Faculdade de Engenharia da Universidade do Porto

Software Architecture for Multiple Smartphones

Renan Correia d'Azambuja Ramos

Master of Science in Electrical and Computer Engineering
Major in Telecommunications

Advisor: Prof. Sergio Barbosa Villas-Boas (Universidade Federal do Rio de Janeiro)
Co-advisor: Prof. Nuno Teixeira Almeida

June 2010

© Renan Correia d'Azambuja Ramos, 2010

Abstract

This thesis describes the concept of interoperability through the development of a SOA-based architecture combined with the use of Web services, with the aim to promote integration between devices of different technologies, emphasizing the introduction of smartphones into the e-business world.

The need for this project comes at a time when the world is pointing out the growing use of mobile devices and there is a lack of systems developed with the capacity to handle these new technologies. The latest models can enjoy the benefits from much of the Web and from the services already available, but there is still plenty space for growth in this sector. Besides, within the universe of mobile phones, one should highlight the fact that there are many people who still use old equipment with few resources, and companies should also see them as potential customers.

In addition, this architecture allows companies with obsolete systems / technologies not to have to rebuild their entire IT infrastructure. Using few resources, new capabilities can be added to a company's system and, thus, expand its business opportunities.

The last aspect of this thesis comes specifically from mobile phones. Because they have limited capacity, it is necessary to work on means of taking advantage of every feature they have. For this, it is proposed a model of cache with a small development on the server side, in order to reduce the use of the resources available in the devices and increase system's efficiency by providing faster response and avoiding the same action to be taken more than once.

As a result of this project, various applications for mobile phones, a few Web services and one Web client were constructed using different technologies in order to test whether the architecture works according to planned in theory. For last, it is proposed a cache system, which was developed to demonstrate that it improves the system's performance.

Keywords: Cache, Enterprise Architecture, Integration, Interoperability, SOA, Web Services

Resumo

Esta tese descreve o conceito de interoperabilidade através do desenvolvimento de uma arquitectura baseada em SOA combinada com o uso de Web services, com o objectivo de promover a integração entre dispositivos de tecnologias diferentes, enfatizando a introdução dos smartphones no mundo do comércio eletrônico.

A necessidade deste projecto surge num momento em que o mundo tem as atenções direccionadas ao crescente uso de dispositivos móveis e há uma falta de sistemas desenvolvidos com capacidade de lidar com essas novas tecnologias. Os modelos mais recentes podem aproveitar os benefícios de muito da Web e dos serviços já disponíveis, mas ainda há bastante espaço para o crescimento deste sector. Além disso, dentro do universo de telefones móveis, deve-se destacar o facto de que muitas pessoas ainda utilizam aparelhos antigos e com poucos recursos e as empresas também devem enxergá-las como potenciais clientes.

Esta arquitectura permite também que empresas com sistemas / tecnologias obsoletos não tenham a necessidade de remontar toda a sua estrutura de TI. Com poucos recursos, pode-se incluir novas capacidades ao sistema da companhia e, com isso, ampliar suas possibilidades de negócios.

O último aspecto desta tese vem especificamente dos telefones móveis. Por terem a capacidade limitada, é preciso trabalhar em meios de tirar proveito de cada característica que eles possuem. Para isso, é proposto um modelo de cache com um pequeno desenvolvimento no lado do servidor, de modo a reduzir o uso dos recursos disponíveis nos aparelhos e aumentar a eficiência do sistema, fornecendo respostas mais rápidas e evitando que se faça a mesma ação mais de uma vez.

Como resultado deste projecto, foram desenvolvidas diversas aplicações para telefones móveis, alguns Web services e um cliente Web usando tecnologias diferentes, de modo a testar se a arquitectra funciona de acordo com o previsto em teoria. Por último, é proposto um sistema de cache, que foi desenvolvido para demonstrar que melhora a performance do sistema.

Acknowledgements

I would like to express my gratitude to my advisor, Professor Sergio Villas-Boas, for his support, patience and encouragement throughout my studies. His suggestions and guidance were essential to this thesis.

My thanks go also to my dear parents and my sister, Renata, for their understanding and support, especially during the months I was writing this thesis. I must highlight the gratitude to my father, who has initiated the countdown at the moment I walked into the university.

A special thanks to my grandparents, Maria and José, for their undeniable support in order to guarantee that I could reach this point. They provided all means for me to get here.

I could not forget to mention my family in Portugal and in Brazil, especially Alberto e Helena, who took me as a son while I was in Porto.

Moreover, many thanks to my teachers, colleagues and friends, who have somehow contributed for me to be where I am now. Caldeira, both Brunos, Pires, Marco, Fred, Raposo, Edu, Carlos and Miguel, thank you. For all not mentioned friends, don't be jealous, there isn't enough space for everybody in one paragraph.

Last, but not least, I would like to thank my fiancée Aline Soares for her understanding, support and love during the past few years. Her family is also part of this. Thank you all!

Table of Contents

Abstract iii

Resumo v

Acknowledgements vii

Table of contents ix

List of figures xi

List of Abbreviations xiii

Chapter 1 1

Introduction 1
- 1.1 - Background 1
- 1.2 - Purpose / Objectives 2
- 1.3 - Limitations 3
- 1.4 - Target Audience 3
- 1.5 - Timeline and Method 3
- 1.6 - Thesis Outline 4

Chapter 2 5

Theoretical Background 5
- 2.1 - Technologies found in mobile phones 5
- 2.2 - XML. 9
- 2.3 - Protocols 9
- 2.4 - Apache Axis, Server and Tomcat 10
- 2.5 - IDEs 10
- 2.6 - SOA, Web Services and Cache 11

Chapter 3 13

The Concept of Interoperability 13
- 3.1 - Services and the Enterprise Service Bus 13
- 3.2 - Web Services 16
- 3.3 - SOA 18
- 3.3.1 - SOA Governance 22
- 3.3.2 - SOA Maturity 23
- 3.3.3 - SOA Security 25
- 3.4 - The Hype of Handheld Devices 27
- 3.5 - What About Old Mobile Phones? 28

3.6 - Technical Implementation 29

Chapter 4 39

The Cache Issue (On Smartphones) 39
4.1 - Cache Level 1 (L1) 41
4.2 - Cache Level 2 (L2) 42
4.3 - Results and Performances Comparison 58

Chapter 5 63

Conclusions and Future Work 63

References 67

List of figures

Figure 1-1: Project Timeline.....3
Figure 2-1: The Java Platform.....6
Figure 2-2: CLDC Platform.....6
Figure 2-3: Document divided into layers 7
Figure 2-4: Mobile OS market share in 2009.....8
Figure 2-5: Example of a XML file.....9
Figure 2-6: Communication layers and protocols.....10
Figure 3-1: Virtual vs. Actual components.....14
Figure 3-2: SOA, CBD and OO in context.....15
Figure 3-3: Example of a SOA-based system.....16
Figure 3-4: Example of a SOA-based system.....16
Figure 3-5: Each application is distributed over four logical machines.....18
Figure 3-6: The Silo approach vs. the SOA approach.....20
Figure 3-7: Technical Evolution of SOA.....21
Figure 3-8: SOA lifecycle.....22
Figure 3-9: SOA Governance Framework.....23
Figure 3-10: SOAMM (Sonic Software Corp.'s SOA Maturity Model).....25
Figure 3-11: IBM's SOA Security Reference Model.....26
Figure 3-12: Comprehensive Model for securing Information Systems.....27
Figure 3-13: Technical implementation timeline.....29
Figure 3-14: System's design (first technical implementation).....29
Figure 3-15: Mobile application constructed using Qualcomm's Plaza Internet Mobile.....30
Figure 3-16: Mobile application constructed using J2ME.....31
Figure 3-17: Mobile application constructed using Flash Lite.....32
Figure 3-18: XML Web service displayed using Internet Explorer.....33
Figure 3-19: XML Web service displayed using Opera.....33
Figure 3-20: XML Web service displayed in Safari.....33
Figure 3-21: XML Web service displayed using Chrome.....34

Figure 3-22: XML Web service embedded in a Web page.....34
Figure 3-23: Authentication via Android..... 36
Figure 3-24: Authentication via Web browser..... 36
Figure 3-25: Temperature service – Mobile client side (Android) 37
Figure 3-26: Temperature service – Web browser..... 38
Figure 3-27: Sum service via Android..... 38
Figure 3-28: Sum service via Web browser..... 38
Figure 4-1: The logic behind L1 – the usual cache system.....41
Figure 4-2: XML file from a famous news-Web service (it contains the timestamp).....43
Figure 4-3: The logic behind L2 – better performance.....47
Figure 4-4: The application constructed to test the cache system (football news service)....58
Figure 4-5: The introduction of a new layer between the client and the Web service.....59
Figure 4-6: Comparison between systems in scenario 1.....60
Figure 4-7: Comparison between systems in scenario 2.....61

List of Abbreviations

API	Application Programming Interface
CICS	Customer Information Control System
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
ebXML / XML	Electronic Business using eXtensible Markup Language
EJB	Enterprise JavaBeans
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
GPRS	General Packet Radio Service
HTML	HyperText Markup Language
HTTP / HTTPS	HyperText Transfer Protocol / HyperText Transfer Protocol Secure
IDE	Integrated Development Environment
Java EE / J2EE	Java 2 Enterprise Edition
Java ME / J2ME	Java Platform, Micro Edition (former Java 2 Platform, Micro Edition)
MIDP	Mobile Information Device Profile
NSTISS	National Security Telecommunications and Information Systems Security
OEM	Original Equipment Manufacturer
PHP	Hypertext Preprocessor
RPC	Remote Procedure Calls
SDK	Software Development Kit
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
TTL	Time-to-Live
UDDI	Universal Description, Discovery and Integration
URI	Unique Resource Identifier
WAP	Wireless Application Protocol
WSDL	Web Service Definition Language

Chapter 1

Introduction

This chapter introduces a few considerations regarding the project so that the reader can get familiar with the topic before having it explained in details. In this first chapter, the relevance of the subject for the world today is presented. Furthermore, it is described what the project entails, the limitations that have been identified even before getting started, the target group and there is also an outline of how this dissertation is arranged. Thus, the aim is to give the reader a short overview of the whole project.

1.1 - Background

Since the first computer was invented, software architecture is a hype topic. Currently, there are many proposed architectures that promise to put an end to the compatibility issue between different technologies, and to maximize the system performance. With the large increase in the use of mobile devices, such as PDAs and smartphones, the system must be designed to work in whatever gadget someone might use.

Among all architectures, the Service-Oriented Architecture (SOA) has proven to be the future. In fact, it is not a new implementation, but widespread use of SOA is. Companies that don't yet adopt the service-oriented architecture model don’t know how it can change their enterprise's IT architecture and are placing themselves at a competitive disadvantage. By 2008, SOA was a prevailing software engineering practice, ending the 40-year domination of monolithic software architecture. SOA needs an interoperability platform to link service consumer applications with services; several technologies, including Web services, can do it, but they must often be used in combination. Although SOA does not require Web services — Common Object Request Broker Architecture (CORBA) is SOA-based, for example —, Web services are based on accepted standards and are driving SOA to the mainstream. Currently, SOA and Web services are implemented together in more than 75 percent of new SOA or Web

services projects [1]. It can be simply described as a method of constructing software and information systems that aims to produce flexible, reliable and future-proof systems. It will be described in details in chapter 3.

The sales of laptops, mobile phones and tablet type devices have grown tremendously over the last few years as these devices have become essential tools for many different types of mobile workers, from sales people to service technicians, who need to access and enter data electronically. In recent years, the market for small handheld computers such as Personal Digital Assistants (PDAs) has been evolving even more rapidly than the market for conventional PCs and laptops, and trends are pointing toward exploiting mobile technology even more [2].

The focus in this thesis can be summarized in one word: interoperability. This word can be defined as the ability of two or more systems or components to exchange information and to use the information that has been exchanged [3] or, going further, as being able to accomplish end-user applications using different types of computer systems, operating systems, and application software, interconnected by different types of local and wide area networks [4]. That means mobile phones can work along with laptops, desktop computers and whatever gadget someone wishes to use.

However, as the world is far from being perfect[1], new problems have come up with the new architecture: all devices have different capabilities. That means limitations may apply to one or more gadgets; others will work perfectly though. This fact will be described in section 1.3 and later along this dissertation.

For last, it is important to notice that all this data traffic requires data plans for mobile users and it makes necessary to find a way to make this cost as low as possible.

1.2 - Purpose / Objectives

This project has two main tasks: to develop SOA-based Web services to show how the interoperability can be achieved using this architecture. Then, a few client applications for smartphones should be constructed to test it along with a Web browser, proving that displaying the information does not depend on the device's technology; the difference is only on how the data is presented. As the focus is on how to integrate mobile phones to the business world it is important to highlight that, in order to maximize the device's performance, the applications should process the least information possible and leave most of it to the server side. The programs should only work on how to display the information, which means working on the graphical issue. The applications were developed using J2ME,

1 George Houston. (2010). *The Correspondent*. Nabu Press

Flash Lite / ActionScript, HTML / PHP, the Qualcomm's Plaza Internet Mobile platform (using HTML and JavaScript) and the Android SDK.

Since the capabilities of mobile phones – even in recent models – are not as good as the capabilities of laptops and desktop computers, an enhancement is proposed in this thesis: the use of caching systems to improve the application's performance. Two different levels of caching systems are proposed and it is done a comparison between them.

1.3 - Limitations

SOA is a concept that covers a huge part of IT; hence it would be impossible to describe all aspects of it. The focus in this report is on how SOA facilitates systems integration and the interoperability between them, especially regarding smartphones, since these mobile devices are the hype of the moment. In fact, as phones have limited capabilities, a study of an enhancement to improve their performance was performed. Actually, this study has gained almost the same importance as the study of the architecture itself (in this report only, of course – the importance of SOA is not measurable).

All tests were executed in emulators, since there were not devices available for testing in "real life".

1.4 - Target Audience

The primary target group is the same of all theses: the examiner and advisors. However, this report is not restricted to a few people; all people interested in SOA and in applications for mobile phones represent the target audience as well as all organizations with at least a basic IT level. Colleagues are somehow part of this project and they might also be interested in reading what is written here.

1.5 - Timeline and Method

The timeline followed to accomplish all tasks is illustrated by the following figure:

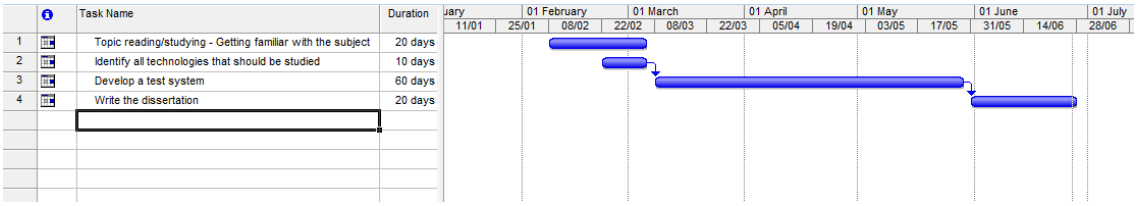


Figure 1-1: Project Timeline

In order to achieve all goals, several applications were developed as well as a few Web services to test the interoperability and show that, using this architecture, different technologies can use the same service; they differ only in how the requested information is displayed on the screen. Moreover, the applications were constructed using different technologies (see sections 1.2 and 3.6), so that each program can process the information and show the results in a particular way other than the rest of the programs. In chapter 3, it is possible to verify those differences.

To develop, test and compare the caching system as an enhancement it was used the Flash Lite application and an existing Web service. The results obtained and the comparisons done are shown in chapter 4.

1.6 - Thesis Outline

This thesis is divided into six main chapters. The first (and current) one gives the reader an overview of the project, introducing the concepts that will be studied and explained later in the following chapters. It describes the project's context, objectives, limitations, target group, timeline, and also the method used to achieve success in every proposed task.

The second chapter discusses the technologies involved in this project. Here, there are explanations on each component, programming language, application and architecture used to construct every piece of the project so that the reader can have a strong and solid knowledge of the topic.

The third chapter is fully dedicated to the combination SOA / Web services and why it represents the best option to be adopted by all organizations today. This chapter emphasizes how important this architecture / combination is to provide services to multiple clients using different technologies. By the end of the chapter, the reader should be completely aware of the proposed solution and understand the benefits everyone (i.e., both companies and clients) can take from it. Also, it is shown in this chapter the applications and services developed to illustrate the scenario described along the same chapter.

The fourth chapter deals with the caching system proposed as an enhancement for applications in mobile phones. The results are described and compared so that it is possible to notice how important and useful this improvement is. As a new proposal, it represents a new idea for organizations.

The fifth chapter discusses the economic impacts that this systems architecture may cause. Of course, as expected, the impacts are positive and the adoption of this architecture by almost all organizations shall be considered a matter of time.

Finally, the sixth and last chapter concludes the dissertation, discussing how well the project has gone and what conclusions can be drawn from the results as well as the future works that may complement this project.

Chapter 2

Theoretical Background

This chapter deals with the theory involved in this project directly and indirectly related to the topic. That means the technologies and protocols as well as a few concepts are explained in the following sections while the main aspects of the thesis are found in the next chapters. Thus, it is not expected SOA and caching systems to be described here. The reader that is not familiar with the topic should read this chapter carefully.

2.1 - Technologies found in mobile phones

This section discusses the technologies involved in the development of mobile applications in this project.

J2ME is a technology that allows software development for mobile devices and embedded systems. It is the Java platform for mobile phones, PDAs, remote controls and other compact devices. Java ME technology was originally created in order to deal with the constraints associated with building applications for small devices with the purpose to fit such a limited environment and make it possible to create Java applications running on small devices with limited memory, display and power capacity. This technology is the most compatible, considering mobile phones (from old to recent models).

Java ME platform is a collection of technologies and specifications that can be combined to construct a complete Java runtime environment specifically to fit the requirements of a particular device or market. This offers a flexibility and co-existence for all the players in the eco-system to seamlessly cooperate to offer the most appealing experience for the end-user. The Java ME technology is based on three elements [5]:

- a configuration provides the most basic set of libraries and virtual machine capabilities for a broad range of devices;
- a profile is a set of APIs that support a narrower range of devices;
- an optional package is a set of technology-specific APIs.

Java ME has been divided into two groups; the following figure illustrates this division:

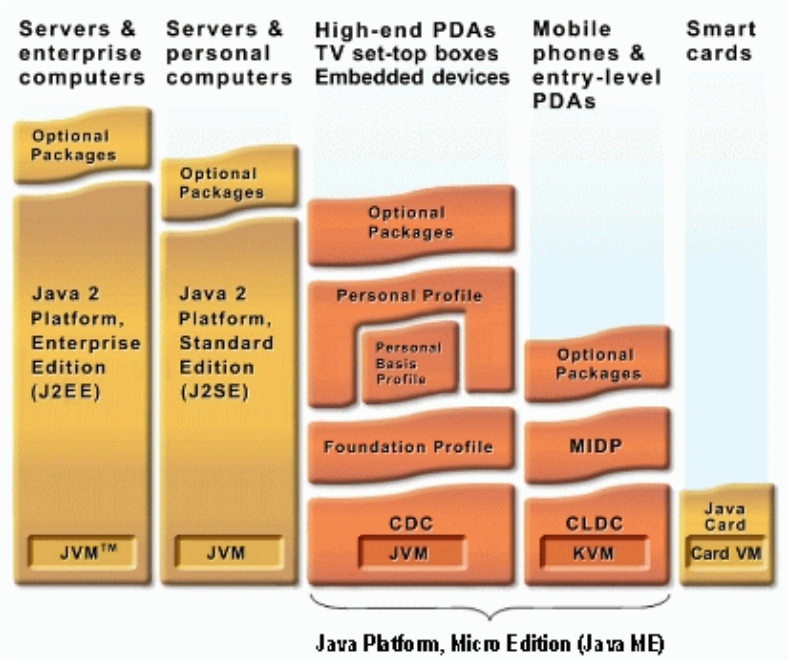


Figure 2-1: The Java Platform [5]

In this project, the chosen configuration was the CLDC (Connected Limited Device Configuration):

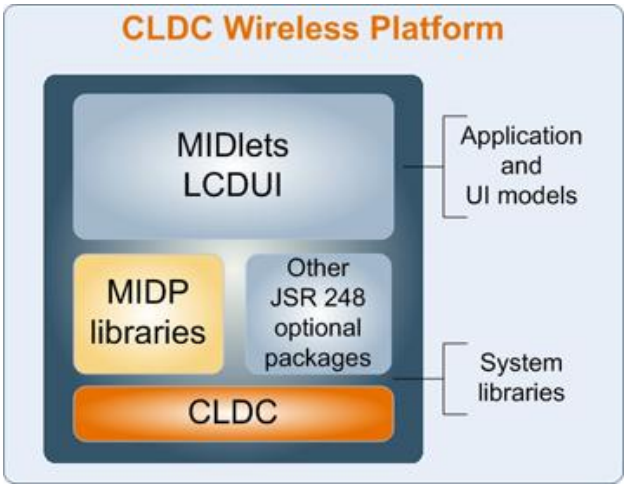


Figure 2-2: CLDC Platform [5]

Flash Lite is a development technology specifically designed for mobile phones. It is highly portable and can be ported to a variety of leading operating systems on mobile and digital home devices. Many of the world's largest OEMs and operators are shipping Flash Lite enabled devices currently. This technology is capable of producing outstanding animations and user-friendly look. It also supports loading and parsing of external XML data in content created with Adobe Flash technology using the same XML handling methods as Flash Player [6].

One of the most important things to state about Flash Lite is how easy it is to manipulate data, such as shapes, images, audio or video files. Besides, it supports the ability to store data locally and retrieve this information, turning on the possibility of using a caching system (like the one described in chapter 4). It also support ActionScript, an object-oriented programming language often used to construct Rich Internet Applications (RIA).

Other technology that can be used in mobile devices is HTML. It is the most usual language used to produce Web pages. It provides means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It allows images and objects to be embedded and can be used to create interactive forms. Qualcomm Plaza Internet Mobile uses HTML to build widgets. To promote the separation between format and content in a document, the Cascading Style Sheets (CSS) is used. It is very usual to find a CSS development to improve HTML documents.

One more technology that was used during the development of this project is JavaScript (JS). It is typically used to enable programmatic access to computational objects within a host environment. It can be characterized as a prototype-based object-oriented scripting language that is dynamic, weakly typed and has first-class functions. It is also considered a functional programming language because it has closures and supports higher-order functions [7].

The following figure illustrates the separation between these technologies in a document:

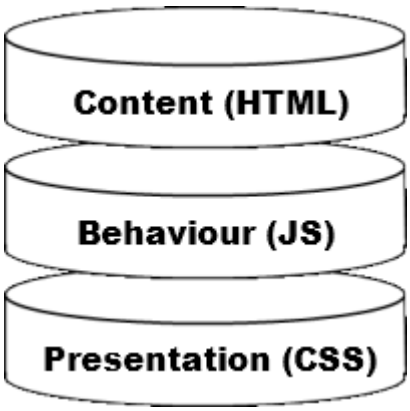


Figure 2-3: Document divided into layers

The technologies described above were directly used in this project. However, there is a need to discuss other technologies that are related to mobile phones, such as mobile operating systems, which is the operating system that controls a mobile device — similar in principle to an operating system such as Mac OS, Linux or Windows, that controls desktop computers or laptops. However, they are currently somewhat simpler, and deal more with the wireless versions of broadband and local connectivity, mobile multimedia formats, and different input methods [9]. The main OS are Symbian, iOS, RIM's BlackBerry, Windows Mobile and Android.

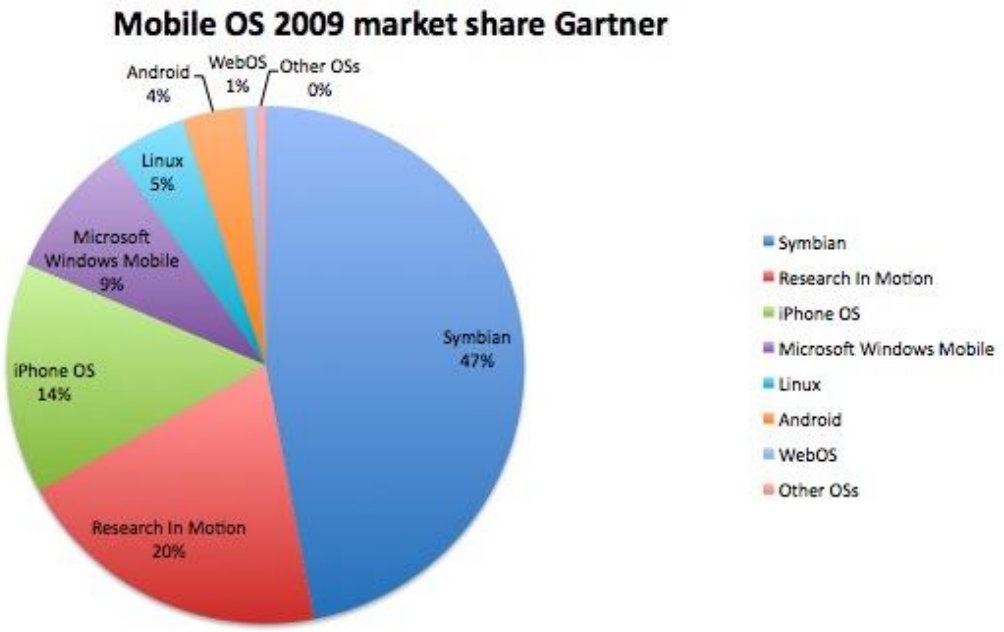


Figure 2-4: Mobile OS market share in 2009 [10]

Regarding the OS, three should be highlighted: Symbian, iOS and Android. Symbian because it is the most used system in the world and it is compatible with C++, J2ME, Flash Lite, Python, Perl and Ruby; no other OS can run applications from so many different technologies. The iOS is the OS developed by Apple and applications for this OS are usually impressive, especially regarding the graphical user interfaces; the number of iOS users is increasing widely. For last, Android, which is an OS developed by Google, is the most recent OS developed and it's an open-source project with a huge increase over the past two years. A problem related to Android is that it does not use established Java standards; it only reuses the Java language syntax instead. It does not provide the full-class libraries and APIs bundled with Java SE or ME [11].

2.2 - XML

XML is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. Although XML's design focuses on documents, it is widely used for the representation of arbitrary data structures, as in Web services [12].

Some of the main XML characteristics are:

- Separation between content and formatting;
- Simplicity;
- Possibility of creating unlimited tags;
- Focus on information, not the look.
- A database can write a XML file using an application and another database can read the data from it.

The figure below illustrates how a XML file looks like:

```
- <xml>
    <title>XML test</title>
  - <text type="test">
    - <body>
      - <p>
          Though this is a very pared
          <lb />
          down XML document, it nonetheless
          <lb />
          provides an example of how an XML
          <lb />
          document displays on the web without
          <lb />
          the intercession of a stylesheet or
          <lb />
          other conversion program.
        </p>
      </body>
    </text>
  </xml>
```

Figure 2-5: Example of a XML file

2.3 - Protocols

Since this topic is not the focus in this thesis, it won't be extended. To keep it simple, it's possible to state that for now the protocols that the reader must be familiar with are: TCP/IP, HTTP and HTTPS. The TCP/IP is a set of protocols regarding the communications

between computers in a network. HTTP is a communication protocol used in information systems. HTTPS is a combination of the HTTP with the SSL/TLS protocol to provide encryption and secure (website security testing) identification of the server [13]. The other important protocols regarding this project are well described in chapter 3. The figure below illustrates the layers and protocols involved in the communication.

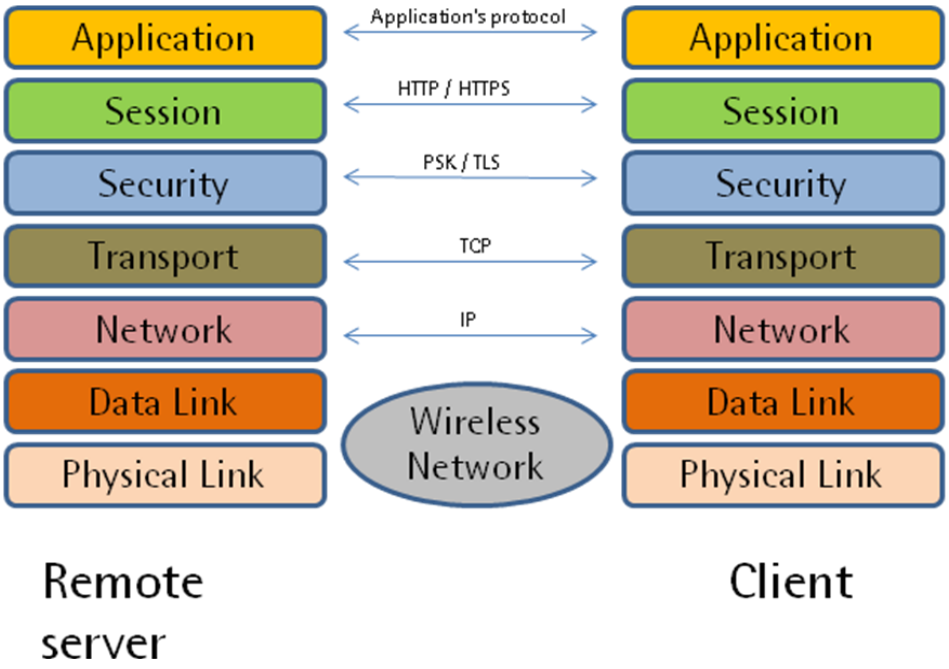


Figure 2-6: Communication layers and protocols

2.4 - Apache Axis, Server and Tomcat

The Apache Axis is an open-source framework, XML-based, consisting of a Java implementation of the SOAP server. By using Axis, developers can construct distributed applications. The Apache Server is most successful free Web server, which is a computer program responsible for dealing with the HTTP requests and responses. The Apache Tomcat is an open-source servlet container that has the ability of acting as a Web server providing a "pure Java" HTTP Web server.

2.5 - IDEs

In this project, the IDEs used to build the applications were NetBeans and Eclipse. As can be inferred from the expression Integrated Development Environment, they are both

development environments that help people construct applications using several programming languages.

In this project, the Web client (see chapter 3) was written in PHP, constructed using NetBeans and tested hosted on a virtual machine. The Android client was developed using the Eclipse IDE enhanced with the official Android plug-in.

2.6 - SOA, Web Services and Cache

These topics are explained in details in the next chapters. SOA and Web services are the primary subjects of chapter 3 and the caching system is the main topic of chapter 4.

Chapter 3

The Concept of Interoperability

This chapter discusses the concept of interoperability as a combination of SOA and Web services, so that people using different technologies can access the same service from several devices and each person using his / her own system. It is shown how this architecture allows enterprises to expand their business targeting more clients, especially those from the mobile world.

3.1 - Services and the Enterprise Service Bus

In a service-oriented architecture, the services are the logic behind the scenes, that is, when the user requests or sends any data, the services are the responsible for treating the response. A service is a contractually defined behaviour that can be implemented and provided by a component for use by another component [16].

The term “services” does not imply Web services; although Web services are well known implementations of services; other specialized implementations include J2EE and .NET. A service presents a description, which consists of the technical parameters, constraints and policies that define the terms to invoke the service. Each service should include a service definition in a standardized format, this enables applications and human actors to examine the service description and determine issues such as, what the service does, how they may bind to it, and what security protocols (if any) must be used with it. The declaration may also include details about any implied process or other legal or business terms that occur when the service is invoked. While the nature of the services themselves may vary, a common standard for declaring a service is desirable when building an infrastructure. Two such standards exist today: Web Services Description Language (WSDL) and ebXML’s Collaboration Protocol Profile. WSDL will be defined later in this report; the other standard won’t be referenced, though, since it is less used and was not studied in this thesis.

There are many definitions for the word "service". For example, CBDI defines a service as:

"*... a vehicle by which a consumer's need or want is satisfied according to a negotiated contract (implied or explicit) which includes Services Agreement, Function Offered...*" [32].

Another good definition of service comes from the World Wide Web Consortium (W3C). It states that a service is:

"*... an abstract resource that represents a capability of performing tasks that represents a coherent functionality from the point of view of provider entities and requester entities. To be used, a service must be realised by a concrete provider agent.*" [33]

CBDI's definition matches the one from Adobe (quoted later in this report) regarding the contractual aspect and they are both often found in SOA-related literature. The concept of service shall not be considered a replacement for the processes used to build systems today; it should be understood as an evolution though.

It is usually found in literature that the concepts involved in SOA are very abstract. To illustrate part of that abstraction, the figures below show the relation between virtual and actual components, as will be mentioned later in section 3.3:

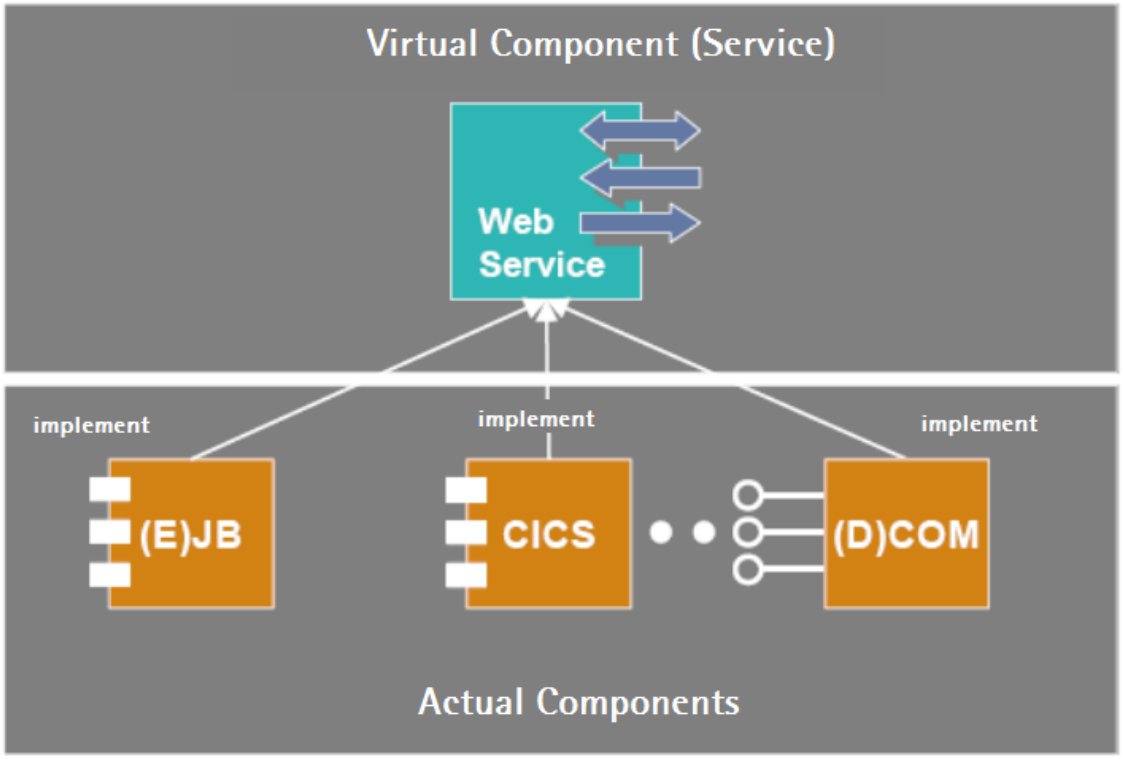


Figure 3-1: Virtual vs. Actual components [34]

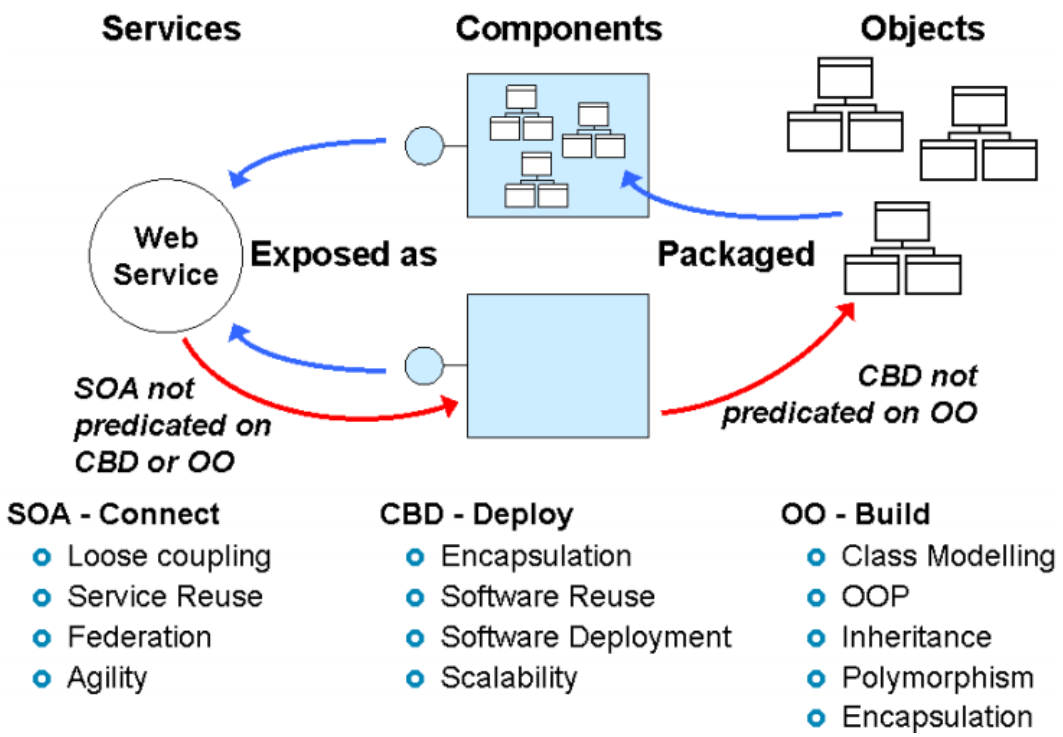


Figure 3-2: SOA, CBD and OO in context [19]

An Enterprise Service Bus (ESB) consists of a software architecture which provides fundamental services for complex architectures via an event-driven and standards-based messaging-engine (the bus). An ESB generally provides an abstraction layer on top of an implementation of an enterprise messaging system, which allows integration architects to exploit the value of messaging without writing code. Unlike the more classical enterprise application integration approach of a monolithic stack in a hub and spoke architecture, an ESB builds on base functions broken up into their constituent parts, with distributed deployment where needed, working in harmony as necessary [35][36].

In an enterprise architecture that makes use of an ESB, the services / applications will communicate throughout the bus, which acts like a message broker between them. It results in a reduction in point-to-point connections to allow applications to communicate; hence the adjustment process in a system needed because of a change in one or more components becomes easier and simpler.

The following figures illustrate two examples of systems operating in a Service-Oriented Architecture:

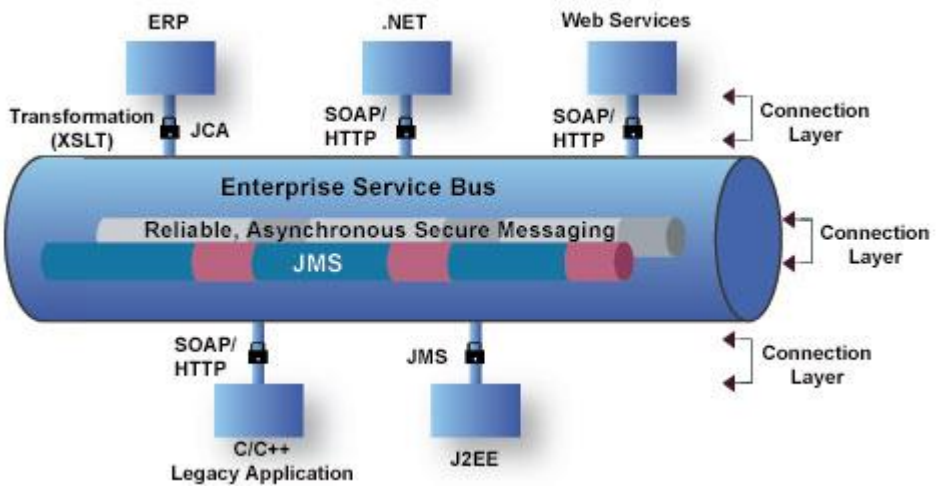


Figure 3-3: Example of a SOA-based system [37]

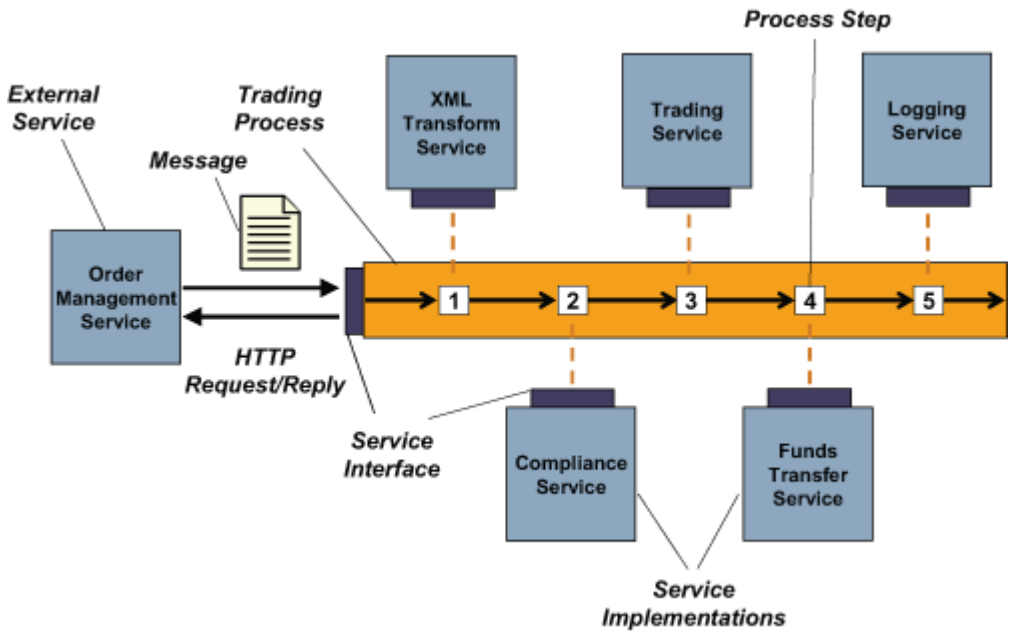


Figure 3-4: Example of a SOA-based system [38]

3.2 - Web Services

Service-Oriented Architecture does not specifically mean Web services, .NET, J2EE, CORBA or ebXML. These are specialized SOA implementations that embody the core aspects of a service-oriented approach [16]. As described before, this thesis focus on the implementation of Web services in order to reach the proposed objective. The definition of a Web service has changed in the past few years. Currently, according to the W3C, Web services are:

"*... a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-to-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialisation in conjunction with other Web-related standards*" [33].

However, in the past, W3C defined Web services as:

"*... a software application identified by an URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML-based messages via internet-based protocols...* " [39].

As discussed before, services present a description which consists of several data from the service itself, such as technical parameters, constraints and policies that define the terms to invoke the service. The Web Service Description Language (WSDL), found many times in this report, is a XML-based language used to describe Web services as a service contract. It is a XML document that describes the service as well as specifies how to access it and which operations or methods are available.

The Simple Object Access Protocol (SOAP), also frequently found in this thesis, is the protocol that specifies the communication between a client and a server in SOA. One more acronym that is usually found when discussing Web services is UDDI, which represents an initiative with the objective to accelerate the use of Web services; it is a Web service to manage information on services providers, implementations and metadata. The figure below illustrates the relation between Web services, protocols, the WSDL and the client:

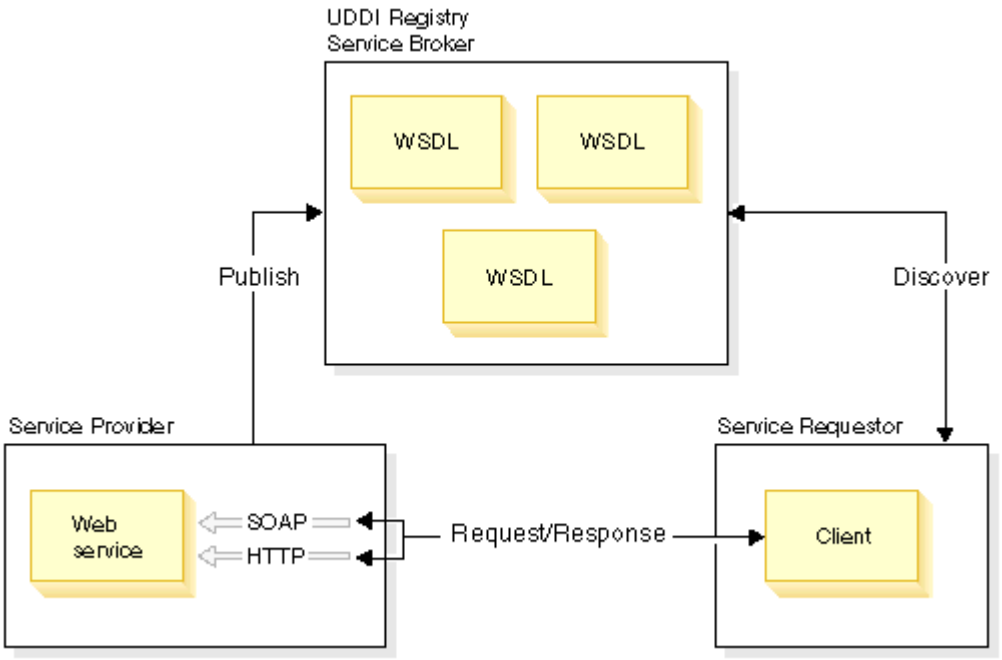


Figure 3-5: Each application is distributed over four logical machines [40]

Web services are identified by an URI and described and defined using XML. One of the reasons that make Web services so attractive is the fact that this model is based on standard technologies, such as XML and HTTP, which are often used to make interactive services available and can be accessed by other applications using, for instance, the Simple Object Access Protocol (SOAP). The data transport is done usually via HTTP (or HTTPS) and the data is transferred in XML format, encapsulated by SOAP.

One of the primary advantages of the XML Web services architecture is that it allows programs written in different languages on different platforms to communicate with each other in a standards-based way. Comparing this approach to older proposals, for instance, if we compare XML Web services to Common Object Request Broker Architecture, it is possible to notice the difference between them due to the fact that SOAP is significantly less complex than earlier approaches, so the barrier to entry for a standards-compliant SOAP implementation is considerably lower. Another significant advantage that XML Web services have over previous efforts is that they work with standard Web protocols, such as XML, HTTP and TCP/IP. Most of the companies today already have a Web infrastructure and people with knowledge and experience in maintaining it; hence it is less expensive to adopt this XML Web services model than adopting other technologies.

3.3 - SOA

Today, the main challenges for a CIO are: to demonstrate that IT adds business value, to align IT with business objectives without increasing costs, to provide quick answers to the

organizations' issues and to help people (and the company in general) to be more efficient. For many years, companies have been trying to find the perfect system design to ease their business models changes and expand their capabilities in order to increase the number of customers, for instance, by aggregating users from different technologies and letting them access whatever service the company provides.

"*A recent survey of Fortune 500 companies indicated that over 80% had altered their business model in a giver two-year period. Two thirds of these – roughly half of the total respondents – claimed that this business change had been constrained by inflexible IT. In a survey by IBM Business Consulting Services, 90% of CEOs expect to transform their enterprise to become more responsive, particularly to customer demand.*" [15]

SOA is the answer to those challenges. The Service-Oriented Architecture is an evolution of the Component Based Architecture, Interface Based Design (Object Oriented) and Distributed Systems of the 1990s, such as DCOM, CORBA, J2EE and the Internet in general [16]. Although there are many different definitions to this architecture, the following has been chosen for its relation to the focus on this dissertation – business and interoperability:

"*SOA is a conceptual architecture for implementing dynamic e-business. It is a way of designing a software system to provide services to end-user applications or other services through published and discoverable interfaces. Services provide a better way to expose discrete business functions and, therefore, an excellent way to develop applications that support business processes.*" [17]

The definition above fits perfectly this report intent of focusing on e-business. It is easy to imply that organizations today are concerned with the dynamic business and with the need to change constantly in order to better fit customers demand. That's part of the keys to implement the SOA model. However, in the past, companies did not think of that architecture. Historically speaking, there has been a great evolution in software architectures. The figure below illustrates the evolution from silos to SOA:

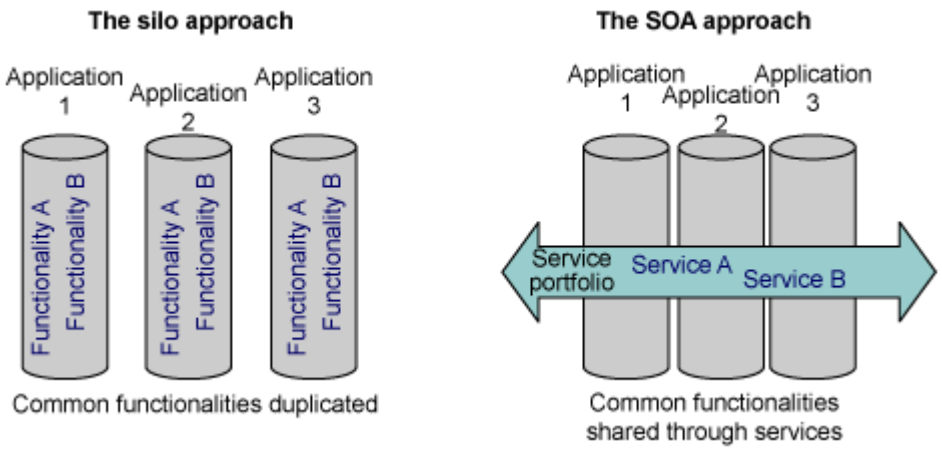


Figure 3-6: The Silo approach vs. the SOA approach [18]

SOA promises to free the business layer from the restrictions found in the technology areas, since it makes business functionalities available to users as a set of reusable services at an enterprise service bus. Here is a list with a few characteristics of SOA [19][20][21]:

- Agile systems: SOA enables organizations to easily change their systems; not only in terms of functionality, but also in terms of changing platform or vendor, for example;
- Easy integration with partners – cutting expenses: SOA has the ability to integrate systems across different platforms;
- Improvement in Return of Investment: SOA can eliminate costs by replacing proprietary software for open standards-based Web services technologies;
- Platform and Geographical independence: SOA works across multiple platforms and physical location can be easily changed without needing to stop any service;
- Reuse: Reuse has been driving the latest programming / development methods. SOA allows the same service to be reused easily no matter the technology. Logic is divided into services intending to promoting reuse.

In addition, one might take a few factors into account when defining a SOA implementation [22]:

- Efficient use of system resources;
- Service maturity and performance;
- Enterprise Application Integration (an integration framework composed of a collection of technologies and services which form a middleware to enable integration of systems and applications across the enterprise).

To better explain the definition of SOA, it's possible to assert that it consists of six main elements:

- Conceptual overview: well-defined business objectives; clear IT strategic objectives; governance model with well-defined policies;
- Services: central artifact of any SOA; promote reuse and interoperability; require a service-design model. Services are virtual components that can be grouped into different categories;
- Enabler technology: IT mechanism that allows services and business processes to be implemented in a SOA. Currently, the most used and accepted mechanisms are Web services. Systems integration aspects, such as EAI, EDI and ESB as well as legacy systems evolution are also important;
- Governance and Policies: see section 3.3.2;
- Metrics: essential to analyse SOA investment, results and performance;
- Organizational model: SOA requires a cultural change within organizations.

To better understand how the process of building different architectures has come to the current stage, the following figure presents a trace from an era where monolithic applications were popular, when the first large-scale systems were deployed by businesses and other enterprises. The evolution from procedural, or unstructured code, to Object-Oriented, or structured code, was the first step in the evolution to a modern SOA:

EVOLUTIONARY STEP	EXAMPLE
Monolithic	Large scale application using a procedural coding methodology
Structured or Object Oriented	Dividing applications into units of logic based on functionality. The first steps of SOA.
Clients and Servers	The logical progression of OO – bundling groups of functions on one computer (server) and invoking them from another (client).
3 Tier	Adding an extra layer to interaction. The primary driver was to create an interface that was agnostic to the specific environment of the server. For example, if you make an HTTP request to a server, the middle tier translates your HTTP get() request into the native format that is needed to get the resource requested.
N^{th} Tier	Layered request-response calls between applications. Portal development relied on this concept.
Distributed Objects	A ubiquitous and heterogeneous system of many distributed, orthogonal objects rather than a simple, 2 or 3 computer interaction.
Components	Aggregating objects into logical components that achieve specific functionality (often mapping to a component or the builder's requirements) and creating interfaces to those components. An example is a database server used as a persistent data-storage component for CRM software.
Service Oriented Components	A ubiquitous environment of orthogonal components interacting in a peer-based environment, often using service provider proxies (interfaces based on widely accepted standards) to offer services.

Figure 3-7: Technical Evolution of SOA [16]

Like every other architecture, SOA has a lifecycle, which can be illustrated as in the figure below:

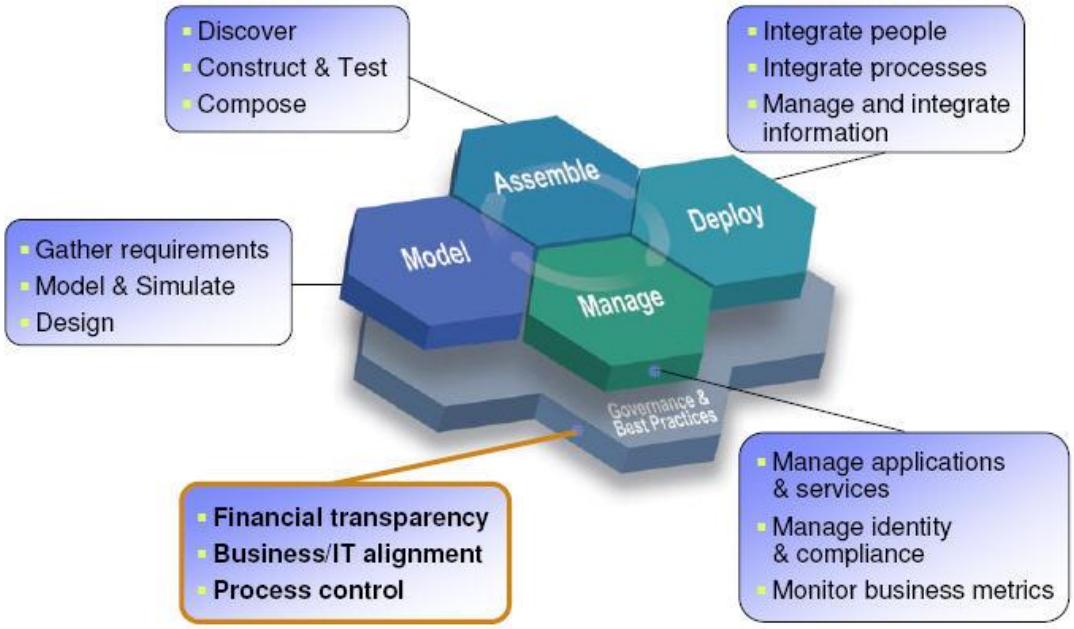


Figure 3-8: SOA lifecycle [23]

After defining the most important concepts related to SOA, the following sections will deal with the explanation of three very important aspects present in SOA: governance, maturity and security.

3.3.1 - SOA Governance

Questions and concerns regarding IT governance have been around since the introduction of IT in companies [43]. The results from governance are well defined structures, processes and procedures in enterprises that have designed those mechanisms, and these factors lead organizations to an increase in quality and in results and, thus, to deliver better products, achieving more competitiveness within the IT world.

SOA governance is considered to be essential to the success of projects and companies today. "*Lack of working governance mechanisms in midsize-to-large (greater than 50 services) post-pilot projects will be the most common reason for project failure*" [24]. In addition, "*it has been said that 70% of the integration infrastructure projects that fail were due to poor management issues, and not because of software and technology difficulties. And while enterprise integration is inherently complex, one of the major stumbling blocks is the lack of a common vocabulary and body of knowledge around the asynchronous messaging*

architectures used to build integration solutions" [25]. Sandy Carter, vice-president of SOA & WebSphere Marketing, Strategy and Channels at IBM Corporation, states that SOA governance's main task is to assist the process establishment of shared services, reinforcing standards and policies [26]. Furthermore, Marks and Bell allege "*an SOA governance model defines the various governance processes, organizational roles and responsibilities, standards and policies that must be adhered to in an SOA conceptual architecture*" [27]. Carter also asserts that an SOA governance framework should be there to follow up with processes. The figure below illustrates this framework:

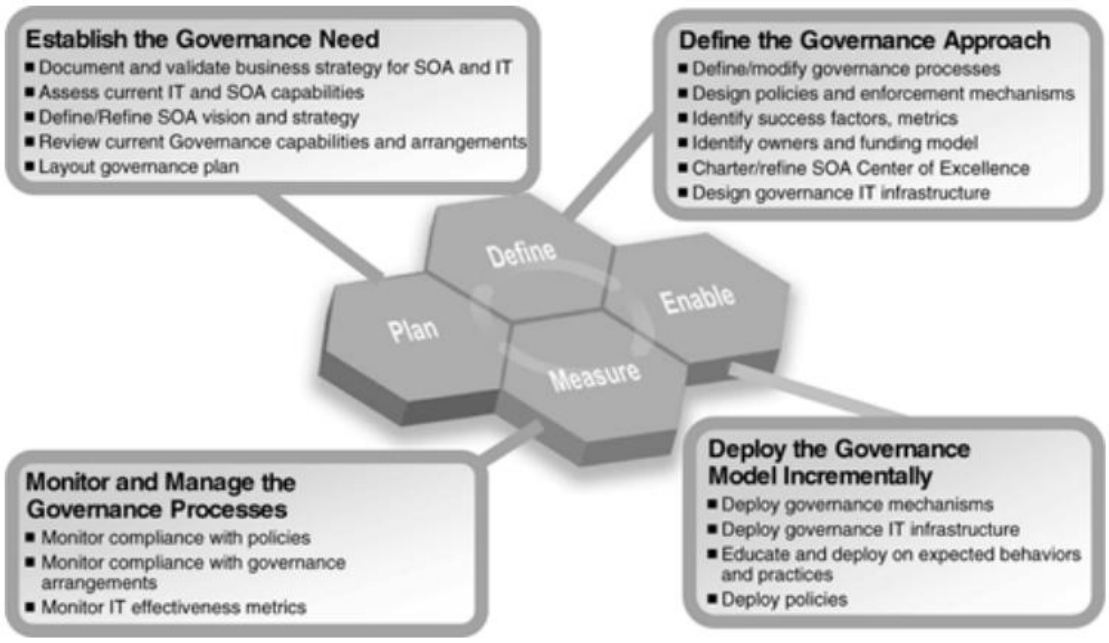


Figure 3-9: SOA Governance Framework [26]

An entire book could be written on this topic. Since it is not the main aspect in this thesis, this subject will not be extended. The objective was, in this section, to show how important governance is to SOA, like it is to all other aspects (in IT or not).

3.3.2 - SOA Maturity

With all of the investments being made in SOA, it is important to measure progress and success of this implementation. The SOA maturity assessment considered in this report takes into account the maturity in terms of technology, people and processes. For example, an organization can be mature in some processes, such as SOA design, and immature in others, such as operational SOA or service quality assurance. Moreover, the people perspective considers not only IT capabilities across multiple disciplines but also collaboration and alignment with the business. All measurements should point out enterprise's strengths and

weaknesses as well as gauge its ongoing improvements. Some SOA measurements should be posted similar to other common IT measurements, such as projects within budget and on time, availability, and system response time. This helps to create a relation between SOA and success by defining and measuring progress and achievements so that everybody can see the results from the adoption of this model.

In order to categorize maturity, CMMI defines six levels, where each maturity level is a well-defined evolutionary plateau that institutionalizes a level of capability for developing the talent within the organization [28]. The six levels are:

0. Incomplete
1. Initial
2. Managed
3. Defined
4. Quantitatively Managed
5. Optimizing

A SOA maturity model can be understood as a collection of key practises to achieve optimal flexibility and a well-defined architecture. There are SOA maturity models defined by many vendors and each model focus on a different aspect. For example, Sonic Software Corp. focuses on the business aspects while IBM focuses on the technical area. As described before, this project is related to how business is (positively) affected by SOA, so Sonic's definition will be used since it fits well the proposed aspect. The structure of Sonic's SOA Maturity Model is demonstrated in the following figure:

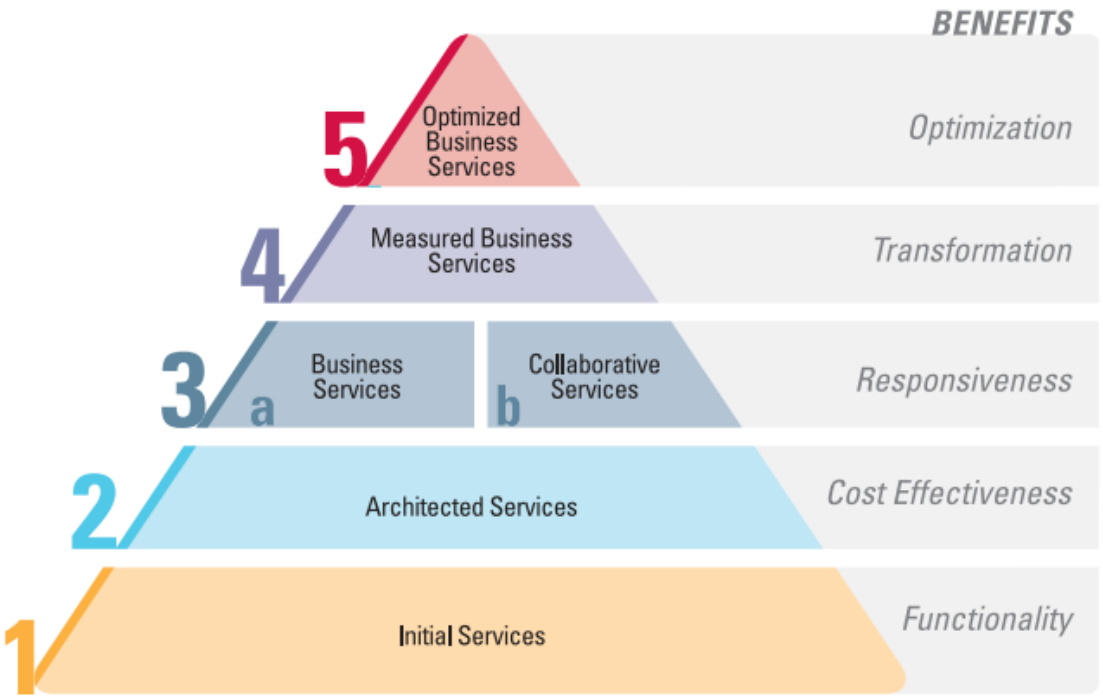


Figure 3-10: SOAMM (Sonic Software Corp.'s SOA Maturity Model) [25]

By defining a SOA maturity model, each project's phase can be easily planned, controlled, measured and managed, reducing risks especially in the long-term. Furthermore, it can facilitate the adoption of a SOA.

3.3.3 - SOA Security

Early in its development, the Simple Object Access Protocol was seen as an HTTP-based protocol, so the assumption that was made was that HTTP security would be adequate for SOAP. After all, there are thousands of Web applications running today using HTTP security so surely this is adequate for SOAP. For this reason, the current SOAP standard assumes security is a transport issue and is silent on security issues [29]. Since this thesis' aim is not to deeply study SOA security, it will be given only an overview on the security models.

To face the security issues in SOA, famous organizations, such as Microsoft, IBM and Verisign have come up with an initiative: WS-Security. Its purpose is the use of XML-Signature and XML-Encryption to provide security to SOAP messages. It also includes a few important components, such as routing and transactions treatment.

In general, the global security requirements for any information system are confidentiality, integrity and availability (CIA). Currently, there are a few known SOA security models. IBM has divided its model into layers of abstraction and its reference model is illustrated in the figure below:

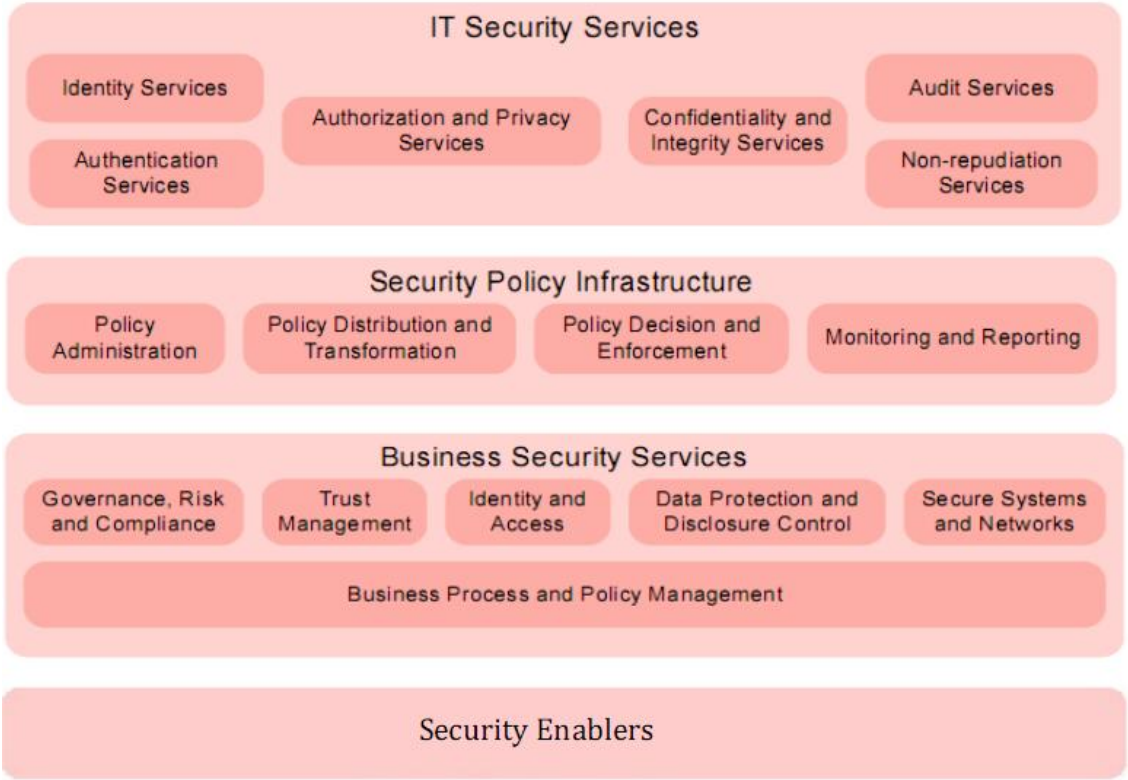


Figure 3-11: IBM's SOA Security Reference Model [30]

Another model comes from the NSTISS. Its committee defines "*a comprehensive model for the security of information systems which also functions as an assessment, system development and evaluation tool*" [31]. The following figure shows the proposed model:

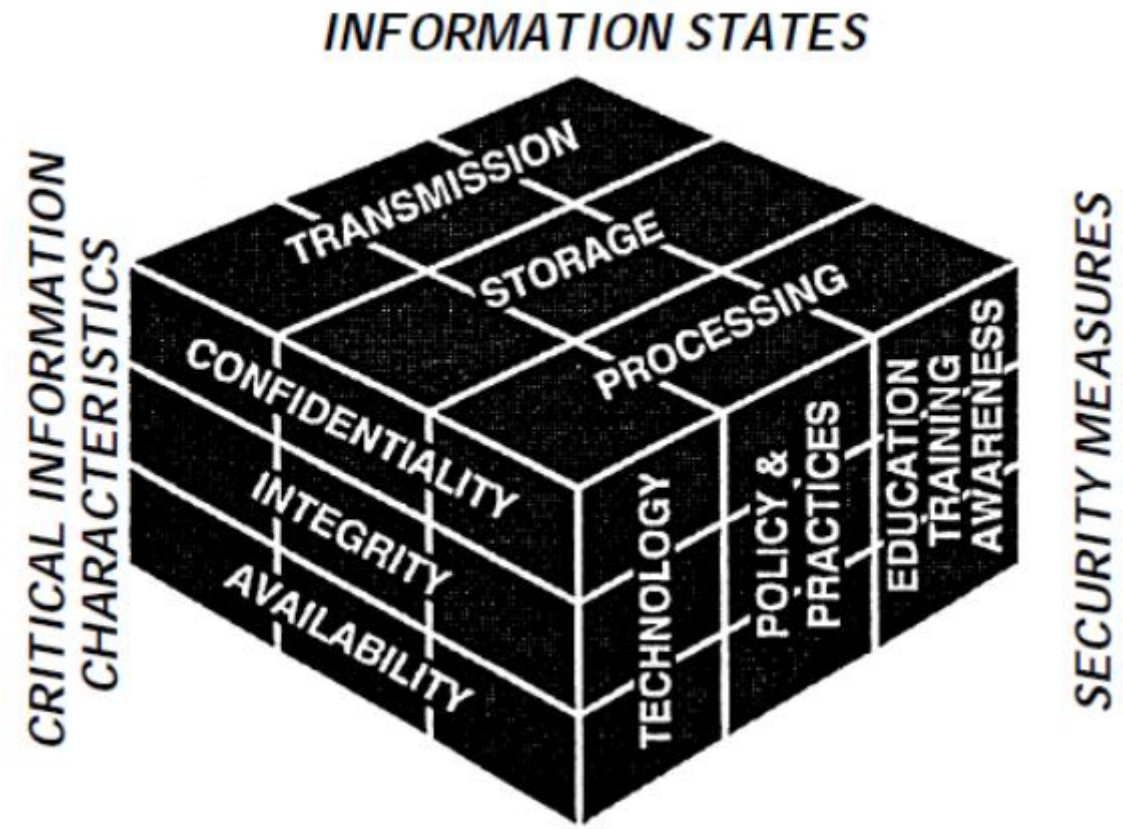


Figure 3-12: Comprehensive model for securing information systems [31]

Besides some models, standard security procedures must be applied in order to reduce the number of vulnerabilities in the system. Some of those important procedures are:

- Authentication: establish and confirm the user identity. The "login system" with a username-password combination should always be part of the system;
- Authorization: limit the access to resources only to people who have the access granted to that specific resource by assigning security levels;
- Security auditing and policies: security policies should be defined and maintained and there should be someone (or a team) assigned to monitor and audit the organization's system in order not to miss any security holes.

3.4 - The Hype of Handheld Devices

In the current reality, corporate applications cannot inherently support access from all devices and will not be able to do so for some time. However, the concept of multi-channel and multi-modal applications has evolved to create application middleware solutions that allow both legacy custom-developed applications and contemporary packaged applications to be accessed from handheld devices of multiple vendors. Obviously, any mobile middleware

solution should have built-in support for as many enterprise applications, old and new, as possible.

As discussed before in this report, corporate information on handheld devices is becoming more necessary than ever. An organization's ability to leverage its legacy systems in combination with newer applications, therefore, will increasingly become a critical success factor in designing and implementing new business initiatives.

In building mobile applications for the enterprise, integration and the corresponding challenges are as central to its design as pure application development. Because mobile applications are created simply to enable a mobile workforce to continuously access corporate business processes and information in real time, they are very rarely standalone applications in their own right. Rather, they apply the existing business logic of these host applications (to ensure data integrity, security, and so forth) as an extension and not re-create this logic in new systems. To provide a complete view of a data set to the user, many mobile enterprise applications must access multiple existing business systems. For example, to supply the sales team with a complete profile of the customer, the mobile application may need to connect to the corporate ERP (e.g., SAP) system for sales information, the Siebel system for customer care issues and a mainframe for customer records.

To design mobile enterprise solutions that execute such information access and integration efficiently, the developer must consider realistically the resource limitations (limited real estate, data bandwidth, and so on) and create the applications to only provide the specific data required for the task and not try to be general purpose. Mobile integration follows the same guiding principles and tenets as do any other integration issues: SOA for integration and XML and Web services for the delivery of integration. With the implementation of XML and Web services, SOA can decouple the end devices and their operating environments from the integration of mobile services with corporate applications [2].

3.5 - What About Old Mobile Phones?

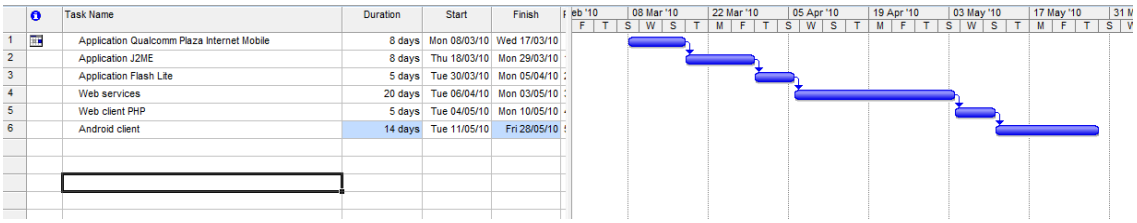
The world is close to reach the phenomenal number of 5 billion mobile phones; this will occur in 2010 [41]. The study done in this thesis aims the recent, brand new less limited smartphone models. Of course, as time goes by the number of old phones decreases and the amount of recent (and with more capabilities) devices models increases widely. However, many people are still somehow attached to their old "dumb" phones and for a lot of reasons they do not plan to change that. Then, the question that is raised is: are those people out of the e-business world?

The answer is no. This project does not go further on this topic, but there is already a framework to integrate those very limited devices to the mobile business world using the Short Message Service (SMS). The user sends a message to the service number with the input

data; the server parses the received message and sends the information to the Web service. Then the reverse process is done to send the response to the user. This framework has been tested and it works [42].

3.6 - Technical Implementation

Now that all theory has been elucidated, this report comes to the section regarding the practical work. The primary aim was to create an environment where it would be possible to test the SOA architecture. As discussed in previous sections, it is very important to make a work plan so that all goals can be achieved successfully on time and with the expected results. The following figure illustrates this work plan:



	Task Name	Duration	Start	Finish
1	Application Qualcomm Plaza Internet Mobile	8 days	Mon 08/03/10	Wed 17/03/10
2	Application J2ME	8 days	Thu 18/03/10	Mon 29/03/10
3	Application Flash Lite	5 days	Tue 30/03/10	Mon 05/04/10
4	Web services	20 days	Tue 06/04/10	Mon 03/05/10
5	Web client PHP	5 days	Tue 04/05/10	Mon 10/05/10
6	Android client	14 days	Tue 11/05/10	Fri 28/05/10

Figure 3-13: Technical implementation timeline

The first step is to design the system to understand what technologies will be used and what the requisites are. The figure below demonstrates the system design done to achieve all proposed goals.

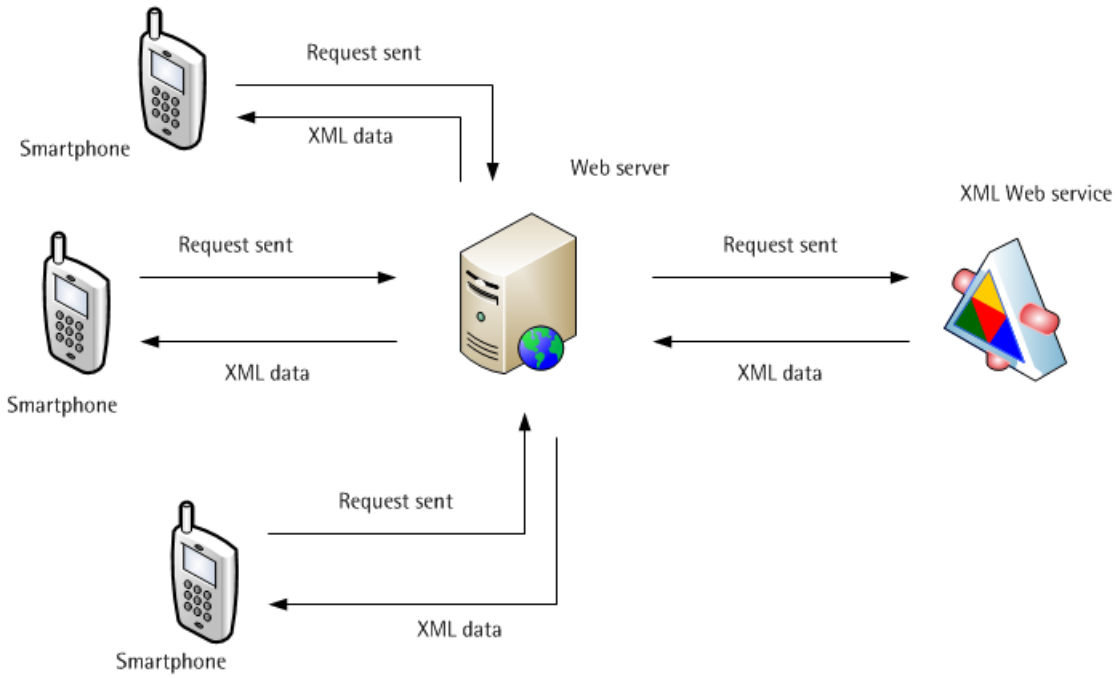


Figure 3-14: System's design (first technical implementation)

As can be inferred from figure 3-14, the first technical development is to construct several applications using different technologies but all pointing to the same Web service. The technologies used in this project were chosen for their capabilities of being executed in smartphones from the most famous vendors. The chosen Web service is about sports news and it can be accessed for free from any browser or gadget in general with Internet access. Basically, what the programs need to do is to read / parse a XML file from the server determined in the application's source code and to display the received information in a proper way decided by the developer. It should be clarified that all tests were conducted using emulators.

Regarding the applications, the first technology that was chosen was the Qualcomm's Plaza Internet Mobile, which uses a mix of HTML, CSS and JavaScript to build widgets. After having the code compiled, the test is held at the Sun Java ™ Wireless Toolkit for CLDC (WTK). The result – what is displayed when the application is executed – is shown in the following figure:

Figure 3-15: Mobile application constructed using Qualcomm's Plaza Internet Mobile

The second application is written in J2ME and is also tested using the WTK. It connects to the same Web service and requests and processes the same information (XML file), the difference lies in how the information is displayed on the phone (the emulator) screen. The result can be seen in the figure below:

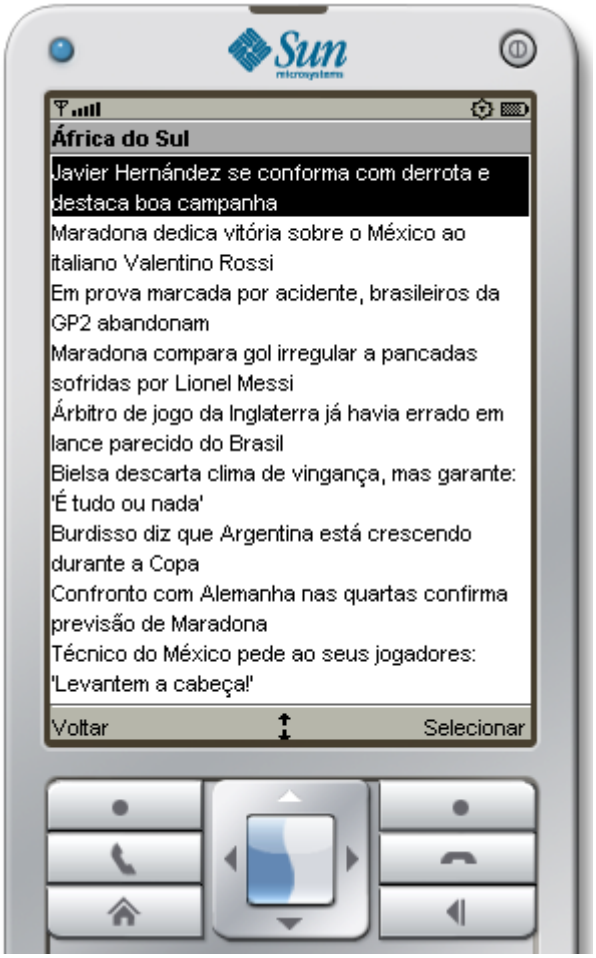


Figure 3-16: Mobile application constructed using J2ME

A third application has also been developed. Since each technology has advantages and disadvantages, it is important execute tests in various environments. For example, this third application is written in Flash Lite / ActionScript, which is focused on outstanding graphical interfaces and it is easier to be delivered as a stand-alone application. On the other hand, Java is more secure, is universally available and usable and is also more scalable. The third result can be seen in the following figure:

Figure 3-17: Mobile application constructed using Flash Lite

Now, three different applications have been developed and the results can be compared. To add more technologies, the service will also be tested using a few Web browsers; each browser displays the information in a specific way, as shown in the figures below:

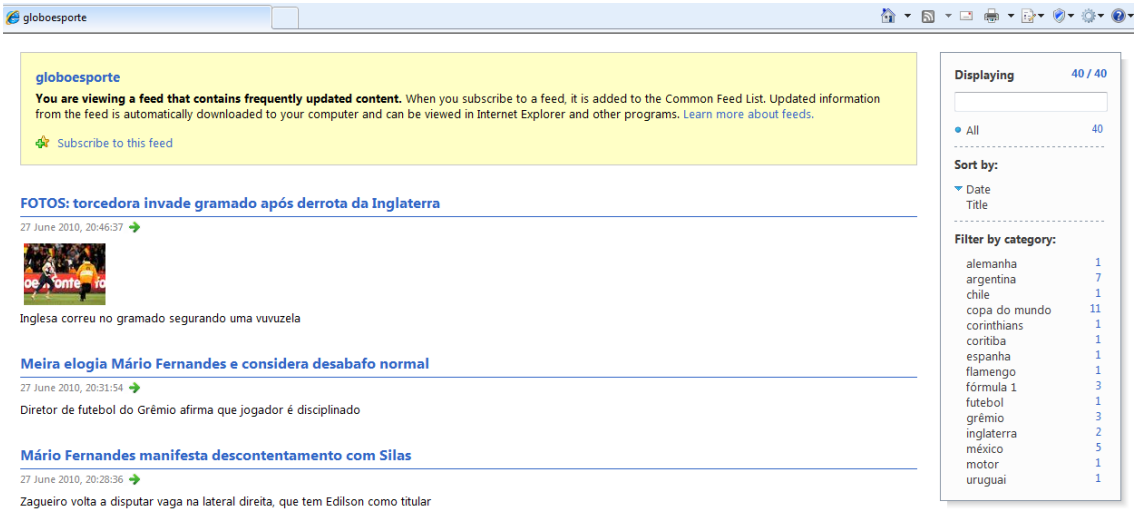


Figure 3-18: XML Web service displayed using Internet Explorer

Figure 3-19: XML Web service displayed using Opera

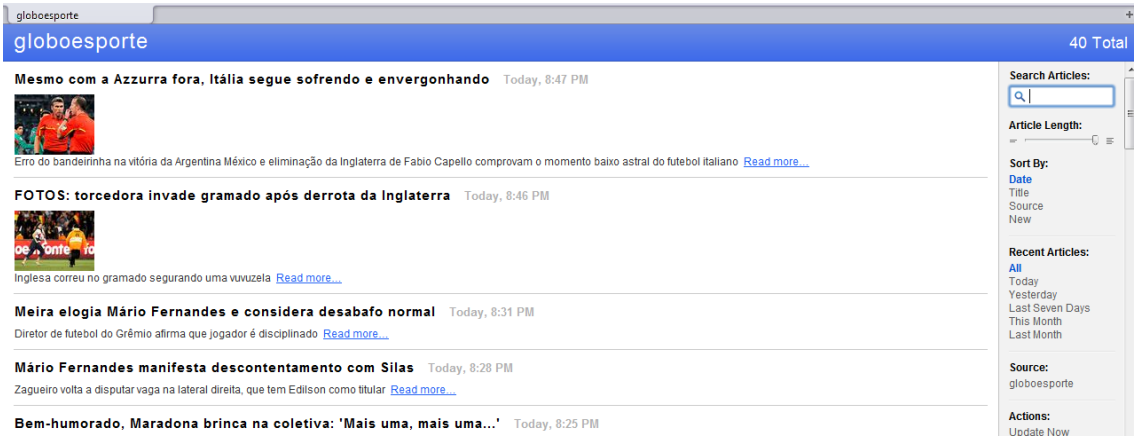


Figure 3-20: XML Web service displayed in Safari

globoesportehttp://globoesporte.globo.compt-BR© Todos os direitos reservados.Sun, 27 Jun 2010 20:47:12 -
0300http://blogs.law.harvard.edu/tech/rsshttp://globoesporte.globo.com/ImageShow/0,,197044,00.gifgloboesportehttp://globoesporte.globo.com14460FOTOS: torcedora invade gramado após derrota da
Inglaterrahttp://globoesporte.globo.com/futebol/copa-do-mundo/inglaterra/noticia/2010/06/fotos-torcedora-invade-gramado-apos-derrota-da-inglaterra.html<a
href='http://globoesporte.globo.com/futebol/copa-do-mundo/inglaterra/noticia/2010/06/fotos-torcedora-invade-gramado-apos-derrota-da-inglaterra.html' alt='FOTOS: torcedora invade gramado após derrota
da Inglaterra'><img border='0' src='http://s.glbimg.com/es/ge/f/90x68/2010/06/27/inglesainvadeestadio_get.jpg620.jpg' alt='FOTOS: torcedora invade gramado após derrota da Inglaterra' title='FOTOS:
torcedora invade gramado após derrota da Inglaterra' />
Inglesa correu no gramado segurando uma vuvuzelainglaterraSun, 27 Jun 2010 20:46:37 -0300Meira elogia Mário Fernandes e considera
desabafo normalhttp://globoesporte.globo.com/futebol/times/gremio/noticia/2010/06/meira-elogia-mario-fernandes-e-considera-desabafo-normal.htmlDiretor de futebol do Grêmio afirma que jogador é
disciplinadogrêmioSun, 27 Jun 2010 20:31:54 -0300Mário Fernandes manifesta descontentamento com Silashttp://globoesporte.globo.com/futebol/times/gremio/noticia/2010/06/mario-fernandes-manifesta-
descontentamento-com-silas.htmlZagueiro volta a disputar vaga na lateral direita, que tem Edilson como titulargrêmioSun, 27 Jun 2010 20:28:36 -0300Bem-humorado, Maradona brinca na coletiva: 'Mais uma,
mais uma...'http://globoesporte.globo.com/futebol/copa-do-mundo/argentina/noticia/2010/06/bem-humorado-maradona-brinca-na-coletiva-mais-uma-mais-uma.html<a
href='http://globoesporte.globo.com/futebol/copa-do-mundo/argentina/noticia/2010/06/bem-humorado-maradona-brinca-na-coletiva-mais-uma-mais-uma.html' alt='Bem-humorado, Maradona brinca na
coletiva: 'Mais uma, mais uma...'><img border='0' src='http://s.glbimg.com/es/ge/f/90x68/2010/06/27/maradona_coletiva_get.jpg300.jpg' alt='Bem-humorado, Maradona brinca na coletiva: 'Mais uma, mais
uma...'' title='Bem-humorado, Maradona brinca na coletiva: 'Mais uma, mais uma...'' />
Depois da vitória sobre o México, argentino estende entrevista e ainda sai com a Jabulani debaixo do
braçoargentinaSun, 27 Jun 2010 20:25:32 -0300Javier Hernández se conforma com derrota e destaca boa campanhahttp://globoesporte.globo.com/futebol/copa-do-mundo/mexico/noticia/2010/06/javier-
hernandez-se-conforma-com-derrota-e-destaca-boa-campanha.html<a href='http://globoesporte.globo.com/futebol/copa-do-mundo/mexico/noticia/2010/06/javier-hernandez-se-conforma-com-derrota-e-
destaca-boa-campanha.html' alt='Javier Hernández se conforma com derrota e destaca boa campanha'><img border='0' src='http://s.glbimg.com/es/ge/f/90x68/2010/06/27/hernandez_reu.jpg300_.jpg'
alt='Javier Hernández se conforma com derrota e destaca boa campanha' title='Javier Hernández se conforma com derrota e destaca boa campanha' />
Autor do gol mexicano vê evolução da 'El Tri'
e lamenta erro do juiz: 'Poderia ter consertado logo depois'mexicoSun, 27 Jun 2010 20:12:46 -0300Maradona dedica vitória sobre o México ao italiano Valentino
Rossihttp://globoesporte.globo.com/futebol/copa-do-mundo/argentina/noticia/2010/06/maradona-dedica-vitoria-sobre-o-mexico-ao-italiano-valentino-rossi.html<a
href='http://globoesporte.globo.com/futebol/copa-do-mundo/argentina/noticia/2010/06/maradona-dedica-vitoria-sobre-o-mexico-ao-italiano-valentino-rossi.html' alt='Maradona dedica vitória sobre o México
ao italiano Valentino Rossi'><img border='0' src='http://s.glbimg.com/es/ge/f/90x68/2010/06/27/maradona_coletiva_get.jpg300.jpg' alt='Maradona dedica vitória sobre o México ao italiano Valentino Rossi'
title='Maradona dedica vitória sobre o México ao italiano Valentino Rossi' />
Piloto de MotoGP sofreu grave acidente e quebrou a perna há três semanasargentinaSun, 27 Jun 2010 20:11:48 -

Figure 3-21: XML Web service displayed using Chrome

Generally, Web developers build sections in Web pages to display the information, which is often recent news on the most various subjects. The following figure illustrates this situation:

Figure 3-22: XML Web service embedded in a Web page

As expected, Web browsers have more capabilities and they can manipulate more data and use more resources than smartphones can. However, this difference is decreasing and in a near future may no longer exist.

There are more technologies to be used, such as Yahoo! Pipes, an amazing free Web application that provides a simple graphical user interface for building a service that can

aggregate various services from many locations, using the Web as a database. It would be impossible to list and test all technologies, but since the focus in this thesis is on smartphones and important technologies related to them have already been tested, there is no need to extend this paragraph.

With all the development done so far, it is possible to test the proposed architecture and prove that it can really insert new customers – the mobile clients - into the e-business world. Using different technologies (Qualcomm's, J2ME and Flash Lite), it was possible to access the same service hosted on the Internet from the developed clients. Any enterprise that wishes to grow and include mobile clients within its customer universe should implement this architecture. This will be discussed later in chapter 5.

Sticking to the plan, the next step is to develop a couple of Web services and a new application – now specific to Android system – to run the test. The Web services can also be tested using any Web browser. Although the whole system was executed in the same computer, it was possible to simulate the proposed architecture.

Regarding the security issue – very important today - both the Web application (i.e., to be accessed from a Web browser) and the Android program have a login system, as typically they would have when done by commercial vendors. Moreover, the concept of session has been developed and the user is kept logged in when navigating through the application and using different services. To allow those two features, the Web services developed could connect to a database system or could simply have a hard coded (inserted into the source code) fake database.

The constructed Web services might seem to be silly, but they are enough to prove they can receive data from the client, process this data and respond; hence anything could be done, that is, enterprises could afford to invest in this architecture and this model would certainly increase their income.

As the Web services have no graphical interface, there won't be a figure in this report to illustrate them; the next figures show the Android client as well as a Web browser executing the test, though. The results will be commented later in chapter 5.

First, it is shown the authentication image, which is the one that comes first when the application is executed, for both cases:

Figure 3-23: Authentication via Android

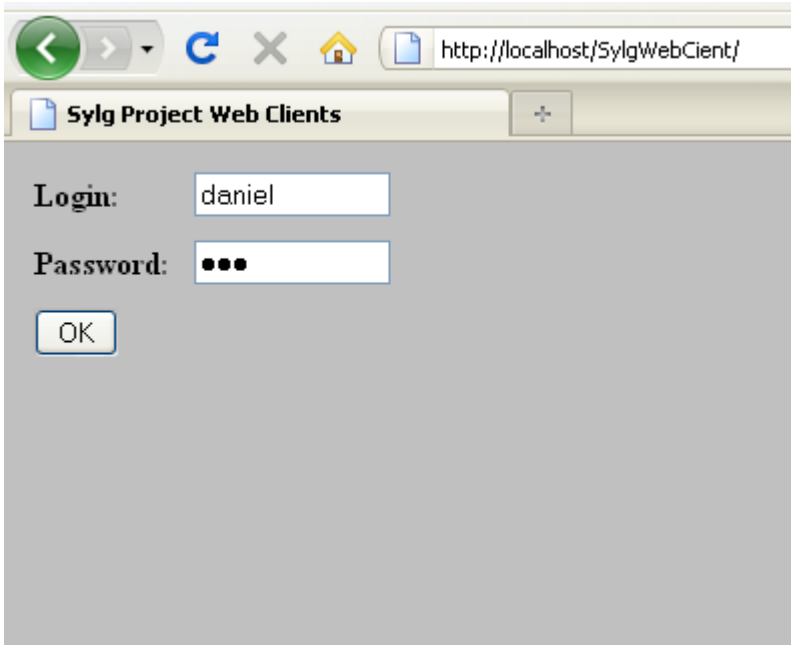


Figure 3-24: Authentication via Web browser

Then, the following images demonstrate the client side of two of the three services created: the temperature service and the sum service. The “hello, World” service will be omitted, since it its image would not add any new value to this report. Although they look

quite simple, they serve as an example of how the system works and it is possible to conclude that any data could be manipulated if the same structure is followed.

Figure 3-25: Temperature service – Mobile client side (Android)

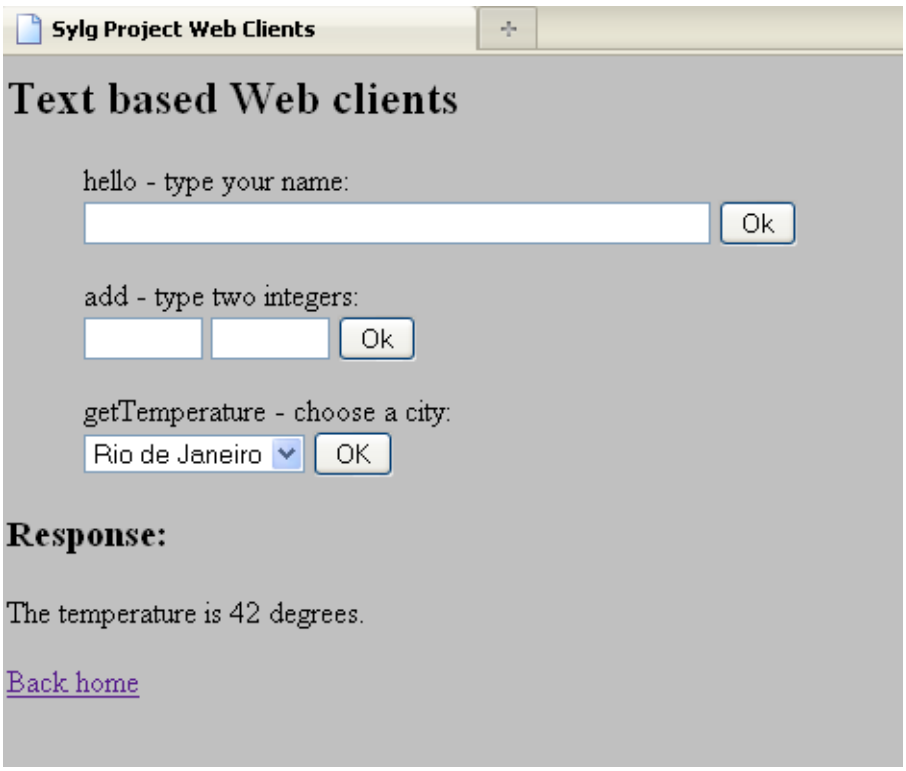


Figure 3-26: Temperature service – Web browser

Figure 3-27: Sum service via Android

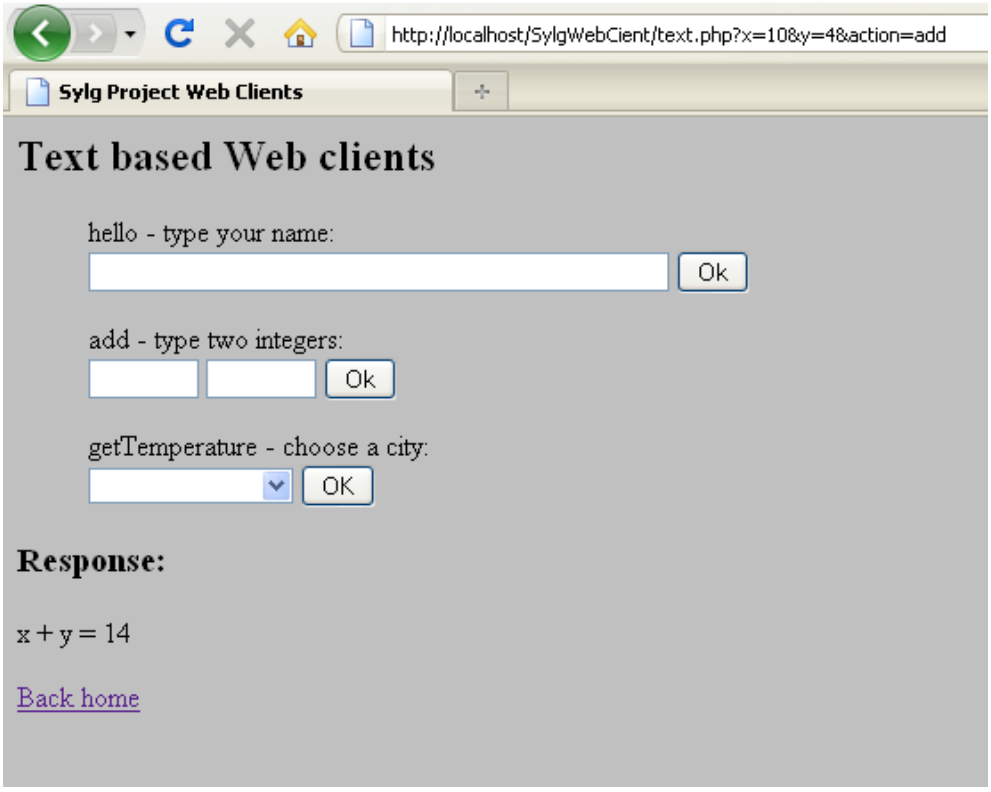


Figure 3-28: Sum service via Web browser

Chapter 4

The Cache Issue (On Smartphones)

After having the system designed and implemented, it was observed that its performance on mobile phones was below expectations. The following step, then, became to improve the system's performance and ensure that it is optimised. As smartphones have become more popular, it has been noticed that the amount of data transferred between mobile applications and services has been increasing widely. This huge consumption is directly related to the unstoppable growth in networks capabilities to meet the new demands of customers and it leads the "mobile world" to a serious problem due to the high data traffic; many believe the Internet is going to collapse soon because of that [46]. On the other hand, the more consumers use, the more they have to pay. Thus, it is expected from applications to use the least amount of data they need so that money can be saved.

Also, mobile phones do not have the same capabilities that computers do – even the most recent models. In order to guarantee the optimal performance, applications should leave most of the processing in the server side and should focus, for example, on how the information will be shown.

To face that issue, several techniques could have been applied in order to improve the system's performance. It is proposed in this dissertation the use of a cache system so that smartphones can benefit from it. Actually, in this project the solution proposed involves not only caching, but also mirroring. The simple fact of having a cache memory allows applications to have an "offline mode", that is, it turns on the possibility of running the program with the data that is stored in the device, so the user does not need to spend any money with data transfer if he / she only wishes to consult some information that is already saved. However, that is only one of the advantages a cache memory offers. Having the information in the device makes the processing faster since the application does not need to connect to the Web service to obtain the information.

Several caching methods are known to help improve a system's performance. Here is a list with a few of these methods:

- Caching on disc;
- Server caching (similar to caching with one or N proxies);
- Caching on protocols;
- Distributed caching;
- Pre-fetching;
- Compression.

Obviously, there are more methods, but those are examples of how a caching system can be designed. In this project, the methods "caching on disc" and "server caching" were used to achieve the proposed goals.

In this project, two levels of cache are proposed: the first – called L1 in this dissertation – refers to the usual cache system used today. The application stores data in the device and a time-to-live (TTL) is assigned to that information. When the application is executed, it checks whether the data stored has expired or not. If the TTL has expired, the program connects to the Web service and downloads the information. If not, the data already stored is displayed. The second level – L2 – refers to a new development on the server side: the mobile device only needs to download a timestamp and compare it to the one stored before displaying the results. It will be better explained in section 4.2.

As expected, the cache system L1 showed some improvement, while L2 has proven to be even better in improving system's performance, though. This chapter is fully dedicated to the description of these both levels and the comparison between them.

Apart from caching, mirroring was used to speed up Web transfers. This technology consists of having the same information on different servers placed in several different locations. If the mirror is closer to the user, then there will be a speedup in the traffic transfer. That means also that the mirrors must be always updated.

The advantage caching has over mirroring is that the cache machine can change the data according to the user's / client's need. Besides, a cache is always closer to the end-user than a mirror, since it is already inside the application. To get the best from both worlds, the solution implemented was a combination of both caching and mirroring. Although it is very important, the topic "security in cache" will not be discussed in this thesis. It should also be highlighted that the proposed caching was defined to be transparent, so that the user would not know of that enhancement. It is possible, then, to conclude that changing the phone's settings (for example, to make the cache space bigger) was out of question. It has been decided that we have to work like an organization would.

4.1 - Cache Level 1 (L1)

What is called Cache Level 1 in this project is the usual caching system found almost everywhere today. This "conventional" cache stores recently accessed files in the device and for each of these files is assigned a TTL. If a requested file is in the cache and its TTL has not expired, it is retrieved from the cache. Otherwise, i.e., if the file is not in the cache or if its TTL has expired, the application needs to request it from the server.

Actually, in this project the application does not store multiple files but data within a specific file created by the program that has been constructed. The idea is also to set a high value to the TTL, turning on the possibility of using the program in an "offline mode", that is, without requesting any data from the server. In this sense, the user can run the application with the data shown in its last execution.

Attempting to improve the quality of this caching system, a little development was proposed: when the application was opened, it was set to load the last results – the data stored in cache. This means the application is always executed first in the offline mode, except in the first time the user ever runs it or when the TTL has expired, once there is no data stored yet. Then, a "Reload" option (e.g., as a button or as a link) is available for the program to connect to the Web service(s) and request all the information from it (or them, if more than one service).

The flux diagram below illustrates the algorithm used in the caching system referred as L1:

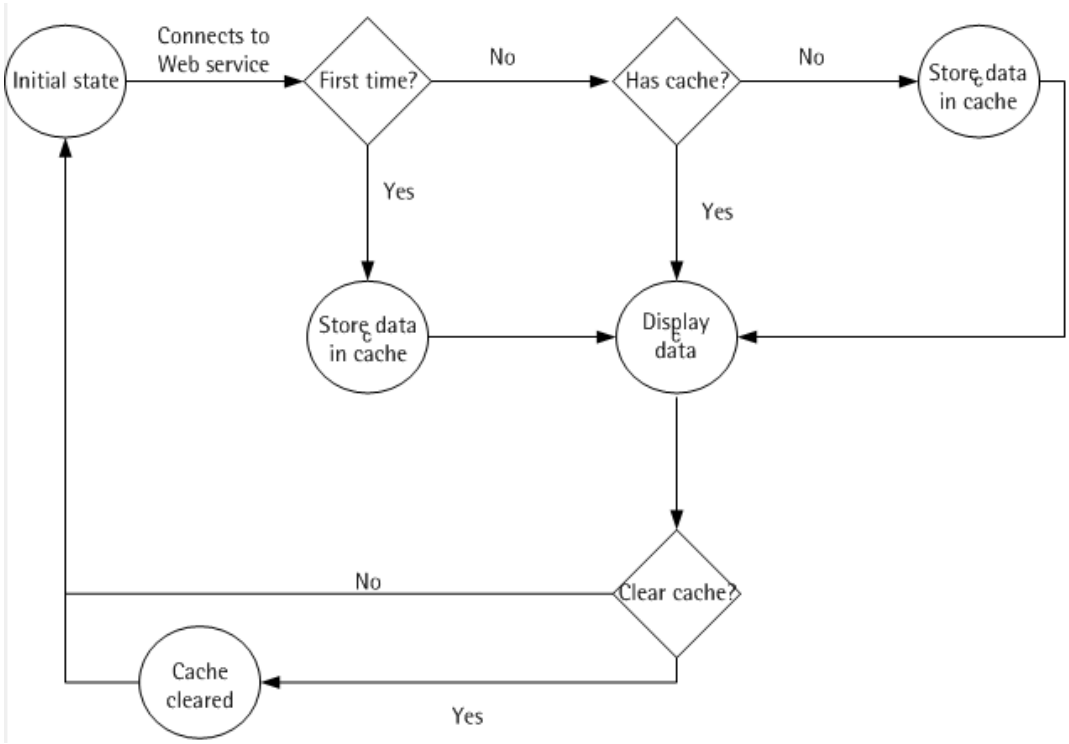


Figure 4-1: The logic behind L1 – the usual cache system

To better explain how the system works, the figure should be described step-by-step:

1. The application is executed;
2. If it is the first time the application is executed or its TTL has expired, it connects to the Web service, receives the information needed and displays it on the screen. Otherwise, the data stored in cache is shown. The communication between the program and the service is via HTTP or HTTPS and the information received is usually a XML file or an image. There are other possibilities, such as the use of a tokenised file, but in this project the focus was on XML Web services only;
3. If it is the first time the application is executed or its TTL has expired, it stores the received data in the cache memory. Otherwise, the user may decide whether he / she wants to reload the data, i.e., connect to the Web service and request the information, or not. The cached information is written and hidden somewhere in the application and it is not a trivial task to find this place. Actually, as it does not really matter, it will not be described here how to find this information or file manually. However, the option to clear the cache has been developed and has been made available for the user so that the use of this memory is optional, although it is recommended to always use a cache memory;
4. The application processes the information (either the received or the stored) and displays it on the screen;
5. If the user presses the "Reload" button, then the cache is cleared and the information is requested from the server.

The use of this cache system indicates some improvement in the application performance, but the results will not be shown here, they will be better described in section 4.3, though.

4.2 - Cache Level 2 (L2)

After having the first level of this enhancement developed, a new issue came up: downloading a big file, such as a video, audio, image or even a XML file with thousands of lines, may be too much to ask mobile devices to store and process every time the program is executed. So, one more need has been presented: to find out when the request for files is needed and when it is not.

Regarding the amount of data transferred, this issue is highly important. For instance, let's imagine the service the application connects to demands a 10MB data transfer. This "simple" fact leads to a few considerations:

1. For smartphones, the download of 10MB is considerably slow and it is known that mobile users want applications to be faster and faster – unnecessary wait is something not acceptable;
2. Thousands or millions of people might want to use the service at the same time and the server could not be able to handle all requests and to respond due to the high demand;
3. The data network may face many problems due to the high traffic;
4. The device may not be able to store that amount of data and the usage of the cache or even of the program can be compromised;
5. The user will probably end up paying a truly expensive bill if his / her data plan does not match all this need.

Once the facts above are clear, the next step is to figure out a way to, as mentioned before, find out whether it is necessary to connect to the server or not. The proposal in this dissertation, called L2, is to make a new development of the server side in order to create the possibility to know when the information stored in cache is up-to-date or outdated. This is achieved by developing a new Web service with a timestamp – and that only – so that the user only needs to download a simple date to process. Going deeper: the server has a scheduler that makes remote calls to the original Web service according to a determined frequency. The server has now 2 services (assuming that it only had one before that): one is the original and the other is the timestamp-service (TS-service). This new service pulls the timestamp from the original service and saves it. That's how the TS-service keeps itself updated.

It is important to highlight that most of the Web services today already have a timestamp within the XML Web service file. However, if someone decided to use this timestamp to compare to the one stored in the phone, he / she would still have to download the whole file and then that comparison (and the cache) would become completely unnecessary. The following figure shows a XML file from one famous news American broadcaster:


```
<title>What do WikiLeak documents tell us?</title>
<guid isPermaLink="false">http://edition.cnn.com/2010/WORLD/asiapcf/07/26/afghanistan.wikileaks.qa/index.html?
eref=edition_world</guid>
<link>http://rss.cnn.com/~r/rss/edition_world/~3/kf_pKytGsdg/index.html</link>
<description>CNN's Nic Robertson analyzes the fallout from the publication of tens of thousands of U.S. military and
diplomatic reports about Afghanistan by whistleblower website WikiLeaks. FULL STORY 1 INSIDER'S VIEW OF WAR&lt;img
src="http://feeds.feedburner.com/~r/rss/edition_world/~4/kf_pKytGsdg" height="1" width="1"/&gt;</description>
<pubDate>Mon, 26 Jul 2010 17:24:46 EDT</pubDate>
<feedburner:origLink>http://edition.cnn.com/2010/WORLD/asiapcf/07/26/afghanistan.wikileaks.qa/index.html?
eref=edition_world</feedburner:origLink></item>
<item>
<title>Scuffles mark AU summit</title>
<guid isPermaLink="false">http://edition.cnn.com/2010/WORLD/africa/07/26/uganda.summit.scuffles/index.html?
eref=edition_world</guid>
<link>http://rss.cnn.com/~r/rss/edition_world/~3/TEyvd-sOEzo/index.html</link>
<description>Scuffles between Libyan President Moammar Gadhafi's presidential guard and Ugandan security personnel have
twice broken out at the African Union summit on the shores of Lake Victoria, officials said.&lt;img
src="http://feeds.feedburner.com/~r/rss/edition_world/~4/TEyvd-sOEzo" height="1" width="1"/&gt;</description>
<pubDate>Mon, 26 Jul 2010 16:50:27 EDT</pubDate>
<feedburner:origLink>http://edition.cnn.com/2010/WORLD/africa/07/26/uganda.summit.scuffles/index.html?
eref=edition_world</feedburner:origLink></item>
```


Figure 4-2: XML file from a famous news-Web service (it contains the timestamp)

From figure 4-2, it is possible to imply that companies already have the timestamp within their services. The proposal is based exactly on this fact: it would not be too much work to create a new Web service with that timestamp only. A new standard should be created so that every company could make this new service available. One possibility is to keep the original name and add "-timestamp" into it. For example, if one service is based on a file named RSS.xml, then the timestamp Web service file would be called RSS-timestamp.xml. That way every programmer / vendor would be able to consider this new standard while building a new application.

In this project, the author implements that separation between the original file and the timestamp that is already within it. The Web service that was chosen to test the proposed system was a football news-Web service from a famous Brazilian news agency, which is available for free on the Internet. Since it would not be possible to ask the news agency to do this development on its side, the decision was to use the author's computer as server, which would act as a middleware. This server was responsible for storing a copy of both the original and the timestamp service. To achieve that, the following code (written in PHP) was used:

```
<?php
/*
 * This file gets the RSS file and timestamp and saves them under
standard
 * URLs. This file should be called periodically and has no user
interface.
 */

// Consts:
define( "RSS_URL" , Web-service URL (omitted) );

// When running from the prompt is required to use the absolute path
in the next constants
define( "ABSOLUTE_PATH" , "C:\\Program Files\\EasyPHP-
5.3.2\\www\\Cache" );

define( "RSS_LOCAL_FILE_URL" , ABSOLUTE_PATH + "public\\RSS.xml" );
define( "TIMESTAMP_FILE_URL" , ABSOLUTE_PATH + "public\\timestamp.xml"
);
define( "RSS_LAST_CHANGE_TIME_URL" , ABSOLUTE_PATH + "rss-last-change-
time.dat" );

// Declarations:
$lastBuildDateLocal = "";

// Get the publication date of the newest item on the RSS.
$doc = new DOMDocument();
$doc->load(RSS_URL);
$lastItem = $doc->getElementsByTagName('item');
$lastItemDateRemote = $lastItem->item(0)-
>getElementsByTagName('pubDate')->item(0)->nodeValue;

// Create the file with the last change time of the RSS or, if it
already
```

```
// exists, get the value.
if( ! file_exists(RSS_LAST_CHANGE_TIME_URL))
{
    $fp = fopen(RSS_LAST_CHANGE_TIME_URL, "w");
    fwrite($fp, $lastItemDateRemote);
    fclose($fp);

}
else
{
    $fp = fopen(RSS_LAST_CHANGE_TIME_URL, "r");
    $lastBuildDateLocal =
fread($fp,filesize(RSS_LAST_CHANGE_TIME_URL));
    fclose($fp);
}

// If the RSS has been updated.
if( $lastBuildDateLocal != $lastItemDateRemote )
{
    // Copy the RSS file to the local server.
    $rssContent = file_get_contents(RSS_URL);
    $fp = fopen(RSS_LOCAL_FILE_URL, 'w');
    fwrite($fp, $rssContent);
    fclose($fp);

    // Update the last change time internal file.
    $fp = fopen(RSS_LAST_CHANGE_TIME_URL, "w");
    fwrite($fp, $lastItemDateRemote);
    fclose($fp);

    // Update the last change time public XML.
    $fp = fopen(TIMESTAMP_FILE_URL, "w");
    $data = "<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\n\t<timestamp>\n\t\t" . $lastItemDateRemote .
"\n\t</timestamp>";
    fwrite($fp, $data);
    fclose($fp);
}


?>
```

It is important to highlight this PHP code was running on an Apache server installed on a Windows 7 machine and the two XML files (services) were saved in a specific folder (in this case, it is possible to conclude from the code that the folder is C:\Program Files\EasyPHP-5.3.2\www\Cache). With this little development, it was possible to reproduce the server side of the system that had been designed in theory.

To make the remote calls, the following batch file was created (batch files are text files that contain several commands to be executed by the command interpreter):

```
"C:\Program Files\EasyPHP-5.3.2\php\php.exe" "C:\Program Files\EasyPHP-5.3.2\www\Cache\index.php"
```

When this batch file is executed, it updates the two files / services. The remote calls make a program running on the server (as seen above) download and compare the information to the one stored. If the downloaded data is more recent than the one stored, then the program replaces the saved copy for the most recent and also updates the timestamp-service, replacing the old XML for a new one with the last date. Since the server is usually a powerful computer with a very fast and reliable Internet connection, it is not a very difficult task to keep this job updating the service like the one just described.

In this project, in order to determine the frequency that the services should be updated, it was created a new task using the standard Windows 7 Task Scheduler. It could be set to call the batch file every 5 minutes or 1 hour, for example. It depends on the user's need.

Having the server to act as a middleware was not only a matter of need; it was also a choice due to the fact that it represents, in addition, a mirroring architecture, that is, the server installed on the author's computer works as a mirror of the news agency server. This is a well-known technique to reduce traffic and latency over networks.

After having this development on the server side, the application, on the client side, does not try to download the original service file every time the user executes the application, it downloads only the timestamp and compares it to the one stored instead. Hence, the amount of data transferred is no more than 1kB and the program takes a second or even less to process this information. In case the timestamp is not the same than the one downloaded, it is updated and so is the information stored in the cache. With this new development, the user can easily use the application in the offline mode – there is no time-to-live – and he / she does not need to worry about the data usage. It will transfer more than 1kB only when there's an update on the service, i.e., only when it is really necessary.

The following figure illustrates how this caching system works:

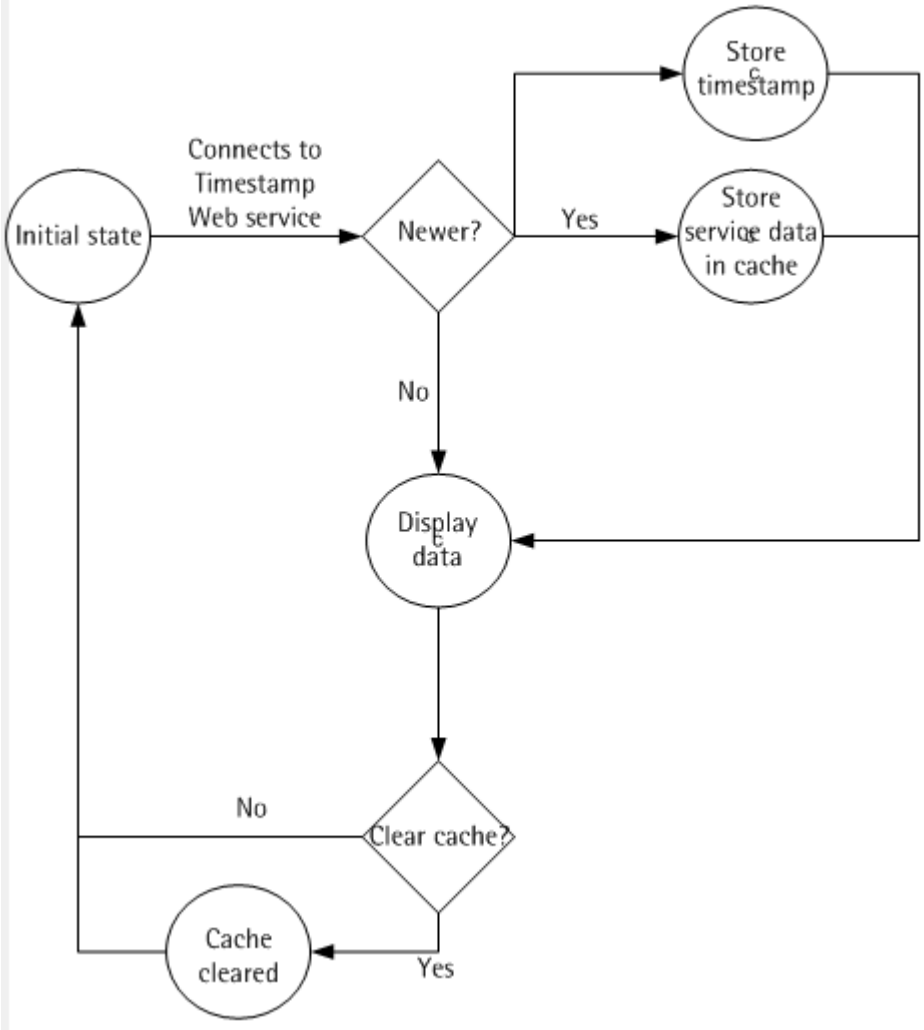


Figure 4-3: The logic behind L2 – better performance

This new enhancement has proven to be the best option so far to achieve the proposed goal: to transfer the least amount of data, to leave most of the processing to the server side and to keep the application as automatic as possible. More details about the results are described in section 4.3.

The client application was constructed using Eclipse with the Android plug-in (Android Development Tools) – it is an Android application. The program was called Sylg RSS Client and it was divided into four classes. The program's source code is demonstrated below. The first class to be shown is the SylgRSSClient.java:

```
package sylg.cachesubproject.cient;

import java.net.MalformedURLException;
import java.net.URL;
```

```
   import android.app.ListActivity;
   import android.content.SharedPreferences;
   import android.database.Cursor;
   import android.os.Bundle;
   import android.util.Log;
   import android.view.Menu;
   import android.view.View;
   import android.widget.ListView;
   import android.widget.SimpleCursorAdapter;
   import android.widget.Toast;

   public class SylgRSSClient extends ListActivity
   {
      public static final int INSERT_ID = Menu.FIRST;
      public static final int NUMBER_OF_ARTICLES = 5;
      public static final String RSS_URL =
"http://10.0.2.2/Cache/public/RSS.xml";
      public static final String TIMESTAMP_URL =
"http://10.0.2.2/Cache/public/timestamp.xml";
      public static final String PREFS_NAME = "SylgPrefsFile";



      private SylgDbAdapter mDbHelper;
      private String serverTimestamp;
      private String localTimestamp;

       @Override
       public void onCreate(Bundle savedInstanceState)
       {
           super.onCreate(savedInstanceState);
           setContentView(R.layout.articles_list);

           // Restore preferences
           SharedPreferences settings =
getSharedPreferences(PREFS_NAME, 0);
           localTimestamp = settings.getString("timestamp" , "");

           RSSHandler rh = new RSSHandler();
```

```
          TimestampHandler th = new TimestampHandler();

          try
          {
           URL rss_url = new URL(RSS_URL);
           URL timestamp_url = new URL(TIMESTAMP_URL);
           serverTimestamp = th.getTimestampFromServer(this,
timestamp_url);
           if( ! serverTimestamp.equals(localTimestamp) )
                       rh.updateArticles(this, rss_url);
          }
           catch (MalformedURLException e)
           {
                 Log.e("SylgRSSCient",e.toString());
           }

          mDbHelper = new SylgDbAdapter(this);
          mDbHelper.open();
          fillData();
          mDbHelper.close();

           SharedPreferences.Editor editor = settings.edit();
           editor.putString("timestamp", serverTimestamp);

           editor.commit();
       }

      @Override
       protected void onListItemClick(ListView l, View v, int
position, long id)
      {
          super.onListItemClick(l, v, position, id);
          Toast.makeText(this, "No action implemented yet.",
1).show();
       }

       private void fillData()
       {
```

```
            // Get all of articles from the database and create the
item list
            Cursor c = mDbHelper.fetchAllArticles();
            startManagingCursor(c);

            String[] from = new String[] { SylgDbAdapter.KEY_TITLE };
            int[] to = new int[] { R.id.text1 };

            // Now create an array adapter and set it to display using
our row
            SimpleCursorAdapter notes =
                new SimpleCursorAdapter(this, R.layout.articles_row, c,
from, to);
            setListAdapter(notes);
        }



        @Override
        protected void onStop()
        {
             super.onStop();

             SharedPreferences settings =
getSharedPreferences(PREFS_NAME, 0);
             SharedPreferences.Editor editor = settings.edit();
             editor.putString("timestamp", serverTimestamp);

             editor.commit();
        }

    }
```

This class is the application's core. It is responsible for defining the program's preferences, such as the number of items that the program should return, and for handling the logic that was previously defined (compare timestamps, get the responses from server and display them and so forth). Another class is RSSHandler.java, which is demonstrated on the following lines:

```
package sylg.cachesubproject.cient;


import java.io.IOException;
import java.net.URL;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;

import android.content.Context;
import android.util.Log;

public class RSSHandler extends DefaultHandler
{

   private int ignoredTitlesCounter = 0;

   // Used to define what elements we are currently in
   private boolean inTitle = false;

   // Feed and Article objects to use for temporary storage
   private String currentArticleTitle = null;

   // Number of articles added so far
   private int articlesAdded = 0;

   // Number of articles to download
   private static final int ARTICLES_LIMIT = 5;

   private SylgDbAdapter droidDB = null;
```

```
	public void startElement(String uri, String name, String qName,
		Attributes atts)
	{
		if (name.trim().equals("title"))
		{
			inTitle = true;
			ignoredTitlesCounter++;
		}
	}

	public void endElement(String uri, String name, String qName)
throws SAXException
	{
		if (name.trim().equals("title") && ignoredTitlesCounter >
2)
		{
			inTitle = false;

			droidDB.createArticle(currentArticleTitle);
			currentArticleTitle = null;

			articlesAdded++;
			if (articlesAdded >= ARTICLES_LIMIT)
				throw new SAXException();
		}

	}

	public void characters(char ch[], int start, int length)
	{
		String chars = (new String(ch).substring(start, start +
length));

		if (inTitle)
			currentArticleTitle = chars;

	}
```

```
      public void updateArticles(Context ctx, URL url)
      {
            try
            {
                  droidDB = new SylgDbAdapter(ctx);
                  droidDB.open();
                  droidDB.deleteAllArticles();

                  SAXParserFactory                spf                =
SAXParserFactory.newInstance();
                  SAXParser sp = spf.newSAXParser();
                  XMLReader xr = sp.getXMLReader();
                  xr.setContentHandler(this);
                  xr.parse(new InputSource(url.openStream()));

                  droidDB.close();




            }
            catch (IOException e)
            {
                  Log.e("SylgRSSClient", e.toString());
            }
            catch (SAXException e)
            {
                  Log.e("SylgRSSClient", e.toString());
            }
            catch (ParserConfigurationException e)
            {
                  Log.e("SylgRSSClient", e.toString());
            }
      }
   }
```

This class handles the information received from the football news Web service. It is essential to the application since it is used by the main class SylgRSSClient. Another important class is SylgDbAdapter.java, which is responsible for adapting the information

received from the XML Web service into a database that is used by another class. The following code illustrates that class:

```
package sylg.cachesubproject.cient;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class SylgDbAdapter
{

    public static final String KEY_TITLE = "title";
    public static final String KEY_ROWID = "_id";

    private static final String TAG = "SylgDbAdapter";
    private DatabaseHelper mDbHelper;
    private SQLiteDatabase mDb;

    private static final String DATABASE_NAME = "data";
    private static final String ARTICLES_TABLE = "tb_articles";
    private static final int DATABASE_VERSION = 2;
    private static final String DATABASE_CREATE =
        "CREATE TABLE " + ARTICLES_TABLE + " (" + KEY_ROWID + "
INTEGER PRIMARY KEY AUTOINCREMENT, "
        + KEY_TITLE + " TEXT NOT NULL);";


    private final Context mCtx;

    private static class DatabaseHelper extends SQLiteOpenHelper
    {

        DatabaseHelper(Context context)
```

```
            {
                super(context, DATABASE_NAME, null, DATABASE_VERSION);
            }

            @Override
            public void onCreate(SQLiteDatabase db)
            {
                db.execSQL(DATABASE_CREATE);
            }

            @Override
            public  void  onUpgrade(SQLiteDatabase  db,  int  oldVersion,
int newVersion)
            {
                Log.w(TAG,  "Upgrading  database  from  version  "  +
oldVersion + " to "
                        + newVersion  +  ",  which  will  destroy  all  old
data");
                db.execSQL("DROP TABLE IF EXISTS " + ARTICLES_TABLE);
                onCreate(db);
            }
        }

        public SylgDbAdapter(Context ctx)
        {
            this.mCtx = ctx;
        }

        public SylgDbAdapter open() throws SQLException
        {
            mDbHelper = new DatabaseHelper(mCtx);
            mDb = mDbHelper.getWritableDatabase();
            return this;
        }

        public void close()
        {
            mDbHelper.close();
```

```
        }

        public long createArticle(String title)
        {
            ContentValues initialValues = new ContentValues();
            initialValues.put(KEY_TITLE, title);
            return mDb.insert(ARTICLES_TABLE, null, initialValues);
        }

        public boolean deleteAllArticles()
        {
            return   mDb.delete(ARTICLES_TABLE,   KEY_ROWID   +   "<"   +
"9999999", null) > 0;
        }

        public Cursor fetchAllArticles()
        {
            return  mDb.query(ARTICLES_TABLE,  new  String[]  {KEY_ROWID,
KEY_TITLE}, null, null, null, null, null);
        }
    }
```

The last class used in this application is TimestampHandler.java, which is very simple – it only gets the timestamp from the server and stores it so that it can be called from the main class.

```
    package sylg.cachesubproject.cient;

    import java.io.BufferedReader;
    import java.io.IOException;
    import java.io.InputStream;
    import java.io.InputStreamReader;
    import java.net.URL;

    import org.xml.sax.helpers.DefaultHandler;

    import android.content.Context;
```

```
   public class TimestampHandler extends DefaultHandler
   {
      public String getTimestampFromServer(Context ctx, URL url)
      {
            StringBuilder sb = new StringBuilder();
            String line = null;
            try
            {
                  InputStream is = url.openStream();
                  BufferedReader   reader   =   new   BufferedReader(new
InputStreamReader(is, "UTF-8"));
                  while ((line = reader.readLine()) != null)
                        sb.append(line).append("\n");
            }
            catch (IOException e)
            {
                  return "";
            }
            finally
            {

            }

            return sb.toString();
      }
   }
```

As a result of all the development just described, the application is constructed and ready to be executed. After running the program using the Eclipse IDE, the Android emulator is opened in order to simulate the mobile phone environment. The following figure illustrates the response the user gets after opening the widget:

Figure 4-4: The application constructed to test the cache system (football news service)

It is possible to imply from figure 4-4 that the client application shows the five most recent news titles from the information retrieved from the server, as described before in this report. Although the application might look simple to the reader, it proves the proposal works and we can conclude any other implementation (following this structure) would work as well. The analysis done to measure the results are described in the next section, 4.3.

4.3 - Results and Performances Comparison

This section describes the results obtained from the proposed system. First, a simple comparison between the original system and the proposed one is illustrated by the figure below:

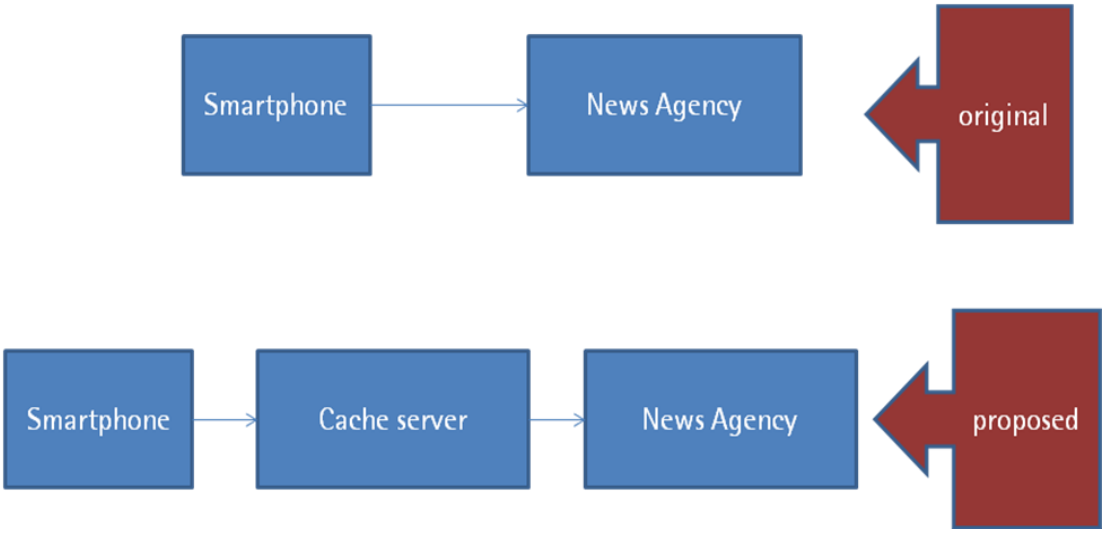


Figure 4-5: The introduction of a new layer between the client and the Web service

As can be inferred from figure 4-5, the cache server (or the timestamp service, depending on the implementation – in this project, it was necessary to use a cache server since everything was tested using the same computer connecting to a Web service hosted somewhere on the Internet) works as a middleware, i.e., it acts as a new layer between the client and the Web service. Actually, it works the closest mirror possible, that is, no other mirror would be so close to the end-user. Thus, the addition of this new layer allows the system to store the latest information – and the timestamp – from the news agency. The idea of the mirroring is similar to the idea of caching with proxies.

To test the system's performance and measure the gain that can be obtained by using the proposed enhancement, two test scenarios were defined:

1. The application connects to a simple "small" 30kB XML Web service;
2. The application connects to a "big" 5MB XML Web service.

To define how to measure the gain obtained from the proposed enhancement, it was inserted the following code into the program's source code:

```
long startTime = System.nanoTime();
// ... the code being measured ...
long estimatedTime = System.nanoTime() - startTime;
```

When cache is the topic, it is always expected the usage of terms such as "cache hit" and "cache miss". However, in the proposed system there will be no miss, so it does not make any sense to use those terms. The timestamp is always compared to the most recent data in

the server and the information in the smartphone is kept updated. That is why the time was the unit chosen in this project; it is the best choice to measure the performance when neither “cache miss” nor “cache hit” fits the evaluation criteria.

For the first time, the two systems – both the original (without cache) and the proposed – had the same performance, since there was no data stored. The differences can be noticed, obviously, afterwards, when there is a cache stored in the smartphone. The systems were tested using the code already described and it is important to state that the Internet connection speed was 600kbps by the time of testing. The following figure shows the results for the first scenario:

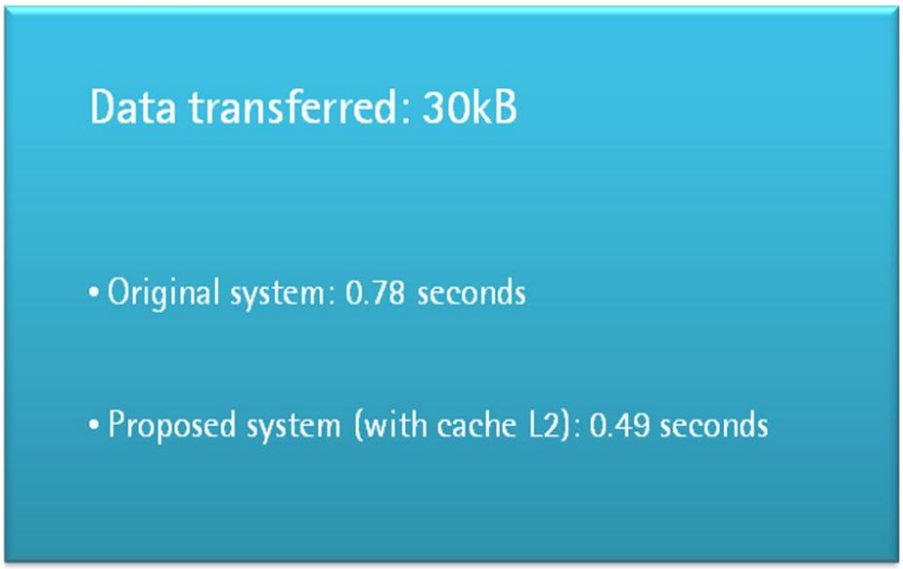


Figure 4-6: Comparison between systems in scenario 1

From figure 4-6, it is possible to conclude that the proposed system really improves the performance, even for a little transfer (only 30kB) – almost 35% of gain. However, in this situation the user would not notice the difference, since for both cases the processing was done in less than one second. The following figure illustrates the results for the second scenario:

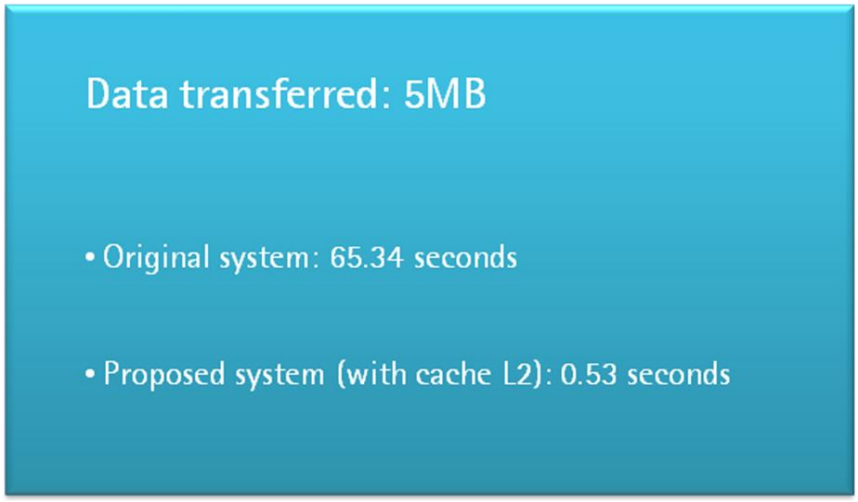


Figure 4-7: Comparison between systems in scenario 2

From figure 4-7, it is possible to imply that the bigger the file is the bigger the boost in the system's performance is going to be. For a 5MB file, the improvement is already huge – almost 124 times faster –, as we can see. Also, L2 deals better with dynamic contents due to the use of the timestamp service enhancement.

Therefore, Cache Level 2 has proven to be much more efficient than Cache Level 1, which may present lots of "cache miss", and than having no cache at all. For services that deal only with small XMLs, it would not matter whether the developer chooses to implement L1 or L2. Actually, for small XML files, even having no cache at all would not represent any trouble. But if the service involves a big amount of images, videos or uses heavy applications data, then one should advise to develop L2.

It is also possible to see a reduction in latency and traffic. The decrease in latency is essential to mobile applications vendors, since this specific audience is very demanding and the time they wait for the program to respond is considered a waste.

Chapter 5

Conclusions and Future Work

This project was about developing and testing a SOA-based architecture by which it could be proven that the use of such architecture brings many advantages to both clients and enterprises. The aim was to show how an integrating system can be constructed using SOA as a guide. This integration was focused on smartphones. Although the services and the applications developed were very simple, it is possible to build a new complete and errors-free solution by working on what has been done.

The topic "SOA" is now clear and, after reading this dissertation, it is possible to believe that is just a matter of time for most companies to adopt this model. Of course, as seen previously, the simply adoption of SOA is nothing; enterprises need to maintain it with well defined processes, practises and metrics. It can be implied from this report that SOA may be viewed as a business conceptual architecture, the IT services within an enterprise should be aligned with other sector, especially the business areas. Although it is considered "the world's salvation", SOA depends on governance, projects portfolios, corporate architectures and corporate business models.

During the project's development phase, a new proposal has been done: to enhance mobile phones performances in the system using a cache combined with a server mirror. This has proven to be very efficient, especially when the user sends or receives a large amount of data to / from the server. Today, the information consumers in the mobile economy are the mobile devices. These mobile devices are no longer simple gadgets for presenting static information such as WAP card decks or Web pages, but they are fully programmable interactive systems with applications built for them by conventional development environments for desktop and laptop computers. In today's world, applications for mobile

devices can consume Web services and work easily with any corporate systems. Mobile technology opens excellent competitive opportunities for companies by enabling crucial corporate information to travel with employees, partners, and customers wherever they go. The "go anywhere" coverage of mobile networks and the exponential growth of Internet-enabled mobile phones and PDAs make the mobile enterprise a feasible and cost-effective proposition.

Regarding the limited capabilities of mobile phones, the answer is to put enough of a client on a phone to serve as an user interface, but to leave the functionality on a server. That way, the phone can have a fully featured application, which essentially resides on a server. That architecture mirrors the Web services architecture; one can have a small application on the phone itself, and the server can populate it with data, and handle communications.

Also, Web services are the keys to the mobile enterprise; they are also the common denominator among business systems, the Java world, and the .NET world. The more a company's application assets are Web-enabled and the business processes are presented as Web services (or perhaps more accurately as business services), the more those company assets can be utilized to easily deliver high-value mobile solutions.

A new approach for future work would be on middleware solutions. Many of the current mobile middleware solutions available today have roots in database synchronization with infrastructures based around this technology; however, unstructured data (the majority of business information) by its nature does not lend itself to a database-centric solution. As a mobile middleware platform should not require the developer to create new logic but build an extension of the existing logic, a new approach is required to deliver true real-time enterprise mobility.

Selecting a mobile middleware platform that provides real-time access to backend business systems and the tools for transforming specific business functions from those systems into Web services is critical in building enterprise mobile solutions. This platform must offer a wide array of business systems connectors out of the box and connect to legacy systems because this is a core business platform for most large organizations. Accomplishing this with the limitations of the screen size and reduced bandwidth on mobile devices requires the mobile solutions to choose and only expose the functions of backend business systems critical to the application's use.

In the future, there will probably be more objects dynamically generated and therefore not good for caching. For future work, the approach could be extended in order to explore other approaches regarding handling complex data types based on XML messages when dynamic invocation is used and perform more complex services and incorporate semantics context-awareness and security aspects.

Furthermore, networks are getting faster and the need for a cache decreases with that. However, there are limitations on how much traffic can flow through a network and the need to store objects in the client may, for instance, allow users to share the network or reduce the delays from it. Besides, new protocols, architectures and security models will probably be investigated, since these areas still have space for growth, although the ideas are not new.

The challenge remains and whoever feels like moving on from this point is “free to go”. More enhancements on the caching systems are welcome as well as new improvements in the proposed architecture. The more people work on this project, the more systems get better (more efficient). It’s like SOA: with more services working and more people maintaining the system quality, the results get always improved.

References

[1] Natis, Yefim V. (2003). *Service-Oriented Architecture Scenario*. Garner, Inc.

[2] Mitchell, Roy. (2006). *Web Services on Mobile Devices*. Available at http://itmanagement.earthweb.com/entdev/article.php/3612721/Web-Services-on-Mobile-Devices.htm (viewed 28 June 2010).

[3] Institute of Electrical and Electronics Engineers. (1990). *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, NY.

[4] O'Brien, James A., Marakas, George. (2005). *Management Information Systems*. McGraw-Hill/Irwin, 7th Edition.

[5] Sun Developer Network. (2010). *Java ME Technology*. Available at http://java.sun.com/javame/technology/index.jsp (viewed 28 June 2010).

[6] Adobe Systems Incorporated. (2010). *Flash Lite Features*. Available at http://www.adobe.com/products/flashlite/features/ (viewed 28 June 2010).

[7] Wikipedia. (2010). *JavaScript*. Available at http://en.wikipedia.org/wiki/JavaScript (viewed 28 June 2010).

[8] Skinner, Jesse. (2006). *Unobtrusive Ajax*. TheFutureOfTheWeb.com

[9] Wikipedia. (2010). *Mobile operating system*. Available at http://en.wikipedia.org/wiki/Mobile_operating_system (viewed 28 June 2010).

[10]Luiz, André. (2010). *Symbian perde mercado, já iPhone duplica participação*. TekiMobile.

[11]Wikipedia. (2010). *Android (operating system)*. Available at http://en.wikipedia.org/wiki/Android_(operating_system) (viewed 28 June 2010).

[12]Wikipedia. (2010). *XML*. Available at http://en.wikipedia.org/wiki/XML (viewed 28 June 2010).

[13]Wikipedia. (2010). *HTTP Secure*. Available at http://en.wikipedia.org/wiki/HTTP_Secure (viewed 28 June 2010).

[14]UpTeq Multimedia (2009). *How is the SCWS managed remotely?*. Gemalto. Available at http://www.gemalto.com/telecom/upteq/multimedia.html (viewed 28 June 2010).

[15]Sprott, David. (2004). *Service Oriented Architecture: An Introduction for Managers*. CBDI Forum Limited.

[16]Nickull, Duane. (2005). *Service Oriented Architecture*. Adobe Systems Incorporated.

[17]Amanuddin, Riz, Ronchi, Diego, Nguyen, James, Hua, Kent, Yee, Eric, Au, Edmond. (2004). *Service-Oriented Architecture in a Pervasive Environment*. IBM Software Group.

[18]Maréchaux, Jean-Louis. (2006). *Combining Service-Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus*. IBM, Software Group.

[19]Wilkes, Lawrence. (2004). *The Essential Guide to Service Orientation*. CBDI Forum Limited.

[20]Sun Microsystems, Inc. (2004). *Assessing Your SOA Readiness*.

[21]Wood, Lamont. (2004). *The Costs and Benefits of SOA*.

[22]Wikipedia. (2010). *Service-oriented architecture*. Available at http://en.wikipedia.org/wiki/Service-oriented_architecture (viewed 28 June 2010).

[23]Wisuttikul, Thanapol (2007). Building SOA Solution and Managing the Service Lifecycle. National Electronics and Computer Technology Center - NECTEC.

[24]Falkl, John, Laird, Robert, Carrato, Tony. (2009). *IBM Advantage for SOA Governance Standards*. IBM Software Group

[25]Progress Software Corporation (2010). *Develop a comprehensive SOA planning and implementation model*. Available at http://web.progress.com/en/sonic-landing/whitepapers/soa-maturity-model.html (viewed 28 June 2010).

[26]Carter, S. (2007). *The New Language of Business: SOA and Web 2.0*. IBM Press.

[27]Marks, Eric A., Bell, Michael. (2006). *Service Oriented Architecture (SOA): Planning and Implementation Guide for Business and Technology*.

[28]Software Engineering Institute | Carnegie Mellon. (2010). *CMMI | Tools and Methods | People CMM*. Available at http://www.sei.cmu.edu/cmmi/tools/peoplecmm/ (viewed 28 June 2010).

[29]Wolter, Roger. (2001). *XML Web Services Basic*. Microsoft Corporation, msdn.

[30]Nagaratnam, N., Nadalin, A., Mostow, J., Muppidi, S. (2007). *SOA Security Reference Model*. STSC CrossTalk. Available at http://www.stsc.hill.af.mil/crosstalk/2007/09/0709NagaratnamNadalinMostowMuppidi.html (viewed 28 June 2010).

[31]NSTISS. (1994). *National Training Standard For Information Systems Security (INFOSEC) Professionals*.

[32]Wolter, Roger, Wilkes, Lawrence. (2004). *Understanding Service-Oriented Architecture*. CBDI Forum.

[33]David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Mike Champion, Christopher Ferris, David Orchard, *Web Services Architecture*, World Wide Web Consortium, Note NOTE-ws-arch-20040211, February 2004.

[34]Mendes, Marco Aurélio S. (2007). *Arquiteturas Orientadas por Serviços*.44o Café Empresarial ASSESPRO-MG.

[35]Wikipedia. (2010). *Enterprise service bus*. Available at http://en.wikipedia.org/wiki/Enterprise_service_bus (viewed 28 June 2010).

[36]Wikipedia. (2010). *Enterprise application integration*. Available at http://en.wikipedia.org/wiki/Enterprise_application_integration (viewed 28 June 2010).

[37]Carvalho, Davi. (2008). *ESB Best Practices (Boas Práticas para ESB)*. Available at http://soasimples.com/blog/?m=200808 (viewed 28 June 2010).

[38]Progress Software Corporation. (2010). *ESB Architecture and Lifecycle Definition*. Available at http://web.progress.com/en/esb-architecture-lifecycle-definition.html (viewed 28 June 2010).

[39]W3C Web Services Architecture Working Group. *Web Services Architecture Requirements*, Working Draft 29 April 2002.

[40]WebSphere Studio | Scenarios (2004). *Overview of the sample application*. IBM Software Group.

[41]CNET Reviews (2010). *Cell phone subscriptions to hit 5 billion globally*. Available at http://reviews.cnet.com/8301-13970_7-10454065-78.html (viewed 28 June 2010).

[42]Ledesma, Bruno de Medeiros. (2007). *Um framework para Web Services através do Short Message Service*. Departamento de Informática e Estatística da Universidade Federal de Santa Catarina

[43]Grembergen, Wim Van (2004). *Strategies for Information Technology Governance*. Idea Group Publishing.

[44]Linthicum, David S. (2007). *How Much Will Your SOA Cost?*. SOAInstitute.org. Available at http://www.soainstitute.org/articles/article/article/how-much-will-your-soa-cost.html (viewed 28 June 2010).

[45]Roch, Eric. (2007). *SOA Costs and ROI*. The Service Oriented Architecture (SOA) Blog. Available at http://it.toolbox.com/blogs/the-soa-blog/soa-costs-and-roi-16794 (viewed 28 June 2010).

[46]Kelly, Spencer (2007). *Warning of 'internet overload'*. BBC News | Programmes | Click. Available at http://news.bbc.co.uk/2/hi/programmes/click_online/6756899.stm (viewed 28 June 2010).

[47]Holm, Veli. (2010). *The Hushed Cost of a Smartphone*. EuropeanCEO. Available at http://www.europeanceo.com/news/finance/expense-management/article809.html (viewed 28 June 2010)

[48]Tom Pisello, Alinean (2006). *Is There Real Business Value Behind the Hype of SOA?*. Computer World. Available at http://www.computerworld.com/s/article/9001155/Is_There_Real_Business_Value_Behind_the_Hype_of_SOA_?nlid=44&source=NLT_ROI (viewed 28 June 2010).

[49]Dias, Ghian V., Cope, Graham, Wijayaratne, Ravi. *A Smart Internet Caching System*. Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka. Available at http://www.isoc.org/inet96/proceedings/a4/a4_3.htm (viewed 28 June 2010).

[50]Woods, Dan, Mattern, Thomas.(2006). *Enterprise SOA: Designing IT for Business Innovation*. O'Reilly.

[51]Gonçalves, Ricardo J., Müller, Jörg P., Mertins, Kai, Zelm, Martin. (2007).*Enterprise Interoperability II: New Challenges and Approaches*. Springer

[52]Pacheco, Diego. (2009). *O Papel do ESB em uma solução SOA*. iMasters. Available at http://imasters.uol.com.br/artigo/12264/desenvolvimento/o_papel_do_esb_em_uma_solucao_soa/ (viewed 28 June 2010).

[53]Natis, Yefim V., McCoy, David W. (2003). *Service-Oriented Architecture: Mainstream Straight Ahead*. Gartner. Available at http://www.gartner.com/pages/story.php.id.3586.s.8.jsp (viewed 28 June 2010).

[54]Povey Dean, Harrison John (1997). *A Distributed Internet Cache*. School of Information Technology, University of Queensland. In Proceedings of the 20th Australasian Computer Science Conference, Sydney, Australia.

[55]Tewari Renu (1998). *Beyond Hierarchies: Design Considerations for Distributed Caching on the Internet*. Department of Computer Science at the University of Texas, Austin.

[56]Knight, Jon, Hamilton, Martin (1996). *Cashing in on Caching*. Available at http://www.ariadne.ac.uk/issue4/caching/ (viewed 28 June 2010).