAC – Free Lossless Audio Codec*, available at http://flac.sourceforge.net/index.html [online], 2008, [accessed 30th July 2010].

Datz, T. *What You Need to Know About Service-Oriented Architectures*, Available at http://www.cio.com/article/32060/What_You_Need_to_Know_About_Service_Oriented_Architecture [online], 2004, [accessed 6th July 2009].

Deru, M. & Torcellini, P., *Performance Metrics Research Project – Final Report*, available at http://www.osti.gov/bridge [online], 2005, [accessed 13th November 2010].

Dey, A. & Sohn, T., *Supporting End User Programming of Context-Aware Applications*, Conference on Human Factors in Computing Systems, Fort Lauderdale, 2003.

Duda, I., Alesky, M. & Schader, M., *Leveraging Different Application Styles in Mobile Business*, Proceedings of MoMM, Linz, Austria, 2008.

Edmunds, A. & Morris, A., *The problem of information overload in business organisations: a review of the literature*, *International Journal of Information Management*, Volume 20, Issue 1, pp 17-28, ISSN 0268-4012, DOI: 10.1016/S0268-4012(99)00051-1, February 2000.

Erl, T., *Service-Oriented Architecture: Concepts, Technology, and Design*, Upper Saddle River, NJ, Prentice Hall PTR, 2005.

Erl, T., *SOA Principles of Service Design, Service-Oriented Computing*, Upper Saddle River, NJ, Prentice Hall, 2008.

Erradi, A., Kulkarni, Anand, S. & Kulkarni, N., *Evaluation of Strategies for Integrating Legacy Applications as Services in a Service Oriented Architecture*, IEEE International Conference on Services Computing (SCC'06), 2006.

EyeOS, *EyeOS*, available at: http://Eyeos.org [online], 2009 [accessed on: 24th March 2010].

eXactMobile, *eXactmobile for the best Digital music, True tones, and Games in South Africa*, available at http://www.exactmobile.co.za/ [online], 2010, [accessed 1st November 2010].

Flickenger, R., Okay, S., Pietrosemoli, E., Zennaro, M. & Fonda, C., *Very Long Distance Wi-Fi Networks*, NSDR ’08, Seattle Washington, USA, 2008.

Fraunhofer, *Fraunhofer-Gesellschaft*, available at http://www.fraunhofer.de/en/ [online], 2010, [accessed 24th September 2010].

Fremantle, P., Weerawanana, S. & Khalaf, R., *Enterprise Services*, Commun. ACM 45(10), 2002.

Gailly, J.-L. & Adler, M., *The gZip Homepage*, available at www.gzip.org/ [online], 2003, [accessed 12th July 2009].

Gartner, *Gartner Hype Cycle*, available at http://www.gartner.com [online], 2010, [accessed 24th September 2010].

Google, *Google Docs*, available at: http://docs.google.com/, [online], 2009, [accessed 24th March 2010].

Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.J., Nielsen, H.F., Karmarkar, A. & Lafon, Y., *SOAP Version 1.2 Part 2: Adjuncts (Second Edition)*, World Wide Web Consortium, available at http://www.w3.org/TR/2007/REC-soap12-part2-20070427/ [online], 2007, [accessed 31st May 2010].

Halonen, T. & Ojala, T., *Cross-Layered Design for Providing Service Oriented Architecture in a Mobile Ad hoc Network*, MUM’06, Stanford, CA, USA, 2006.

Howard, P.G. & Vitter, J.S., *Arithmetic Coding for Data Compression*, Proceedings of the IEEE Vol. 82 No. 6, June 1994.

Huang, K.-Y., *Challenges in Human-Computer Interaction Design for Mobile Devices*, Proceedings of the World Congress on Engineering and Computer Science, Vol. I, San Francisco, USA, October 20-22, 2009.

Huffman, D.A., *A Method for the Construction of Minimum-Redundancy Codes*, Proceedings of the I.R.E., pp 1098-1101, 1952.

Hurwitz, J., Bloor, R., Baroudi, C. & Kaufman, M., *Service Oriented Architecture for Dummies*, Wiley Publishing Inc., Indianapolis, Indiana, ISBN-13: 978-0-470-05435-2, 2007.

IBM, *IBM – United States*, available at http://www.ibm.com/us/en [online], 2010, [accessed 29 th June, 2010].

Investintech, *What is Compression?*, available at http://www.investintech.com/resources/articles/whatcompression/ [online], 2006, [accessed 1st May 2010].

International Telecommunication Union (ITU), *Information Technology – Digital COMPRESSION and Coding of Continuous-Tone Still Images – Requirements and Guidelines*, Terminal Equipment and Protocols for Telematic Services, 1992.

Johnson, B.C., Manyika, J.M. & Yee, L.A., *The Next Revolution in Interactions*, The McKinsey Quarterly, Number 4, 2005.

Johnsrud, L., Hadzic, D., Hafsøe, T., Johnsen, F. & Lund, K., *Efficient Web Services in Mobile Networks*, Sixth European Conference on Web Services, 2008.

Jørstad, I., Dustdar, S. & van Thanh, D., *Service-Oriented Architectures and Mobile Services*, UMICS, available at www.infosys.tuwien.ac.at/Staff/sd/papers/Service-OrientedArchitecturesAndMobile%20Services_UMICS2005.pdf [online], 2005, [accessed 10th March 2009].

Jørstad, I., van Thanh, D. & Dustdar, S., *An Analysis of Service Continuity in Mobile Services*, available at http://www.infosys.tuwien.ac.at/Staff/sd/papers/AnAnalysisOfServiceContinuityInMobileServices.pdf [online], 2004, [accessed 10th March 2009].

Juul, N.C. & Jørgensen, N., *Security Issues in Mobile Commerce Using WAP*, 15th Bled Electronic Commerce Conference, Bled, Slovenia, 2002.

Kajko-Mattsson, M. & Chapin, N., *SOA-zation Framework (SF)*, Proceeding of PESOS '10, Cape Town, South Africa, 2010.

Kanneganti, R. & Chodavarapu, R., *SOA Security*, Greenwich, CT. Manning Publications Co., 2008.

Levene, M., Wood, P., *XML Structure Compression*, available at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.824&rep=rep1&type=pdf [online], 2002, [accessed 31st January 2010].

Lewis, G.A. & Wrage, L., *Model Problems in Technologies for Interoperability: Web Services*, Technical Report CMU/SEI-2006-TN-021. Carnegie Mellon Software Engineering Institute, available at http://www.sei.cmu.edu/reports/06tn021.pdf [online], 2006, [accessed 22nd January 2010].

Lian, W., Cheung, D.W., Mamoulis, N. & Yui, S.-M., *An Efficient and Scalable Algorithm for Clustering XML Documents by Structure*, IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 1, January 2004.

Li, Z.-N. & Drew, M.S., *Fundamentals of Multimedia*, *School of Computing Science*, Simon Fraser University, Pearson Prentice Hall, Upper Saddle River, NJ, ISBN: 0-13-061872-1, 2004.

Li, X., Zaki, Y., Weerawardane, T., Timm-Giel, A. & Goerg, C., *HSUPA Backhaul Bandwidth Dimensioning*, TZI-ikom, Communications Networks, University of Bremen, Germany, 2008.

Liefke, H., *XMill, An Efficient Compressor for XML*, available at http://www.liefke.com/hartmut/xmill/xmill.html [online], 2004, [accessed 12th July 2009].

Liegl, P., *The Strategic Impact of Service Oriented Architectures*, Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, 2007.

Lowe, H.J., Lomax, E.C. & Polonkey, S.E., *The World Wide Web: A Review of an Emerging Internet-based Technology for the distribution of Biomedical Information*, Journal of the American Medical Informatics Association, Volume 3, Number 1, 1996.

Madhavapeddy, A. & Tse, A., *A Study of Bluetooth Propagation Using Accurate Indoor Location Mapping*, Springer, UbiComp, LNCS 3660, pp. 105-122, 2005.

Meeker, M., *Mobile Internet Will Soon Overtake Fixed Internet*, available at http://gigaom.com/2010/04/12/mary-meeker-mobile-internet-will-soon-overtake-fixed-internet/ [online], 2010, [accessed 24th September 2010].

Miano, J., *Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP*, Addison-Wesley, ISBN: 978-0201604436, 1999.

Microsoft, *DotNetZip* Library, Copyright © 2006-2010 Microsoft Corporation, available at http://dotnetzip.codeplex.com/ [online], 2010a, [accessed 17th March 2010].

Microsoft, *SQL Server 2008*, available at http://www.microsoft.com/sqlserver/2008/ [online], 2010b, [accessed 31st October 2010].

Microsoft, *.Net Framework Conceptual Overview*, available at http://msdn.microsoft.com/en-us/library/zw4w595w(v=VS.90).aspx [online], 2010c, [accessed 31st October 2010].

Microsoft, *Visual Studio 2008*, available at http://www.microsoft.com/visualstudio/en-us/products/2008-editions [online], 2010d, [accessed 31st October 2010].

Microsoft, *Windows Mobile 6*, available at http://msdn.microsoft.com/en-us/library/bb278115.aspx [online], 2010e, [accessed 31st October 2010].

Microsoft, *Windows Mobile 6 Professional and Standard Software Development Kits*, available at http://www.microsoft.com/downloads/en/details.aspx?FamilyID=06111a3a-a651-4745-88ef-3d48091a390b&displaylang=en#Top [online], 2010f, [accessed 31st October 2010].

Microsoft, *The Official Microsoft IIS Site*, available at http://www.iis.net/ [online], 2010g, [accessed 31st October 2010].

Microsoft, *Windows Server 2008*, available at http://www.microsoft.com/windowsserver2008/en/us/default.aspx [online], 2010h, [accessed 31st October 2010].

Mirchandani, C., *Evaluating Performance Factors in System Development*, MAPLD, 2001.

Natchetoi, Y., Wu, H. & Babin, G., *A Context-Dependent XML Compression Approach to Enable Business Applications on Mobile Devices*, Proceedings of Euro-Par, 2007.

Natchetoi, Y., Wu, H., Babin, G. & Dagtas, S., *EXEM: Efficient XML data exchange management for mobile applications*, Springer, 2007.

Natchetoi, Y., Kaufman, V. & Shapiro, A., *Service-Oriented Architecture for Mobile Applications*, SAM'08, Leipzig, Germany, May 10th 2008.

Nelson, M. & Gailly, J.-L., *The Data Compression Book*, 2nd Edition, M&T Books, ISBN: 978-1558514348, 1995.

Nezhad, H.R.M., *Model-driven Adapter Development for Web Services Interactions*, available at http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-169/paper7.pdf [online], 2006, [accessed 21st May 2010].

OASIS, *Reference Model for Software Oriented Architectures*, available at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm [online], 2006, [accessed 17th April 2009].

Ojala, O., *Master's thesis, Service Oriented Architecture in Mobile Devices: Protocols and Tools*, Helsinki University of Technology, 7th November 2005.

Othman, A.K., Zakaria, M. & Hamid, A., *TCP Performance Measurement in GPRS Link Adaption Process, International Journal of Engineering*, Volume 2, Issue 1, 2008.

Palmer, J.W., *Web Site Usability, Design, and Performance Metrics, Information Systems Research*, ABI/INFORM Global, Vol. 13, No. 2, pp 151-167, 2002.

Papazoglou, M.P. & van den Heuvel, W-J., *Service Oriented Architectures: Appoaches, Technologies and Research Issues*, The VLDB Journal 16, pp 389-415, 2007.

Pennebaker, W.B. & Mitchell, J.L., *The JPEG Still Image Data Compression Standard*, New York, Van Nostrand Reinhold, 1993.

Phu, P. & Yi, .M., *A Service Management Framework for SOA-Based Interoperability Transactions*, Proceedings from KORUS, pp 680-685, IEEE, 2005.

Poslad, S., *Ubiquitous Computing: Smart Devices, Environments and Interactions*, Jonh Wiley & Sons Ltd., ISBN 978-0-470-04560-3, 2009.

Preece, J., Rogers, Y. & Sharp, H., *Interaction Design: Beyond Human-Computer Interaction*, 2nd Edition, New York, NY, John Wiley & Sons, 2007.

Quynh, P.T. & Thang, H.Q., *Dynamic Coupling Metrics for Service-Oriented Software*, International Journal of Computer Science and Engineering, 2009.

Ravi, N., Scott, J., Han, L. & Iftode, L., *Context-Aware Battery Management for Mobile Phones*, PERCOM '08, Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications, 2008.

Rist, T. & Brandmeier, P., *Customizing Graphics for Tiny Displays of Mobile Devices, Personal and Ubiquitous Computing*, Vol. 6, No. 4, pp. 260-268, 2002.

Sanchez-Nielsen, E., Martin-Ruiz, S. & Rodriguez-Pedrianes, J., *An Open and Dynamical Service Oriented Architecture for Supporting Mobile Services*, ICWE'06, Palo Alto, California, USA, ACM 1-59593-352-2/06/0007, pp 121-128, July 11-14 2006.

Sanders, D., Hamilton, J. & MacDonald, R., *Supporting A Service-Oriented Architecture*, SpringSim, pp 325-334, 2008.

Savini, M., Ionas, A., Meier, A., Pop, C. & Stormer, H., *The eSana Framework: Mobile Services in eHealth using SOA*, 2007.

Sedoyeka, E., Almasri, S., Rahman, A. & Hunaiti, Z., *HSDPA Wireless Broadband Link Performance*, Faculty of Science and Technology, Anglia Ruskin University, Chelmsford, UK, 2008.

Seeck, R., *Data Compression, Binary Essence*, available at http://www.binaryessence.com/dct/en000003.htm [online], 2005, [accessed 24th September 2010].

Senga, E., *A Service-Oriented Approach to Implementing an Adaptive User Interface*, Masters Thesis, Nelson Mandela Metropolitan University, Department of Computing Sciences, 2010.

Seong, S.-W., *Dictionary-Based Code Compression Techniques Using Bit-Masks for Embedded Systems*, Masters Thesis, University of Florida, Department of Computer and Information Science and Engineering, 2006.

Seward, J., *bZip2*, available at www.bzip.org/ [online], 2007, [accessed 12th July 2009].

Shannon, C.E., *A Mathematical Theory of Communication*, available at http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf [online], 1948, [accessed 24th July 2010].

Shen, H.T., *Service-Oriented Architecture*, University of Queensland, available at http://www.itee.uq.edu.au/~infs3204/Lecture_Notes/M1.pdf [online], 2007, [accessed 29th September, 2010].

Shrestha, A., *MobileSOA Framework for Context-Aware Mobile Applications*, Eleventh International Conference on Mobile Data Management, IEEE, 2010.

Shudong, W. & Higgins, M., *Limitations of Mobile Phone Learning*, The JALT CALL Journal Vol. 2, No. 1, pp 3-14, 2006.

Skibinski, P., *XWRT32*, available at http://www.ii.uni.wroc.pl/~inikep/ [online], 2009, [accessed 12th July 2009].

SOABooks.com, *The Prentice Hall Service-Oriented Computing Series from Thomas Erl*, available at http://www.soabooks.com [online], 2010, [accessed 30th May 2010].

Spanos, N., *100 IT Performance Metrics*, available at www.itmpi.org/assets/base/images/itmpi/Spanos-Metrics.pdf [online], 2009, [accessed 13th November 2010]

StatSoft, *ANOVA/MANOVA*, available at http://www.statsoft.com/textbook/anova-manova/ [online], 2010, [accessed 2nd May 2010].

Tokgoz, Y., Meshkati, F., Zhou, Y., Yavuz, M. & Nanda, S., *Uplink Interference Management for HSPA+ and 1xEVDO Femtocells*, proceedings from IEEE GLOBECOM, 2009.

Tso, F.P., Teng, J., Jia, W. & Xuan, D., *Mobility: A Double-Edged Sword for HSPA Networks*, MobiHoc ’10, Chicago, USA, 2010.

U.S. Census Bureau, *Population Division*, available at http://www.census.gov/ipc/www/popclockworld.html [online], 2010, [accessed 1st May 2010].

van Gurp, J., Karhinen, A. & Bosch, J., *Mobile Service Oriented Architectures (MOSOA)*, available at http://www.janbosch.com/mosoa.pdf [online], 2005, [accessed 9th February 2009].

Velde, E.T., Atsma, D.E., Hoekema, R., Luijten, J.E., Buddelmeijer, C.I., Spruijt, H.J. & Putten, N.H.J.J., *A Multicenter PDA Project to Support the Clinical Decision Processs*, Proceedings of the IEEE Conference on Computers in Cardiology, pp. 177-179, 2004.

Venu, R., *A Project Report on Webkit Port of Mobile SOA*, Master's Thesis, Department of Computer Science, Cochin University of Science & Technology, June 2008.

W3C, *Web Services Architecture*, available at http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/ [online], 2004a, [accessed 3rd May 2010].

W3C, *Web Services Architecture*, available at http://www.w3.org/TR/2004/NOTE-ws-i18n-scenarios-20040730/ [online], 2004b, [accessed 3rd May 2010].

W3C, *XML: Extensible Markup Language*, available at http://www.w3.org/XML/ [online], 2004c, [accessed 24th September 2010].

W3C, *XML: Extensible Markup Lanuage 1.1 (Second Edition)*, available at http://www.w3.org/TR/2006/REC-xml11-20060816/ [online], 2006, [accessed 24th September 2010].

W3C, *XML Schema*, available at http://www.w3.org/standards/xml/schema [online], 2010, [accessed 24th September 2010].

Wandre, S., *EDGE: Enhanced Data Rates for GSM Evolution*, Illinois Institute of Technology, Broadband Networks, 2002.

Wang, L. & Sajeev, A.S.M., *Roller Interface for Mobile Device Applications*, Proceedings of the Eighth Australian Conference on User Interface, pp. 7-13, 2007.

WebSphere, *WebSphere Version 5.1, Application Developer 5.1.1., Web Services Handbook*, available at http://www.redbooks.ibm.com/redbooks/pdfs/sg24689.pdf [online], 2004, [accessed 23rd June 2009].

Yongming, G., Dehua, C. & Jlajin, L., *Clustering XML Documents by Combining Content and Structure*, International Symposium on Information Science and Engineering, pp. 583-587, 2008.

Initial Testing Appendices

Appendix A:

The list of XML files used during the initial testing, described in Section 6.2, were downloaded from the following Web sites:

- http://xml-file.smartcode.com/
- http://download.wikimedia.org/enwiki/
- http://jeffkubina.org/data/wikipedia/dumps/enwiki/
- http://www.w3schools.com/XML/xml_examples.asp
- http://www.ii.uni.wroc.pl/~inikep/research/Wratislavia/index.htm
- http://infolab.stanford.edu/pub/movies/dtd.html

The list of XML files used is divided into three categories of small, medium and large XML files.

Small Sized XML Files					
Number	**File**	**Size (MB)**	**Number**	**File**	**Size (MB)**
1	9-9.xml	0.00029	**16**	j_caesar.xml	0.17503
2	AL_meta.xml	0.00034	**17**	much_ado.xml	0.18617
3	applicationxmltest.xml	0.00061	**18**	all_well.xml	0.20001
4	hotcop.xml	0.00093	**19**	r_and_j.xml	0.20839
5	books1.xml	0.00112	**20**	othello.xml	0.23725
6	AD_meta.xml	0.00521	**21**	CY_meta.xml	0.24565
7	TR_meta.xml	0.00546	**22**	hamlet.xml	0.26671
8	1998shortstats.xml	0.00807	**23**	BA_meta.xml	0.40947
9	4-2.xml	0.00880	**24**	MT_meta.xml	0.58403
10	schedule20030703.xml	0.00886	**25**	1998stats.xml	0.61255
11	1998styledshortstats.xml	0.01097	**26**	1998styledstatistics.xml	0.61381
12	LI_meta.xml	0.09757	**27**	1998statistics.xml	0.63830
13	dream.xml	0.13832	**28**	LU_meta.xml	0.71661
14	macbeth.xml	0.15552	**29**	IS_meta.xml	0.80913
15	merchant.xml	0.17361	**30**	RS_meta.xml	0.92141

Medium Sized XML Files					
Number	**File**	**Size (MB)**	**Number**	**File**	**Size (MB)**
1	nt.xml	0.00029	**16**	NO_meta.xml	0.17503
2	System.Design.xml	0.00034	**17**	HU_meta.xml	0.18617
3	index.xml	0.00061	**18**	ot.xml	0.20001
4	LV_meta.xml	0.00093	**19**	SE_meta.xml	0.20839
5	bom.xml	0.00112	**20**	RO_meta.xml	0.23725
6	EE_meta.xml	0.00521	**21**	DK_meta.xml	0.24565
7	wsu.xml	0.00546	**22**	casts124.xml	0.26671
8	mondial-3.0.xml	0.00807	**23**	orders.xml	0.40947
9	SI_meta.xml	0.00880	**24**	GR_meta.xml	0.58403
10	partsupp.xml	0.00886	**25**	mains243.xml	0.61255
11	file1.xml	0.01097	**26**	SK_meta.xml	0.61381
12	uwm.xml	0.09757	**27**	BG_meta.xml	0.63830
13	LT_meta.xml	0.13832	**28**	FI_meta.xml	0.71661
14	IE_meta.xml	0.15552	**29**	mscorlib.xml	0.80913
15	nt.xml	0.17361	**30**	NO_meta.xml	0.92141

Large Sized XML Files					
Number	**File**	**Size (MB)**	**Number**	**File**	**Size (MB)**
1	CH_meta.xml	10.72262	**12**	GB_meta.xml	48.82530
2	PT_meta.xml	11.33062	**13**	IT_meta.xml	58.83995
3	PL_meta.xml	14.81124	**14**	ES_meta.xml	72.26563
4	NL_meta.xml	18.89466	**15**	FR_meta.xml	73.48315
5	BE_meta.xml	21.29284	**16**	treebank_e.xml	82.09469
6	CZ_meta.xml	21.67564	**17**	SwissProt.xml	109.50109
7	initial_search_entries.xml	23.30371	**18**	dblp.xml	127.66145
8	nasa.xml	23.88995	**19**	DE_meta.xml	129.09689
9	lineitem.xml	30.79937	**20**	enwikibooks-20061201-…xml	149.05986
10	enwikinews-20061201-…xml	44.26847	**21**	AirBase_v3_statistics.csv	182.41079
11	AT_meta.xml	47.83838	**22**	psd7003.xml	683.64431

Appendix B: List of Image Files

Test File	Full Name	Original Size
Test1	G1-07.bmp	17
Test2	PUNKSKULL.bmp	47
Test3	G1-06.bmp	106
Test4	Man on the Moon-02.bmp	176
Test5	The Idea-01.bmp	250
Test6	pug.bmp	376
Test7	Untitled.bmp	641
Test8	G1_2.bmp	733
Test9	Negatives___.bmp	770
Test10	Orange IMAGE-02.bmp	1135
Test11	Be Strong.bmp	1609
Test12	Untitled.bmp	1877
Test13	The Great One-01.bmp	2305
Test14	The Great One-18.bmp	2368
Test15	Image1.bmp	3165
Test16	Image2.bmp	3601
Test17	Trees and Clouds-01.bmp	3601
Test18	Waves Amidst the Clouds.bmp	3627
Test19	CS-01.bmp	3841
Test20	Image-04.bmp	4922
Test21	The Chronicles-01.bmp	5499
Test22	Leaves.bmp	8001
Test23	Believers.com.bmp	8760
Test24	Wheels.bmp	9751
Test25	Sweet_dreams.bmp	10355
Test26	The Phoenix - Blue.bmp	11075
Test27	Image1.bmp	11251
Test28	Wishlist_for_a_New_World.bmp	11602
Test29	Fish_1.bmp	12830
Test30	Porshe_Fire.bmp	21241

Appendix C: List of Audio Files

The list of Audio files used during the initial testing, described in Section 6.2, was downloaded from the following Web sites:

- http://www-mmsp.ece.mcgill.ca/documents/audioformats/wave/Samples.html
- http://www.glooped.com/
- http://www.firstpr.com.au/0-big/
- http://www.simplosive.com/freesamples.htm
- http://www.dogstar.dantimax.dk/testwavs/
- http://compression.ca/act/act-files.html

File Name	Original Size
addf8-GSM-GW.wav	5
WIND.wav	11
M1F1-Alaw-AFsp.wav	47
M1F1-int24-AFsp.wav	138
LaidBackGroove2_140.wav	333
Shamisen-C4.wav	497
Brown_Noise.wav	862
simplosive-mb201-125bpm.wav	993
SharpBassTune_140.wav	1551
3stepoct.wav	2717
ChopsyGuitar_Together_125.wav	2977
Summer_Nt3_110.wav	3195
Summer_SM57_110.wav	3195
ChopsyGuitar_OneAfterTheOther_12.wav5	4466
PowerChords_Together_75.wav	5789
Summer_Together_110.wav	6390
every.wav	6831
HighWall2_90.wav	7351
GuitarNoise2_75.wav	8821
GuitarNoise1_75.wav	9647
cc.wav	23824
ky.wav	35055
be.wav	42562
eb.wav	42968
hi.wav	54609
bm.wav	57751
sl.wav	83037
bi.wav	86772
si.wav	87316
ce.wav	173157

Appendix D: Results for Image Compression

		GIF		JPEG		PNG	
Image	Original Size (KB)	Compr. Size	Compr. Ratio	Compr. Size	Compr. Ratio	Compr. Size	Compr. Ratio
Test1	17	4	23.52941%	3	17.64706%	4	23.52941%
Test2	47	4	8.51064%	4	8.51064%	6	12.76596%
Test3	106	13	12.26415%	12	11.32075%	15	14.15094%
Test4	176	13	7.38636%	15	8.52273%	18	10.22727%
Test5	250	10	4.00000%	9	3.60000%	21	8.40000%
Test6	376	55	14.62766%	24	6.38298%	143	38.03191%
Test7	641	7	1.09204%	9	1.40406%	9	1.40406%
Test8	733	5	0.68213%	12	1.63711%	3	0.40928%
Test9	770	9	1.16883%	39	5.06494%	4	0.51948%
Test10	1135	50	4.40529%	41	3.61233%	94	8.28194%
Test11	1609	74	4.59913%	47	2.92107%	213	13.23804%
Test12	1877	978	52.10442%	883	47.04315%	797	42.46137%
Test13	2305	90	3.90456%	76	3.29718%	107	4.64208%
Test14	2368	167	7.05236%	132	5.57432%	306	12.92230%
Test15	3165	302	9.54186%	48	1.51659%	159	5.02370%
Test16	3601	954	26.49264%	826	22.93807%	2635	73.17412%
Test17	3601	685	19.02249%	217	6.02610%	1297	36.01777%
Test18	3627	571	15.74304%	171	4.71464%	1296	35.73201%
Test19	3841	15	0.39052%	45	1.17157%	28	0.72898%
Test20	4922	70	1.42219%	245	4.97765%	43	0.87363%
Test21	5499	212	3.85525%	104	1.89125%	361	6.56483%
Test22	8001	2185	27.30909%	998	12.47344%	5043	63.02962%
Test23	8760	58	0.66210%	202	2.30594%	97	1.10731%
Test24	9751	227	2.32797%	143	1.46652%	226	2.31771%
Test25	10355	1310	12.65089%	384	3.70835%	2942	28.41140%
Test26	11075	694	6.26637%	380	3.43115%	2031	18.33860%
Test27	11251	971	8.63034%	2420	21.50920%	419	3.72411%
Test28	11602	2047	17.64351%	447	3.85278%	6989	60.23961%
Test29	12830	2645	20.61574%	881	6.86672%	6041	47.08496%
Test30	21241	595	2.80119%	551	2.59404%	1162	5.47055%
Mean			**10.69007%**		**7.59941%**		**19.29410%**
Median			**7.21936%**		**4.28371%**		**11.49662%**
Std. Dev.			**11.21166%**		**9.37601%**		**20.85987%**

Appendix E: Quality Ratings for Image Compression

	GIF	JPEG	PNG
Image		**Rating**	
Test1	3	3	3
Test2	3	3	3
Test3	3	3	3
Test4	3	3	3
Test5	3	3	3
Test6	3	3	3
Test7	3	3	3
Test8	3	3	3
Test9	3	3	3
Test10	3	3	3
Test11	3	3	3
Test12	1	2	1
Test13	3	3	3
Test14	2	3	3
Test15	3	3	3
Test16	3	3	3
Test17	3	3	3
Test18	2	2	2
Test19	3	3	3
Test20	2	3	3
Test21	2	3	3
Test22	2	3	3
Test23	3	3	3
Test24	3	3	3
Test25	1	3	2
Test26	3	3	3
Test27	3	3	3
Test28	3	3	3
Test29	2	2	2
Test30	2	3	3
Mean	2.633333	2.9	2.833333
Median	3	3	3
Std. Dev.	0.614948	0.305129	0.461133

Appendix F: Results for Audio Compression

		Compressed Size (MB)				Compression Ratio (%)				Compression Time		
File Name (*.wav)	Original Size (KB)	MP3	FLAC	MAC		MP3	FLAC	MAC		MP3	FLAC	MAC
WIND	11	9	13	5		81.818%	118.182%	45.455%		0.59	0.42	0.21
M1F1-int24-AFsp	138	48	55	94		34.783%	39.855%	68.116%		0.89	0.59	0.21
LaidBackGroove2_140	333	31	174	155		9.309%	52.252%	46.547%		1.33	0.47	0.32
Shamisen-C4	497	39	127	110		7.847%	25.553%	22.133%		1.95	0.54	0.32
Brown_Noise	862	79	575	563		9.165%	66.705%	65.313%		2.55	0.60	0.43
simplosive-mb201-125bpm	993	62	163	196		6.244%	16.415%	19.738%		1.23	0.59	0.32
SharpBassTune_140	1551	142	422	363		9.155%	27.208%	23.404%		0.30	0.59	0.54
ChopsyGuitar_Together_125	2977	272	1150	992		9.137%	38.629%	33.322%		1.79	0.74	0.87
Summer_Nt3_110	3195	291	1846	1717		9.108%	57.778%	53.740%		1.84	0.80	0.87
Summer_SM57_110	3195	291	1835	1678		9.108%	57.433%	52.520%		1.79	0.99	0.98
ChopsyGuitar_OneAfterTheOther_125	4466	406	1853	1557		9.091%	41.491%	34.863%		1.29	0.84	1.31
PowerChords_Together_75	5789	526	2411	2178		9.086%	53.584%	37.623%		2.66	1.51	1.64
Summer_Together_110	6390	581	3424	3152		9.092%	72.888%	49.327%		2.94	1.46	1.75
every	6831	621	4979	4661		9.091%	24.078%	68.233%		2.78	1.11	1.96
HighWall2_90	7351	668	1770	1575		9.087%	24.078%	21.426%		2.69	1.11	1.09
GuitarNoise2_75	8821	801	3645	3216		9.081%	41.322%	36.458%		4.80	1.39	2.29
GuitarNoise1_75	9647	877	4027	3614		9.087%	41.744%	37.462%		3.90	1.54	2.40
cc	23824	2163	13065	12466		9.079%	54.840%	52.325%		8.36	2.85	5.57
ky	35055	3181	24592	23080		9.074%	70.153%	65.839%		11.54	3.68	8.32
be	42562	3862	17330	15040		9.074%	40.717%	35.337%		12.89	4.30	9.84
eb	42968	3899	28758	27950		9.074%	66.929%	65.048%		13.55	5.11	10.17
hi	54609	4955	19242	20597		9.074%	35.236%	37.717%		16.48	5.91	12.79
bm	57751	5240	39281	37460		9.073%	68.018%	64.865%		17.69	5.93	13.67
sl	83037	7533	33512	27448		9.072%	40.358%	33.055%		28.10	6.72	18.37
bi	86772	7872	50511	48231		9.072%	58.211%	55.584%		26.08	7.28	20.56
si	87316	7922	43804	41157		9.073%	50.167%	47.136%		26.07	7.31	19.68
ce	173157	15708	70536	66524		9.072%	40.735%	38.418%		51.05	17.66	38.39

Performance Evaluation Appendices

Appendix G: Performance Evaluation Results

Server Side									
	Performance Evaluation Test 1			Performance Evaluation Test 2			Performance Evaluation Test 3		
Metric	**Web Service 1**	**Web Service 2**	**Web Service 3**	**Web Service 1**	**Web Service 2**	**Web Service 3**	**Web Service 1**	**Web Service 2**	**Web Service 3**
Original Size (KB)	112.000	9609.000	7439.000	112.000	9609.000	7439.000	112.000	9609.000	7439.000
Compressed Size (KB)	7.950	1105.900	576.000	7.950	1105.900	576.000	7.950	1105.900	576.000
Compression Ratio (%)	7.098%	11.509%	7.743%	7.098%	11.509%	7.743%	7.098%	11.509%	7.743%
Compression Time (sec)	1.027	3.340	2.991	0.018	3.440	3.001	0.035	3.430	2.568
Processing Power (%)	13.967	56.500	55.938	13.188	52.965	59.063	11.625	46.985	58.281

HTC s710									
	Performance Evaluation Test 1			Performance Evaluation Test 2			Performance Evaluation Test 3		
Metric	**Web Service 1**	**Web Service 2**	**Web Service 3**	**Web Service 1**	**Web Service 2**	**Web Service 3**	**Web Service 1**	**Web Service 2**	**Web Service 3**
Decompression Time (sec)	7.267	NA	4.573	6.553	NA	6.333	7.133	NA	4.218
Memory Footprint (MB)	0.082	0.088	0.088	0.070	0.078	0.091	0.074	0.088	0.078
Battery Consumption (%)	0.093%	0.045%	0.067%	0.087%	0.039%	0.078%	0.100%	0.036%	0.052%
Task Completion Rate (%)	100%	100%	100%	100%	100%	100%	100%	100%	100%
Overall Time (sec)	10.625	7.753	10.014	10.002	7.175	11.876	11.057	7.073	10.980

HTC TyTN II									
	Performance Evaluation Test 1			Performance Evaluation Test 2			Performance Evaluation Test 3		
Metric	**Web Service 1**	**Web Service 2**	**Web Service 3**	**Web Service 1**	**Web Service 2**	**Web Service 3**	**Web Service 1**	**Web Service 2**	**Web Service 3**
Decompression Time (sec)	4.17522	NA	3.625	5.012435	NA	4.005	4.6542522	NA	3.7125
Memory Footprint (MB)	0.084	0.086	0.084	0.075	0.076	0.088	0.074	0.082	0.079
Battery Consumption (%)	0.023%	0.009%	0.035%	0.037%	0.015%	0.042%	0.032%	0.008%	0.033%
Task Completion Rate (%)	100%	100%	100%	100%	100%	100%	100%	100%	100%
Overall Time (sec)	7.594	7.887	8.913	11.122	6.588	10.015	7.822	6.457	9.758