Università degli studi di Salerno

Dipartimento di Studi e Ricerche Aziendali

(Management & Information Technology)

Dottorato Internazionale di Ricerca

Sistemi Informativi e Ingegneria del Software

XII Ciclo, Nuova Serie

Tesi di Dottorato in

Enhancing Ubiquitous Computing Environments Through Composition of Heterogeneous Services

Doctoral Dissertation of

Pasquale Di Giovanni

Ph.D. Coordinator

Prof. Filomena Ferrucci

UNISA Advisor

Prof. Giuliana Vitiello

UCD Advisor

Dr. Michela Bertolotto

Anno Accademico 2014-2015

Ai miei genitori,
i pilastri della mia vita a cui devo tutto.

A Demia,
che mi ha fatto capire il significato della parola Amore.

Abstract

In recent years the substantial advancements in Information and Communication Technologies enabled the development of original software solutions that can provide support to problems people face in their daily activities. Among the technical advancements that have fostered the development of such innovative applications, the gradual transition from stand-alone and centralized architectures to distributed ones and the explosive growth in the area of mobile communication have played a central role. The profitable combination of these advancements has led to the rise of the so-called Mobile Information Systems. Unfortunately, fulfilling such a type of systems is very challenging and several aspects have to be taken into account during the design and development of both the front and back ends of the proposed solution. Within this context in this thesis we investigate two main aspects: 1) the elicitation of requirements and the design of usable mobile User Interfaces and 2) the information exchange in a back end combining heterogeneous services, more specifically services based on the standards of the World Wide Web (W3C) and Open Geospatial Consortium (OGC).

In particular, we develop a methodology to support the design of mobile solutions when usability requirements play a key role for the success of the whole system. We also present a solution for a seamless integration of services developed according to different standards with specific focus on the issue

of proper management of geospatial metadata in a W3C standards-oriented infrastructure. The result of our investigation is an extension for a key W3C standard for the metadata retrieval to support OGC metadata.

The case study considered in our work is a Mobile Information System to be used by a community of farmers in Sri Lanka.

Acknowledgements

The present work would not have been possible without the help of so many people in so many ways.

First and foremost, I would like to express my deepest gratitude to my supervisors, Giuliana and Michela, for their invaluable help and support during these years. I am also in debt to Monica for all the time she dedicated to me and for all the fruitful exchange of ideas. My gratitude also goes to Professor Ginige and all the members of the Social Life Networks for the Middle of the Pyramid project. I had the opportunity to work with an incredible team on a wonderful research project. Special thanks go to my friend Marco Romano, who guided me during my first faint steps in scientific research. Finally, it is impossible for me to not mention my family, my beloved Demia and all the friends who surrounded me with their support and love during these years. I have been tremendously fortunate to meet each of you.

To you all, thank you.

Contents

List of publications

Journal papers

Sebillo, M., G. Tortora, M. Tucci, G. Vitiello, A. Ginige and P. Di Giovanni. 2015. Combining personal diaries with territorial intelligence to empower diabetic patients. J. of Visual Languages and Computing 29(2015): 1–14.

Conference proceedings

De Chiara, D., P. Di Giovanni, M. Sebillo, G. Tortora and G. Vitiello. 2011. Geomarketing policies and augmented reality for advertisement delivery on mobile devices. Proc. 17th International Conference on Distributed Multimedia Systems (DMS 2011).

Ginige, A., M. Romano, M. Sebillo, G. Vitiello and P. Di Giovanni. 2012. Spatial data and mobile applications - general solutions for interface design. Proc. International Working Conference on Advanced Visual Interfaces (AVI 2012). pp. 189–196.

Diozzi, F., P. Di Giovanni, G. Pezzullo, R Sannino and A. Vozella. 2012. Usability Issues for an aereospace digital library. Proc. Interna-

tional Working Conference on Advanced Visual Interfaces (AVI 2012). pp. 604–607.

Sebillo, M., G. Tortora, G. Vitiello, P. Di Giovanni and M. Romano. 2013. A Framework for Community-Oriented Mobile Interaction Design in Emerging Regions. pp. 342–351. In: M. Kurosu (ed.). Human–Computer Interaction. Users and Contexts of Use, 15th International Conference, HCI International 2013. Springer.

Bertolotto, M., P. Di Giovanni, M. Sebillo, G. Tortora and G. Vitiello. 2014. The Information Technology in Support of Everyday Activities: Challenges and Opportunities of the Service Oriented Computing. Mondo Digitale 13(49): 1–12.

Bertolotto, M., P. Di Giovanni, M. Sebillo and G. Vitiello. 2014. Standard-Based Integration of W3C and GeoSpatial Services: Quality Challenges. pp. 460–469. In: S. Casteleyn, G. Rossi, M. Winckler (eds.). Web Engineering, 14th International Conference, ICWE 2014. Springer International Publishing.

Ginige, A., L. De Silva, T. Ginige, P. Di Giovanni, A. Walisadeera, M. Mathai, J. Goonetillake, G. Wikramanayake, G. Vitiello, M. Sebillo, G. Tortora, D. Richards and R. Jain. 2014. Towards an Agricolture Knowledge Ecosystem: A Social Life Network for Farmers in Sri Lanka. Proc. 9th Conference of the Asian Federation for Information Technology in Agricolture (AFITA 2014).

Sebillo, M., M. Tucci, G. Tortora, G. Vitiello, A. Ginige, P. Di Giovanni. 2014. Combining personal diaries with territorial intelligence to empower diabetic patients. Proc. 20th International Conference on

Distributed Multimedia Systems (DMS 2014).

Vitiello, G., G. Tortora, P. Di Giovanni and M. Sebillo. 2014. Practicing Mobile Interface Design Principles through the Use of HCI Design Patterns – a Training Strategy. Proc. XI Conference of the Italian Chapter of AIS (ItAis 2014).

Book chapters

Di Giovanni, P., M. Romano, M. Sebillo, G. Tortora, G. Vitiello, L. De Silva, J. Goonethilaka, G. Wikramanayake, T. Ginige and A. Ginige. 2013. Building Social Life Networks through Mobile Interfaces - the Case Study of Sri Lanka Farmers. 399-408. In P. Spagnoletti (ed.). Organizational Change and Information Systems, Lecture Notes in Information Systems and Organisation 2. Springer–Verlag Berlin Heidelberg.

Di Giovanni, P., M. Bertolotto, G. Vitiello and M. Sebillo. 2014. Web Services Composition and Geographic Information. pp. 104–142. In: E. Pourabbas (ed.). Geographical Information Systems: Trends and Technologies. CRC Press Taylor & Francis Group.

Vitiello, G., M. Sebillo, G. Tortora, P. Di Giovanni and A. Ginige. 2015. Overcoming the Digital Divide in Europe – Let′s Learn from Emerging Countries!. pp. 209–-220. In: L. Mola, F. Pennarola, S. Za (eds.). From Information to Smart Society, Environment, Politics and Economics. Springer International Publishing.

Workshops

Di Giovanni, P., M. Romano, M. Sebillo, G. Tortora, G. Vitiello, L. De Silva, J. Goonethilaka, G. Wikramanayake, T. Ginige and A. Ginige. 2012. User Centered Scenario Based Approach for Developing Mobile Interfaces for Social Life Networks. Proc. First Int. Workshop on Usability and Accessibility focused Requirements Engineering (UsARE 2012). pp. 18–24.

Vitiello, G., P. Di Giovanni and M. Sebillo. 2013. Integrating Mobile Design Patterns within a User Interface Development Environment. Proc. Workshop on Prototyping to Support the Interaction Designing in Mobile Application Development (PID–MAD 2013). pp. 1–4.

De Silva, L. J. Goonetillake, G. Wikramanayake, A. Ginige, T. Ginige, G. Vitiello, M. Sebillo, P. Di Giovanni, G. Tortora and M. Tucci. 2014. Design Science Research Based Blended Approach for Usability Driven Requirements Gathering and Application Development. Proc. Second Int. Workshop on Usability and Accessibility focused Requirements Engineering (UsARE 2014). pp. 17–24.

List of Tables

List of Figures

Acronyms

CQL Common Query language

DSR Design Science Research

DTD Document Type Definition

GDP Gross Domestic Product

GIS Geographic Information System

GML Geography Markup Language

HCI Human Computer Interaction

HTML Hyper Text Markup Language

HTTP Hyper Text Transfer protocol

IANA Internet Assigned Number Authority

ICT Information and Communication Technologies

INSPIRE Infrastructure for Spatial Information in Europe

IRI Internationalized Resource Identifier

IS Information System

ISO International Organization for Standardization

J2EE Java Enterprise Edition

JAXB Java Architecture for XML Binding

JAX-WS Java API for XML Web Services

JAX-WS RI Java API for XML Web Services Reference Implementation

KVP Key-Value Pair

m-health Mobile health solutions

MIME Multimedia Internet Mail Extensions

Mobile IS Mobile Information System

MTOM Message Transmission Optimization Mechanism

OGC Open Geospatial Consortium

OWS OGC Web Service

QName Qualified name

RPC Remote Procedure Call

SaaS Software as a Service

SE Software Engineering

SII Spatial Information Infrastructure

SIM Subscriber Identity Module

SLD Styled Layer Descriptor

SLN4MoP Social Life Networks for the Middle of the Pyramid

SOA Service Oriented Architecture

SOC Service Oriented Computing

SSL Secure Sockets Layer

SwA SOAP with Attachment

TSL Transport Layer Security

UCD User Centered Design

UDDI Universal Description Discovery and Integration

UE Usability Engineering

UI User Interface

URI Uniform Resource Identifier

URL Uniform Resource Locator

USD Urban Services Data

UX User Experience

W3C World Wide Web Consortium

WCS Web Coverage service

WFS Web Feature service

WIMP Windows, Icons, Menu,Pointers

WMP Web Map service

WPS Web Processing service

WWW World Wide Web

WS-BPEL Web Services Business Process Execution Language

WSCI Web Service Choreography Interface

WSDL Web Services Description Language

WS-I Web Services Interoperability Organization

WSIT Web Services Interoperability Technologies

WS-Metadata Web Services Metadata Exchange

WS-Security Web Services Security

XML Extensible Markup Language

XOP XML Binary Optimized Packaging

Chapter 1

Introduction

This dissertation deals mainly with the issues related to the design and development of Information Systems where data provided through service-based solutions are consumed using mobile applications.

In recent years the substantial advancements in Information and Communication Technologies (ICT) have enabled the development of original software solutions that, by aggregating and harnessing the rich information available from the most diverse data sources and transforming final users from passive information consumers to information producers, can provide support to problems people face in their daily activities.

Among the technical advancements that have fostered the development of such innovative applications, we can surely mention the gradual transition from stand-alone and centralized architectures to distributed ones and the explosive growth in the area of mobile communication. Indeed, while on one side each advancement in the distributed computation field has represented a significant step towards the ability for "*computing to occur virtually anywhere*" (Goodchild et al., 2004), on the other side the opportunities made possible by the advent and fast spread of mobile devices like smartphones

and tablets have revolutionized the way people can exploit the available information.

From a more technical point of view, the profitable combination of the advantages resulting from the transition to distributed environments with the new possibilities of information exploitation made possible by the mobile revolution has led to the rise of the so called Mobile Information Systems (Mobile IS, for short) where"*access to information resources and services is gained through end-user terminals that are easily movable in space, operable no matter what the location, and, typically provided with wireless connection*" (Pernici, 2006). In this context, mobile applications, lightweight software solutions to perform the most diverse task, represent probably the element that mainly contributed to shape the design of current mobile and ubiquitous environments.

Nevertheless, despite the undeniable advantages, the peculiar characteristics of mobile ISs make their design and development a very complex activity since "*the information, services and user interfaces available may vary depending on the context of the utilization of the system*" (Pernici, 2006).

First of all, the provided information is consumed using a mobile terminal, making it necessary to carefully take into account the peculiar characteristics of such devices such as the limited screen size and the different interaction modalities. Another distinctive characteristic is represented by the consistent number of available sensors that allow the development of context-aware solutions, namely the "*ability of application to extract, interpret and use situational information and adapt functionality to the current context of use*" (Alatalo et al., 2001). Identity of the user, spatial and environmental data as well as social situation constitute typical examples of contextual information currently available in modern mobile applications (Korkea-Aho, 2000).

Indeed, the ability to seamlessly exploit the surrounding context and the geographic reference of the provided data represents, probably, the most distinguish feature of mobile solutions when compared with traditional desktop applications. Geospatial data, in particular, have become totally widespread in mobile applications and represent a component of the utmost importance in the context of mobile computation. Location-based services for critical tasks like health-care or disaster management, the use of Google or Bing maps to find shops or restaurants or the geotagging of photos or position sharing represent well-known examples of the importance assumed by geospatial data in our daily activities.

However, to fully exploit the capabilities of mobile devices to act as a gateway to access heterogeneous information it is paramount for mobile applications to rely on the proper support of a back end infrastructure able to offer fundamental functionalities such as data storage, authentication or information integration.

To efficiently support such functionalities and overcome traditional issues like heterogeneity of available resources or the representation, encoding and translation of data, the majority of supporting infrastructures are currently designed and developed around the principles of Service Oriented Computing (SOC) paradigm. The key concept of this paradigm is the notion of service, an independent software module that performs a well-defined set of operations. A service exposes its capabilities through its public interface whose functionality can be invoked by any type of software system: traditional desktop applications, mobile applications and even other services. By hiding the service's internal structure and business logic, such a characteristic contributes to make the interaction independent of the technical details of the specific platform that hosts the actual service implementation. Platform

independence represents one of the SOC distinguishing features, when compared with other proposals, and constitutes the fundamental basis for the other key characteristic behind the success of this paradigm: services composition, namely the ability to compose different services, possibly developed by different organizations to provide complex functionalities. Moreover, since each service that contributes to the composition can expose, in a platform-independent way, the functionalities of an existing software system, such a computational approach promotes a strong reuse of software components and constitutes one of the *de facto* options for the fulfillment of interoperability among heterogeneous technologies, architectures and data representations and for creating a framework for application-to-application interaction.

Nevertheless, the fulfillment of a Mobile IS that combines a service-based approach for information provision and the use of mobile terminals for its ubiquitous exploitation still presents several issues that require research efforts both in academia and industry. Among them, in this thesis we focus, in particular, on the set of problems related to the elicitation of requirements, the design of usable mobile User Interfaces and information exchange in a back end based on heterogeneous services.

1.1 Problem Statement

As previously mentioned, the ability of mobile-based IS to provide access to information and resources at the right place and right time has promoted the development of new ranges of solutions that from health-care to business management can create benefits for organizations and people. Yet there are many research challenges that need to be resolved before such systems can be successfully developed. In this Section we provide an overview of those

issues related to the fulfillment of a mobile IS that motivated the investigation conducted in this thesis.

The first major research challenge to consider is how the physical characteristics of mobile terminals may affect the most critical component for a successfully exploitation of the features offered by a mobile IS, namely its User Interface (UI). A good user interface design is, in fact, a fundamental aspect to the success of a system (Sommerville, 2011).

However, when compared with traditional desktop applications, the well-known limitations of mobile world such as the reduced screen size and specific input modalities make the *usability gulf* between desktop and mobile environments still wide and deep (Blackberry Limited, 2014).

In traditional desktop computer applications, a user is expected to approach the system with a clear intuition to carry out some activity or achieve some goal. Actions are purposeful and the result is expected and evaluated. The design emphasis is on making the affordances of the interaction unambiguous and available and ensuring that system feedback and state are clearly visible. For the mobile world, this is not sufficient and new interaction paradigms different from the traditional WIMP (Windows, Icons, Menu, Pointers) interfaces are needed. Such paradigms have to consider new important variables that play a leading role during the design of the mobile UI. Concrete examples of such variables are:

- The importance of context around the user that can influence his/her attention on the specific task,
- The frequent interruption of the task as users switch attention among competing activities, and
- The reduced screen size, which causes lack of room for displaying data.

Squeezing data to fit the display often results in the loss of relevant information, especially if the meaning of the displayed data depends also on their spatial components. Moreover considering that, with the advent of the *touch screen* technology, the display is not only used to visualize information but also as the primary source for input, the right trade-off should be reached between data visualization and data input areas.

With the increasing capability of such devices to process a growing amount of data, the difficulties to effectively visualize the derived information is intended to become an increasingly serious problem for the near future.

Based on these considerations it is clear how an active involvement of final users throughout the whole design and development process becomes a factor of paramount importance in order to improve the effectiveness of a mobile solution and help them to make sense of all the information available to them. In particular, in addition to the required functionality, the development of the mobile UI should consider factors related to the users level of literacy, familiarity in using the device, users cultural background, language beliefs and the sophistication and functionality of the mobile device users can afford.

The second major challenge considered in this thesis is related to the development of a SOC-based infrastructure supporting heterogeneous types of data. An important aspect that requires an active investigation concerns the communication issues among services developed according to different standards.

As previously outlined, the functionalities of a service are exposed through its public interface. A complete description of such an interface is the only thing that a service client needs to know in order to invoke and use the service features. Once the potential client knows the operations supported

by the service, all the communication between the two entities is based on various messages exchange mechanisms. However, since the actual implementation of services and clients might be realized using different platforms and programming languages, the use of proprietary formats for the interface definition and information exchange is simply unfeasible. In addition, since services composition and interoperability are highly influenced by these two key elements, it is fundamental to describe them in a neutral manner using globally accepted standards and provide their concrete implementation with technologies available to every computational platform. In this context, the majority of current solutions based on the SOC principles have embraced the proposals of the World Wide Web Consortium (W3C) that has established a series of universally accepted standards for the development of enterprise-class distributed applications. In order to guarantee their independence from a specific platform the specification and implementation of such standards rely on the use of the Extensible Markup Language (XML).

Nevertheless, a quite significant and interesting exception is represented by the choices of the geospatial community that, under the guidance of the Open Geospatial Consortium (OGC), has defined its own set of standards explicitly tailored on the specific characteristics of geospatial data. Although sharing some characteristics like the use of XML for the information exchange, OGC proposals are based on different design assumptions, and as a result incompatible with the W3C ones. Indeed, the differences affect all the key sections of the concrete service development namely the management of the service public interface, the binding type and time of the operations, and the discovery mechanism of the actual service capabilities (i.e., the way metadata describing the operations supported and the accepted data types are exposed and retrieved). As a direct consequence, a seamlessly composi-

tion of W3C and OGC services is not directly achievable. However, a better integration between the two proposals could be of interest for both the W3C and geographic communities. In fact the former could access and process, more easily, the wide amount of geospatial data currently available by invoking OGC services, while the latter could benefit not only from standards such as those for access management and security, but also from the huge amount of supporting infrastructure developed for W3C services. The use of several fundamental W3C standards is also endorsed by important international initiatives, such as the Infrastructure for Spatial Information in the European Community (INSPIRE) (Villa et al., 2008a) and OGC itself has set a special working group to provide general recommendations and guidelines for adding the support of W3C standards to existing and future OGC services. Finally since the OGC still lacks a true standard for the management of the services workflow, with a deeper integration of the two worlds, the standard that manages the Orchestration of W3C services, namely the Web Services Business Process Execution Language (WS-BPEL), could be used.

1.2 Research Questions and Scientific Contribution

The research issues outlined in the previous Section guided the investigation conducted throughout this thesis. Specifically, this work comes up with contributions towards the following research questions.

- **How to design usable mobile UIs tailored to meet the expectations and needs of the intended user base of a Mobile Information System?**

In the mobile context, the physical screen constraints and the peculiar mobile interaction modalities dramatically influence the ability to design user interfaces able to support the exploitation of provided information. To minimize the impact of these restrictions on the development of an effective UI it is necessary to perform an accurate requirements elicitation and address user requirements since the early stages of the development process. The proposed solution should, in fact, not only meet the functional requirements but also be easy to use and meet the ability and expectations of intended users. Such aspects can be particularly critical during the development of mobile applications for ad-hoc business processes when functional requirements are unclear and intended users are not aware of the potential of ICT-based solutions. In this context, the thesis contribution concerns the proposal of a methodology useful to support the design of mobile solutions when usability requirements play a key role for the success of the whole system.

- **Is it possible to exploit geospatial data established on the OGC standards inside a W3C-based infrastructure?**

The different design choices make W3C and OGC services incompatible each other. One of the few available options to make such services communicate seamlessly consists of mapping one set of standards to the other at both the syntactic and semantic levels. A complete fulfilment of such a mapping is still an open research issue. The currently most accepted solution to this aim is represented by the development of a software wrapper, usually a service itself, that *translates* the requests and responses messages from the W3C services format to a format suitable for the OGC services and vice-versa, while keeping the structure of the original services unchanged. Although the idea of developing a wrapper has been already discussed in the literature (Ioup et al., 2008), the design and development of such a software solution

represents a quite complex task involving several aspects that cannot be automated by a straightforward mechanical translation process. The thesis addresses, in particular, the fundamental issue of a proper management of geospatial metadata in a W3C standards-oriented infrastructure. The proposed contribution extends a key W3C standard for the metadata retrieval by adding the support for OGC metadata.

The investigation of the research questions discussed above was carried out in the context of a Mobile IS meant to support the research goals of an international collaborative project. The Social Life Networks for the Middle of the Pyramid (SLN4MoP) is an international collaborative research program that, by combining the computation flexibility of distributed systems with the advantages offered by mobile devices, aims to provide real-time information to support activities related to livelihood, targeted to meet the needs of people living in developing countries. In this context, a pilot research study concerning the design and development of a software solution to support Sri Lankan farmers improving the quality of their cultivations and related earnings is being carried out. The achievement of the project objectives involves some research challenges for both the front and back ends of the proposed system. For what concerns the system front end, while the majority of final users will mainly use smartphones or tablets as primary way to access the desired information, simply developing a mobile application to show the desired data is not a suitable solution. In fact, apart from the traditional technical challenges that arise during the mobile development, there are other, non-trivial, factors that must be taken into account. In particular, issues such as environmental and contextual requirements as well as specific cultural characteristics of the target population are of paramount importance for the development of a usable user interface. In this context, collecting the

complete list of interface requirements can be a challenging task in itself.

The design of the back end also requires taking several aspects into account. The system must be able to provide not only different types of information to the various potential stakeholders but must provide them according, for example, to the specific user's preferences or security policies. Therefore flexibility is one of the main requirements. Other important requirements include the ability to add new features without affecting the existing components, the independence of the system functionality from the specific format of the various data sources and the possibility to seamlessly add new data sources or replace existing ones without modifying the behavior of existing implementations. Finally, in order to accelerate the development process, the opportunity to use different development tools from different vendors represents a desirable addition. With these requirements in mind, the SOC paradigm, in compliance with the services standards proposed by the W3C seems like a natural approach. However, although W3C service are one of the best alternatives for the development of enterprise-class distributed applications, as previously mentioned, they are not the best choice for the management of geospatial data. Geospatial data are of utmost importance for the purposes of our system. For example, suggestions for a specific user on the best crops to grow have to take into account not only the current market trends but also specific characteristics of the soil and of the area where his/her farm is located. Moreover, since geospatial information could be useful also for third parties (e.g., a government agency interested in visualizing the various soil types available in a specific region), geospatial services developed following the OGC are required. Therefore a seamless integration of the W3C and OGC services is needed for the specific purposes of the SLN4MoP project.

1.3 Organization of this Thesis

The outline of this thesis is as follows:

In Chapter 2 we provide an overview of related work in the areas of interest for our investigation. The Chapter is divided into two main Sections. In the first Section we discuss the issues concerning the elicitation of user requirements, the User Experience and the development of usable mobile applications. In the second Section we focus on the main concepts behind the SOC paradigm and challenges in the integration of W3C and OGC services.

In Chapter 3 we describe our approach to successfully design and develop usable mobile applications. After a brief introduction about the case study of Sri Lankan farmers, we describe, first of all, the Design Science Research (DSR) that served as foundation for our methodology. Subsequently we introduce and discuss our approach for usability driven requirements gathering and application development. The proposed methodology blends a range of technologies by using the DSR approach. Thirdly, we describe in detail how we applied the steps of our methodology to our case study. The Chapter ends with a discussion about the advantages of our methodology.

In Chapter 4 we describe our contribution to the exploitation of geospatial metadata in W3C-based services infrastructures. Firstly we introduce the fundamental role that metadata play in the context of Service Oriented Computing and discuss some existing approaches for their retrieval in W3C environments. Secondly we focus on the fundamental W3C Web Services Metadata Exchange protocol (WS-Metadata, for short) and compare its underlying design choices against the completely different design decisions adopted for the development of OGC services. Thirdly we detail our approach to extend the WS-Metadata protocol with the support for OGC metadata. Finally we discuss the development of a concrete software implementation

supporting the extended version of the W3C protocol.

Chapter 5 reports an empirical evaluation of the research contributions proposed in the thesis. Firstly we discuss the users response towards the mobile prototype that was developed based on our design methodology. Subsequently we provide an example of the applicability of our design approach in a different application domain. Finally we present the initial evaluation results of our software implementation supporting the geospatial metadata exchange in W3C environments.

In Chapter 6 we present the conclusion of the thesis discussing the main outcomes and limitations and point out future research directions of our work.

Chapter 2

Background and Related Work

This chapter introduces the preliminary background of the fundamental characteristics and open issues related to the topics of interest to this thesis. An overview of related research approaches is also provided. The chapter contains two main Sections. Section 2.1 highlights basic issues that arise when developing mobile-based solutions, ranging from user needs and usability requirements to current mobile technology limits. Section 2.2 covers the main concepts and standards behind the Service Oriented Computing paradigm. A summary of the key issues is included in Section 2.3

2.1 User Requirements and HCI issues in Mobile Applications Development

In this section we provide an overview of the main challenges that must be taken into account during the elicitation of user requirements and the several aspects that might influence the design phase and the actual development of usable mobile applications.

2.1.1 User Experience and Requirements Elicitation

According to the ISO 13407 standard, an interactive system is a *"combination of hardware and software components that receive input from and communicate output to a human user in order to support his or her performance or a task"* (ISO/IEC 13407, 1999).

Among the various elements that nowadays determine the success and influence the overall *quality in use* of such a type of software systems, usability and User Experience (UX) certainly represent two complementary factors of the utmost importance.

However, developing a software solution that is easy to use and, at the same time, increases the productivity and satisfaction of final users still represents a challenging task in the context of Software Engineering (SE) (Seffah and Metzker, 2009).

First of all, as clearly discussed in (Seffah and Metzker, 2009) traditional Software Engineering methodologies do not properly consider:

a) user needs and usability requirements, and

b) the requirements testing and validation with end-users before the actual deployment of the system.

Moreover, as pointed out by (Law et al., 2009) to further extend the traditional usability framework that focuses mainly on user congition and performance, it is necessary to focus also on non-utilitarian aspects of user interaction.

To overcome such issues, several methods and engineering approaches have been proposed to better integrate usability and UX techniques into the overall software development life cycle.

As for the former aspect, from an informal point of view, Usability Engineering (UE) can be seen as *"the process by which usability is ensured for an interactive application, at all phases in the development process"* (Hix and Gabbard, 2002). In their seminal work Good et al. (Good et al., 1986) describe UE as a process grounded in classical engineering which aims at developing a software product by taking into account a set of early specified measurable characteristics that the product is expected to have. The process succeeds if the product is demonstrated to have the intended characteristics. The authors also highlight that the specification of measurable usability characteristics is fundamental to determine the usability requirements of a product, or to measure whether the finished product fulfils such requirements. In this context, a critical point is represented by the requirements elicitation, namely the process of understanding the problems and the needs of intended users. Such an information gathering represents a crucial aspect for the development of a software solution and presents an inherent number of difficulties. Among them (Davis, 1982) identifies three main types of issues:

1. the constraints on humans as information processors and problem solvers,
2. the variety and complexity of information requirements, and
3. the complex patterns of interaction among users and analysts in defining requirements.

However, although practitioners are aware of the importance of including usability requirements, it is often the case that the elicitation, specification and evaluation of such requirements occur late in the development process (Dix et al., 2009).

Several techniques have been proposed in the literature to formally specify requirements addressing user's goals and to test the developed prototypes against established usability measures (Juristo et al., 2001; Bygstad et al., 2008). The common main goal of these UE models is to provide tools and methods for the implementation of the users needs and to guarantee the efficiency, effectiveness and users satisfaction of the solution. Among them we can mention, for example, the User Centred Design-Process Model (IBM, 1996) and the UE Lifecycle (Mayhew, 1999). Usually, such models are not directly applied; they are indeed adapted to meet the specific conditions of a certain organization (Nebe et al., 2008).

A well-established methodology that plays a major role in the context of UE is the scenario-based design namely *"a family of techniques in which the use of a future system is concretely described at an early point in the development process."* (Rosson and Carroll, 2002). In other words, scenarios are stories made up of a sequence of actions, events and a final outcome. All these elements are set in a usage context along with the goals, the plans and the reactions of people that take part in the sequence of events. According to the authors, the design of an interactive system is still an ill-defined problem that, usually, designers try to solve according to the so-called solution-first strategy. In this methodology, a candidate solution is generated and analysed in order to clarify the problem state, the allowed operations and the final goals. However, although useful, this methodology presents some problems: the proposed solution is often generalized too quickly; it could be difficult to abandon an approach when it is no longer useful and the reuse of pieces of solutions might not be appropriate for the task under consideration. (Rosson and Carroll, 2002). On the other hand, the scenario-based approach has the following strengths:

- It is concrete, in order to avoid indeterminacy but flexible to manage ambiguous and dynamic situations;
- It is orientated towards people and their needs. In addition, systems are described in terms of what the users will do with them;
- It helps designers reflect on their ideas during design.

Therefore it may constitute a viable methodology to minimize the issues of the solution-first approach. Moreover, among other advantages, this technique provides a means for interactive systems designers to rapidly communicate the various uses of a tool and to share ideas with all the involved stakeholders.

Another well-used approach is the Persona technique, *fictitious, specific, concrete representations of target users* (Pruitt and Adlin, 2006). The advantages of this approach in the requirements engineering process are well-described in (Schneidewind et al., 2012). According to the authors "*The persona technique enables a better understanding of users characteristics and thereby highlights the user needs in software development*" and in the context of requirements elicitations, it contributes to:

- Model and prioritize actors,
- Identify and illustrate scenarios,
- Prioritize and illustrate use cases,
- Specify relationships among actors and use cases,
- Identify and specify non-functional requirements,
- Illustrate the conceptual model,

- Prioritize requirements,
- Support the requirements specification, and
- Support the approval of the requirements validation.

For what concerns User Experience (UX), as previously mentioned, in addition to traditional usability techniques, it represents, nowadays, the second key-aspect for the success of a modern interactive system. The definition proposed by the ISO 9241 standard describes UX as "*A person's perceptions and responses that result from the use or anticipated use of a product, system or service*" (ISO 9241-11, 1998). Among the main reasons of its wide and fast acceptance in the Human Computer Interaction (HCI) community, we can surely mention the awareness that "*product development is no longer only about implementing features and testing their usability, but about designing products that are enjoyable and support fundamental human needs and values*" (Väänänen-Vainio-Mattila et al., 2008). An interesting discussion about the various UX facets can be found in (Hassenzahl and Tractinsky, 2006). The authors analysis focuses on three main perspectives. First of all they argue that the HCI community should not only focus on the traditional aspects of an interactive product (like the achievement of goals in work settings) but also take into account hedonic ones such as stimulation, identification and evocation. Secondly they discuss the importance of providing more affective systems focusing on the need of understanding "*the role of affect as an antecedent, a consequence and a mediator of technology use*"(Hassenzahl and Tractinsky, 2006). Finally they discuss two fundamental aspects of technology use: situatedness and temporality stating that the actual experience can be seen as a unique combination of different elements like the product and the user's internal states. Such interrelated elements interact and modify

each other.

Given its growing importance, in the last decade a significant number of UX models and frameworks have been proposed. Although all the contributions address the UX key features (e.g., its subjective nature) they diverge on other fundamental aspects (e.g., its scope) making it difficult to get a universal definition of UX also due to its connection with a broad range of dynamic concepts such as emotional, experiential or aesthetic (Law et al., 2009). Despite such differences, there is a general consensus in the scientific and industrial communities about the need for UX to be "*manageable and measureable*" (Väänänen-Vainio-Mattila et al., 2008) by providing a set of evaluation methods. In fact, although, due to its richness, UX cannot be measured in its totality this aspect is of the utmost importance in order to effectively contemplate UX in the actual product development lifecycle.

In this context, a first contribution can be found in (Blythe et al., 2007). The authors propose a framework to anylize UX studies. The proposed framework consists of three approaches: Grid Analysis, Citation Analysis and Content Analysis. Since UX is a broad and interdisciplinary topic, the authors identify five aspects that can be used to analyze the scientific contributions, namely theory, purpose, method, domain and application.

In (Väänänen-Vainio-Mattila et al., 2008), a discussion on the requirements for practical UX evaluation methods is provided. The paper is partly based on the results of the *UX evaluation methods in product development workshop*. The authors argue, first of all, that User Centered Development (UCD) still represents the key-factor to design good User Experience since it is necessary to "*understand users' needs and values first, before designing and evaluating solutions*" (Väänänen-Vainio-Mattila et al., 2008). Moreover they recommend an early and frequent evaluation of the software product

since "*the earlier the evaluations can be done, the easier it is to change the product to the right direction*"(Väänänen-Vainio-Mattila et al., 2008). For what concerns the actual requirements for practical UX evaluation methods, the authors propose the following recommendation:

- Valid, reliable, repeatable
- Fast, lightweight, and cost-efficient
- Low expertise level required
- Applicable for various types of products
- Applicable for concept ideas, prototypes and products
- Suitable for different target user groups
- Suitable for different product lifecycle phases
- Producing comparable output (quantitative and qualitative), and
- Useful for different in-house stakeholders.

However, the authors recognize that it is unfeasible to have one single method that fulfills the proposed requirements list.

In (Hassenzahl and Ullrich, 2007) the authors analyze UX under a different perspective. The authors argue that, depending on the context of use, the basis of product evaluation may change. In particular, they discuss how the presence or absence of instrumental goals may impact on the evaluation of an interactive product. By using as test case a general platform fo the generation of interactive stories, they analyze the differences in user experience when the thirty participants experience the system with *no-goal* condition or by finding answers to particular questions (*goal condition*). Two types of

variable are measured: Experiential and Retrospective. For the former, the mental effort and affect (pleasure or displeasure) are investigated while evaluation and acquired knowledge are envisioned for the latter. The study results show that the presence or absence of specific goals highly impact on the experience of an interactive product. In particular, instrumental goals increase the mental effort of the participants. The mental effort was also negatively related to affect. In such a condition, the software product was mainly evaluated in terms of its capacity to support the goal achievement (Hassenzahl and Ullrich, 2007). Moreover the study shows how, without goals, participants changed their set of criteria to evaluate the software product.

2.1.2 HCI issues in Mobile Applications Design and Development

In the context of mobile applications development, the peculiar characteristics of mobile devices as well as the ever-changing users context, make the creation of usable solutions a challenging task and have a fundamental impact on the proper design of what is recognized as the most important component of a mobile application, the User Interface (UI).

UIs represent an essential aspect for every modern interactive system and the effectiveness of their design has a fundamental impact on the overall usability of the system itself. This crucial role is well described by the famous assertion stating that, for the final non-technical users, the UI is perceived as the entire system. Such a claim is even more valid for the mobile world due to the key role of the UI in the profitable exploitation of an application.

When compared with the maturity reached in traditional desktop applications, the development of mobile solutions in general and mobile UIs in particular still lacks globally accepted rules and design guidelines. A first

important reason can be identified in the excessive hardware and software fragmentation within the various mobile platforms that usually leads developers to implement radically different user interfaces depending on the potential mobile platform on which the application will be executed. This problem is increased also by the fact that the various mobile platforms, such as Google's Android, Apple's iOS or Microsoft's Windows Phone, have their own conventions and guidelines for the development of mobile applications in general and the user interface in particular (Apple, 2012; Google, 2013; Microsoft, 2013). In addition to the technical differences among the various mobile operating systems, other well documented problems concern the reduced screen size of mobile devices, the impossibility of applying the graphical techniques used for designing desktop interfaces to mobile devices (Brewster, 2002; Chittaro, 2010), and the context surrounding the user during the actual interaction with the mobile application.

A useful theoretical usability model that takes into account the unique characteristics of mobile systems is presented and discussed in (Hassanein and Head, 2003). According to the authors four main elements namely the user, the task, the environment, and the interface need to be considered. The user types can be partitioned into two main categories: novice or expert users. In general, expert users behave quite differently from novice users and can quickly perform sequences of actions to achieve a certain goal. However, there are other fundamental factors, such as the memory and visual capacities of human beings, to take into account. The memory capacity, in particular, is directly related to the amount of information a user can recall. Since such a capacity is quite limited, an interface that forces to remember long sequences of commands should be avoided. Also the potential tasks a user might perform are subdivided by the authors into several typologies: closed

or open tasks, accessing or authoring tasks. A closed task is usually related to as a specific objective and the final goal can be divided into sub-goals. On the other hand, an open task is more vague and exploratory (e.g., the exploration of a web site). In an access task, the mobile user performs a sequence of know steps in order to retrieve information, while in an authoring task the mobile user generates new information. This latter case can be complicated by the typical limited input mechanisms of mobile devices and the environmental conditions. In fact, while traditional desktop applications are usually used in quiet and static locations, mobile applications are likely to be used in dynamic environments that impose more challenges or limitations for the user and the user interface (Hassanein and Head, 2003). Among the issues we can mention, for example, the lack of attention due to the presence of other individuals, or suboptimal environmental conditions such as poor/high luminance or noise. Finally, although the rise of the touchscreen displays has made some of this papers considerations obsolete, several aspects related to the UI types analysed, such as the unsuitability of mobile devices to display text-intensive content and to insert long text sequences, are currently still valid.

The intrinsic limitations of mobile devices for the execution of complex interactive tasks and the pressing need to overcome the highest possible number of environmental issues led researchers to explore new modalities to both improve the capabilities and enrich the usability of mobile UIs. For example, the effective integration of sound into the design of mobile interfaces could help users during all the tasks (e.g., driving or walking) that require less visual attention. An interesting analysis of sound exploitation in the design of mobile user interfaces is available in (Brewster, 2002). In this paper, a detailed description of two formal experiments shows how a sonically enhanced

UI can also help to reduce the size of the various graphical widgets with consequent availability of a wider amount of screen space. However it is worth noting that the previously mentioned environmental issues might reduce the usefulness of the proposed solution.

Another analysis of how the unique characteristics of mobile interaction have a direct impact on the design and evaluation of mobile interfaces can be found in (Chittaro, 2010). The author first analyses the distinctive aspects of mobile interaction and then discusses their implications on the design of mobile multimodal UIs. The analysis starts with a comparison of mobile devices with the traditional desktop systems. The first important difference is represented by the hardware limitations of mobile devices that *"make it harder to develop powerful interaction techniques on mobile devices* (Chittaro, 2010). The second difference is constituted by the surrounding context that has profound effects, at various levels, both on the final users and on the type of UI he/she can use. For example, the use of sound-enhanced UIs is not tolerable during a conference or a meeting with others. Moreover, due to the external stimuli, the level of attention during the interaction with the handset might be very limited, making the use of a mobile device *a secondary rather than a primary task*. Another important aspect analysed by the author concerns the adoption of a proper strategy to mitigate the cognitive workload during the execution of a task. In the traditional desktop environments the strategies to reduce this issue can be subdivided into two main categories. In the first category users are trained to enforce their stress resistance or a more skilled person is selected to accomplish the task. In the second category the interface of the system is designed to impose a smaller cognitive workload letting, for example, the system assume the control of some task functions when the workload for the user is excessive. Unfortunately, these two types

of strategies are only partially valid in a mobile context. First of all mobile devices are meant to be used by everyone and not only by experts, secondly the idea of receiving a training to use a mobile device is not popular among final users. Therefore, according to (Chittaro, 2010) the majority of design efforts should aim at making the tasks less demanding. As for the concrete effect of the mobile context on the actual design, the discussion is focused on the analysis of eyes-free interfaces i.e., interfaces that prevent the user from looking at the physical device when he/she does not need to. Although such an interaction modality could be useful for simple tasks, it is important to carefully analyse the cognitive perspective and the context of use of the specific application. In fact, an eyes-free interface simply *shifts human acquisition of information from one sense to another and it does not guarantee that the user will be able to better handle the task* (Chittaro, 2010). Therefore the best solution for final users is to choose an appropriate combination of modalities taking into account that, although each of them can be effective in an isolated context, their combination can lead to serious inconsistencies. Moreover, by exploiting greater context awareness, only the visualization of the functions and the notification of information relevant for the current task should be supplied.

As previously highlighted, the unique characteristics of mobile devices and the environmental conditions have a substantial impact on the UI types a mobile user can use and interact with. However, the great majority of efforts devoted to UI design has been oriented to the needs of people living in the developed world. Little has been done for people living in developing countries where a considerable amount of the population is still constituted by semi- or non-literate users. In these countries, mobile phones are quite widespread as they can be far less expensive than traditional personal computers and serve

a variety of useful purposes (such as financial or basic healthcare services). However, the specific culture, habits and the peculiar technological context require the planning of detailed ethnographic studies with (possibly in-situ) observations of the potential users. This task represents a fundamental step in order to pull out ad-hoc UI guidelines or training useful to make effective and usable mobile applications. For example, it has been shown that for what concerns the text entry accuracy, if novice users received at least an hour of training they would achieve an error rate less than 2% (Patnaik et al., 2009). When such training is not available the error rate can be considerably higher representing a serious problem for all those tasks where data accuracy is a very important parameter (e.g., the fulfilment of a medical form by a user living in a remote region).

A first example of the potential of an SMS-based system to provide useful information for everyday activities can be found in (Jayaweera and Senaratne, 2011). In this paper, the authors describe the development of an application aimed at facilitating the trade activities and to provide useful information to Sri Lankan fishers. Using the proposed system, users can send and receive messages about, for example, the current prices and/or demand and offer of fish. Moreover the weather information module offers to subscribers on-demand notifications about current weather conditions.

A more detailed investigation of the potential benefits of mobile phones for people living in developing countries can be found in (Danis et al., 2010). The need to exploit new approaches to enhance the prevention of diseases in countries with a very limited healthcare delivery system represents the basis for a study of the usefulness of an SMS-based HIV/AIDS education system aimed at increasing awareness in the population. The study was carried out in Uganda and the choice of using an SMS-based system is justified

by the wide spread of traditional mobile phones among the population. The authors' research goals were, first of all, to study the feasibility of SMS-based UIs for users that do not receive an explicit training and secondly to analyse the participation rate to the SMS-based quiz. This last aspect is crucial to understand how to incentivize people's participation in this kind of efforts. The lesson learned is twofold since *"structured SMS messages can be used effectively with untrained users in an application where errors are tolerable"* (Danis et al., 2010) and a subsidy as a reward for participation seems to be a good way to increase the use of such a type of systems.

Another important contribution in this context is presented in (Medhi et al., 2011), where different kinds of interfaces are compared in order to study the most common usability issues and improve the design of mobile UIs. In particular the analysis involves traditional text-based interfaces with different text-free interfaces such as a spoken dialog system, a graphical interface and a live operator. The evaluation is performed with the aid of two independent experiments, the former concerning the completion of a mobile banking transaction, the latter with the insertion of health data with minimal errors. The first step of the investigation, involving 90 subjects from India, the Philippines and South Africa, concerned the study of mobile phone usage patterns by semi- or non-literate users. The common traits of the involved subjects were

- functional illiteracy or semi-literacy but partial numeracy,
- low levels of formal education, and
- zero experience with personal computers.

The first outcomes reveal *"several barriers to using existing text-based interfaces, including difficulties understanding or utilizing hierarchical struc-*

tures, soft keys, scroll bars, non-numeric inputs, and specialized terminology" (Medhi et al., 2011). In particular, for each issue, the following motivations were identified:

- *Hierarchical navigation.* The majority of the subjects were initially unable to understand or navigate hierarchical menus even for simple tasks such as calling back a number from which a missed call was received.

- *Discoverability.* Structuring functionality into deep hierarchies makes it less discoverable. Moreover, additional issues may arise from poor interaction design, when the various functions are categorized under seemingly unrelated categories.

- *Scroll bars.* The role of vertical scrollbars was not initially understood by the majority of the subjects.

- *Soft-key function mapping.* The use of *hard-keys* seems to be more suitable for the involved subjects regardless of whether they owned mobile phones or not. Several difficulties arose with the use of soft keys (e.g., those appearing directly below the screen) that have different functions in different contexts, or numeric keys when used to choose from an enumerated list on screen.

- *Non-numeric inputs.* The difficulties of inserting text sequences are a direct consequence of the fact that a good amount of the subjects use mobile phones just to make and receive voice calls only.

- *Language difficulties.* This last issue is particularly relevant during technical tasks such as the use of mobile banking services since information messages are always entirely in English. A directly related

problem concerns the use of technical terms such as get balance or change pin that can cause confusion in absence of detailed explanation. Moreover the presence of an instruction manual is often not sufficient due to the complexity of the banking jargon.

To contribute to mitigate such problems, the paper's authors formulate a set of design recommendations and evaluate them with the two above-mentioned experiments comparing text-based interfaces with both automatic solutions (spoken dialog and text-free interfaces) and a live human operator:

- Provide graphical cues,
- Provide voice annotation support wherever possible,
- Provide local language support, both in text and audio,
- Minimize hierarchical structures,
- Avoid requiring non-numeric text input,
- Avoid menus that require scrolling,
- Minimize soft-key mappings, and
- Integrate human mediators into the overall system, to familiarize potential users with scenarios and UIs.

In (Terrenghi et al., 2013), the authors discuss an explanatory study concerning the use of tablets in emerging markets. Although the authors recognize that the small sample of participants (17 people) could limit the generalization of the results, some useful suggestions can be gathered. According to the study results, a tablet:

- is an additional gadget, not a replacement of a laptop or a smartphone,

- provides new contexts of use, and

- is more a portable than a mobile internet device.

Moreover, the authors observe that:

- the handling of documents is still cumbersome,

- almost none in the sample population felt comfortable purchasing goods or services using the tablet, and

- the *single user* approach of the majority of mobile application, limits the way people share the device.

Finally, in (Terrenghi et al., 2014) the user requirements for exploiting mobile devices as a mean to enhance mobile payments in emerging markets are analyzed. According to the authors,"*there is a high potential for designing novel mobile payment experiences in emerging markets that create value by cutting costs, increasing accuracy and transparency, and improving budgeting*" (Terrenghi et al., 2014). Therefore, the following design principles are proposed:

- predicate new mental models on those that already exist,

- embrace and augment physicality,

- design for a diverse community of stakeholders (since the level of technical understanding and literacy among users can be extremely broad),

- put trust and privacy control at the center of the experience, and

- appreciate the importance of speed and latency.

2.2 The Service Oriented Computing Paradigm

Being one of the currently most used approaches for the development of distributed solutions, the various aspects that characterize the Service Oriented Computing (SOC) paradigm are well discussed and analysed in the literature. Nevertheless, despite its increasing popularity, several aspects of this methodology still require investigation both in academia and industry. This Section reviews work carried out in this field with reference to the issues that were most relevant for the purposes of this thesis.

The basic concepts behind the Service Oriented Computing are explained in (Papazoglou, 2003). The author, in the context of the transition from traditional monolithic applications to the new vision of Software as a Service (SaaS), introduces the idea of Service as a self-describing computational element that, exposing its public interface using open standards, supports the development of distributed applications. Since the potential clients of a service can be either traditional applications or other services, fundamental characteristics such as technology neutrality, loose coupling and the support for location transparency are also described. The concept of Service Oriented Architecture (SOA) as a concrete mean to effectively combine services is also discussed. This architectural approach is particularly useful when different software solutions, developed with various technologies, need to communicate with each other and its principles rely heavily on the previously mentioned requirements of technology neutrality, loose coupling and location transparency. Detailed information about the main characteristics of contemporary SOAs is discussed in (Papazoglou and Van Den Heuvel, 2007). The important aspect for our research is that an SOA, as a design philosophy, represents a standard way to realize actual SOC based solutions. Although an SOA presents a flexible architecture, one fundamental key point

is represented by the fact that all services are autonomous and an external user does not need to know the actual service implementation since the operational details are hidden behind the service public interface. This separation of the public interface from the internal details represents the keystone for the integration of solutions developed by different entities. Once again, the need to expose the service public interface using globally accepted standards is of paramount importance. To effectively deploy an SOA, various aspects must be taken into account, including:

- The need to expose each new or existing application as a service,
- The need to properly configure and manage services, and
- The need and ability to combine services together in order to provide joint results.

In the remainder of this section, after a brief overview of the main standards that enable the effective development of SOC-based systems, we focus on the fundamental issue of services composition by presenting existing solutions for service integration in general, with particular emphasis on the composition of OGC and W3C services.

2.2.1 Web Services

The majority of current service-based solutions rely on the proposals of the World Wide Web Consortium (W3C).

According to the definition of the W3C working group (Booth et al., 2004), a Web service is *"a software system designed to support interoperable machine to machine interaction over a network. It has an interface described in a machine processable format. Other systems interact with the*

Web service in a manner prescribed by its description using messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."[1]

Standards are one of the main reasons behind the massive adoption and deployment of Web services as a means to create a framework for application-to-application interaction through the Web publishing of business functionalities and the universal access to them. In other and simpler words, a Web service can be described as *"a piece of software application whose interface and binding can be defined, described, and discovered as XML artifacts"* (Yu et al., 2008).

Web services have become, over time, the most common implementation of SOA principles so that some concepts and technologies behind them have influenced and contributed to a number of new SOA characteristics. The whole Web services framework can be analyzed under different points of view, namely the participants to the interaction, the basic activities offered by a service oriented environment and protocols that allow for the effective exchange of information.

As for participants, we can identify three main roles (Yu et al., 2008). The service provider is the organization that owns a Web service and makes it accessible through the network. The service client represents a potential user of a Web service and can be either a human or a software agent (a traditional application, a mobile application or another Web service). Finally, the service registry is the entity that allows service clients to locate the service and obtain information on how to invoke its functionalities.

Nevertheless, a Web service is not an autonomous entity but, in order

[1]For an overview of the the Hyper Text Transfer Protocol and Extensible Markup Language see Appendix A

to be able to offer its functionalities, it requires a supporting infrastructure, the service oriented environment, which has to guarantee the following basic activities (Tsalgatidou and Pilioura, 2002):

- Web service creation,
- Web service description,
- Web service publishing to intranet or the Internet repositories for potential users to locate,
- Web service discovery by potential users,
- Web service invocation, binding,
- Web service un-publishing in case it is no longer available or needed.

Finally, the interaction protocols can be divided into the following three categories:

- Communication protocol,
- Service description,
- Service discovery.

For each of these areas, XML-based standards have been defined: the SOAP protocol for the communication, the Web Services Description Language (WSDL)[2] for the description and the Universal Description Discovery and Integration (UDDI) standard for the discovery. In particular, SOAP defines the communication protocol for Web services; WSDL provides a manner for service providers to describe their application while UDDI offers a registry service for advertisement and discovery of Web services (Figure 2.1).

[2]A detailed discussion of the main characteristics of these two fundamental standards is provided in Appendix A

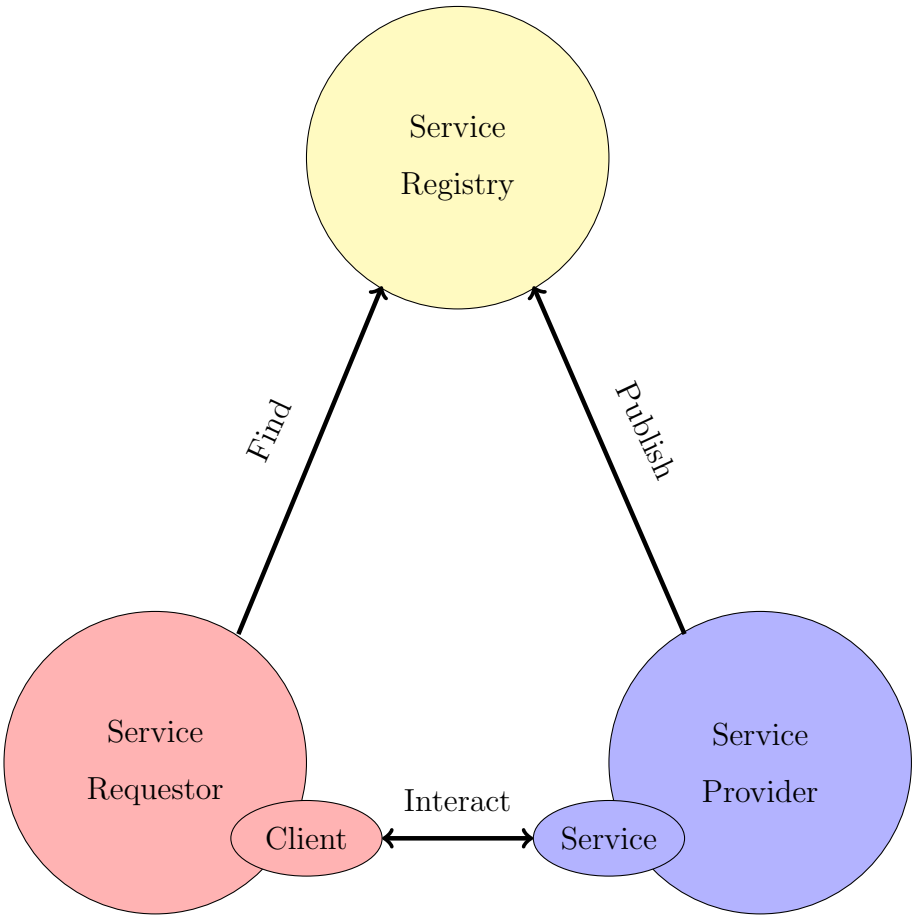

Figure 2.1: The basic Service Oriented Architecture

2.2.2 The Open Geospatial Consortium Services

Despite the success of the W3C services and the huge amount of documentation and frameworks for services development available, the geographic community has developed, over time, its own set of services. Among the various available solutions, the broad consensus around the proposals of the Open Geospatial Consortium (OGC) have made OGC services the de facto standard for what concerns the exchange of geospatial data in distributed environments. The OGC is *"an international industry consortium of more*

than 479 companies, government agencies and universities participating in a consensus process to develop publicly available interface standards". The OGC main goals are to promote the benefits of integrating location resources into commercial and institutional processes and to facilitate the collaboration of developers and users of spatial data products.

Therefore, the aim of OGC services is to represent *"an evolutionary, standards-based framework that enable seamless integration of a variety of online geo-processing and location services"* (Doyle and Reed, 2001). Although the OGC focused on the use of distributed platforms such as CORBA, the explosive growth of the World Wide Web led the consortium towards a communication model that would utilize established technologies such as XML or HTTP. By using these open standards, OGC services can provide a *"vendor-neutral, interoperable framework for web-based discovery, access, integration, analysis, exploitation and visualization of multiple online geo-data sources, sensor-derived information, and geo-processing capabilities"* (Doyle and Reed, 2001). Many factors have influenced the design of the OGC services; these include, for example, the need to interconnect services often provided by different organizations, the need to ensure requirements such as access control or security or the necessity for a client, to know what service can be used with a specific type of data.

However, although OGC services use HTTP as the transport protocol and XML as the lingua franca for the exchange of data, they are often incompatible with the platform proposed by the W3C mainly due to the fact that these two sets of standards were developed in parallel by different organizations. First of all, unlike W3C services, each OGC service represents a separate standard designed to handle a specific kind of data (Ioup et al., 2008). Each of these standards describes, in detail and in a technology implementation

neutral manner, how a service should perform its tasks, describes the service public interface and specifies additional parameters and data structures needed in all request and response operations. Therefore, the first important difference between the two types of services is represented by the strong standardization imposed by OGC regarding the public interface of a geospatial service. Furthermore each W3C service can expose, in a WSDL document, its own interface so two services offering the same functionality could have two totally different public interfaces. On the contrary each OGC service that implements a particular standard presents a fixed interface whose functionality (and the type of data returned) is defined a priori. OGC has proposed, over time, a quite wide and complete set of specifications but those most widespread and commonly used are: Web Map Service (WMP), Web Feature Service (WFS) and Web Coverage Service (WCS)[3]. In particular, with regard to these three types of services, in order to facilitate the development process, the OGC has developed, in addition to their single operational requirements, a Common Standard (Whiteside and Greenwood, 2010) that defines all aspects that should be common to all three implementations. These common structures relate to some of the parameters and data types used in the various request and response operations. Moreover, every OGC service must provide a standard way to describe its capabilities to its clients and a client may obtain such description by invoking the standardized GetCapabilities operation. The implementation of such an operation is mandatory and, for a client, there is no other way to know what the capabilities offered by a specific server are.

[3]A detailed discussion of the three main OGC standards is provided in Appendix A

2.2.3 Services Composition

One of the primary properties that should be guaranteed when realizing services consists in their seamless integration. Indeed, in addition to being platform or operating system independent, one of the hallmarks of service philosophy is the possibility to compose two or more services, developed by different entities in order to use their functionalities to *"build networks of collaborating applications distributed within and across organizations"* (Nano and Zisman, 2007). A composed service can be the result of the composition of elementary services, previously composed services or a combination of the above. The main advantages of service composition include better reuse of the service functionalities since the same service can be used in different contexts and better flexibility since the internal representation of a service can be modified (e.g., for refactoring or optimization reasons) without affecting the behaviour of the whole system (provided that the service public interface remains unaltered). Therefore instead of developing a whole application from scratch, an organization can realize its system by composing different types of services using, for example, third-party services for certain functionalities and focusing only on the development of those functionalities that represent its core business (Di Nitto et al., 2008).

However, despite the simplicity behind the general idea, services composition is a complex task and several facets must be wisely considered. As clearly explained in (Dustdar and Schreiner, 2005), the business logic does not lie entirely in a single monolithic entity, that is the operations and the algorithms that handle data and information exchange may also result from the composition of different services. This fundamental characteristic leads to the need of considering some key issues, such as the necessity to coordinate the sequence of operations to ensure the accuracy of computation and

avoid inconsistencies, and the necessity to use a transaction protocol (e.g., the WS-Transaction (OASIS, 2009) protocol used in Web services stack) suitable either for short or long running tasks. Besides these general problems, another important challenge described in (Dustdar and Schreiner, 2005) is related to the choice of the composition strategy that could be automated or manual, static or dynamic, that is whether the composition takes place during the design phase of the whole architecture or at run time. In addition, any composition mechanism, besides to provide a dynamic and flexible composition model, has to satisfy several non-functional requirements such as scalability, security and dependability (Milanovic and Malek, 2004).

Among the various approaches proposed in literature, two types of service composition are widely used, namely choreography and orchestration. As for service choreography, each service involved in the composition knows its role in the whole interaction. Choreography implementation is fairly simple, but discovering the source of a malfunction can be a difficult task. The Web Service Choreography Interface (WSCI) is one of the most widespread standards for the choreography of services (Barros et al., 2005). Service orchestration, instead, describes how Web services can interact by exchanging messages, including the business logic and the execution order of the interactions, thus benefiting from loosely coupled services. For simple orchestrations involving few components a possible and quite basic approach consists of the usage of a traditional programming language to link the various components. However, since programming languages are mainly focused on the definition of classes, methods or structures rather than on the overall execution process, for complex orchestrations it has been necessary to develop service composition standards accepted and used by all the involved entities. Among the current standards and proposals for Web services orchestration, the Web

Services Business Process Execution Language (WS-BPEL)[4] represents a robust and widely adopted solution. In the following subsections in order to build the basis for an integrated solution, requirements and properties for the OGC service composition are detailed and several approaches are presented. Finally, an initial analysis of issues related to the Web services and OGC services orchestration is performed also with relation to the OGC program for the interoperability.

Composing OGC Services

Due to its strong reliance on WSDL, it is quite difficult to use WS-BPEL for the composition of OGC services. As a matter of fact, there is no need for OGC services to be equipped also with a WSDL document. Moreover, WS-BPEL lacks support for the direct management of non-XML data, e.g., the binary data, such as the image files returned by a WMS. For these reasons researchers have investigated other modalities to manage orchestration of geospatial services. A valid example of an OGC services orchestration is provided in (Stollberg and Zipf, 2007). After describing difficulties of using WS-BPEL for the orchestration of OGC services, in this paper the authors propose the use of the OGC Web Processing service (WPS) standard introduced in (Schut, 2007) as a feasible solution.

The WPS standard provides for a standardized interface that aims at supporting publication and discovery of geospatial processes. According to the standard (Schut, 2007), a geospatial process includes *"any algorithm, calculation (e.g., polygon intersection) or model that operates on spatially referenced data"*. In particular, the purpose of the WPS interface is to standardize the way processes and their input/output are described, how a client can invoke

[4]See Appendix A

the execution of a process and how the output should be treated. Data that can be used during the elaboration can vary from image data formats to data exchange standards such as GML.

To achieve this goal, three mandatory operations are specified:

- GetCapabilities,
- DescribeProcess that allows a client to request and receive detailed information about processes that can be run on the service instance,
- Execute that allows a client to run a specified process implemented by the WPS.

In the proposed example, based on the search for adequate evacuation shelters in a bomb threat scenario, the authors first show how the use of WPS can supply all the traditional well-know GIS functionalities. Then, they also point out the need to combine all the developed services to represent them as a single application, that is the main idea behind the service composition (Stollberg and Zipf, 2007). To implement such a composition, they observe that, although the specification for the WPS mainly focuses on the implementation of geoprocessing methods, there are no restrictions on what can actually be implemented as a *WPS process*, that is a WPS can also be used as a service that coordinates an orchestration of geospatial services (Stollberg and Zipf, 2007). Moreover, they observe that by using WPS for composition purposes, three possible approaches are possible. The first two belong to general categories known as Centralized Service Chaining and Cascading Chaining. In the former a single service controls the entire workflow invoking all other services in order to achieve a goal; in the latter services communicating each other can directly exchange data. It is clear that these concepts are quite similar to the general definitions of orchestration and

choreography. The third option available with WPS is to combine all functionalities into a single WPS implementation. The ability to compose and orchestrate OGC services is of paramount importance for the development of Spatial Information Infrastructures (SII), distributed systems based on SOA principles that allow the processing of spatial data in order to provide useful information to the final user (Rautenbach et al., 2013).

Some guidelines that describe the general structure of a service-oriented SII can be found in an OGC best practice paper (Whiteside, 2005) which *"summarizes the most significant aspects of the Open Geospatial Consortium (OGC) web services architecture"*. The proposed architecture is based on the following properties (Whiteside, 2005):

a) A multiple tiers subdivision of the various service components.

b) Support for Transparent, Translucent or Opaque service chaining.

c) Use of open standards for the definition of the service interface.

d) Use of open Internet standards for the service communication.

e) Independence from specific hardware or software platforms.

While the last four points have been previously mentioned and discussed, the technical choices carried out for the first point deserve some further considerations. From a high level perspective, geospatial services are loosely arranged in four tiers as shown in Figure 2.2. These tiers are geospatial data independent although each of them includes specific services that deal with geospatial data. Each tier can offer its functionality both to other tiers and also directly to final clients. In fact, a complete separation in four tiers is not always required and clients can bypass the tiers that are not needed.

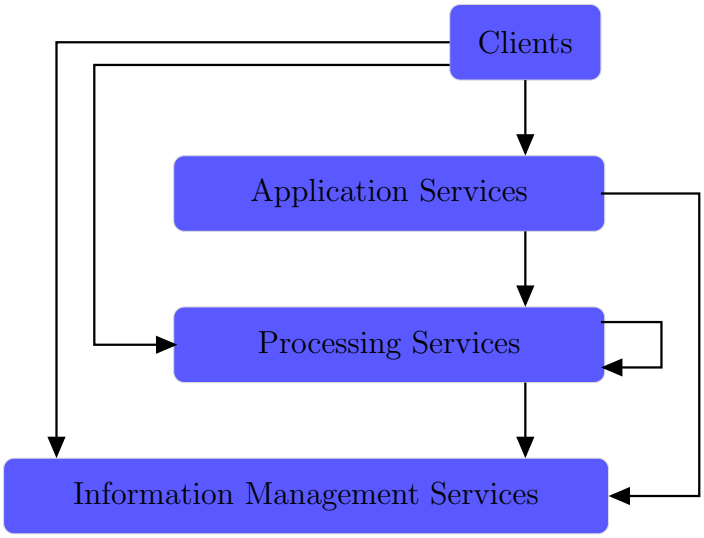

Figure 2.2: OGC Proposed Architecture. Service tiers subdivision

Such a characteristic is useful especially when a complete separation could be inefficient for the purposes of the final system.

The first tier, the Information Management Services includes services that provide access to geospatial datasets and are usually used to retrieve a subset of such data useful for the client. WMS and WFS are typical examples of services that belong to this layer. The Processing Services tier contains services designed to process data. Services in this tier can either use other services of the same tier or services from the Information Management Services layer. The Application Services tier encompasses those services useful to provide support functions to final clients of the system, in particular web browser clients.

The development of a SII is a non-trivial task exhibiting different challenges that, currently, can be overtaken only by adding some form of customization to the various services functionalities, thus proving the lack of flexibility and generality of current standards.

A complete example that demonstrates how the use of the various OGC standards can constitute a possible basis to perform distributed geoprocessing and that investigates the difficulties that arise during the development of such a type of systems can be found in (Friis-Christensen et al., 2007). In this paper the authors describe a use case for the development of an application that computes forest fire statistics. By it they present their architecture for the distributed computation and analyze and discuss some challenges related to the use of an SOA to perform geospatial computation. The proposed architecture is a traditional multilayer system where the data access services (WMS, WFS) provide access to data stored in the various distributed geodata repositories. An additional discovery service layer is used to provide users with a catalog service to allow them to discover what data could be useful for their needs. The execution of the diverse steps for the actual calculation of the various statistics is performed in the geoprocessing service layer. Finally, the client of the application is a traditional web application.

The prototype and the underlying architecture are used by the authors to make some interesting considerations that is worth mentioning here. First of all, the various steps needed for the effective calculation of the statistics are performed in a single geoprocessing service. This is a simple choice and guarantees good performance but limits the general flexibility since the different operations required cannot be reused in another application. A good solution could be the development of a single service for each functionality, which then have to be chained together in order to get the desired result. The three types of service chaining identified by the authors correspond to those supported by the proposed OGC architecture (transparent, translucent, opaque). In the first type the workflow is managed by a human user, in the second type a service that controls the chain is invoked and the human

is aware of the various steps, in third type an aggregated service is invoked and the user does not know the single steps. The first two types of service chaining are analyzed.

The first approach provides a high flexibility but the client application must continuously interact with the service and the continuous transmission of input data can increase the total computation time. The translucent approach could solve some of the transparent approach issues, since the entire workflow is sent to the service instance in a unique step. However the WPS specification would require some adaptation in order to support such an approach. The second important observation concerns the amount of time required to perform the processing of voluminous spatial data. This issue does not only affect the final user experience but might have also an impact on the services involved in the process. According to the authors the problem is caused by the synchronous communication mechanism on which the proposed architecture is based. A possible solution is the use of asynchronous messaging (supported by the WPS specification) for time-consuming operations. In such a modality the service response is provided at a later time in a different communication session (Friis-Christensen et al., 2007). The retrieval of information about the status of the process can be performed in two different ways: the pull and the push mechanisms. In the former the client carries out a periodic check, in the latter the service provider sends a notification about the status of the process. According to the authors the push mechanism is more convenient to alert humans (e.g., by email or SMS) while the pull mechanism is more suitable for the machine-to-machine communication. The pull mechanism is directly supported by the WPS standard. In such a mechanism the response of a WPS to the Execute request is an XML file containing a link pointing to a constantly updated Execute

response document. During the processing this document contains the status of the operation, when the processing is finished the provided URL contains the link to the final result. However, the asynchronous approach raises other research questions, such as what happens when an asynchronous call is performed in a service chain or the need to define a policy to deal with the data referenced by the URL contained in the XML response file.

Another type of difficulty that can arise during the development of an SII is represented by the lack of supporting infrastructure for geodata processing. An analysis of available frameworks for the orchestration of geospatial services can be found in (Rautenbach et al., 2012) where two different platforms are discussed and analyzed: the 52 North framework and the Zoo project. The characteristics of these two solutions are tested by using the production of thematic maps as case study. Both the frameworks are operating system independent, available as open source and compatible with the current WPS standard. The 52 North framework is Java-based and offers a web admin tool that helps the uploading of WPS processes, an orchestration API and a graphical modelling tool for the organization of the geoprocessing workflow. On the other hand, the Zoo project is made up of three main components: the Zoo kernel, the Zoo services and the Zoo API. The Zoo kernel is the module that allows the WPS creation and management. The Zoo services communicate with the Zoo kernel and are composed of two parts: a configuration file that describes the service, and the code that the final user wants to turn into a Web service. Finally, the Zoo API is a Javascript library that can be used for process creation and chaining. The two frameworks are compared against a wide set of characteristics, from the available documentation to the ease of integration with other GIS applications. Each of them has strengths and weaknesses (e.g., the lack of semantic information and support to the

WS-BPEL standard) but the results show that it is possible to use OGC standards such as WMS, WFS and WPS in order to orchestrate a thematic map service.

A more detailed example of OGC services orchestration to produce thematic maps can be found in (Rautenbach et al., 2013). In addition to the traditional OGC services, for the development of thematic Web services the authors discuss also the use of the Common Query language (CQL) and some non-standard extensions to the Styled Layer Descriptor (SLD). The former is a formal language to express queries to information retrieval systems, the latter is an OGC XML Schema useful for describing the appearance of a map. The described extensions can be used in conjunction with GeoServer, a Java-based server that supports the editing and sharing of geospatial data on the Web. The previously mentioned 52 North framework has been used for the orchestration of OGC services. The issues raised by the authors include the impossibility to run asynchronous operations, the need to perform some programming tasks for wrapping, in a WPS, all the statistical processing needed for the generation of a thematic map and the use of some GeoServer extensions to SLD. As such customizations are not part of an OGC standard, the portability of the proposed solution is limited.

The OGC Program for the Interoperability

When compared to W3C services, OGC services represent a totally different standard. However, the growing popularity of SOAP, WSDL and WS-BPEL and the awareness of the great advantages that could result from the possibility of seamlessly combining these two worlds, led the OGC to set a special working group in order to provide general recommendations and guidelines for adding WSDL/SOAP support to existing and future OGC services.

The first result is the awareness that for OGC services there is the need to define an Interface Definition Language, that is a language to describe the interface of a software component, usually in a language-independent way (Schäffer, 2008). A possible choice is, of course, the use of the WSDL. Currently, in OGC services the role to describe the available operations is carried out by the GetCapabilities function although a complete intersection with the WSDL specification is not possible. The main difference is that WSDL focuses mainly on the description of the explicit interface providing for both the list of available operations and the types of input and output messages, while the GetCapabilities provides only for the list of all operations along with meta-information. The proposed solution is that the GetCapabilities should list a path to a WSDL file that describes the OGC service. A complementary approach is the possibility for an OGC service to be discovered by a WSDL document and then, additional metadata could be fetched by using the traditional GetCapabilities operation. Other important differences outlined in (Schäffer, 2008) are the binding type and the binding time of operations. In W3C services the message payload is completely defined at design time while in the OGC services the type of a response message can dynamically vary based on the client requests (e.g., in the WFS GetFeature operation).

The last problem here discussed is how the SOAP protocol can be used in conjunction with traditional binary data, such as the images returned by a WMS. Binary data, usually called *opaque data* (Powell, 2004) often constitute a problem for (Web) services-based solutions. First of all, the serialization (i.e. the translation of an object into an XML stream) of such data into XML documents is not always an easy or feasible solution. For example, documents with digital signatures could lose their integrity (Pow-

ell, 2004). To deal with this problem two predominant techniques can be followed (Bosworth et al., 2003), namely either embedding, in some way, the opaque data in an XML element or referencing it as an external entity. The former is the currently used solution because the latter is inapplicable in a W3C compliant service environment, being based on XML features prohibited by the SOAP standard (such as Document Types Declarations). In XML, the support to binary data is usually achieved by using the base64 or the hexadecimal text encoding. The result for both is a sequence of octects (Bosworth et al., 2003). However, although these two encoding solutions are very simple to implement, a well-known problem concerns the size increase of the binary data as well as the overhead caused by the processing time needed to perform the encoding and decoding operations. To overcome these performance issues other proposals have been suggested, such as SOAP with Attachment (SwA) (Barton et al., 2000). SwA relies on the fundamental concept of MIME multipart messages, which simply means that a message is split into two or more parts and hence can include multiple attachments. The MIME standard specifies how these parts should be combined to form a single message. In SwA, the traditional SOAP message constitutes the root part of the MIME multipart message and the SOAP Body element contains explicit references to other parts of the MIME multipart message, which may contain arbitrary data (Schäffer, 2008). A problem with this approach is that it does not work with other fundamental components of the W3C service stack, such as the WS-Security protocol (Powell, 2004; Nadalin et al., 2004). Therefore the solution currently used (recommended also by the previously mentioned OGC working group) is represented by the Message Transmission Optimization Mechanism (MTOM). MTOM is a W3C recommendation that provides an efficient mechanism for exchanging large binary data by using

SOAP messages and is based on another W3C recommendation, namely the XML binary Optimized Packaging (XOP) that, among other advantages, does not require the time consuming task of the base64 encoding (Schäffer, 2008). Opaque data in MTOM are treated in a manner similar to what happens in SwA but, in this case, the SOAP message consists of the whole MIME multipart message (Schäffer, 2008). This makes it compatible with all other high level protocols of the W3C service stack (Powell, 2004).

Unfortunately, due to the large number of existing geospatial services not supporting SOAP or WSDL, the integration of W3C and OGC services is still challenging. However, as discussed in the remainder of this chapter, the growing necessity to integrate the two worlds has led researchers to investigate possible solutions that would allow the current W3C and OGC services to communicate.

W3C and OGC Services Integration

The opportunity to use the enormous amount of geospatial information accessible via OGC services within W3C services is one of main reasons stimulating the efforts to seamlessly combine these two different worlds. In fact, geographic community recognizes that a more complete integration with the SOAP and WSDL protocols would allow for employing all standards specified for W3C platform, such as those relating to security and rights management. To reach this aim, some issues have to be faced which cannot be solved by a mere mechanical process meant to make a translation from one service standard to another by simply transforming an OGC service interface into a WSDL document. Indeed, other aspects should be considered, e.g., the management of the metadata returned with GetCapabilities documents. In addition, it is relevant to avoid, as much as possible, both moving the compu-

tational complexity of the whole operation to the service client, and making any substantial changes to already existing services.

The focus of the following subsection is on a review of current solutions presented in literature. They share, as underlying idea, the design of a service wrapper or a proxy meant to provide for geospatial information in a W3C-compliant way, keeping the structure of existing W3C or OGC services unchanged. Then, the Inspire directive is presented, which aims at creating a common spatial data infrastructure for facilitating the sharing of environmental information.

An Overview of Current Solutions

A first proposal can be found in a discussion paper from OGC (Gartmann and Schäffer, 2008). In this work, the authors propose a generic approach to equip OGC services with a SOAP binding that allows for the transformation of any HTTP GET or POST request into a SOAP request. The proposed solution might be used as a basis for the construction of a wrapper to be applied to all OGC services, thus making the SOAP transformation completely transparent to the client application. The proposed architecture consists of a server side proxy and a client side proxy. The client side proxy receives the HTTP GET and POST requests and transforms them into a SOAP message, while the server side proxy receives this SOAP message and restores the original HTTP GET or POST request. For the response messages, the contents must be properly XML encoded in order to incorporate them in a SOAP document and, in particular, the previously discussed MTOM standard could be used for binary data and for those services that can also return plain or Html text, such as the WMS.

A more complete analysis of problems arising during the integration of

W3C and OGC services can be found in (Ioup et al., 2008). In this paper, some techniques for *"dividing OGC services and mapping capabilities into multiple SOAP services"* are presented and discussed. The core idea is to split a single OGC service into multiple atomic W3C services each representing a single geospatial dataset. As for the effective implementation, also these solutions are based on the creation of a wrapper aimed at solving a series of integration problems that the authors categorize into Data Handling, Functionalities Mapping and Metadata Management issues. The authors' work arises from a critique to the solution proposed in (Gartmann and Schäffer, 2008). Their evaluation is based on the observation that OGC services are based on a two-step process: a generic client first queries a server to know its capabilities and then uses the various functionalities of the OGC service to get the real data. The authors argue that by adding a simple SOAP transformation a W3C service client should perform three steps: get the WSDL, get the server capabilities and then get the real data. Therefore, on the basis of such a possibility, they discuss issues that have to be faced to realize this split. The first issue to deal with concerns the Data Handling since, as previously discussed, OGC services are not limited to XML as data exchange format. Moreover, different OGC services may return different data types. A viable solution could be the possibility for a W3C service to return only string data types. Although simple, this option is not admissible when it is necessary to return data in binary format. A different solution for binary encoded data could be to simply return a URL pointing to actual data. Through it, a client could directly contact the OGC service and retrieve the binary information. By this simple approach, a Web service is not in charge of managing data in binary format and might not act as a proxy for OGC services specific data. However, this approach transfers the compu-

tational complexity to the clients, a not always performing solution. Finally, a relevant part of the communication occurs outside the W3C service stack, thus blocking the use of the aforementioned standards, such as WS-Security. These motivations make this simple to implement proposal unfeasible. The only way to overcome these issues is the creation of a full wrap around the OGC service, thus forcing clients to communicate exclusively with the W3C compliant wrapper. Based on such a solution, a W3C service could be able to handle even binary data. For this task the authors propose to use the standard MTOM or, in case the Web service environment does not support this standard, the use of Base64 encoding (Ioup et al., 2008).

The second problem the authors discuss concerns the mapping of functionalities. In this case, some issues arise due to the fixed set of functionalities that OGC services have and that are described in the Capabilities document. To solve this problem, in (Ioup et al., 2008) two main methods are discussed. In the first method the WSDL document simply lists all the available operations of an OGC service, thus the same WSDL specification can be used for all the OGC services of the same type. However, a drawback exists, namely a direct mapping would lose information since the exact dataset of each OGC service can be retrieved only by parsing the document returned by the GetCapabilities function. For this reason the authors propose an alternative solution which represents the second method that they discuss in the paper. It consists in a direct mapping between the OGC service dataset and the W3C service functionalities, i.e., the mapping does not occur at individual functionality level. In particular, a wrapper can expose directly the OGC service data layer. This task can be performed in two different ways: 1) all data layers available in a single OGC function are mapped into a single and atomic W3C service; 2) each data layer is mapped into a different W3C

service; in this case function names will be the same for each Web service, but every service will return a single and different data layer. However, from a Metadata Management point of view, in the W3C services realm there is no standard for the management of spatial metadata while almost every OGC service requires them, thus removing them would mean to lose a lot of their usefulness. Then, it needs to provide a way to offer such metadata in a WSDL document and in (Ioup et al., 2008) the authors suggest to consider that, although WSDL documents do not contain metadata (except for those contained in functions provided by the service) such documents are, to some extent, extensible. Therefore, the proposed method includes metadata in the extensible part of a WSDL document. In particular, they suggest to include them in the <SERVICE> element of the document and to use XML Schema to encode the limits of the input parameters. Some constraints have nevertheless to be satisfied, namely the whole set of available metadata of an OGC service should be supported and they must not interfere with the proper use of the WSDL document. A further benefit of this solution concerns the possibility of preserving the two steps process of getting the capabilities and then executing an operation.

The difficulties to integrate efficiently Web services and OGC services are analyzed also in (Amirian et al., 2010). By using as a case study the development of a software system for the management of the Urban Services Data (USD) of a large city, the paper investigates the communication issues that arise in environments where the underlying data must be accessed by different types of users, in a reliable and up to date manner. In this case, further difficulties are represented by the needs of each system user and their way to access such data, namely different computing platforms and communication technologies, while performance and reliability aspects must

also be taken in account. As for the integration issues, a classification into Data Types handling, Functionality Mapping and Metadata Delivery issues is adopted as in (Ioup et al., 2008). To deal with the first task (Data Types handling), four different approaches are discusses and evaluated, namely the translator Web service, the physical Web service, the wrapper Web service and the common back end.

In the first approach, a Web service receives SOAP messages from a client and translates them in a format suitable for the target OGC service. This suitable format is sent back, in a SOAP message, to the client which uses it to retrieve data directly from the OGC service. In the physical Web service approach, the Web service translates the SOAP request and sends it directly to the OGC service; the OGC service generates the binary file and stores it in a permanent location on the server. The link to this location is sent back to the W3C service, which forwards it to the client in a SOAP message. Finally, the client uses this physical address to retrieve the binary data. In the wrapper Web service approach, the wrapper service catches the response message from the OGC service and sends it back to the client by using only W3C encoding. The whole communication between client and service is totally Web service-based. As for the binary data returned by an OGC service, the preferred choice is, again, the use of the MTOM standard. In the last approach, the common back end, a W3C service and an OGC service provide for *two direct gateways to the same server engine* (Amirian et al., 2010). A client can either send requests directly to the OGC service or can query the W3C service, which will provide responses in a W3C compliant way. Moreover in the latter case, according to the need of the client, the W3C service can return the possible binary data using their physical address, the Base64 encoding or the MTOM standard. Flexibility and performance are the main

advantages of this approach. As for the mapping of functionalities these two approaches and those discussed in (Ioup et al., 2008) are quite similar. In the former approach, a Web service replicates the same methods of an OGC service in a SOAP compatible way. In this case a client must first invoke the GetCapabilities method, parse the Capabilities document and finally invoke the desired OGC functionality replicated by the W3C service. The main disadvantage of this method is that a service consumer is able to exploit a service by exclusively using the published service contract, namely its interface. In the latter approach, the one-to-many mapping, each data layer of an OGC service is mapped by a specific Web service resource. In particular, a single Web service can expose a single data layer (multiple services method), alternatively, a Web service can expose a function for each single OGC service data layer (facade service) (Amirian et al., 2010). Finally, for the metadata delivery, three possible solutions are analysed, namely GetCapabilities function, WSDL extension, and Metadata exchange. In the first approach, each Web service provides a GetCapabilities function and a client has to parse the returned values. In the second approach, all relevant metadata of an OGC service are put in the extensible part of a WSDL document. The third approach, instead, proposes the use of Web Services Metadata Exchange (WS-Metadata) specification (Ballinger et al., 2008), that describes a standard format to encapsulate metadata. The main advantage of the third solution is that the usage of a standard and documented way for delivering service metadata drastically reduces the need of developing customized solutions for metadata retrieval.

In (Sancho-Jiménez et al., 2008), the integration of OGC and W3C services is addressed from another point of view. A method to automatically retrieve the SOAP interfaces and the WSDL metadata starting from the

mandatory operations (GetCapabilities, Describe Process, Execute) of any WPS is proposed. In order to provide these interfaces, the underlying idea is the creation of an intermediary proxy which is made up of two sub-modules. The former generates the WSDL metadata used to describe the WPS interface, the latter adapts a SOAP message in a request suitable for the WPS interface. As the proposed solution is an automatic derivation method, in order to generate the WSDL document, a request to the proxy must contain the URL of the WPS. Through this URL, the proxy uses the WPS public interface to retrieve all the information needed for the generation of the WSDL document. Moreover, the proxy offers the possibility to get both one document containing all the operations offered by the WPS, and a single document for each of them. Each generated WSDL document will include both the GetCapabilities and the DescribeProcess specification along with an Execute method for each operation offered by the particular WPS. In addition, in order to accurately define the parameters of the Execute operation, a parsing of the DescribeProcess response is performed. Finally, on receiving of a SOAP request, the proxy first parses the message, gets the WPS URL and then generates the request and invokes the WPS public interface.

The INSPIRE Directive

Another important example of difficulties met when combining the two standards in a heterogeneous environment handling large amounts of data can be found in the extensive documentation provided by the Infrastructure for Spatial Information in Europe (INSPIRE) (Villa et al., 2008a,b, 2009).

The INSPIRE project is an effort of the European Community to create a common spatial data infrastructure aiming at facilitating both the sharing of environmental spatial information among public sector organizations and

the public access to spatial information across Europe. The fulfilment of such a common spatial data infrastructure is a non-trivial task that requires overcoming some fundamental issues, such as the fragmentation of datasets and sources, the lack of harmonization between datasets and the duplication of information. Moreover, the open standards compliance, the definition of a common contract for all interfaces, and the data description and representation constitute other key aspects addressed by INSPIRE. In particular, for the pressing need to use wide adopted standards, the INSPIRE Network Services SOAP Framework (Villa et al., 2008b) describes core ideas behind the proposal of a SOAP framework for the INSPIRE infrastructure as well as issues and solutions related to different geospatial domains. In (Villa et al., 2008b) the authors observe that although based on open standards the existing OGC services support a mix of protocols and technology bindings. Unfortunately, such a technology mix could slow the integration and the implementation process, thus it should be avoided in order to get the maximum benefit from the offered services. Then, on the basis of all possible solutions, risks and requirements, SOAP has been proposed as the default communication protocol and binding technology for the INSPIRE services. In (Villa et al., 2008b) the authors also present some criticisms to the OGC about the Consortium decisions concerning the main aspects of a hypothetical SOAP framework, namely there is not a common choice for data encoding, data transport and representation and for a profitable use of SOAP Headers. The proposed INSPIRE framework focuses on the following topics (Villa et al., 2008b):

- Standard compliances,
- Underlying protocols binding,

- Use of the SOAP Header,
- Exception report in SOAP encoding,
- Data encoding style and use, and
- Binary data transport and representation.

As for the standard compliance, beside the choice of SOAP as communication protocol, significant role is given to the Web Services Interoperability Organization (WS-I) Basic Profile recommendations (Ballinger et al., 2004) in order to provide the highest level of interoperability. The WS-I Basic profile consists *"of a set of non-proprietary Web services specifications, along with clarifications, refinements, interpretations and amplifications of those specifications which promote interoperability"*. In contrast, the OGC proposals are not WS-I compliant. As for the use of the SOAP Headers, (Villa et al., 2008b) presents some concrete examples of how such headers can be effectively used. In fact, although their use in the OGC integration proposals is not compulsory, the authors propose to use them to manage, in a modular way, some common aspects of the INSPIRE Web services, including for example the use of SOAP Headers for security purposes. In fact, a header block could be used for carrying security-related information to a specific recipient. Another way to take advantage of SOAP Headers consists of using them for checksums and signature purposes: a message producer could want to provide recipients with a message along with a means to determine whether a message was altered during its path. Moreover, a Header block could carry information useful to a receiver to check the integrity of the binary data attached to the SOAP message.

A SOAP Header could also be used to provide for human readable information without interfering with the content of the Body of the SOAP

message. Finally, a SOAP Header could be used to transport all metadata or information not strictly connected with the message encoded in the Body element. INSPIRE suggests this solution also to solve the multilingualism issue.

2.2.4 Chapter Summary and Key-points

In this chapter we introduced and discussed the fundamental characteristics and issues that may influence the design of both the front and back ends of modern ubiquitous Information Systems. We started our investigation by analysing the two challenging tasks of user requirements elicitation and mobile UI design. To support the accurate collection of users requirements and to better integrate HCI and usability techniques into the overall software development lifecycle, several methods have been proposed. Among them two useful methodologies are the *scenario-based approach* and the *persona technique*. In addition to the correct requirements elicitation, the design choices carried out for the UI development represent the other main factor that greatly influences the overall usability of interactive software systems. Developing an effective UI is a particularly challenging task especially in the mobile world where factors such as the peculiarities of the intended final users of the mobile application, type of task to be performed or the environment surrounding the final user have, more than in the traditional desktop environments, a substantial impact on the design choices of usability engineers. Current research efforts investigate the exploitation of the various sensors available on modern mobile devices to develop new interaction modalities such as the combination of human voice to provide an input and the use of vibration patterns to report an output. However the best combination of such modalities needs to be wisely chosen in order to avoid serious design

inconsistencies. As for the *user variable* mentioned above, apart from well-known aspects such as, for example, the need to avoid an excessive cognitive workload, it is necessary to observe that the design of a usable mobile UI cannot exclude the social context where it will be mostly used. Finally, it is worth noting that a considerable amount of potential and effective users of mobile services is made up of semi-literate people or people living in developing countries. Therefore the specific usability needs of these users (e.g., avoiding the use of specialized terminology or the use of complex hierarchical structures) must be carefully taken into account during the design phase.

In the second part of the chapter, we focused on the SOC paradigm, the currently most adopted approach for the design and development of large scale and distributed supporting infrastructures. This new computing platform is based on the idea of service, an independent software module that performs certain, more or less complex, operations. Technology neutrality, loose coupling and the support for location transparency are some of the main reasons behind the SOC wide adoption. Another fundamental characteristic is represented by services composition namely the ability to compose different services, developed also by different organizations to provide with complex functionalities. Services composition is a quite challenging task. Some relevant problems are represented by the need to coordinate the sequence of operations to ensure the correctness of the computation and avoid inconsistencies and the need to have a commonly accepted composition model and a language to specify the services involved in the composition. As for the actual technologies used for the development of SOC-based solutions, they mainly rely on the proposals of the W3C that has defined a series of universally accepted XML-based standards. Among them, the two cornerstones of the whole W3C services stack are the SOAP protocol for the exchange of mes-

sages and the WSDL for the description of the service interface. The benefits of the SOC paradigm soon become clear also to the GIS community who saw in this technology a possible way to overcome some common problems such as rapidly sharing data between distributed and heterogeneous sources and the achievement of spatial interoperability. In this context, the proposals of the Open Geospatial Consortium have become the *de facto* standard for developing distributed geospatial applications. Unfortunately, although geospatial services share some common principles with traditional W3C proposals (e.g., the use of the XML for the encoding of messages) the other core technologies they rely on are quite different. The first important difference between the two sets of standards is represented by the strong standardization imposed by the OGC regarding the public interface of a geospatial service. Another difference is that W3C services usually rely on pure XML documents while OGC services can return also binary data (as well as XML documents) and, finally, the last fundamental difference concerns the binding type and the binding time of operations. In the OGC services the type of a response message can dynamically vary based on the client requests while in W3C services the message payload is completely defined at design time. Since a better integration of the two sets of standards would benefit the two worlds, several proposals have been made to promote a better communication between W3C and OGC services. A possible solution could be the adaptation of a set of standards to the syntactic and semantic rules of the other set. Unfortunately such a proposal is not feasible in practice due to the large amount of services developed according to the current standards. Therefore common research efforts rely on a common underlying idea related to the design of a service wrapper that *translates* the request and response messages from services developed according to the W3C standards to services developed according to

the OGC standards and vice-versa. However translating from one standard to another is not straightforward since three fundamental problems have to be addressed namely the properly management of binary data and metadata returned by geospatial services and the mapping of the specific functionalities offered by each service.

Chapter 3

Requirements Gathering and Application Development for New Types of Problems

The requirements elicitation phase is of central importance to information systems development. Nevertheless, irrespective of the problem being addressed, gathering correct user requirements is a challenging task especially when the developed system should not only meet the functional requirements but also be easy to use by the intended users.

A correct elicitation is even harder when designing innovative ICT-based solutions for new and specific types of problems. The ad-hoc nature of the business processes and the absence of any existing solutions, make gathering the requirements for this latter case a very complex task. In this chapter we discuss the rationale that led us to propose a novel solution to successfully incorporating both functional and usability requirements when developing information systems for such a new type of problems. The proposed contribution results from our approach to the development of a mobile information

system aimed at helping Sri Lankan farmers with their daily activities such as crop selection or cultivation planning. The problem we dealt with greatly differed when compared to usual software engineering projects. We heavily depended on the user to get a clear understanding of the problem domain. But we were unable to ask users about possible ICT based solution or the requirements to develop such a solution as the users were not aware of the real possibilities.

The chapter is structured as follows. We introduce, first of all, the case study and discuss the main characteristics of the Design Science Research (DSR) approach (Peffers et al., 2006), a pragmatic research paradigm that guided the design and helped us to capture the knowledge created during the design process. Subsequently we introduce and discuss our approach for usability driven requirements gathering and application development. The proposed methodology blends a range of technologies using the DSR approach. Thirdly we describe in detail how we applied the steps of our methodology to the case study of Sri Lankan farmers. Some reflections are finally provided.

3.1 Case Study

Several reasons led us to consider Sri Lanka as the country where a pilot research study could be carried out. In Sri Lanka, mobile technology is occupying increasing importance in society, while the era of personal computers has been skipped by most population. At the end of March 2012, the number of Cellular Mobile Subscribers was around 91.3% of the total population (Telecommunications Regulatory Commission of Sri Lanka, Statistics, 2012).

In this context, during the last four years after the end of civil war, the government has promoted several initiatives with the goal of supporting

social, cultural and economic development. One of the major challenges has been to find strategies to solve digital divide directly by the use of mobile technology and smartphones. People from Government, private sector and other stakeholders have been firmly convinced that ICT is a key determinant for the competitive advantage of nation and may lay the foundation for a society with equitable distribution of opportunities and knowledge. However, rural Sri Lanka, where nearly 70% of the population lives, is yet to profit from these developments. In such regions, agriculture employs the largest share of the workforce yet it contributes the least to the Gross Domestic Product (GDP) when compared to the industry and services sectors.

After a preliminary analysis, the main problem that strongly limits the development of this crucial sector appeared to be the overproduction of some vegetables and under supply of others due to the fact that many farmers grow the same crop in the same area without having an awareness of what others are growing (Hettiarachchi, 2011, 2012). Such an issue has a negative impact on the farmers expected income and is one of the main causes of the continuous labor lost, within the last two years, in the agriculture domain of Sri Lanka. In late 2011, when we discovered this problem, there was no long-term solution. Most of the time, the Government of Sri Lanka implemented temporary solutions to protect the farmers. Such solutions neither could address the issue nor could reduce the damage caused. During our investigations we saw a potential long-term solution using the latest advances in mobile technology.

3.2 The Design Science Research approach

The Design Science Research (DSR) is a matured research methodology in Information Systems (IS) (Peffers et al., 2006; Hevner and Chatterjee, 2010; Hevner et al., 2004). This is a *problem-solving paradigm* (Hevner and Chatterjee, 2010) which seeks for an innovative solution to increase the efficacy through the development of an artifact. In literature we can find various DSR process models aligned with various disciplines (Peffers et al., 2006; Hevner and Chatterjee, 2010; Hevner et al., 2004; Vaishnavi and Kuechler, 2004), although DSR has been successfully applied and validated in the IS research community. As shown in Figure 3.1, this process model includes six major steps, namely Problem identification & motivation, Objectives of a solution, Design and development, Demonstration, Evaluation and Communication.

Problem identification and motivation represents the stage where a specific research problem is identified and the importance of the solution is justified. Resources, such as *knowledge of the state of the problem and the importance of its solution* (Peffers et al., 2006), are required at this stage. Based on the problem identified next is to identify the main objectives of the solution. Knowledge with respect to the problem, existing solutions and its importance is used to infer the objectives. Objective of a solution will then transform to an artifact within the phase of Design and development. Artifact produced at this stage could include construct, models, methods or instantiations (March and Smith, 1995). Once an artifact is created, its efficacy to solve the problem is demonstrated by using some methods, such as experiments, case studies, and proofs (Peffers et al., 2006). The results observed in demonstration stage are further evaluated to check whether the objective of the solution is successfully achieved. Based upon reflections and feedback this process would iterate as shown in Figure 3.1 to increase the

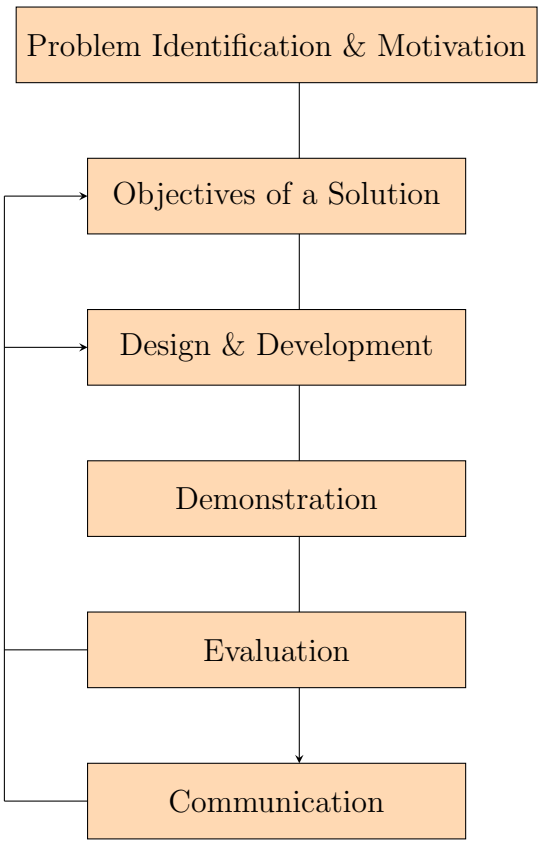

Figure 3.1: Design Science Research process

efficacy of the solution. The knowledge acquired by performing these steps will be communicated further to the research community and other relevant personals.

3.3 The Proposed Methodology

The framework we propose (Figure 3.2) is built on top of the DSR methodology (since our aim was to design an artefact; a mobile based information system for farmers) and focuses on usability aspects from the beginning of the design process. In the following, by describing the steps shown in Figure 3.2, we provide a discussion of the approach we followed.

The first step involved the recognition of the existing issues in a systematic and in-depth manner. To gain a broader understanding of the application domain and identify the issues that were contributing to the problem, we reviewed related literature, conducted surveys, interviewed farmers and agricultural officers. In this context, findings further confirmed the feasibility of a mobile based solution (Figure 3.2, Understand the problem domain). Based on the data collected from 2 surveys with farmers and one with Agriculture officers we performed a casual analysis to obtain a deeper view of the problem domain. We identified crop choosing, growing and selling stages as the key phases that create a direct impact on the farmer revenue (Figure 3.2, Understand the causes). To visualize the physical form of the conceptual solution we formulated, we used the scenario-based approach to design the first set of interfaces. In this approach we created a set of tyical scenarios and personas of actors based on data gathered from the surveys. Next we investigated how information deficiencies in these scenarios can be mitigated by providing missing information compared to the information needs identified in the

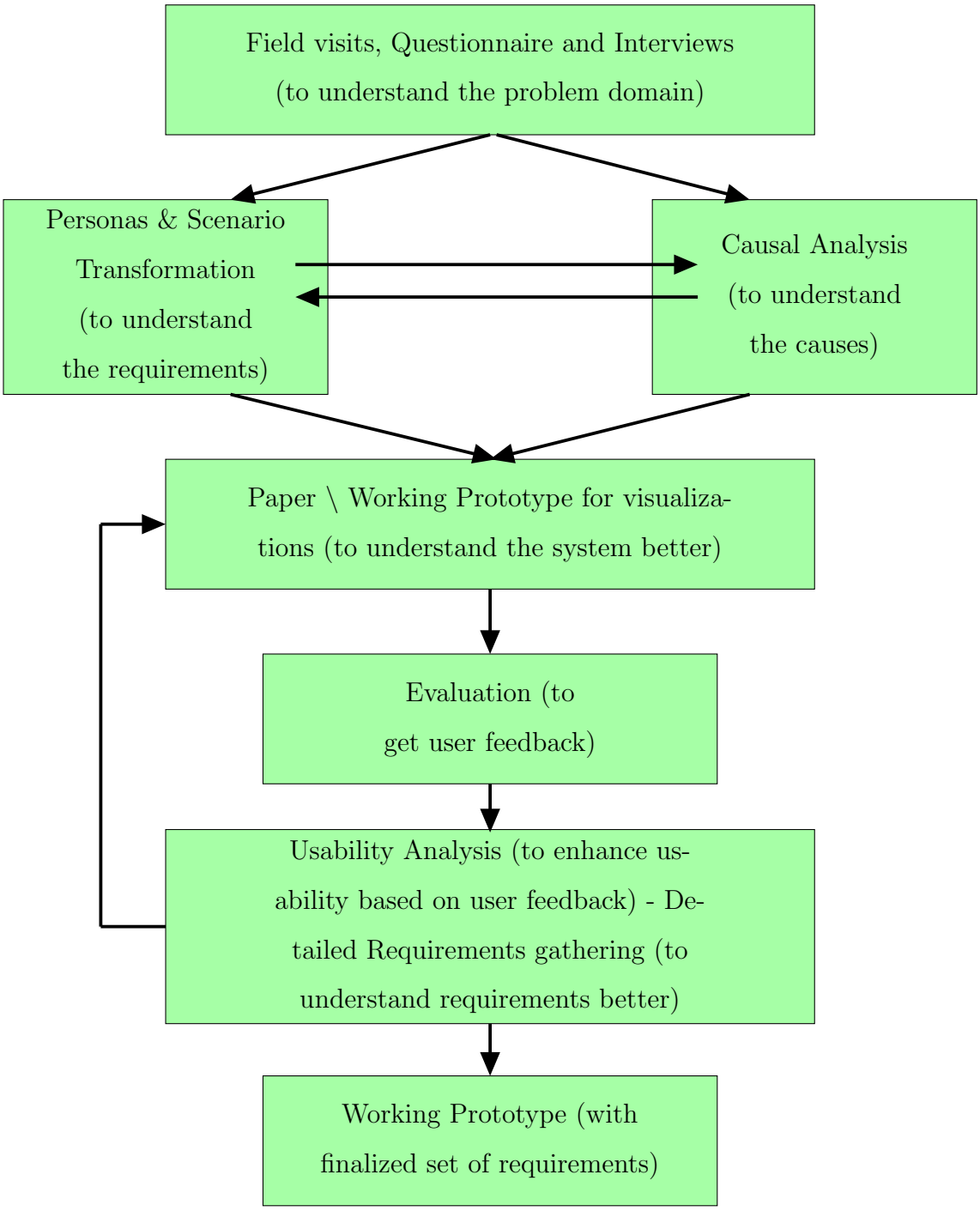

Figure 3.2: Process steps followed in the research

causal map. In particular, in few brainstorming sessions on how best farmers can be empowered enabled us to identify new scenarios not only to empower the farmers but also to generate the dynamic information that is required by farmers. Moreover, based on such scenarios and personas we also identified the usability requirements. We used transformed scenarios and usability requirements to develop the first set of user interfaces. This design gave the whole research team a good idea about how information can be visualised and user input can be captured (Figure 3.2, Understand the requirements). Analysing these scenarios we were able to identify that our system should assist farmers to diversify the crop production to prevent possible oversupply situation which drastically affect the selling price. Not knowing what neighbouring farmers are producing was identified as a major negative factor that gives rise to current price fluctuations and over production situation. We created the first functional prototype by combing the artefacts designed in the previous phases. Moreover, in order to gather user requirements for our future design, we asked farmers to specify their requirements. Starting from the scenarios, design claims and application requirements we derived a list of interface requirements that we used to design the actual application (Figure 3.2, Understand the system better). This prototype was field trialled in December 2012 in a main vegetable producing region in Sri Lanka with 32 farmers (Figure 3.2, Evaluation). We analyzed the findings in detail and refined the application accordily (Figure 3.2, Usability Analysis - Detailed requirements gathering). Finally we extended the system to provide information required to support decision making at various stages of the farming lifecycle (Figure 3.2, Working Prototype).

The blended nature of several techniques within a DSR framework differentiates our work with the rest in the literature (Hevner and Chatterjee,

2010; Vaishnavi and Kuechler, 2004; March and Smith, 1995). In particular, based upon the findings of the first DSR cycle we planned three more DSR cycles to meet usability characteristics. Thus, while gathering user requirements our methodology enabled us to meet the usability aspects of the designed solution. This is an achievement since today most of the designs fail due to the lack of focus on usability aspects. Then, our study will further contribute to the DSR knowledge repository for the future IS researchers. Another relevant result is represented by the possibility offered by the DSR framework to share the experience and knowledge within the different user groups (ranging from end user, developers, designers and researchers).

The gap between the extremes of end user and researcher were minimized in this work. Constant interactions with the users facilitated throughout the DSR process enabled us to identify their expectations and goals. Visual interfaces used in the latter stages, allowed us to communicate with farmers successfully. Through these methods we succeeded in arriving at a realistic solution. Farmers also found that this is an easy mechanism in providing requirements more freely. We observed in the later part of the project that the farmers proactively expressed the requirements compared to early stages of our research. The incremental techniques used throughout the DSR process led us to easily accommodate the evolving requirements with less effort.

In addition, it helped the research team spread over four continents to share their expertise to derive the solution. Moreover, the method that we suggest in this chapter focuses on usability aspects from the beginning. Then, we argue that this would increase the success in designing any type of ICT based solution even the stakeholders are more familiar with the requirements.

3.4 Profiling the target Users Community

Designing a mobile software solution for a specific community of users requires a deep knowledge of that community. The peculiar social and cultural constraints may in fact invalidate existing design guidelines and HCI patterns issued to address common usability challenges or provide guidance in the applications design. To properly take into account the importance of such constraints, the first step of our investigation has been the formulation of a set of guidelines that, independently of the specific problem domain, can be used to profile crucial aspects of a community (Figure 3.3). The collected information can then be exploited to better formulate usability goals and taken into account throughout the requirements, design, prototyping and testing phases (Sebillo et al., 2013). The structure of these general rules is as follows.

- Social Context (Social Organization, Ethical beliefs, etc.)
  - Consider the social organization of the community. It *describes the collection of values, norms, processes and behaviour patterns within a community that organize, facilitate and constrain the interactions among community members* (Mancini et al., 2003)
    - Find out the social necessities and limitations.
    - Find out social relationships between individual subjects belonging to the same community.
    - Find out possible participation in governative or voluntary organizations.
- Cultural Context (Language, Education)

- Consider the average cultural level of the target community. The designed interface should be easily understood by any stakeholder.
- Consider the average education level of community members. Never make a design choice based on incorrect knowledge assumptions about the user.
- Different languages may affect the use of text in the visual design of the interface. This is especially true for communities living in some Eastern countries and where the official spoken language uses an alphabet different from the common Latin alphabet. Consider that:
  * sentences could be hard to represent on small screens,
  * the (virtual) keyboard could be missing some characters of the language alphabet. This would again affect the choice of the mobile device.
- In some countries more than one language is spoken. So, consider the necessity to design a multi-language interface.
- Consider the semiotics of the target community. Signs, colors, symbols, metaphors can have different meanings in different environments.

- Technological Context (Available technology, Familiarity with mobile devices)
  - Consider the technological means available in the geographic area of the community. Also consider the average degree of familiarity with the mobile technology you are planning to use and the attitude to learning new technologies.

- Mobile devices operations often depend on remote services. Therefore connectivity issues are paramount in this context:
  - consider the data quantity that the system needs to transmit on the wireless networks,
  - bad connections can cause loss of reliability and can make the application progressively slower causing usability problems too.
- The application should run on the majority of the devices available in the community. The world market trends can suggest some devices, however in specific communities particular technological ecosystems could be found.
- Consider the device models available to that community in the specific country. Advanced devices may be present in rich communities (but this is not a rule), in some countries some models could not be available.

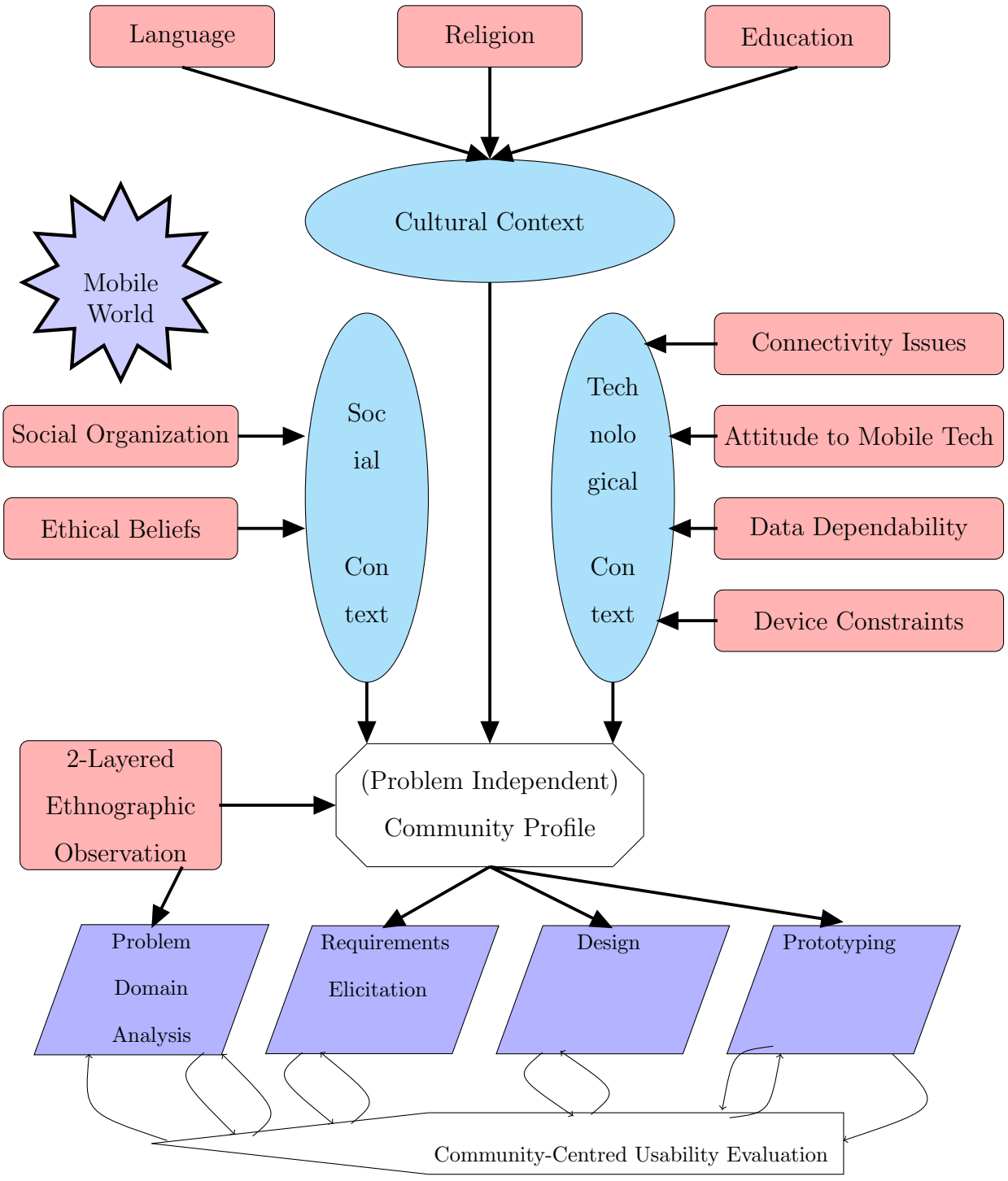

Figure 3.3: Community-oriented design and development

3.4.1 The Case of the Sri Lankan Community

In the following we present some relevant aspects of the Sri Lankan community that directly impacted on the design choices concerning the information system front end. The Sri Lankan society presents opposing aspects: people reflect both some of the typical facets of the modern advanced western societies and some aspects strongly connected to a multiethnic culture, full of ancient traditions. As a result of the initial ethnographic study, we were able to depict a general community profile for people living in Sri Lanka who may benefit from the use of mobile devices for their daily activities. This represents the common knowledge about the social, cultural and technological contexts, which could be exploited for several application domains.

Social Context - Young people have not been extremely influenced by modernity and the effects of westernization. 80% of them have religious beliefs, the majority being Buddhists. Mobile devices are quite widespread but there are still barriers to their adoption, especially among parents who are often concerned with security and reputation issues. Moreover, peoples general attitude to ICT is often influenced by the opinion of prestigious members of the community, such as local temple priests.

Cultural Context - Sri Lanka is a real multicultural nation. The community is made up of two main ethnic groups, namely the Sinhalese and the Tamil. 83% of the members speak Sinhalese and the remaining 17% speak Tamil. However, English is the third official language, mainly spoken in the cities.

Technological context - Like in many developing societies, a significant gap in ICT can be observed in Sri Lanka, especially among people living in villages, sometimes even missing electric power supply. In order to encourage the adoption of technology, the local government has recently created

the so-called *Nenasala telecenters* (Telecenters in Sri Lanka: The Nenasala project, 2010). Computer desktops and laptops are absent for the majority of individuals, while mobile devices are widespread enough. Statistics say that 86.5% of the population owns at least one SIM (Subscriber Identity Module). The Sri Lankan mobile network currently covers almost the totality of the urban areas although temporary lacks of connectivity may affect services that heavily rely on network uptime. For what concerns the farming population, in particular, we can mention a relatively low computer usage (20%) when compared to the mobile phone usage (92%). Moreover, the possibility among the farmers to buy a smartphone was higher than buying a computer. Low cost and accessibility are the main reasons for their choice.

3.5 Problem Domain Analysis

We conducted 2 surveys with framers to better understand the context related to Sri Lankan farmers and factors that influence their selection of a crop(s) to grow. We confirmed the finding by discussing these with a group of agriculture officers at the Department of Agriculture. The initial investigation was carried out involving 12 farmers representing different cultivation regions in Sri Lanka. This was mainly conducted to identify issues faced by the farmers at different stages of farming cycle. With the aim of further exploration the second survey was carried out in a rural village few miles away from Dambulla (largest agro-based area in Sri Lanka). Fourteen farmers took part in this survey. This survey was planned based on the knowledge gained after analysing the responses provided for the first survey. The details about the characteristics of the interviewed farmers are shown in Figure 3.4.

From the data gathered from the interviews it became clear that the over-

Characteristics	Unit of Measurement	Categories	Respondents Survey 1		Respondents Survey 2	
			Number	%	Number	%
Age	Years	Young (<36)	4	33%	0	0%
		Middle aged (36-45)	4	33%	2	15%
		Old(>46)	4	33%	12	92%
Literacy Level	Year of Schooling	Graduate	1	8%	0	0%
		A/L	4	33%	0	0%
		O/L	7	58%	3	23%
		Above 8th Grade	0	0%	4	31%
		Below 8th Grade	0	0%	6	46%
	R/W Capability (Native Language)	Ability to Read	12	100%	12	92%
		Ability to Write	12	100%	11	85%
	R/W Capability (English)	Ability to Read	1	8%	0	0%
		Ability to Write	1	8%	0	0%
Mobile Phone Usage	Model	Basic	11	92%	12	92%
		Sophisticated	0	0%	0	0%
	Purpose	Taking calls	11	92%	12	92%
		Sending SMS	11	92%	9	69%
		Playing games	0	0%	1	8%
		Getting price details	8	67%	6	46%

Figure 3.4: Farmers Characteristics. Source: (De Silva et al., 2012)

production problem was only a symptom of a much deeper problem. Most of the interviewed farmers were traditional farmers and they were compelled in growing the usual crops that they are well familiar. The knowledge on what crop to grow is gained mainly based on the practical experience as well as from the elders. It was also observed that farmers had no way of knowing what the current production levels for a crop is at the time of deciding a crop to grow except observing what the neighboring farmers are growing. In addition, we also discovered that farmers in Sri Lanka adopt different selling mechanisms. Some bring the harvest directly to the market, while around 90% depend on a middle person, namely the transport agent or the shop-keeper. However, none of them get help from the Government to sell the harvest. Another interesting fact is the behavior with respect to the selling prices. The selling price is a dynamic value that changes very frequently at a particular market. The farmers reported that they were often unable to predict the price as it changed vigorously within few hours. The major problem was that they were unable to gain a good price for their harvest at the market, because all farmers tend to grow the same crop at the same time. Below, we summarize the most important claims about the domain-specific issues identified from the initial interviews:

- Users were disposed to use some technological instruments provided that they are not invasive,
- Governmental centers aimed at supporting agricultural activities are located all around the farms, and
- There was a very low level of trust among the members of the same community, insomuch as not sharing basic information about their crop production.

3.6 Extracting User Requirements

By exploiting the knowledge gained from the surveys and community profile, the research team developed a causal map (Figure 3.5) and several personas (Table 5.1, page 154) and scenarios (Table 5.2, page 157 and Table 5.3, page 157) for the problem domain in order to better understand the impact of not having right information at the right time (Di Giovanni et al., 2012). As shown in Figure 3.5, in view of farming domain, revenue is determined by the selling price of the harvest. There are three main price determinants for a specific crop yield. Yield quality, supply and demand create a huge impact on price fluctuations at the market level (De Silva et al., 2012). As for scenarios, they were used to better analyze the survey findings and, combined with the wider context, to gain insight into farm activities during the sowing and selling times of crops and identify some of the factors that contribute to the oversupply of a crop. The proposed scenario of existing working practices gave us the opportunity to reason about what were the major requirements that emerged from the rural Sri Lankan context and helped us to start our brainstorming activity for the formalization of initial requirements and the design of a possible solution. At this stage, we were primarily interested in deriving a list of farmers requirements that we could take into account throughout the design process.

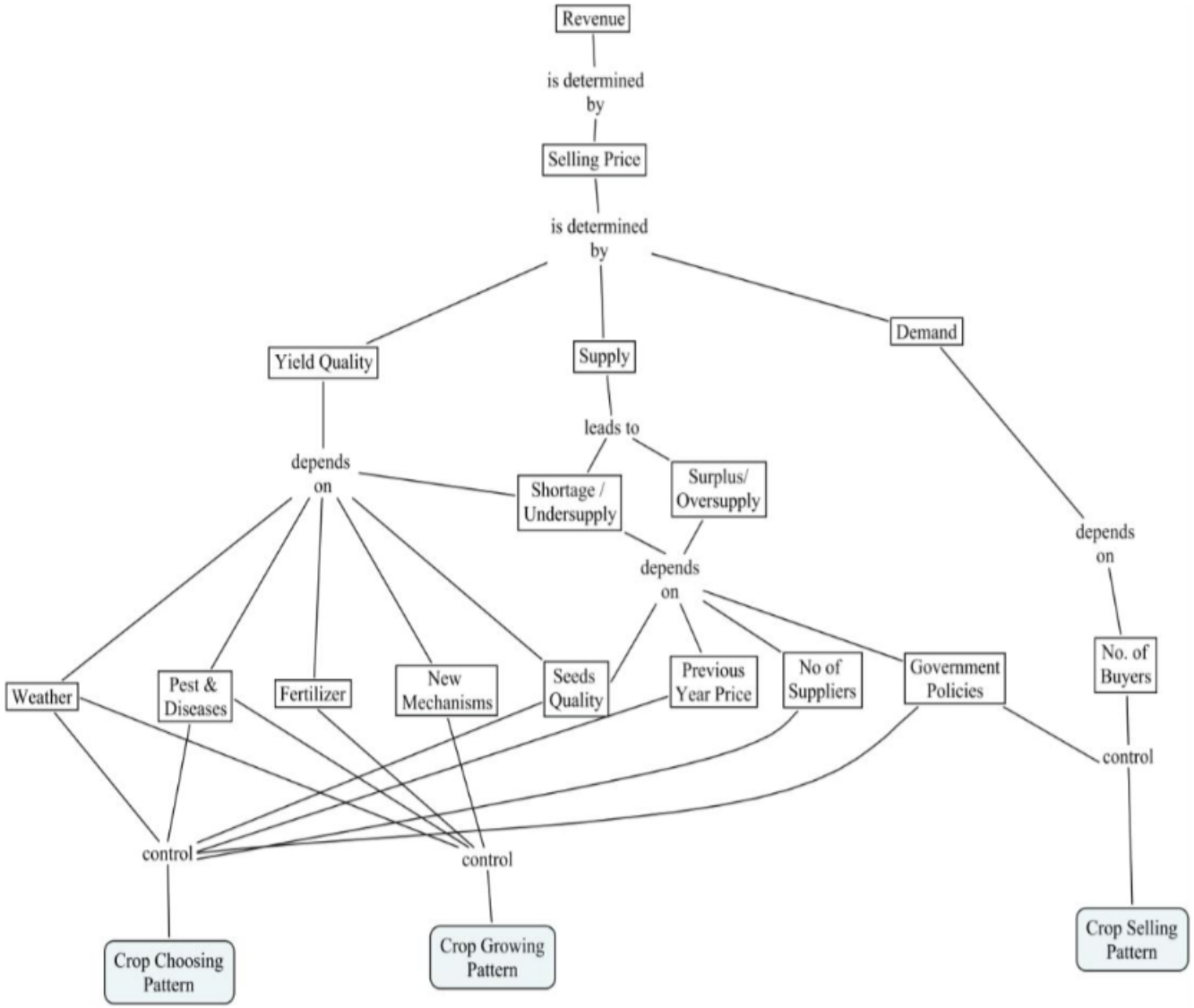

Figure 3.5: Causal Analysis of the Problem Domain. Source: (De Silva et al., 2012)

Summing up all the considerations and discussions we had upon the field-work completion, and reasoning on the derived scenario, we were therefore able to elicit an initial set of requirements divided into five categories according to the classification described by (Preece et al., 2015). (Table 5.4, page 158) and (Table 5.5, page 160) explain the rationale for each requirement.

From the above analysis we identified a software application aimed to assist farmers in diversifying the crop production to avoid selling prices of crops changing vigorously in few hours. Since users are distrustful of technology with the exception of mobile devices, developing a mobile application seemed to be the right direction. The application should receive geographic coordinates of the users location of the farm and should provide them with valuable information about the kinds, the quantity and last selling prices of neighboring crops. Users may use this information in selection of crops for cultivation. The idea behind this application led us towards our scenario transformation activity (Table 5.6, page 161).

Based on the described scenarios we had some brainstorming meetings to identify the most important design claims.

Design claim 1. By retrieving the field coordinates, automatically provided by the integrated GPS module, farmers access only information about estimated quantities and the last selling prices of neighboring crops. This allows farmers to select the appropriate crops in order to provide a larger variety of products and to make selling prices live up their expectations.

Design claim 2. Data presentation should be provided in easy and immediate way exploiting the communicative power of images and color language. This allows small screen of the mobile device to provide users with complex information.

Design claim 3. The UI should provide users with a small number of

menu levels and operations. Farmers use the application just few times a year so that they need to be able to use the application without requiring a long training effort.

By scenario development and transformation methods we got a better understanding of the system goals. We were able to arrive at a potential solution where the production quantities along with cultivated crops are shared by the farmers via mobile-based system. In fact the major functional requirement was that the proposed application would allow the exchange of heterogeneous data between neighboring farmers guaranteeing anonymity. By sharing valuable information on crop cultivations, users would be able to make better decisions during the crop selection activities. Furthermore, because of social phobia and the competition among farmers, users were more conformable sharing information while preserving anonymity. Other requirements derived from the surrounding environment. For example, users might be using noisy production equipment or they might be speaking with co-workers while using the application in the field. Figure 3.6 describes the information architecture of the mobile application. As a first step the application requires users to log into the system. Next the geographical coordinates of their farm location need to be provided. These are two technical steps are very much prone to user errors. After these two steps users reach the crops catalog and select the desired product. Once they select the product a new view of the interface allows them to insert the planned quantity of the crop. Using these findings we designed the first UI version for a mobile application to inform the farmers on prevailing supply and demand situation for different crops.

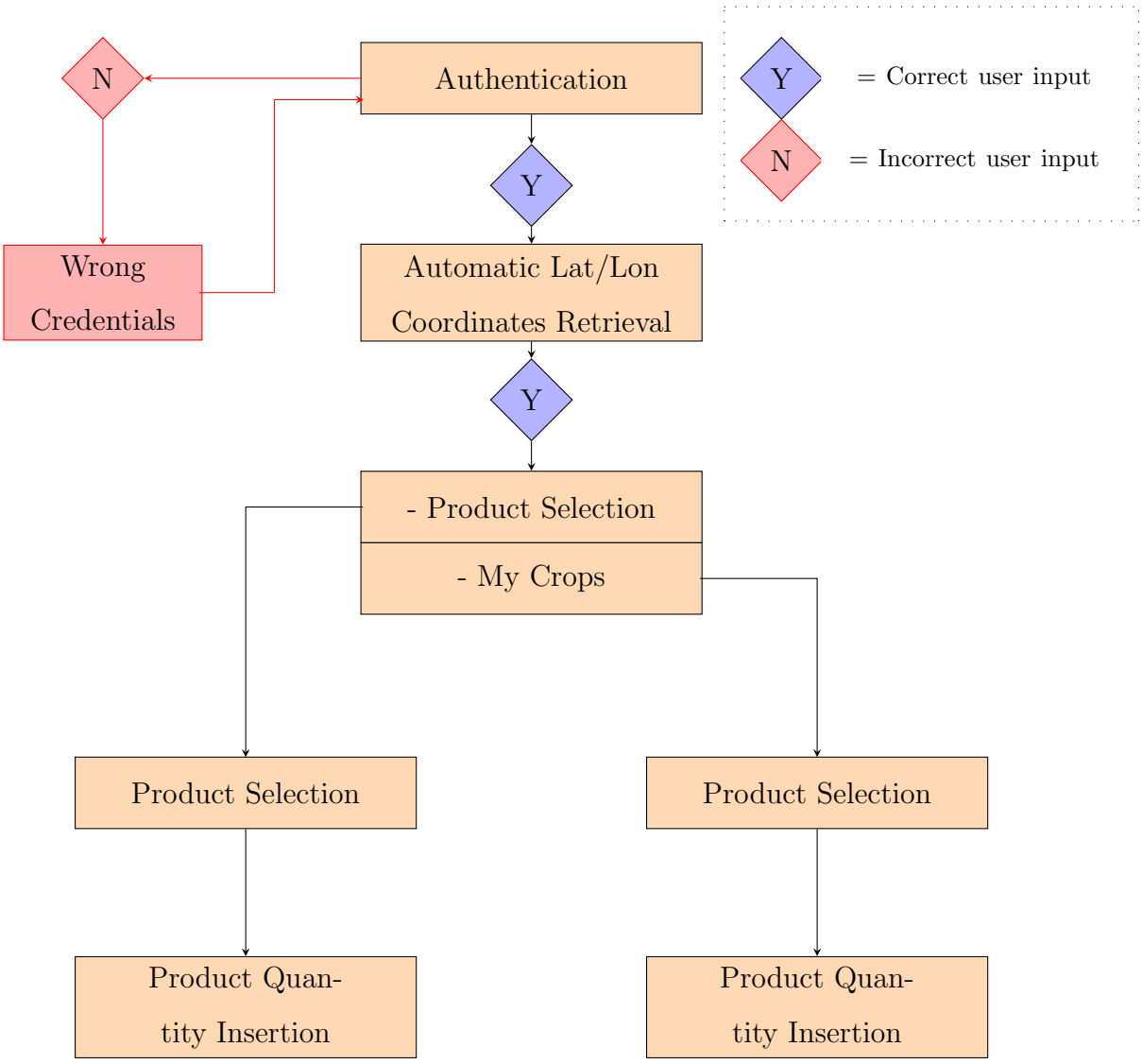

Figure 3.6: Information architecture for the mobile application

3.7 Designing the User Interface

Starting from the scenarios, claims and requirements we derived a list of UI requirements that we used to address the design of our application. We categorized them in two separate categories namely:

1. data presentation
2. data entry

The two categories cover the two aspects of a traditional UI. The former is related to the information output that has to be clear and fully comprehensible for users of different cultural levels while the latter is related to the input modalities; indeed, this stage can be really hard and annoying for users and can lead users to make mistakes as described in (Longoria, 2004). (Table 5.7, page 162) provides the rationale for the two categories.

To meet the *Data Entry* requirements, we decided to automate the first steps of the application workflow shown in Figure 3.6. For example, the field position is automatically retrieved by using the integrated GPS receiver and sent to the system. Figure 3.7 shows the crops catalog. Here the design meets the *Data Visualization* requirements. The catalog was divided into crop categories. Opening a category page triggers a verbal description of the category. We used icons to describe crops and a colored background (based on the universally understood traffic lights metaphor) to indicate the approximate quantity of each crop already in production. The color scale ranges between green = Zero production and red = Intensive production. A local language text label is added to each icon. By selecting the product users reach the last step of the workflow (Figure 3.8).

Opening the new page users receive a verbal description of the selected product. Users are provided with a more detailed description of the product.

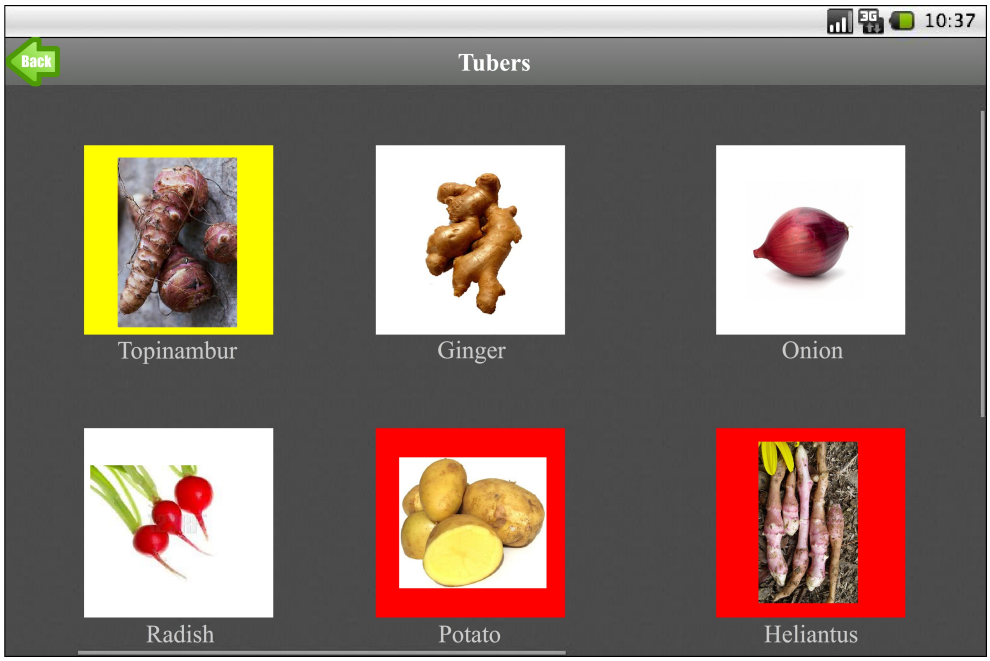

Figure 3.7: The crops catalog

By taking into account data entry requirements we provided check box items to allow users to insert information about the quantity of product that they want to cultivate without typing errors. Testing the initial mobile prototype with targeted user groups not only revealed the importance of designing the user interfaces according to the cognitive, education and cultural background of our community but also highlighted the need to focus on fundamental quality attributes such as effectiveness , efficiency and user satisfaction (De Silva et al., 2014).

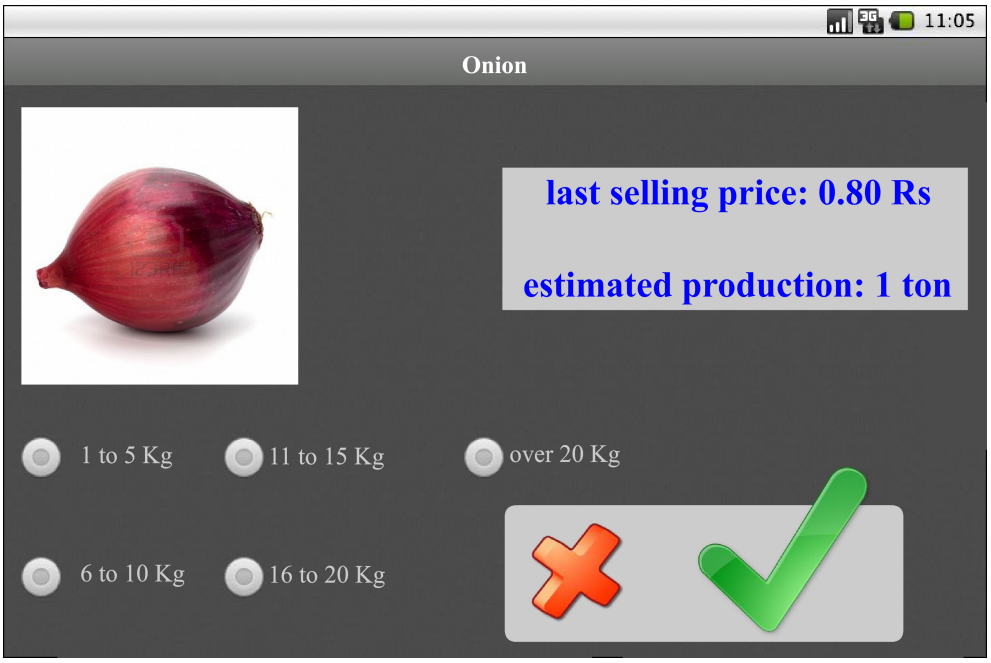

Figure 3.8: Product selection interface

3.8 Designing for Effectiveness

Effectiveness of a solution is the *accuracy and the completeness with which users achieve certain goals* (Frøkjær et al., 2000). Stakeholders of an ICT based system value accurate and complete information. This is one of the characteristics that would empower the user to use the system. Users will get motivated to share information when they experience the benefit that they would receive in return.

As for our case study, farmers need accurate and complete information at the time of making decisions in their farming life cycle. In order to support them, the system should be capable of providing the right information at the right context. We evaluate the effectiveness of the solution throughout the process at different stages by measuring the quality of the designed solution.

A thorough literature review enabled us to identify information needs of the farmers at different stages of the farming life cycle. Based on an extensive survey conducted by Lokanathan and Kapugama (Lokanathan and Kapugama, 2012) we further mapped these needs to different stages and identified the need of personalised information to increase accessibility. Then, we designed user registration system that captures some aspects of their context, such as the farm location, to provide personalized information to the farmers. For example, a farmer can use the system to query what crops will grow in his farm. To answer this query geo-coordinates based on the farm location captured at the time of registering can be used to identify the corresponding agro-ecological zone. Based on the agro-ecological zone, we can also obtain the related environmental factors, such as temperature, rainfall and soil type relating to the farm. By using this information we query a crop ontological knowledge base to find a list of crops that will grow in that particular farm.

To reach the above design level, we started with the crop selection stage of the farming life cycle to identify detailed information needs of the farmers. This is the stage where farmers make critical decisions in identifying what crop to grow in which quantity. The decision made at this stage will influence the revenue at the selling stage. Thus, providing accurate and complete information at this stage is a crucial need. Therefore we conducted several field visits and surveys to gather detailed requirements. Grounded on these findings, we further designed the mobile interfaces for the crop selection stage. These were iteratively tested with farmers to ensure that the provided information corresponds to their requests. During such evaluations farmers stressed the importance of getting accurate and complete information.

Feedback received during these surveys enabled us in designing the back-end databases, crop knowledge repository and Web services for the front end

mobile applications to retrieve and store information. The first working mobile artefact emerged as a result of these combined activities (Figure 3.9). This was developed for a mobile running on Android 4.0 targeting the crop selection stage. This mobile artefact was tested with a sample of 32 farmers in Sri Lanka. Working mobile prototype, questionnaire comprised of multiple choice, Likert scale and open-ended questions, and interviews were used as main research instruments in this evaluation of the effectiveness of the solution. We gathered statistics in relation to the completeness of the information provided. This is to ensure and to further identify detailed requirements needed by the users. In addition to that we also gathered data on the effectiveness of the features provided in the system in decision making. Further details on this evaluation can be found in Chapter 5.

3.9 Designing for Efficiency

Efficiency is another major characteristic for the success of the solution. Since the targeted population is new to this type of solution, achieving efficiency will empower the user to use the system. The efficiency of a solution is measured using *the resources such as time, money or mental effort that have to be expended to achieve the intended goals* (Bevan, 1995). Task completion time or effort will rely on the efficiency of user interfaces that should be designed for user to complete the required task with minimum effort. Otherwise the users will get less motivated. Thus, a user-friendly interface with an easy navigation scheme is necessary to increase the speed in getting and sharing information.

Having identified this need, we designed user-friendly interfaces while iteratively testing the interfaces with farmers. We employed the paper prototype

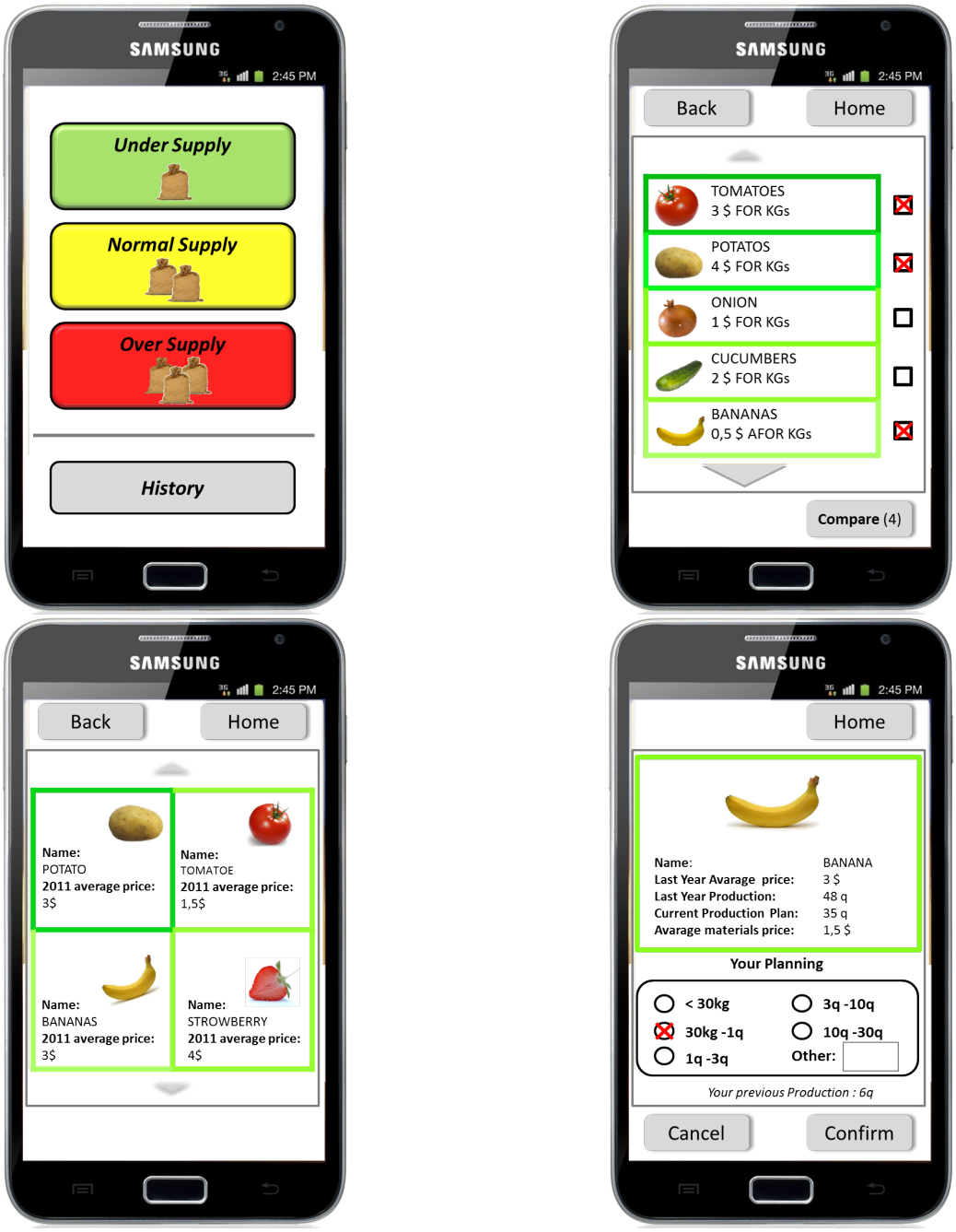

Figure 3.9: The crop selection activity in the first working prototype

technique in HCI to get a quick feedback on the design. Further, we have applied several methods, such as knowledge injection and UI design based on HCI techniques, to enhance the look and feel of the interfaces. This iterative process enabled us to design a better set of UIs to provide the requirements of the user. Our main intension of this iterative process is to produce a set of UIs that will minimize the effort of using the system. Thus, we evaluated our first working mobile prototype to explore the extent to which we achieved efficiency. We used indicators, such as time to complete a task and the required effort to achieve the intended goals in this evaluation.

This evaluation was carried out using a sample of 32 farmers. We gave them three tasks to attend after a training session. More information on these tasks and the evaluation findings can be found in Chapter 5.

3.10 Designing for User Satisfaction

In addition to achieving effectiveness and efficiency, the user should also be satisfied with the system. This parameter has a huge impact on the success of the solution, thus, it is essential to measure it to identify user comfort and attitudes towards the system (Frøkjær et al., 2000). User attitudes can be measured by using standardized methods, such as SUMI (Kirakowski and Corbett, 1993), whose five subscales are efficiency, affect, helpfulness, control and learnability. For the initial evaluations we used affect, helpfulness and learnability to measure user attitudes towards the system.

Mobile prototype including paper and the mobile versions enabled us to iteratively evaluate the solution for user satisfaction. During such evaluations affect or the likeliness to use the system was recorded to be 100%. Their attitude towards learnability is positive. However, some of the users stressed

the need of a training session to better identify the features to reap the benefits. We also observed that these user groups need more guidance to use the application by providing various help facilities.

3.11 Discussion

In this section, some reflections on advantages of the overall development methodology evolved from this research project are described. The characteristic of usability effectiveness, efficiency and satisfaction were achieved through usability requirements which evolved from the beginning of the research as mentioned above. The methodology we propose, by blending several techniques within a DSR approach, enabled us to derive detailed requirement needs of the user. As shown in Figure 3.10, the research team has successfully iterated through four DSR cycles to find a solution to the problem associated with vegetable over-production in Sri Lanka. This problem required an innovative ICT based solution. The characteristics, such as ad-hoc nature, lack of user exposure to ICT and user unawareness to system requirements, made the identification of the solution a much harder task. However, within the first iteration of DSR we succeeded in identifying the user requirement goals. In addition to the requirements we further identified the need of user empowerment. This need initiated several DSR cycles in which we achieved effectiveness, efficiency and user satisfaction. Thus, through a series of iterative cycles in DSR methodology we were able to derive both the functional and usability requirements for a problem where no prior ICT based solution nor clear set of processes that can be enhanced by ICT was defined. This was possible due to the blended techniques of SE and HCI used within the DSR cycles.

In addition to traditional surveys and interviews, we iteratively generated the user requirements through the use of causal maps and scenario transformation methods. Further, grounded on initial requirements, we designed the first mobile prototype to gather deeper requirements. Both paper-based and functional prototypes were used during the surveys to acquire feedback from the end users of the system. Figure 3.11 and Figure 3.12 are two instances where we demonstrated the system to farmers by using the paper-based and the functional prototype of the proposed solution, respectively. Paper-based prototype was designed to gain more grounded user requirements and feedback for the proposed solution. This enabled us to rapidly incorporate requirements to the design. We traversed through design, development, demonstration and evaluation phases in DSR methodology to refine the prototype and design the real working mobile prototype. This gave a real look and feel to the solution so that users found easy in expressing their requirements. As such, we incorporated incremental development techniques described in SE (Sommerville, 2011), throughout the DSR cycles to speed up the design and development process of the mobile artifact.

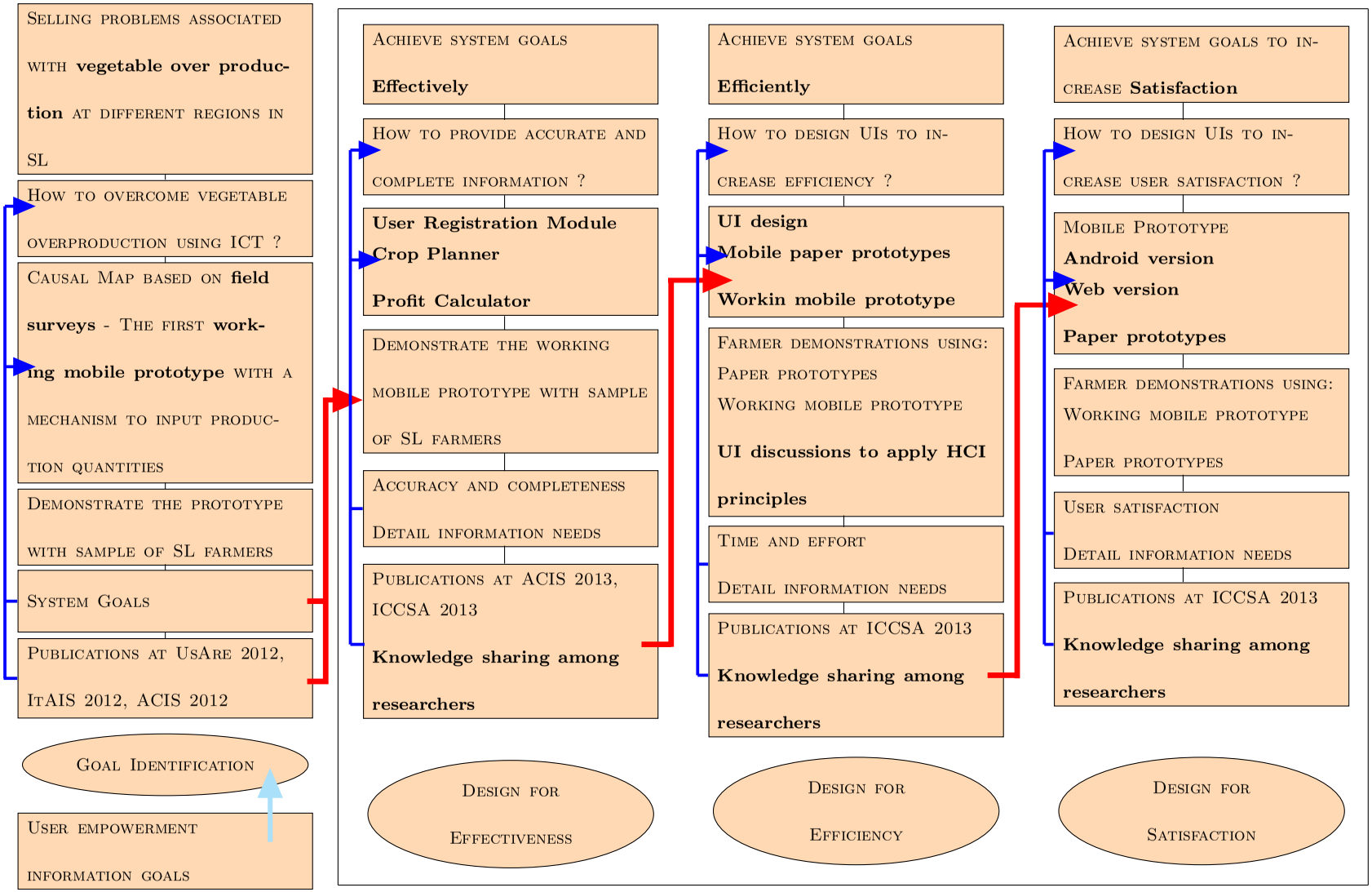

Figure 3.10: DSR process to capture user requirements and design for usability and accessibility

Figure 3.11: Farmers evaluating the paper prototypes

Figure 3.12: Farmers evaluating the working mobile prototype

Further, this approach enabled us to rapidly design each aspect of usability. As shown in Figure 3.10, we iteratively concentrated on how the design goals could be achieved efficiently, effectively and satisfactorily.

Visual interfaces in this regard played an important role in enabling us to achieve such goals. They were constantly used during our field visits except for the initial investigations of the problem identification stage. We observed that the users participation enhanced and the users trend to provide us with

useful requirements by using these instruments. In each DSR cycle demonstrations and evaluations were carried out with the real users of the system, because the focal point of our work was the user himself. Thus, our research facilitated the User Centered Design (UCD) (Rubin and Chisnell, 2008) and throughout the methodology we gave priority for the experience of the users. As stated in ISO standard 13407 (ISO/IEC 13407, 1999), UCD is *"characterized by (1) the active involvement of users and a clear understanding of user and task requirements; (2) an appropriate allocation of function between users and technology; (3) the iteration of design solutions; (4) a multidisciplinary design.* Then, the DSR framework in this regard was a better framework to facilitate such an active user participation to identify the user and their requirements. It is also obvious that having constant interaction with users can make them aware of the solution to a great extent.

Table 3.1: Personas involved in the scenarios as stakeholders

Personas
1. Sirisena is a 45-year-old farmer with long experience in truck farming. Sirisena is part of Sinhalese ethnic group. He has a basic education level; he attended the primary school, he can read and write Sinhalese and he has a basic knowledge of English. Sirisena does not have advanced technical skills; the only technological instrument is his mobile phone that he uses everyday. Moreover he is pretty distrustful of the technological support and, during his work, is accustomed to rely on his farmer experience. Sirisena lives in Sigiriya, a village in the central Matale District of Sri Lanka, where he owns four acres of farmland. Since the property is quite large, eight collaborators support Sirisena in his work. Since Sirisena has a long experience in truck farming he manages the crop production of his family farm. His role is to make decisions on critical aspects of the production. He takes decisions on the kind of production and the time to start it. Moreover he establishes an indicative selling price.
2. Premasiri is a 40-year-old low price fertilizers seller. Like Sirisena, Premasiri is part of the Sinhalese ethnic group and lives near Sigiriya. In order to raise his revenues he also acts as market middleman. Since he can speak English as good as Sinhalese and has a basic knowledge of Tamil, his intermediary role is well recognized by the farmers of the area. During the market activities his responsibility is to negotiate the best selling price of the product trying to match the expectations of his clients.

Table 3.2: Scenarios of current practices

Scenarios
1. *Sirisena manages the crop production of the farm* Sirisena is planning the new crop production. The decision will be made on the basis of three factors. He takes into account: (a) the period of the year, (b) the crop producing high yield within a short time, and (c) the crop selling prices of the last year. Since the period of the year is suitable for potatoes cultivation and it gives the highest yield in a short time, Sirisena decides to produce mainly potatoes. Anyway, Sirisena makes his decision without interacting with his neighbors because he does not trust them. Three months later the crop is ready to be harvested. He establishes an indicative price of fifty rupees for one kilogram of potatoes, on the basis of the last year selling price and the expenses incurred during the cultivation period. Sirisena does not have means to take the harvest to the market and moreover he could not well communicate with potential tamil buyers because of his language limitations. He decides to call Premasiri asking him to mediate during the market activities.
2. *Premasiri acts as middle man to get the harvest sold* Premasiri agrees to sell Sirisenas harvest to the local market. Before starting the market activities, all the farmers decide to raise or reduce the estimated harvest prices considering the presence of competitors. Premasiri notices that many farmers have cultivated large amount of potatoes. He is forced by the local market-law to reduce the estimated price cutting his profit. Moreover he notices that just a few farmers are selling onions so that the onions prices are noticeably higher than the last year prices.

Table 3.3: Claims from the scenarios of current practices

Claims	
Situation Features	**Pros(+) and Cons(-)**
The farmer selects the crop cultivation on the basis of: 1. The period of the year, 2. The crop producing high yield within a short time, 3. The crop selling prices of the last year.	+ The process knowledge is transmitted from father to son as cultural heritage. + Cultural level of Sri Lankan farmers is enough to perform basic computations. - The choice is based on a few factors without considering the neighboring crops that are paramount in the market business. - The way farmers make decisions provides clients with a little variety of products. - Over supply may result from this strategy.
The selling price is fixed when the middle man gets the local market and estimates it on the basis of competitors products.	+ The production is oriented to the local market economy. - The selling prices of crops change vigorously in few hours.

Table 3.4: Functional, Environmental and Data requirements

Functional requirements	
The application allows the exchange of heterogeneous data between neighbouring users guaranteeing anonymity.	By sharing valuable information on crop cultivations, users would be able to make better decisions during the crop selection activities. Furthermore, because of social phobia and the competition among farmers, users are more conformable by preserving anonymity.
Environmental and contextual requirements	
The application could be used in a noisy environment.	Users might be using noisy rural instruments or they might be speaking with co-workers while using the application in the field.
The environment could be sunny, rainy and powdery.	The application is used manly in open environments in particular in agricultural areas that can be really powdery.
Users may need training provided by experts.	Users are not in habit to work with advanced technological instrumentation. Therefore, they may need training.
Data requirements	
The application has the access to data related to the distribution of crops located around users farm.	Farmers are interested to get information just about neighbouring crops that are supposed to be sold to the same market.
Data must be accurate and updated frequently.	Users make crucial decisions on the basis of provided data.

Table 3.5: User profiles and Usability requirements

User profiles	
Most users will be in the range of 20 to 45 years.	All of the information is carried out from the initial survey conducted directly in situ.
The range of the instruction level varies between Ordinary Level and Master Degree.	
A large number of people can speak English well enough.	
Most users are not familiar with technology, with the exception of mobile phones that are quite widespread.	
Users are disposed to use some technological instruments on condition that they are not invasive.	
Usability requirements	
The application should be easy to use and should require a little training effort.	The application is used manly in specific and not frequent tasks.
The UI should be effective: it should provide a simple management of users mistakes.	The application provides support to a critical task on the basis of data updated directly by users. Therefore, it is paramount to reduce the number of possible unintentional user mistakes.

Table 3.6: Scenario transformation activity

Scenarios
1. Even Sirisena is distrustful of technology he is persuaded by a Nenasala officer to experiment a new mobile application aimed to assist selection of crops. Sirisena has been persuaded because of his familiarity with mobile devices and because he has been reassured about the non-intrusiveness of the software. Before starting to use the new application, Sirisena attends a one-day training course in the Nenasala center. Now Sirisena is ready to start planning the new crop production assisted by the new system. He gets to the farm, launches the crop assisting application and sends coordinates of the farm location. The application then informs him about the neighboring crop productions. The decision will be made on the basis of few factors. 1. the period of the year, 2. the crop producing high yield within a short time, 3. last years market prices, and 4. types of crops that grow in the area. Despite potatoes are suitable for the current period of the year and they give the highest yield in a short time, Sirisena notices that their production is already high in the area. In contrast onions production is pretty low and the last years selling price did not live up to expectations. This is because of the overproduction at the time. Therefore Sirisena decides to produce onions hoping the selling price can live up to his expectations. Based on the first three factors, potatoes seem to be the best choice, but the forth factor leads Sirisena towards a different direction. Four months later the crop is ready to be harvested, he establishes an indicative price of fifty rupees for one kilogram of onions, on the basis of expected selling price of the last year and the expenses incurred during his work. Sirisena contacts Premasiri asking him to get the harvest sold. 2. Premasiri gets to the market and checks the variety of the present products. He notices that vendors provide a large variety of vegetables. The variety of the market allows him to save the selling price estimated by Sirisena.

Table 3.7: User Interface Requirements

UI Functionality	**Rationale**
Data Visualization	
List of products, each one associated with a significant icon.	At any cultural level the user is able to quickly identify the visualized product.
A visual colored cue associated to each product to indicate the estimated harvest quantity.	The visual cue is needed to inform a participant of the quantity of a given product of the list. It provides an intuitive way to understand the magnitude of the quantity.
The UI provides users with language support, both in text and audio forms.	The UI can exploit the multimodality in order to guarantee the right comprehension of the information in each situation. For example when the user is semi-literate or when he/she is working in a sunny or noisy environment.
Data Entry	
The UI avoids requiring text input. Whenever it is possible multiple radio buttons are used.	Text input is a common annoying source of mistakes. Radio buttons are easier to interact with.
The UI limits the number of interactions, hiding operations that can be automatized.	Some operations require unnecessarily user interactions exploiting the advanced device features.

Chapter 4

Improving Metadata Exchange in the Composition of Heterogeneous services

As previously mentioned, a service is an autonomous software module that performs a well-defined set of operations. Nowadays, service-based solutions represent the backbone of a growing number of information systems. Platform independence and the ability to compose different services, possibly developed by different organizations, to provide complex functionalities constitute the main reasons behind the success of this computational paradigm. However, as discussed in Chapter two, one of the crucial requirements to successfully compose two or more services is that all the involved entities must adhere to the same set of standards. When services developed according to different sets of standards are involved in the composition process, several interoperability issues may arise that need to be overcome. This is the case when trying to compose services developed according to the recommendations of the World Wide Web Consortium and geospatial services proposed

by the Open Geospatial Consortium. In light of the important differences between the two types of services, one of the few available options to make W3C and OGC services communicate seamlessly consists of mapping one set of standards onto the other at both the syntactic and semantic levels.

The research project we are carrying out with Sri Lankan farmers represents a concrete example where, effective integration between OGC and W3C services is needed. The blueprint of the back end architecture for the purpose of SLN4MoP project has been organized by exploiting the principles of the SOC paradigm, to easily satisfy several fundamental design goals such as the ability to add new features without affecting the existing components, the independence of the system functionality from the specific format of the various data sources and the possibility to seamlessly add new data sources or replace existing ones without modifying the behavior of existing implementations. While the majority of our platform was developed in compliance with the standards proposed by the W3C, for the purposes of the project, geospatial data are of utmost importance. For example, suggestions for a specific user on the best crops to grow have to take into account not only the current market trends but also specific characteristics of the soil and of the area where his/her farm is located. Moreover, since the geospatial information could be useful also for third party entities (e.g., to visualize the various soil types available in a specific region using a traditional desktop GIS application such as uDIG) the services that deal with geospatial information were developed following the standards proposed by the OGC.

In order to guarantee the highest level of interoperability among the various software modules, following the solutions currently discussed in literature (Ioup et al., 2008), we developed a service wrapper to provide a syntactic translation from OGC to W3C, which exploits existing orchestration mid-

dleware and the well-established services orchestration in W3C environments (Bertolotto et al., 2014a,b). In particular, the proposed wrapper translates SOAP-based messages into OGC-compliant requests and vice-versa. A critical aspect during the development of such a software module relates to the proper management of geospatial and OGC metadata in a W3C-compliant architecture. Given the critical role played by metadata for the actual exploitation of both geospatial information and OGC services, this is a crucial aspect for viability of the proposed solution. Indeed the design philosophies for metadata exploitation and exchange in OGC and W3C environments are very different. For OGC services, exposing their metadata represents a mandatory task and every service must provide, through its public interface, a well-standardized operation for metadata retrieval. W3C services, instead, can use several options to expose their metadata. Unfortunately, none of the existing proposals has been designed to directly manage non-W3C services metadata. Therefore, in order to offer a seamlessly interoperability, it is necessary to expose OGC metadata using current W3C standards. However, to be effectively deployable in a real-world infrastructure, an implementation of such an option should not modify the behavior and semantics of such standards and should be totally transparent for those services that do not deal with geospatial information. To the best of our knowledge, a suitable solution to overcome this issue has not been implemented and validated yet. In this chapter we discuss our approach to seamlessly exchange OGC metadata exploiting a W3C-compliant standard.

This chapter is structured as follows. We firstly provide an overview of metadata management in the context of the SOC paradigm, analyze the various available options for their actual retrieval and focus on the key differences between the standard ways to directly retrieve them from the intended

services in W3C and OGC environments respectively. Subsequently we illustrate our proposal to provide a W3C standard that deals with the direct metadata retrieval with the support for OGC metadata and, finally, we discuss its technical feasibility by showing how one of the most used software implementations of the W3C services stack can be modified to seamlessly support the new metadata types.

4.1 The Primary role of Metadata in the SOC paradigm

Generally speaking, metadata are *data about data*. With the growing complexity of software systems, metadata represent, nowadays, an essential component not only for the development and management of these infrastructures but also for the effective exploitation of the information available when using an information system.

Metadata have gained a primary role also in every key-aspect of the SOC paradigm where they provide fundamental support in the whole life-cycle management of a complex system. Indeed, metadata constitute a founding element of every modern SOA due to their extensive use in almost every facet, from the configuration of lower-level components and description of all the non-functional aspects of a service, to the provision of information for the global SOA governance. WSDL documents along with XML Schemas specifying the data type of the exchanged messages or the list of features that a WFS can serve represent classic examples of metadata in both traditional W3C and OGC infrastructures. Among the core functionalities that extensively make use of metadata, the initial retrieval of all the information (such as the list of capabilities, access rules, additional policies, etc.) that

allow a generic client to properly interact with a service and begin the actual messages exchange is surely one of the most important.

However, to properly fulfill this task it is necessary to overcome two fundamental issues, namely how to actually provide such information to the clients and the need for all the involved entities to interpret it in the same way. As for the first issue, several possible solutions are available (Erl, 2005). A client might, for example, look for such metadata searching through the published service documentation but this option does not represent a standardized, globally accepted or easy to implement solution. Another way for a provider to publish the functionality of its services and for clients to look for software components that match their needs is the use of a public registry. In W3C environments, the attempt to provide a standardized directory for service discovery resulted in the definition of the Universal Description, Discovery and Integration (UDDI) specification, a platform-independent framework for the publishing and discovery of information about services (Clement, 2005). The registration information stored into an UDDI registry can be sub-divided into four main data structures namely business entities, business services, binding templates and tModels (Tsalgatidou and Pilioura, 2002). The binding template represents the technical description of a service. It contains the service URL and one or more references to the tModels that *"is the mechanism used to exchange metadata about a Web service, such as the Web service description or a pointer to a WSDL file"* (Newcomer, 2002). In addition, since each UDDI registry can be seen as a SOAP-based service (Tsalgatidou and Pilioura, 2002) each of them offers a set of SOAP-based API for the registration and discovery of services. Although UDDI was expressly designed to be the standard way to publish and discover W3C services information, in the context of making *"geospatial content and services more universally discoverable*

and consumable by non-GIS users" (Lieberman et al., 2003), the OGC investigated the possibility of discovering the capabilities of its geospatial services through the UDDI interface using SOAP messages. The experimental results of this attempt of using UDDI can be found in(Lieberman et al., 2003). Unfortunately, the study shows that UDDI *"is less well suited for obtaining the information to bind to a service, and even less well suited to discovering specific contents or capabilities of individual service instances"* (Lieberman et al., 2003). The intrinsic complexity of UDDI restricted its global adoption also among traditional W3C providers. Other important limitations include the fact that when using UDDI a client cannot *"query a service by its interface signature"* (Fang et al., 2006) and its lack of support for metadata annotation and metadata-based service discovery (Fang et al., 2006).

A feasible and more flexible alternative to the use of public registries is the acquisition of metadata directly from the intended services. Instead of retrieving metadata from a registry, a client can directly interact with a service provider to get the desired information related to the offered services. The only essential thing that a requester needs to know *a priori* is the location where a provider offers the metadata about its services. However, in order to be an effective alternative for metadata retrieval, similarly to what happens with the definition of the public interface and the messaging system, every involved entity has to agree on a common protocol. This greatly simplifies the task preventing clients from interacting, every time, with proprietary retrieval systems offered by services providers (Erl, 2005). Flexibility with the typology of metadata that can be retrieved constitutes an additional desirable requirement. In a W3C-based environment, the standard way to directly retrieve metadata about a service is by using the Web Services Metadata Exchange (WS-Metadata) specification (Davis et al., 2011). Although WS-

Metadata has not been expressly designed to deal with geospatial metadata in general and OGC metadata in particular, its underlying design choices make it a scalable solution, adaptable to arbitrary forms of metadata.

In the following subsection, we provide an overview of the key-points of the WS-Metadata specification and analyze the general structure of a generic OGC Capabilities document.

4.1.1 The Web Services Metadata Exchange specification and the GetCapabilities operation

The Web Service Metadata Exchange specification provides a standardized, SOAP-based, way for the encapsulation, insertion, retrieval and removal of metadata associated with a W3C service. The specification defines, in addition, also a bootstrap mechanism to get started with the actual metadata retrieval, the ability to support future versions of current metadata and the possibility to add further metadata formats. In its simplest form, a typical WS-Metadata message exchange pattern consists of a SOAP request that a requester sends to a service provider and a SOAP response sent back by the provider. The only thing a client needs to know *a priori* to initialize such a communication is the Service Endpoint, namely a location where the requester can send a SOAP message containing the request for the desired metadata. The discovery of an Endpoint address and the application of security policies to the bootstrapping phase are outside the scope of our discussion. The effective metadata encapsulation is achieved by the use of the <Metadata> Element whose outline is shown in Figure 4.1.

In the remainder of the text, to improve the overall readability, when there is no ambiguity, we will omit the *mex* prefix. It is worth noting that as stated in the standard, the choice of the prefix is arbitrary and not semantically relevant. The <Metadata> Element can be seen as a container for one or more metadata units. Each metadata unit is represented by a <MetadataSection> and can be embedded into the <MetadataSection> Element or referenced using the <MetadataReference> or <MetadataLocation> Elements. The Dialect and Identifier attributes of a <MetadataSection> are mandatory. The former specifies the type and version of the metadata embedded into the XML Element while the latter is an absolute Internationalized Resource Identifier (IRI) that identifies the specific metadata. Typical values for common types of dialects are, for example, *"http://www.w3.org/2001/XMLSchema"* for the XML Schema metadata or *"http://schemas.xmlsoap.org/wsdl"* for WSDL 1.1 documents. The interpretation of the Identifier attribute is Dialect-specific (Davis et al., 2011). As for the actual metadata retrieval from a Service Endopoint, the current version of the WS-Metadata protocol defines two mechanisms: the GetWSDL operation, suitable to directly retrieve the WSDL document of a service, and the GetMetadata operation useful when the requester *"whishes to obtain a specific metadata document"* (Davis et al., 2011). Figure 4.2 shows the typical structure of a GetMetadata request.

```
<mex:Metadata ...>
   <mex:MetadataSection
      Dialect='xs:QName'
      Identifier='xs:anyURI' ...>
      (
      <mex:MetadataReference ...>
      endpoint-reference-type
      </mex:MetadataReference>
      |
      <mex:MetadataLocation ...>
      xs:anyURI
      </mex:MetadataLocation>
      |
      DialectSpecificElement
      )
   </mex:MetadataSection> *
      xs:any*
</mex:Metadata>
```

Figure 4.1: The general structure of the <Metadata> Element

```
<mex:GetMetadata Content='xs:any' ? ...>

   <mex:Dialect
      Type='mex:QNameSerialization'
      Identifier='xs:anyURI' ?
      Content='xs:anyURI' ? .../> *

      xs:any*

</mex:GetMetadata>
```

Figure 4.2: The <GetMetadata> Element

The optional Content attribute specifies the IRI for the request. When absent the default value is *"http://www.w3.org/2011/03/ws-mex/Content/Any"*. The <Dialect> Element instead, although optional, deserves some additional considerations. According to the standard, when this Element is absent, all the available metadata must be returned. However, when it is included in a <GetMetadata> request, the response must return only those metadata that match the values specified by the combination of the mandatory Type and the optional Identifier and Content attributes. If there are no available metadata for the specified combination the response must not return any metadata for that Dialect element. Finally, if an Endpoint can accept a GetMetadata request, it must send a GetMetadataResponse message whose skeleton is shown in Figure 4.3.

```
<mex:GetMetadataResponse ...>

   <mex:Metadata ...>
      ...
   </mex:Metadata>

   xs:any*

</mex:GetMetadataResponse>
```

Figure 4.3: The <GetMetadataResponse> Element

As shown in the picture, for a GetMetadataResponse, one <Metadata> Element must be contained in the Body of the SOAP response message.

Unlike what happens with W3C services, the design choices that characterize the three main OGC services (WMS, WFS and WCS) impose the exposure of metadata describing their capabilities. The only way for an OGC client to retrieve such metadata is through the invocation of the GetCapabilities operation whose aim is to allow *"any client to retrieve metadata about the capabilities provided by any server that implement an OWS* [1] *interface implementation specification"* (Whiteside and Greenwood, 2010). The typical response to a GetCapabilities request is an XML file, the Capabilities document, that contains metadata about the specific abilities of the invoked service. The structure of a Capabilities document is rigorously defined by the

[1] OGC Web Service

Consortium (although particular implementations *"can provide additional operations returning service metadata"* (Whiteside and Greenwood, 2010) and should be ideally divided into two main parts, namely the aspects that are common to all the OGC services and the sections that provide metadata necessary for the specific functionalities of each single service type. For example, for a WFS the GetCapabilties operation *"must indicate which feature types it can service and what operations are supported on each feature type"* (Vretanos, 2005). The Capabilities document's parts that should be common to all types of OGC services can be grouped into five main sections:

- Service identification: Provides metadata about the specific service. The standard prescribes that the general structure of this section should be the same for every OGC service. Typical attributes are, for example, the service type (e.g., WFS, WMS) or its version e.g., 1.0.0, 1.1.0.

- Service provider: Provides metadata about the organization that manages the intended service.

- Operation Metadata: Includes the list of operations supported by the specific service (e.g., getFeature for a WFS) along with the URLs for their invocation.

- Contents: The section specifies the actual data served by the specific service. The content of the section is specific to each OGC service.

- Languages: This section contains the list of languages supported by the OGC service.

The GetCapabilities operation supports also several input parameters (such as the service version with which a client expects to interact, the list of languages desired for any human-related types of information, etc.) that

can be used to better clarify the information needed avoiding retrieving the whole Capabilities document. Finally, for the purposes of our discussion, it is also worth noting that the Consortium envisages that an XML encoded GetCapabilities request may be embedded in a SOAP 1.2 message.

4.2 Extending the Web Services Metadata Exchange specification

As discussed above, each OGC service offers by default the ability to retrieve all the metadata useful for its profitable exploitation, while W3C services rely on additional protocols for the fulfillment of this task. However, in the context of integrating OGC services in a W3C environment, forcing a SOAP-based client to directly query the OGC service and obtain the desired metadata is simply unfeasible for several reasons. In particular, this would cause an unnecessary additional complexity for the client since it would require to query non SOAP-based entities and be able to process non-SOAP messages. Another and probably most important reason concerns the fact that, as this direct retrieval occurs outside the W3C service stack, such option is incompatible with the current SOAP-based standards (such as the Web Service Security (Nadalin et al., 2004) that guarantee the security of information. Therefore, providing geospatial metadata in a W3C-compliant way is a major challenge in order to support a better interoperability between the two proposals.

As discussed in Chapter two, the GetCapabilities operation of a generic OGC service could be simply exposed using the <Operation> Element of a WSDL document. However, from a semantic point of view, this option does not represent the right choice. In an OGC environment, the metadata

retrieval is usually a two-step process: invocation of the GetCapabilities operation and parsing of the returned Capabilities document. Embedding the GetCapabilities operation into a WSDL document forces a client to retrieve the WSDL document, parse it, and invoke the *new* GetCapabilities operation. Even when the client is not interested in the metadata offered by the original OGC service, a three-step process (i.e., additional overhead) is performed.

If the WS-Metadata specification directly supported the exchange of geospatial metadata, it would mean that a generic client could retrieve them during, for example, the bootstrapping phase along with the information about the WSDL document. In fact, the wrapper could be configured as an Endpoint and, on receiving a metadata request, transform the Capabilities document into a WS-Metadata compliant message and send it back to the requester. The actual Capabilities document could be retrieved by the wrapper either by directly querying the source OGC service or from a local cache. Unfortunately, as we have seen, the specification has not been designed to directly support the exchange of non-W3C metadata such as those related to OGC services. However, the WS-Metadata underlying design choices not only support future versions of known metadata formats but, more importantly, allow the addition of new formats. In this context, providing the WS-Metadata specification with the native support for OGC metadata essentially impacts on three main aspects:

- The client has to be able to send to an Endpoint a GetMetadata message that explicitly refers to geospatial content
- The GetMetadata request has to discriminate among the various types of OGC services (e.g., whether the request concerns the metadata for a WFS or a WMS)

- The Endpoint has to support the request type and, in case it is unable to support the new metadata types, it has to simply reply with a fault message (as stated in the WS-Metadata specification).

In light of these considerations, in order to provide the possibility to exchange OGC metadata using the WS-Metadata specification, our proposed approach consists of extending the protocol by adding a new set of combinations of values for the *Type* and *Identifier* attributes of the <Dialect> Element. Two important constraints concern the format of the attributes that must be compliant with the WS-Metadata specification requirements and the fact that each combination must uniquely refer to a specific OGC service. As for the *Type* attribute, it must be specified by a QName (Qualified name, a valid identifier for elements and attributes). Since the standard specifies that a QName must be serialized as {*namespace-uri*} *localName*, we chose the form *"{OGC service XML Namespace URI} Capabilities"* where the string between brackets represents the unique URI used in the namespace declaration of each type of OGC service.

Finally, for the *Identifier* attribute, instead, we chose the actual URL of the XML Schema that, for each type and version of OGC service, defines the structure of every admissible Element and its attributes. As for the third <Dialect> attribute i.e., the *Content*, we chose to use the default value defined in the specification, namely *http://www.w3.org/2011/03/ws-mex/Content/Any*. The combination of values for the current versions of the three main OGC standards is shown in Table 4.1.

However, in order to be effectively exploitable in a real context, such protocol extensions require the support of the underlying service infrastructure that must properly manage the new types of requests and responses. In the next section we show, as a concrete example, how the proposed changes to

Table 4.1: New values for the Type and Identifier attributes to support the three main OGC standards

OGC Service Type	Type attribute value	Identifier attribute value	Content attribute value
WFS	{http://www.opengis.net/wfs}Capabilities	http://schemas.opengis.net/wfs/1.1.0/wfs.xsd	http://www.w3.org/2011/03/ws-mex/Content/Any'
WMS	{http://www.opengis.net/wms}Capabilities	http://schemas.opengis.net/wms/1.3.0/capabilities_1_3_0.xsd	http://www.w3.org/2011/03/ws-mex/Content/Any'
WCS	{http://www.opengis.net/wcs}Capabilities	http://schemas.opengis.net/wcs/1.0.0/getCoverage.xsd	http://www.w3.org/2011/03/ws-mex/Content/Any

the WS-Metadata protocol impact on the official Web services stack of the Java Enterprise Edition (J2EE, for short) platform.

4.3 Extending the Java EE Web services Stack to Support the Direct retrieval of OGC Metadata

The Java API for XML Web Services (JAX-WS) specification (Kotamraju, 2010) represents, in the J2EE platform, the default approach to develop service-based solutions established on the SOAP and WSDL standards. The main design goal of this API was to hide (both on the server and client sides) the need of directly dealing with a WSDL document or SOAP message from the developers. From a server side point of view, the operations exposed into a WSDL document are mapped into traditional methods of Java classes. On the client side a proxy, namely an object representing the intended service, is created and a client simply invokes its methods. The JAX-WS runtime will then convert requests and responses into the corresponding SOAP messages. Although the JAX-WS programming model can be implemented by any software vendor, during our investigation we focused on its reference implementation namely the JAX-WS RI (JAX-WS RI Project, 2013). The JAX-WS RI constitutes, in turn, the core layer of Metro, an open source project representing the official Web service stack of the J2EE platform (Figure 4.4).

The other fundamental layer of the Metro stack is represented by the Web Services Interoperability Technologies (WSIT) subsystem. WSIT is built on top of JAX-WS RI and provides the concrete implementation of several additional Web service specifications dealing with enterprise-level features such as reliability, security or transactions. In addition, in order to foster interoperability among service-based solutions developed with different software technologies (e.g., the Microsoft's .NET framework), WSIT provides a bootstrapping mechanism that supports the actual retrieval of the service's WSDL

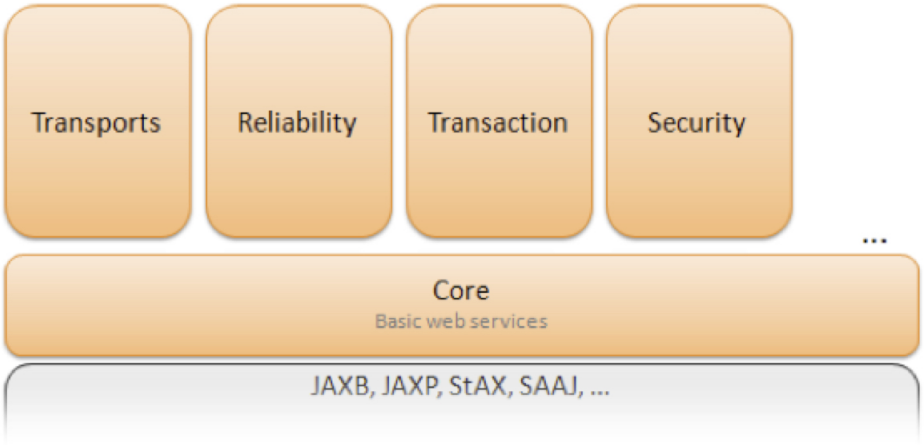

Figure 4.4: The Building blocks of the Metro project. Source: (`https://metro.java.net/`)

document and metadata. Such a functionality extensively relies on the WS-Metadata protocol. The open source nature of the Metro stack has not only simplified the analysis and design of the changes required to support the management of geospatial metadata but has been also of fundamental importance for their development and testing. However, to better contextualize the scope of our changes, in the next subsection we first provide a high-level overview of the global structure of WS-Metadata module available in WSIT and then briefly describe the role played by the most important Java classes that actually support the metadata exchange process.

4.3.1 An Overview of the WSIT Metadata Exchange module

From an implementation point of view, the entire Metro project is made up of several Java packages that handle the core W3C standards as well as the

WS-* additional specifications. Its latest public release is the 2.3 version. In particular, the actual implementation of WS-Metadata protocol is organized, in four packages (*client, server, mex and client.schema*) that group all the source classes according to their role in the whole metadata exchange process.

The first two packages, *client* and *server*, contain the classes that actually let Java-based Web services clients exchange metadata with third party WS-Metadata enabled solutions. The mex package contains only the MetadataConstants class whose aim is to store several useful fixed strings such as the supported versions of the SOAP or WSDL specifications or the prefix of the mandatory XML namespaces. Finally, the *client.schema* package encloses the Java classes that map the fundamental components of the WS-Metadata specification, namely the <Metadata>, <Metadata Section>, <Metadata Reference> and <Get Metadata> Elements.

The core component of the *client* package is the MetadataClient class whose main aim is to provide developers with a convenient way to handle XML-based metadata elements as Java object as well as to obtain additional service information such as its port QNames. The most important part of this class is the *retrieveMetadata()* method that performs the two-step process of retrieving a metadata set from a service Endpoint and convert it into Java instances. The first step, the retrieval phase, is implemented as follows. The method initially attempts to make a request using the SOAP 1.2 protocol. If such a version is not supported by the service Endpoint, it retries using the version 1.1. In case both the attempts fail, it tries to retrieve metadata by adding the *mex* suffix to the Internet address of the service. For actual metadata retrieval, *retrieveMetadata()* internally invokes the *GetMetadata()* method from the MetadataUtil class. The purpose of this utility method is to make the WS-Metadata request to a server. To accomplish its task, it invokes

in turn the *getMexWsdlRequest()* method, that concretely builds the SOAP message containing the request. Once the *retrieveMetadata()* obtains a full response from the service, it performs the second process step, by invoking the *createMetadata()* method that basically removes the metadata from the SOAP message and returns a Metadata object.

The source class of the Metadata object is contained in the *client.schema* package. During the metadata retrieval phase, the above-mentioned methods take advantage of the functionalities provided by several utility classes, in particular the HttpPoster and the PortInfo classes. The former performs the task of making an HTTP POST request to a service while the latter holds information such as the name of a service, the qualified name of its port and the port address. Finally the ServiceDescriptorImpl class is a utility class that can be invoked by the underlying JAX-WS layer to access and use service metadata from an Endpoint.

For what concerns the various server side components, the MexEndpoint class was of particular importance for our purposes. As the name suggests, this class acts as an Endpoint entry for a Web service. The class implements the *invoke()* method of the Provider interface, introduced in the JAX-WS specification to provide a more fine-grained control over XML-based messages. The main goal of the *invoke()* implementation available in the MexEndpoint class is to act as a dispatcher that invokes additional auxiliary methods according to the request type. In particular, to differentiate among such requests, it parses the value of the Action Element available in the Header field of a SOAP request message. Unfortunately, only the code for the management of the GET requests aimed at obtaining a WSDL document and XML Schema documents was implemented. In this specific case the *processGetRequest()* method is invoked. Such a method (by exploiting

the WSDLRetriever class) first obtains the requested WSDL and then writes it in the Body of a SOAP response Message.

Nevertheless, the source code of the above mentioned methods has constituted the scaffolding to provide a service Endpoint with both the support for *GetMetadata* requests as well as the ability to manage geospatial metadata.

4.3.2 Adding the support for OGC metadata to the WSIT Metadata Exchange module

As mentioned in the introduction of this Chapter, each implementation that aims at providing W3C standards with the support for geospatial metadata should be totally transparent for existing clients and services that do not deal with geospatial information. To achieve this aim we left, whenever possible, the actual methods implementation and semantic of the original WSIT classes unaltered. The support for OGC metadata [2] has been provided by adding, to the involved classes, additional methods specifically designed to manage geospatial metadata requests and responses.

The overall modifications were, however, influenced by several aspects. First of all, the WSIT subsystem lacks the support for the latest version of the WS-Metadata specification. In fact, according to the information available in the source code, the supported specification is the WS-Metadata 1.1, dated September 2004. Nevertheless, this issue did not have a notable impact on the purposes of our discussion since the semantic and behavior of the <Metadata>, <MetadataSection> and <GetMetadata> Elements and the *Dialect* and *Identifier* attributes do not differ from the brief description

[2]The code snippets provided in this section use the retrieval of the WFS Capabilities as example. Of course, the whole discussion can be easily extended to every OGC service that can provide its metadata using a standard-compliant Capability document.

provided in subsection 4.1.1 Secondly, to accelerate the development process, we made an initial simplification. In particular we assumed that a requesting client knows a priori that the receiving service supports the geospatial metadata request and the related dialect. In a real context, such information could be retrieved, for example, from a service registry or the provided documentation. Finally, the lack of an official documentation for several WSIT modules has complicated in a consistent manner the whole design and development process. In the next two subsections we describe the main changes required to the original WS-Metadata modules to let both client and service deal with the new metadata types.

4.3.3 Adding the Support for OGC Metadata to the WSIT Metadata Exchange module - Client changes

Figure 4.5 shows a SOAP-based GetMetadata request message to retrieve the capabilities of an OGC compliant WFS. To provide a generic Java application with the ability to make such a type of request, properly understand the service response and effectively use the retrieved metadata, the following changes to the client package classes are required.

First of all, a unique way to address the new Dialect inside the Java source code is necessary. Therefore our first step concerned the addition, to the list of constants included in the MetadataConstants class, of a new CAPABILITIES_DIALECT string uniquely identifying the WFS dialect (Listing 4.1).

Listing 4.1: The CAPABILITIES_DIALECT string identifying the WFS dialect

```
1 public static final String CAPABILITIES_DIALECT =
    "http://schemas.opengis.net/wfs/1.1.0/wfs.xsd";
```

As for the actual retrieval and use of the WFS capabilities, a desirable requirement was to keep the semantic of the original two-step process, namely getting the service metadata and instantiating related Java objects, unaltered. One of the advantages of this choice was the ability to entirely reuse the *createMetadata()* method to return Metadata objects allowing us to focus only on the changes necessary for performing a GetMetadata request using the new *Dialect*. To accomplish this task we modified the main classes of the *client* package by adding several methods meant to both seamlessly handle the retrieval of the new metadata types and overcome some minor restrictions of the WSIT original implementation. Figure 4.6 shows the interactions sequence occurring during the retrieval process while Table 4.2 shows the classes affected by our changes.

```
<s11:Envelope
xmlns:s11='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:wsa='http://www.w3.org/2005/08/addressing'
xmlns:wfs='http://www.opengis.net/wfs'
xmlns:mex='http://www.w3.org/2011/03/ws-mex' >
<s11:Header>
   <wsa:To>
      http://some_address/mex
   </wsa:To>
   <wsa:Action>
      http://www.w3.org/2011/03/ws-mex/GetMetadata
   </wsa:Action>
   <wsa:MessageID>
     urn:uuid:73d7edfc-5c3c-49b9-ba46-2480caee43e9
   </wsa:MessageID>
   <wsa:ReplyTo>
      <wsa:Address>
        requesting_client_address
      </wsa:Address>
   </wsa:ReplyTo>
</s11:Header>
<s11:Body>
   <mex:GetMetadata>
   <mex:Dialect
   Type='{http://www.opengis.net/wfs}Capabilities'
   Identifier='http://schemas.opengis.net/wfs/
   1.1.0/wfs.xsd'/>
   </mex:GetMetadata>
</s11:Body>
</s11:Envelope>
```

Figure 4.5: A GetMetadata request to retrieve a WFS Capabilities document

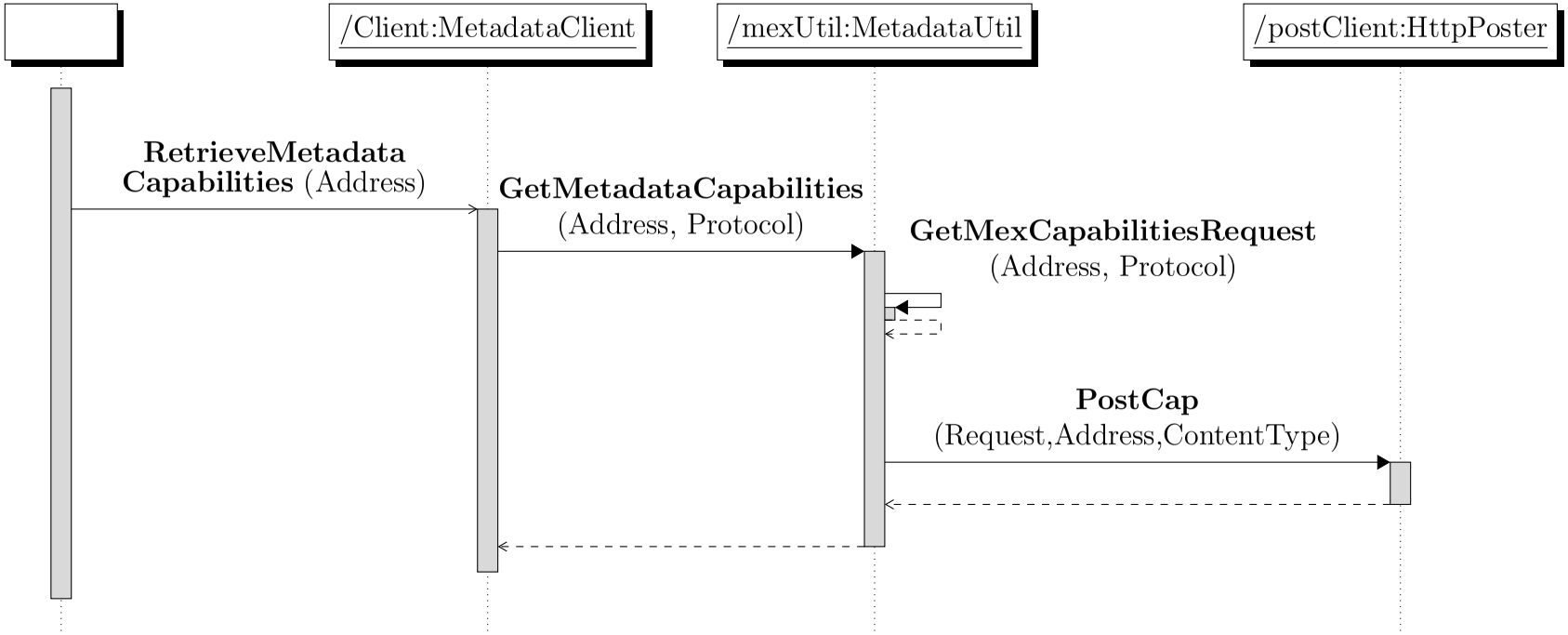

Figure 4.6: Classes and methods involved in the metadata retrieval phase

Table 4.2: Java classes of the client package affected by the changes

Class Name	Methods added or modified
MetadataClient	retrieveMetadataCapabilities()
MetadataUtil	getMetadataCapabilities(); getMexCapabilitiesRequest()
HttpPoster	postCap()
ServiceDescriptorImp	Class constructor; handleXML(); handleLocation(); getCapabilities()

The invocation of the *retrieveMetadataCapabilities()* method (Listing 4.2) is the first step that a Java client performs to begin the geospatial metadata exchange process. The methods body implements the above-mentioned two-step process of getting metadata and creating the corresponding Java objects. In particular, to carry out the first task an additional method, *getMetadataCapabilities()*, in the MetadataUtil class is invoked (Listing 4.3).

Listing 4.2: The `retrieveMetadataCapabilities` method

```
1  public Metadata retrieveMetadataCapabilities(@NotNull final
       String address) {
2      for (String suffix : suffixes) {
3          final String newAddress = address.concat(suffix);
4          for (Protocol p : Protocol.values()) {
5              InputStream responseStream = null;
6              try {
7                  responseStream =
                       mexUtil.getMetadataCapabilities(newAddress,
                       p);
8                  return createMetadata(responseStream);
```

```
9               } catch (IOException e) {
10                  logger.log(ERROR_LOG_LEVEL,
11                   MessagesMessages.MEX_0006_RETRIEVING_MDATA_FAILURE(
12                          p, newAddress));
13                  continue;
14              } catch (Exception e) {
15                  logger.log(Level.WARNING,
16                   MessagesMessages.MEX_0008_PARSING_MDATA_FAILURE(
17                          p, newAddress));
18                  continue;
19              }
20          }
21      }
22      logger.log(ERROR_LOG_LEVEL,
23          MessagesMessages.MEX_0007_RETURNING_NULL_MDATA());
24      return null;
25  }
```

The goal of *getMetadataCapabilities()* is to return an InputStream object that encodes the XML stream retrieved from the service Endpoint. The goal is fulfilled with the in-sequence invocation of two other methods, namely *getMexCapabilitiesRequest()* (Listing 4.4) and *postCap()* from the HttpPoster class (Listing 4.5).

Listing 4.3: The `getMetadataCapabilities` method

```
1  InputStream getMetadataCapabilities(final String address,
2          final Protocol protocol) throws IOException {
3
4          final String request = getMexCapabilitiesRequest(address,
```

```
            protocol);
        if (logger.isLoggable(Level.FINE)) {
            logger.fine("Request message:\n" + request + "\n");
        }
        String contentType = "application/soap+xml"; // soap 1.2
        if (protocol == Protocol.SOAP_1_1) {
            contentType = "text/xml; charset=\"utf-8\"";
        }
        return postClient.postCap(request, address, contentType);
    }
```

The purpose of the first method is to build the SOAP message to be sent to the service Endpoint. The two parameters needed are the Web service address along with its *mex* suffix and the version of the SOAP protocol used. The actual value of the GET_METADATA_REQUEST constant used to fill the Action field of the SOAP Header is shown in Listing 4.6. Finally, the *postCap()* invocation makes the real HTTP POST request to the service Endpoint. It is worth noting that we needed to slightly modify the method source code to provide the HTTP request with the ability to properly manage a GET_MDATA_REQUEST. Such a feature was not available in the original implementation.

Listing 4.4: The `getMexCapabilitiesRequest` method

```
private String getMexCapabilitiesRequest(final String address,
        final Protocol protocol) {

        // start with soap 1.2
        String soapPrefix = "s12";
        String soapNamespace = SOAP_1_2;
```

```
7       if (protocol == Protocol.SOAP_1_1) {
8           soapPrefix = "soap-env";
9           soapNamespace = SOAP_1_1;
10      }
11      return "<" + soapPrefix + ":Envelope " +
12          "xmlns:" + soapPrefix + "='" + soapNamespace + "' " +
13          "xmlns:" + WSA_PREFIX + "='" +
                AddressingVersion.W3C.nsUri + "'>" +
14          "<" + soapPrefix + ":Header>" +
15          "<" + WSA_PREFIX + ":Action>" +
16          GET_MDATA_REQUEST +
17          "</" + WSA_PREFIX + ":Action>" +
18          "<" + WSA_PREFIX + ":To>" + address + "</" + WSA_PREFIX
                + ":To>" +
19          "<" + WSA_PREFIX + ":ReplyTo><" + WSA_PREFIX +
                ":Address>" +
20          WSA_ANON +
21          "</" + WSA_PREFIX + ":Address></" + WSA_PREFIX +
                ":ReplyTo>" +
22          "<" + WSA_PREFIX + ":MessageID>" +
23          "uuid:778b135f-3fdf-44b2-b53e-ebaab7441e40" +
24          "</" + WSA_PREFIX + ":MessageID>" +
25          "</" + soapPrefix + ":Header>" +
26          "<" + soapPrefix + ":Body/>" +
27          "</" + soapPrefix + ":Envelope>";
28  }
```

Listing 4.5: An excerpt of the `postCap` method

```
InputStream postCap(final String request, final String address,
       final String contentType) throws IOException {

       final URL url = new URL(address);
       final HttpURLConnection conn = createConnection(url);
       conn.setDoOutput(true);
       conn.setDoInput(true);
       conn.setRequestMethod("POST");
       conn.setRequestProperty("Content-Type", contentType);
       conn.setRequestProperty("SOAPAction", "\"" +
          GET_MDATA_REQUEST + "\"");
```

Listing 4.6: The GET_MDATA_REQUEST constant

```
 public static final String GET_MDATA_REQUEST =
     XMLSOAP_2004_09 + "mex/GetMetadata/Request";
```

4.3.4 Adding the Support for OGC Metadata to the WSIT Metadata Exchange module - Server changes

Providing a service Endpoint with the ability to correctly manage the SOAP request message shown in Figure 4.5 and create a feasible response (or a proper fault message) constituted the main theme that led the design of the changes we made to the *server* package in general and to the MexEndpoint class in particular. In addition, similarly to what happened with the client package, during their actual implementation we tried to not alter the overall business logic and to reuse the functionalities available into the original package implementation.

Even though, in order to be effective, such modifications are tightly tied each other, from a logical point of view, we can group them into three main categories, namely:

1. Overcoming the lack of support for GET_METADATA requests inside the *invoke()* method,

2. Making all the involved classes and methods aware of the new metadata dialects, and

3. Enabling the service Endpoint to seamlessly access and parse the original Capabilities documents supplied by remote OGC-compliant services and encapsulate their content into a canonical MetadataSection Element.

As for the first task, on receiving a GET_METADATA request, the original implementation of the *invoke()* method simply returns a fault message (GET_METADATA_NOT_IMPLEMENTED). Therefore, the initial step concerned the need to modify the method's source code by adding the support to properly manage and dispatch such a type of requests (Listing 4.7).

Listing 4.7: Changes to the `invoke` method

```
1       /.............../

2

3       String action = headers.getAction(AddressingVersion.W3C,
            soapVersion);

4

5       /.............../

6

7       else if (action.equals(GET_REQUEST)) {
```

```
8           final String toAddress = headers.getTo(wsaVersion,
                soapVersion);
9           return processGetRequest(requestMsg, toAddress,
                wsaVersion, soapVersion);
10      }
```

The Action field embedded in the Header Element of a SOAP message constitutes the discriminating factor used by the *invoke()* native implementation to distinguish among valid metadata request types. Hence, on receiving a well-formed metadata request we compare the current value of the Action field with the value of the GET_METADATA_REQUEST constant (Listing 4.6) stored into the MetadataConstants class.

If the content of the action object is equivalent to the above-mentioned constant, the *processGetMetadataRequest()* method is invoked (Listing 4.8). The method's goal is to construct the response message containing the geospatial metadata requested by the client. The fulfillment of this task requires, as mentioned, the execution of two different activities. First of all, it is essential to access and parse the Capabilities document containing the intended metadata and return its Java-based representation. Subsequently an automated procedure that performs all the low-level steps to correctly embed the retrieved information into a SOAP response message is required. We overcame such issues by improving the functionalities of the server package with the addition of two new classes, namely CapabilitiesRetriever and CapabilitiesUtility containing several utility methods (Table 4.3).

Listing 4.8: The `processGetMetadataRequest` method

```
1   private Message processGetMetadataRequest(final Message request,
2       String address, final AddressingVersion wsaVersion,
```

Table 4.3: Additional classes added to the server package

Class added	Methods
CapabilitiesRetriever	addDocuments(), writeDoc()
CapabilitiesUtility	writeDocTo()

```
3       final SOAPVersion soapVersion) {
4
5       try {
6
7           WSEndpoint ownerEndpoint = findEndpoint();
8
9           if (ownerEndpoint != null) {
10
11              final MutableXMLStreamBuffer buffer = new
                    MutableXMLStreamBuffer(1024000);
12              final XMLStreamWriter writer =
                    buffer.createFromXMLStreamWriter();
13
14              address = this.getAddressFromMexAddress(address,
                    soapVersion);
15              writeStartEnvelope(writer, wsaVersion, soapVersion);
16              CapabilitiesRetriver cr = new
                    CapabilitiesRetriver(ownerEndpoint);
17              cr.addDocuments(writer, null, address);
18              writeEndEnvelope(writer);
19              writer.flush();
20
21              /....... OTHER CODE ...../
```

```
22
23              headers.add(Headers.create(new
                    QName(wsaVersion.nsUri, "Action"),
                    GET_MDATA_RESPONSE));
24              return responseMessage;
25          }
```

The *addDocuments()* method (Listing 4.9) retrieves a local copy of the Capabilities document from a predefined disk location. Such a byte stream constitutes, in turn, one of the mandatory inputs for the following invocation of the *writeDoc()* method (Listing 4.10). The invoked method builds the MetadataSection element that will contain the Capabilities document obtained in the previous step. As described in subsection 4.1.1, in addition to the actual content, MetadataSections Elements have to contain also the *Dialect* and *Identifier* attributes of the metadata unit. To accomplish this task we simply re-used the CAPABILITIES_DIALECT string (Listing 4.1). Finally, to improve the code reusability and maintainability, we chose to separate from the remaining part of the methods body, the Java code that deals with the addition of the Capabilities documents content to the payload of a MetadataSection Element. This specific task is accomplished by the *writeDocTo()* method (Listing 4.11).

Listing 4.9: The `addDocuments` method

```
1 void addDocuments(final XMLStreamWriter writer, final Packet
     request,
2         final String address) throws XMLStreamException {
3         FileInputStream fileInputStream = null;
4         try {
5             fileInputStream = new
```

```
            FileInputStream("GetCapabilitiesResponse.xml");

        /.............../

        writeDoc(writer, fileInputStream, address);

        /.............../
    }
}
```

Listing 4.10: The `writeDoc` method

```
private void writeDoc(XMLStreamWriter writer, FileInputStream
    file, String address) {
    try {


            writer.writeStartElement(MEX_PREFIX,
                "MetadataSection", MEX_NAMESPACE);
            writer.writeAttribute("Dialect",
                CAPABILITIES_DIALECT);
            writer.writeAttribute("Identifier",
                CAPABILITIES_DIALECT);

            CapabilitiesUtility utility = new
                CapabilitiesUtility();
            utility.writeDocTo(file, writer, address);

            writer.writeEndElement();
```

```
14
15
16          } catch (XMLStreamException ex) {
17              /.............../
18          }
```

Listing 4.11: The `writeDocTo` method

```
1  public void writeDocTo(File file, final XMLStreamWriter writer,
       final String address) {
2          try {
3              JAXBContext context =
                   JAXBContext.newInstance("net.opengis.wfs.v_1_1_0");
4
5              Unmarshaller unmarshaller = context.createUnmarshaller();
6              ObjectFactory wfs_factory = new ObjectFactory();
7
8              JAXBElement<WFSCapabilitiesType> wfscapabilitiesElement
                   = (JAXBElement<WFSCapabilitiesType>)
9               unmarshaller.unmarshal(new StreamSource(file),
                    (wfs_factory.createWFSCapabilitiesType()).getClass());
10
11             WFSCapabilitiesType capabilitiesElement =
                   wfscapabilitiesElement.getValue();
12
13             Marshaller jaxbMarshaller = context.createMarshaller();
14             jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
                   Boolean.TRUE);
15             jaxbMarshaller.marshal
```

```
16          (wfs_factory.createWFSCapabilities(capabilitiesElement),
                writer);
17
18
19      } catch (JAXBException ex) {
20          Logger.getLogger
21          (CapabilitiesUtility.class.getName()).log(Level.SEVERE,
                null, ex);
22   }
23 }
```

A critical aspect of such an operation that it is worth to analyze is represented by the binding process of GML objects into Java classes. The direct mapping from OGC Schemas into usable Java objects is, in fact, not a straightforward process.

In the Java platform, one of the most common ways to access and manage XML documents is by using the Java Architecture for XML Binding (JAXB) API (Ort and Mehta, 2003), a specification whose main aim is to simplify the integration of XML and Java technology by providing an object-oriented representation of XML documents. According to the official documentation, the actual use of this API can be seen as a two-step process:

- Binding the Schema of an XML document into a series of Java interfaces and classes representing the Schema elements. Each JAXB compliant implementation provides a binding compiler, namely a tool that automatically performs this task.
- Unmarshalling of the XML document, namely creating an objects tree that represents the actual content and structure of the XML document.

The objects in the tree are instances of the classes produced in the binding step. During the unmarshalling phase, it is also possible to validate the source data against the target XML Schema.

Unfortunately, the proper use of the JAXB API (in particular the use of XJC, the binding compiler of JAXB Reference Implementation) with the OGC XML Schemas represents a challenging task. Major issues to solve include the version proliferation of several Schemas, Schemas that are not valid (e.g., GML version 3) or depend on several others Schemas. Such a complexity makes it difficult, for an OGC Schema, to be directly compiled as is. In this context, a support that dramatically simplifies the Java-based implementation of the OGC specifications can be found in the bindings and libraries provided by the *OGC Schemas and Tools Project* whose aim is to *"compile all of the OGC XML Schemas with JAXB schema compiler"* (OGC Schemas and Tools Project, 2009). According to the documentation, the project supports the 1.0.0 and 1.1.0 versions of the WFS Schemas and the latest versions of the 3.X.X branch of GML (from 3.1.1 to 3.2.1).

In this context, the core elements in Listing 4.11 are the JAXBContext, the ObjectFactory, the WFSCapabilitiesType and the Marshaller classes. The general role of the first class is to provide a customized binding that will provide the necessary information for the marshalling, unamarshalling and validation phases. The ObjectFactory class contains factory methods for each Java element that has been generated starting from the *net.opengis.wfs.v_1_1_0* package. The WFSCapabilitiesType represents the Java class that corresponds to the *WFS_CapabilitiesType* complex type namely the XML Element defined in the WFS Schema that specifies the structure of the service metadata. Finally, with the Marshaller class we convert in-memory Java objects into the XML file that will constitute the Capabilities document.

4.4 Discussion

The ability to compose different services, possibly developed by different organizations to provide complex functionalities represents one of the key features behind the success and wide adoption of the SOC paradigm. However, one of the requirements to successfully compose two or more services is that all the involved entities must adhere to the same set of standards. When services developed according to different sets of standards are involved in the composition process, several interoperability issues may arise that need to be overcome. Such issues cannot be fixed by a mere mechanical process and require the development of *ad-hoc* solutions. A typical example is represented by the composition of W3C and OGC services. In the context of Software Engineering, the development of a wrapper represents a common approach to allow the communication among heterogeneous software systems. In particular, for what concerns the Service Oriented Computing, the development of an abstraction wrapper that abstracts as much as possible the inner details and logic of the wrapped service surely represents a solution that well embraces the low-coupled nature of the entire paradigm. Moreover, the adherence to accepted standards and the ability to reuse the implemented functionality, represent additional desirable core features for such a type of software solution. As for the specific case of OGC and W3C services, the development of a wrapper that exposes OGC functionality in a W3C compliant way represents a feasible manner to overcome all the technical differences marking the two types of services. Unfortunately, the wrapper development process does not represent a straightforward task since a complete mapping from the OGC standards to the W3C ones is still an open research question.

In this chapter we specifically addressed the issues related to the proper management of geospatial metadata into a W3C-based environment. Due to

their fundamental role for the proper exploitation of the information conveyed by an OGC service, it is essential to provide an adequate mechanism for their discovery and use. To efficiently accomplish this task, we proposed a flexible and standard compliant mechanism for the retrieval of such metadata directly from a wrapper avoiding the use of third-party entities like UDDI registries. Our proposal exploits the scalable and adaptable nature of the WS-Metadata specification letting W3C services retrieve geospatial metadata by using an existing and well-accepted W3C standard.

The proposal we discussed extends and improves the solutions proposed in the current literature and its scientific contribution can be summarized as follows:

- A tight adherence to the syntactic and semantic requirements of the WS-Metadata protocol.

- The ability to seamlessly support different types of OGC-compliant services along with different versions of the same OGC service.

- Capability to esily reuse the proposed functionality in a wider development context since the proposed theoretical approach has been concretely implemented as an extension of an official module of the Java 2 Enterprise Platform.

Additional direct consequences of the entire design approach are the reduced effort required to adapt the developed module to other types of services that exchange their metadata using XML-based formats and its strong integration with other features of existing service middlewares. As mentioned, to better satisfy such requirements we decided to investigate the possibility of extending the functionalities of existing software libraries instead of developing a completely new module. In particular we carefully considered how

our proposal might impact on the reference implementation of the Java API for XML Web Services and focused on the changes required to provide the packages that implement the WS-Metadata protocol with the ability to support the proposed extensions. In the following table we summarize the main characteristics of our approach and compare them with the existing literature contributions analyzed in Chapter 2.

		Existing Literature		
	Our Approach	**Ioup et al., 2008**	**Amirian et al., 2010**	**Sancho-Jiménez et al., 2008**
Broader Purpose	Enhance information exchange between OGC and W3C services			
Approach for Geospatial Metadata Management	Service Wrapper - Extension of the Web Services Metadata Exchange specification	Service Wrapper - Encoding geospatial metadata inside the extensible portions of the WSDL document	Service Wrapper - Extension of the Web Services Metadata Exchange specification	A Proxy to derive SOAP and WSDL Metadata from the mandatory metadata in a WPS
Supported OGC Services	OGC services compliant with the Common Standard			WPS
Technology Used for Concrete Development	Java 2 Enterprise Platform	Information not provided	Microsoft .Net	Information not provided - A tool to test the viability of the proposed approach is described
Limitations and differences Compared to our approach		Embedding additional metadata into a WSDL document does not represent a standard way to exchange such data in W3C-based infrastructures	The proposed solution is a custom-developed metadata retrieval framework. The discussed GetMetadata request does not stricly adhere to the guidelines provided into the WS-Metadata specification	The proposed solution is limited to the WPS standard. Additional information has been added using all WSDL and SOAP implementation resources in order to approach SOAP and mandatory requests

Table 4.4: Comparison of our approach with the existing literature

Chapter 5

Evaluation

This chapter is devoted to provide an assessment of our research contributions to the front and back ends of the information system developed for the SLN4MoP project. The chapter is organized as follows. Section 5.1 discusses the users' response towards the proposed mobile prototype. Section 5.2 shows an example of the broader applicability of our approach to user requirements elicitation and application development. Finally, Section 5.3 presents the initial results concerning the black-box testing of the changes we made to the client and server packages of the WSIT Metadata Exchange module and considerations regarding the full implementation and integration of our wrapper solution within the SLN4MoP architecture.

5.1 Evaluating the Mobile prototype

Usability is a major concern of any interactive software solution. According to the ISO 9241-11 standard, the term usability is defined as "*the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*"(ISO

9241-11, 1998). Given the growing importance of addressing usability requirements for the success of a software solution, usability testing is acquiring a primary role in development processes. Moreover usability testing is essential in the context of UCD processes, since "*the only way to be sure about the effectiveness of some design decisions is to build and evaluate them, through the use of application prototypes.*"(Matera et al., 2006).

In this context, the detailed evaluation of the choices that influenced the design and the offered features of our prototype represented a factor of the utmost importance for the purposes of our research project. In order to devise an appropriate usability plan and get the most out of *in situ* testing activities with Sri Lankan farmers, we conducted a preliminary pilot study whose main idea was to tune the usability evaluation goals (Di Giovanni et al., 2013). The pilot study involved people belonging to a Sri Lankan ethnic group living in Salerno, in Italy.

The remainder of this Section is structured as follows. Firstly we recall the main characteristics of the mobile prototype used for both the preliminary pilot study and *in situ* evaluation and present one of the exemplary interactions scenarios we developed to specify the mechanisms for accessing and manipulating task information and show the effectiveness of the developed interface. Subsequently we describe the design and main goals of the pilot study and discuss the elicited results. Finally results from the *in situ* trials are provided.

5.1.1 The Mobile Prototype used for Evaluation

The mobile prototype used to perform our usability evaluation (Figure 5.1, Figure 5.2 and Figure 5.3) targeted mainly the crop selection stage. After a basic login facility (Figure 5.1, left and center), the farmer is directed to

an interface where the 6 main stages of the farming life-cycle are included (Figure 5.1, right).

By selecting the first stage (crop planning), the farmer is provided with the ability to choose among vegetables, fruits or other types or crops as well as to refer to the history functionality (Figure 5.2, left). The *History* functionality shows the farmer what he/she has cultivated in the recent past.

Once the farmer has selected the desired crop variety, the mobile application shows a list of available products (Figure 5.2, center). The list is sorted according to the traffic light metaphore (from green to red). The farmer can select one or more products to compare details like the current level of production (Figure 5.2, right). By selecting a particular product the application provides additional details (Figure 5.3, left). Finally, by tapping on the *select quantity to grow* button the farmer can insert the desired amount of product he/she wishes to cultivate (Figure 5.3, center and right).

The prototype used for the evaluation was in Sinhala language, which is the native language in Sri Lanka. An example scenario is described in Table 5.1.

Figure 5.1: The UI of the mobile prototype

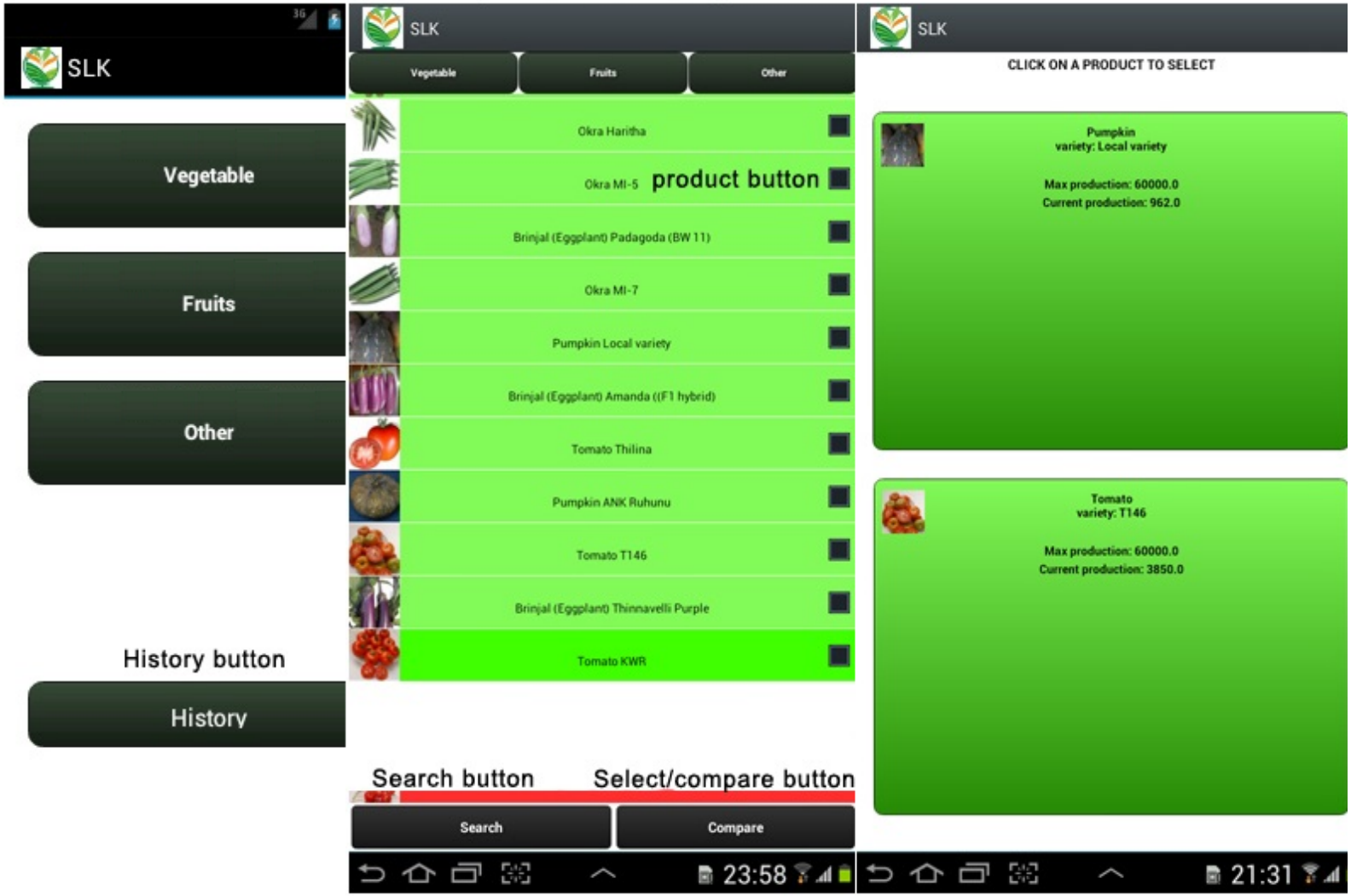

Figure 5.2: The UI of the mobile prototype

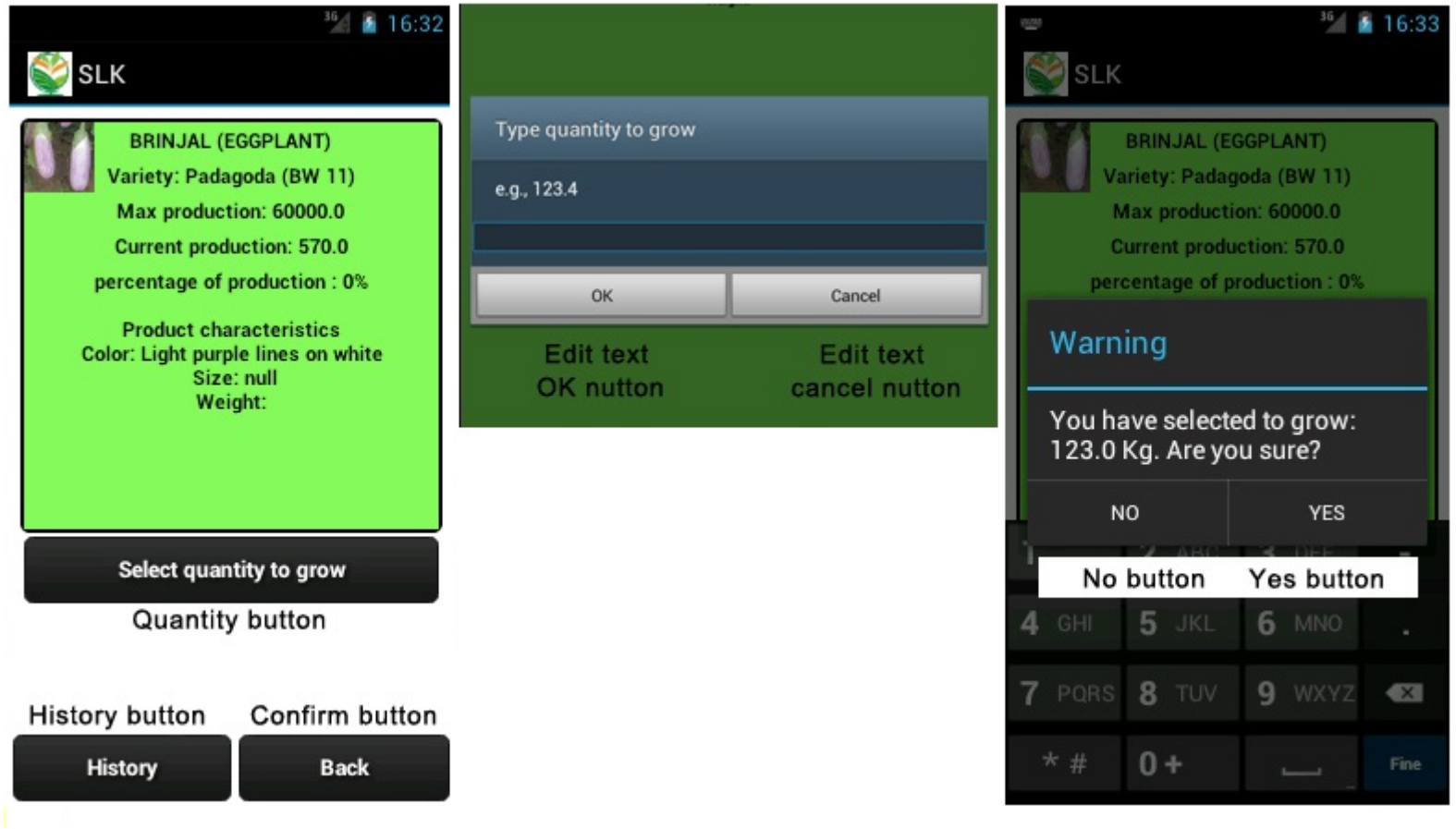

Figure 5.3: The UI of the mobile prototype

Table 5.1: The exemplary interaction scenario

Persona
Sirisena is a 45-year-old farmer with long experience in truck farming. Sirisena is part of Sinhalese ethnic group. He does not have advanced technical skills; the only technological instrument is his mobile phone that he uses everyday. Sirisena lives in Sigiriya, a village in the central Matale District of Sri Lanka, where he owns four acres of farmland. Since Sirisena has a long experience in truck farming he manages the crop production of his family farm. His role is to make decisions on critical aspects of the production. He takes decisions on the kind of production, the time to start it and he establishes an indicative selling price
Scenario
Sirisena is interested in producing convenient crops, i.e., crops that produce high yield within a short time. He launches the application and, after the Login phase, chooses the Crop Planning activity that displays three buttons related to under-supplied, average and oversupplied products. Siresena selects the button for under-supplied products and the list of products in that category is displayed. Products are framed in a variety of decreasing intensity of green, with the darkest corresponding to the highest demand and the lightest to the lowest demand. Observing the proposed list, Sirisena is primarily interested in the production of tomatoes and potatoes and secondly in the production of bananas and strawberries. So he selects these four products of interest and taps on the *compare* button. On the basis of the information provided by the application, Sirisena decides to grow bananas. Selecting the desired product, the application displays the form where he can enter the amount of product he wishes to cultivate. The green frame around the desired product still confirms that the selected quantity does not move crop production level from under supply level to a higher level. Therefore Sirisena decides to confirm his choice.

5.1.2 The Pilot Usability Study

The Sri Lankan group living in Salerno is composed of approximately one hundred people. After the presentation of the SLN4MoP project during a

plenary meeting of that community, we were able to select twenty people who were motivated and knowledgeable enough to play the role of farmers in our experiment. In order to gauge the potential effectiveness and usefulness of the application, we applied a *think-aloud* technique (Van Someren et al., 1994), inviting participants to perform some representative tasks with the prototype with the twofold aim to gain precious suggestions on how to improve it and to verify whether the appropriate data are being collected. During the pilot study, we focused on three aspects of the interface, namely

1. the capability to understand the interface: we checked whether the information clues and their layout are effective to guide users in accomplishing their tasks. The interface must not confuse user or make him/her feel awkward.

2. the capability to navigate the menu: we needed to determine the best way to structure the menu in order to avoid user getting lost during the interaction.

3. the capability to analyze results: we wanted to evaluate whether the information provided by the application is appropriate to lead users towards the selection of the most convenient crops.

Users and instrumentation

The pilot test was performed simulating the real environment in the rural area around the city of Salerno. We used a Huawei Ideos smartphone, an entry-level model with a 600 Mhz processor, 256 megabytes of central memory and a 2,8 inches screen with a resolution of 320x240 pixels. This choice is consistent with the kind of smartphones really available in countries where people cannot afford big expenses.

The 20 participants belong to the Sinhalese ethnic group and are members of an acceptance center for Sri Lankan immigrants. All of them are well integrated in the local Italian society. Nevertheless, they are all knowledgeable about the rural issue in Sri Lanka, in most cases being the reason for their emigration, and they appeared to be highly motivated in performing the tests.

As part of the study, starting from the illustrative scenario described in Table 5.1, we identified three critical subtasks (Table 5.2). Each subtask represents a key feature that influences users capability. To this aim, we asked subjects to adopt the perspective of Sirisena and to simulate the scenario activity by carrying out the tasks while commenting any critical step. We decided to start the experiment with no preliminary training on the application module, overall aiming to verify the learnability degree of the interface.

During the evaluation, working mobile prototype, questionnaire[1] and interviews were used as main research instruments. Each session lasted 20 minutes. An excerpt of the instruction provided to each participant can be found in Table 5.3

The results were compared against a set of usability specifications, which we elaborated for the envisaged scenario (Table 5.4). As usual, the performance measures are based on time to perform a subtask and number of errors. Satisfaction is measured on a 5-point attitude scale. For example, *predictability* after the first subtask was rated on a scale from 1= *not at all predictable* to 5 =*fully predictable*.

Some results are reported in Table 5.5. For the first subtask, we can see that the use of the traffic light metaphor to group the products combined with the intensity color scale, proved to be a good choice, also supporting

[1]See Appendix 2 for questionnaire details

Table 5.2: Subtask decomposition and usability evaluation goals

Subtask Description	**Usability Evaluation Goal**
1. The user explores the application. He/she presses one of the three buttons related to product supply and visualizes the list of products in the corresponding category. Considering the first ten items in the catalog he/she annotates what is the current *production level* he/she perceives for each item.	Understand whether the color intensity scale used in the list is perceived as a measure of the current product demand.
2. The user finds and selects four specific kinds of crop displayed in the catalog. He/she compares them.	Verify whether the user is able to navigate the single level catalog menu.
3. He/she ultimately makes a selection.	Verify whether the user can correctly interpret the feedback received from the application and make the right choice.

Table 5.3: General Instructions for the Farming Prototype

In the next 20 minutes or so, you will be carrying out 3 tasks with the Farming prototype, which are related to the crop selection activity. Each of you will perform different tasks within a common scenario reproducing the situation when a farmer has make a decision on which is the most convenient crop to seed, for a given land and a given season. Note that we intentionally leave out some of the detailed task steps so that we can determine how well the system can guide your interactions with it. If you are confused at any point, please make your best guess on how to proceed, using the information that you have been given. We will intervene if necessary to help you make progress. At the start of each task, please say loud: Beginning Task followed by the number of the task. When you are done, please say: Task Complete. Also, please remember to think out loud as you work. It is very important for us to understand your goals, expectations, and reactions as you work through the task. Any further questions?

Table 5.4: Excerpt from the usability specifications related to the given scenario

Subtask	*Worst Case*	*Planned*	*Best Case*
1. Deduce the expected production level for the envisioned crops	3 errors	1 error	0 errors
	2 minutes	1 minute	30 seconds
	2 on predictability	3 on predictability	4 on predictability
2. Select and compare 4 specific products	3 errors	1 error	0 errors
	4 minutes	2 minutes	1 minute
	3 on complexity	2 on complexity	1 on complexity
3.Select the best crop out of 4 specific products	bad choice	optimal choice	optimal choice
	5 minutes	2 minutes	1 minute
	3 on confusion	2 on confusion	1 on confusion

predictability of the interface. Users were able to easily guess the production level of a crop by simply observing the color of the frame around it. Regarding the second subtask, while the average number of errors is comparable to that of the first subtask, the execution time suggests that the use of the virtual page design pattern to search for a product might be too complex for our target users. This was confirmed by the rating assigned by users to complexity and by their comments during the think-aloud session. Finally, the last subtask shows how the majority of participants was able to select the product for which the highest income can be expected.

The interviews revealed that participants were satisfied with the choice of graphics and color usage. High appreciation received the possibility to compare two or more products. When information such as average price or quantity sold in the previous year were similar for more than one product, the frame indicating the production level was used as a discriminating factor in selecting the best product. Finally, the probably most interesting outcome of the experiment was the identification and tuning of the main usability goals to take into account during the *in situ* evaluation. Thus, for example,

we realized that before starting the testing activities in Sri Lanka, the usability evaluation goal "*verify whether the user is able to navigate the single level catalog menu*", should be tuned towards a comparative evaluation of the virtual page solution against alternative design solutions, such as a grid layout interface, with or without the suggested subdivision of products into categories. The test also revealed that the ease of use should be considered as a primary usability specification feature and its qualitative measure added to the satisfaction parameters to be tested during the *in situ* evaluation.

5.1.3 The first in situ Field Trial

The general approach adopted in the pilot study was replicated when the actual field trial with Sri Lankan farmers was performed in December 2012. As initially mentioned in Chapter 3, this evaluation was carried out using a sample of 32 farmers. More extensive evaluations with a bigger sample of farmers are planned and will be performed in the near future.

The actual in situ investigation and the elaboration of results were performed by other members of the SLN4MoP project. A complete overview of the elaboration setup and a detailed analysis of farmers reaction towards the mobile prototype we designed can therefore be found in (De Silva et al., 2013). In the remainder of this subsection we describe the evaluation setup and summarize its main outcomes.

Users and Instrumentation

The evaluation study was conducted in the Matale District, a main vegetable producing region in Sri Lanka. The mobile application was field trialled with eighteen farmers from the Dambulla Agrarian Division and fourteen farmers from the Galewela Agrarian Division. In such agrarian service divisions, a

Table 5.5: Some relevant results from the pilot study

Subtask no	*Participants results*
1	Errors (Mean): 2
	Errors (Median): 1.07
	Errors (SD): 1.09
	Errors (Mode): 2
	Time to Completion (Mean): 74 seconds
	Time to Completion (Median): 67.5 seconds
	Time to Completion (SD): 36.17 seconds
	Time to Completion (Mode): 55 seconds
2	Errors (Mean): 1
	Errors (Median): 1
	Errors (SD): 0.98
	Errors (Mode): 1
	Time to Completion (Mean): 170 seconds
	Time to Completion (Median): 156 seconds
	Time to Completion (SD): 72.12 seconds
3	Percentage of participants who made the right choice: 70%
	Time to Completion (Mean): 105 seconds
	Time to Completion (Median): 93 seconds
	Time to Completion (SD): 41.92 seconds

high percentage of the population in engaged in the farming industry. The farmers were selected with the help of agricultural officers at Dambulla and Galewela. The trial was conducted in the farmers native language Sinhala.

The aim of the trial was to assess farmers ability to use Smartphones, identify general usability issues in using our application and evaluate how farmers perceive crop selection. In particular, in order to investigate usability

issues, three individual tasks were assigned (De Silva et al., 2013):

1. Select a crop and a quantity to be cultivated,
2. Compare two or more crops using the comparison facility, and
3. Use the history functionality to check the cultivated or planned crops in recent past

The evaluation was conducted over two consecutive days (the first day at the Dambulla Ggrarian Division and the second one at the Galewela Agrarian Division). Five researchers took part in the process. Each researcher worked with one or two farmers at a time.

During the evaluation, working mobile prototype, questionnaire and interviews (Table 5.6 and Table 5.7) were used as main research instruments.

Table 5.6: Crop Planner Evaluation - Interview Questions (5 likert Scale) Source: (De Silva et al., 2013)

1 All information for the crop choosing stage is provided
2 Information is sufficient for decision making
3 Knowledge on history is important
4 Market prices of the previous year are important in deciding a crop
5 Crop comparison facility is essential in deciding a crop
6 Color code usage is important in deciding a crop

The starting and end time were recorded during each task. When the three tasks ended a questionnaire was provided to get farmers' feedback. The questionnaire included both Likert scale and open ended questions to encourage and capture wide range of answers based on the participants knowledge. This gave us the capability to capture farmers ideas freely. One such open

Table 5.7: Crop Planner Evaluation - Interview Questions (Open ended)

1 Name three main factors that you would consider in prior deciding a crop for cultivation.
2 What three things did you like most about the Farming prototype?
3 What three things did you like least about the Farming prototype?
4 If the Farming prototype was made available to you, would you like to use it or not? Why?
5 What do you suggest as changes to the design of the Farming prototype?
6 Do you have any other comments or reactions?

ended question was to identify the factors/functionality which attracted the farmers towards using this application.

We started our evaluation by gathering the general demographic details of the farmers. Subsequently, we gathered information about how they access information, their mobile phone and internet usage. The survey was filled by each farmer which took nearly an hour. As most farmers have not used a Smartphone before, we first gave them some activities to get them used to a Smartphone. This involved dialling the mobile phone number of the researcher, answering the phone when it rings, sending a text message to the researchers Smartphone, locating and reading the text message, taking a picture and looking at the picture that was taken. Next the crop planning module in the application was demonstrated while illustrating the key features described earlier. Then a set of activities aiming to assist the farmer to select a crop for the next season was given in order to evaluate the prototype. At the end farmers were asked to answer few more questions.

Evaluation Results and Discussion

The basic demographic characteristics of the sample population are shown in Figure 5.4

The great majority of sample population is between 21 and 50 year old.

Respondents (32) based on					
Region		%	Education		%
Galewela	14	44%	Primary	2	6%
Dambulla	18	56%	Secondary (O/L)	22	69%
Age		%	Secondary (A/L)	5	16%
< 21	0	0%	Diploma	0	0%
21 - 30	4	13%	Graduate	2	6%
31 - 40	9	28%	Post Graduate	0	0%
41 - 50	11	34%	No Proper Education	1	3%
> 51	8	25%	Phone Availability		%
Sex		%	Yes	26	81%
Male	32	100%	No	6	19%
Female	0	0%	2 Phones	9	28%
			3 Phones	1	3%

Figure 5.4: Basic characteristics of the sample population. Source: (De Silva et al., 2013)

Most farmers are well-educated since 91% of them completed at least secondary education. As discussed also in Chapter 3, mobile phone usage is very high among the farming community although the main use was restricted to phone calls Figure 5.5.

However, despite such a basic use almost all the farmers that tried a mobile smartphone got used to it within 5-10 minutes. In addition, although the majority of them were non-internet users (Figure 5.6) more than 50% were aware of services that can be accessed via the internet (Figure 5.7).

A summary of the findings can be found in Table 5.8 while Table 5.9 summarizes the completion time (in minutes) for each of the three tasks.

According to the farmer response summarized in Table 5.8, around 57% was very attracted to the idea presented using the colour coding scheme. Farmers were bit concern on the accuracy of yield information through this method and the ability to make a correct decision based on the colour code. A

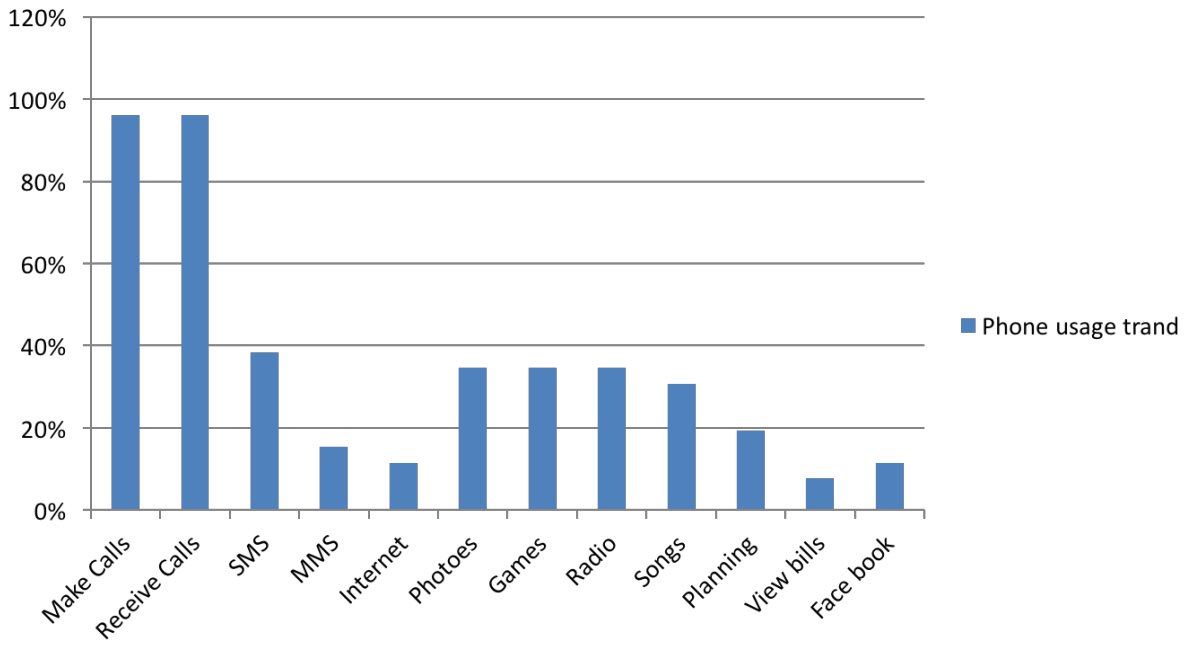

Figure 5.5: Phone usage trend among farmers

Computer / Internet Usage		%
Computer Users	8	25%
Non- Computer Users	24	75%
Internet Users	3	9%
Non- Internet Users	29	91%
use internet via other people	2	6%
Not Heard about Internet	1	3%

Figure 5.6: Technology Usage - Computer/Internet usage among farmers

percentage around 46% farmers found the information provided with respect to crop types and different varieties very useful for them to continue using the application. As our initial prototype contains all possible vegetables and their

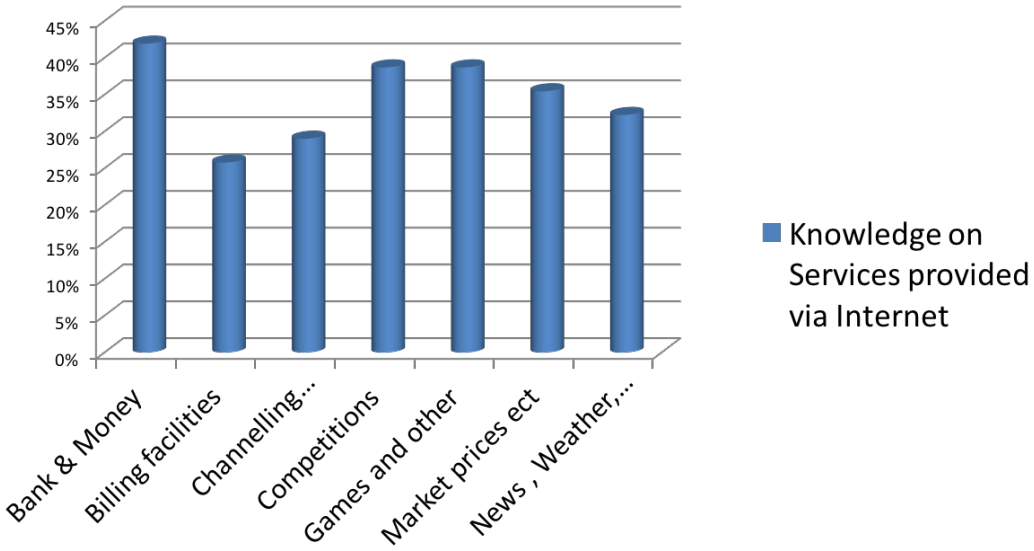

Figure 5.7: Farmers Knowledge on Services Provided via Internet

different varieties on the same screen without having a better classification, farmers found it difficult to select or search for the crop varieties that they were looking for. 62% responses were received in favour of the comparison facility. Some have mentioned the importance of showing more information such as the price sold and the issues faced with respect to the selected crops in the previous season highlighting the need to maintain historical data. Some mentioned that they were attracted due to the language used and the presentation of information which is clear for any novice users to learn and understand. Similarly, some have liked the application as it provides more valuable information which can be accessed in less time and cost.

Table 5.10 shows farmers response about the three main factors they would consider before deciding a crop for cultivation.

Table 5.11 and Table 5.12 show the three main things that farmers liked

Table 5.8: Percentage responses received from a sample of 32 farmers on the information provided and the effectiveness of the features provided in decision making (SA-Strongly Agree, A-Agree, MA-Moderately Agree, DA- Disagree and SDA- Strongly Disagree

Question	**SA**	**A**	**MA**	**DA**	**SDA**
All information for the crop choosing stage is provided	7%	57%	37%	0%	0%
Information is sufficient for decision making	3%	43%	50%	3%	0%
Knowledge on history is important	53%	47%	0%	0%	0%
Market prices are important in deciding a crop	47%	33%	20%	0%	0%
Crop comparison facility is essential in deciding a crop	62%	34%	3%	0%	0%
Color code usage is important in deciding a crop	57%	37%	3%	3%	0%

Table 5.9: Task completion time in minutes based on sample of farmers

	Task 1	**Task 2**	**Task 3**
Median	4	2	1
Mean	4.90	3.14	1.77
SD	3.61	2.68	2.60
Mode	2	1	1

the most and the least about the farming prototype. In particular, around 81% of the correspondents mentioned that there is nothing they can identify as an unwanted feature. However, some have mentioned that it is difficult to find the next action due to the lack of clarity, making it harder to use the application.

Table 5.10: Crop Planner Evaluation - Interview Questions (Open ended)

Name three main factors that you would consider in prior deciding a crop for cultivation	
Revenue	**53%**
Market Prices	**41%**
Weather Factors	**38%**
Crop Variety / Seeds Quality	**28%**
Seasonal Crop Types	**28%**
Water Availability	**22%**
Demand in the Market	**16%**
Current Cultivation Trend	**13%**
Nursery Preparation	**13%**
Less Potential to Diseases	**9%**
Other	**25%**

Table 5.11: Crop Planner Evaluation - Interview Questions (Open ended)

What three things did you like most about the Farming prototype?	
Usage of Color code	**56%**
Seasonal crop varieties for the area/ Information regarding crops	**47%**
Comparison Facility	**34%**
History	**25%**
Current production statistics/Percentages	**25%**
All functionality	**9%**
All functionality	**9%**
Presentation of clear information	**6%**
Ability to select crops	**3%**
Ability to get information easily in less time	**3%**
Language used(Sinhala)	**3%**

Table 5.12: Crop Planner Evaluation - Interview Questions (Open ended)

What three things did you like least about the Farming prototype?	
N/A	**81%**
Difficulty in finding the next action	**3%**
Less Services	**6%**
Limited Number of Crops	**3%**
Unavailability of information w.r.t. pest and diseases	**3%**
How to safe guard the cultivation	**3%**

5.2 Assessing the Effectiveness of our Design Approach in a different Application Domain

The goal of this Section is to show the broader applicability of the design approach described in Chapter 3 by providing an example of its application in a domain different from the agricultural and environmental ones. To this aim, a multidisciplinary collaboration in the healthcare domain (Sebillo et al., 2015), provided us with a well established test bed where the diverse stakeholders could play a relevant role since the initial steps of a system life cycle. In particular, the research we are conducting in this field is addressed to provide patients with personalized services based on a technology with a low-level invasive impact.

This goal represents one of the main achievements of a complete patient-centered care, namely the ability to provide care "*that is respectful of and responsive to individual patient preferences, needs, and values, and ensuring that patient values guide all clinical decisions*"(Institute of Medicine (US). Committee on Quality of Health Care in America, 2001). In this context,

promising results are coming from the development of mobile health solutions (m-health, for short) that, by exploiting the unique features of modern mobile devices, help people perform daily activities necessary to control and handle their health condition.

When compared with the SLN4MoP case study, the development of mobile solutions for the healthcare domain in general and the diabetes management in particular represents a well-studied field. However, most of the proposed solutions, although supporting the majority of the basic tasks needed by a diabetic patient, still suffer from several limitations that undermine their global effectiveness. On one side, the lack of integration and interoperability with existing clinical information systems limits a seamlessly exchange of information among all the involved actors (patients, physicians, pharmacists, etc.). On the other side, although mobile applications are usually preferred to traditional computer or Web-based solutions, their lack of usability represents a considerable obstacle to the achievement of healthcare platforms for the diabetes management. Among the main usability issues that affect existing solutions, we can mention data entry difficulty, lack of support in the automation of repetitive tasks, and absence of personalized feedback (El-Gayar et al., 2013).

Then, to successfully develop personalized services with a low-level invasive impact, the engagement of all the potential stakeholders is needed from the beginning. Such an engagement is, in fact, of the utmost importance to better clarify functional requirements and, more important, to take into account the set of usability attributes that play a key role in such a type of software solutions.

In the following, we summarize the noteworthy points that show how our proposal can be successfully exploited for the requirement elicitation phase

in the healthcare domain.

5.2.1 Requirements Analysis and Application Development

In the very first step of our investigation we focused on the analysis of the existing m-health solutions and the open issues that may prevent them to get consensus from real users. Subsequently, to better identify representative groups of all the involved stakeholders and understand how technology is currently supporting diabetic people, we carried out an initial inquiry at the Diabetes care centre of Salerno. With the help of the physicians, we identified a group of 40 people. In particular, 6 specialized physicians, 24 patients and 10 informal caregivers (family members) were selected. Among the patients, 8 were young people aged between 14 and 23, 6 were working adults and 10 were retired.

A combination of informal interviews and direct observations allowed us to obtain a better understanding of the care processes and management activities as they occur in real healthcare settings. As for clinicians point of view, a low level of interoperability among actors represented the main drawback of existing adopted procedure. Such issue makes the availability of desired operations difficult, such as real time access to patients data in case of first aid interventions.

From patients point of view, although the expectations about the ability of mobile technology to improve their quality of life is high, the main perceived limitation of existing solution was a lacking support to personalization of care management. The personalization should be done by taking into account the patients profile as well as the specific characteristics of their geographical context. The insight gained from the combination of literature

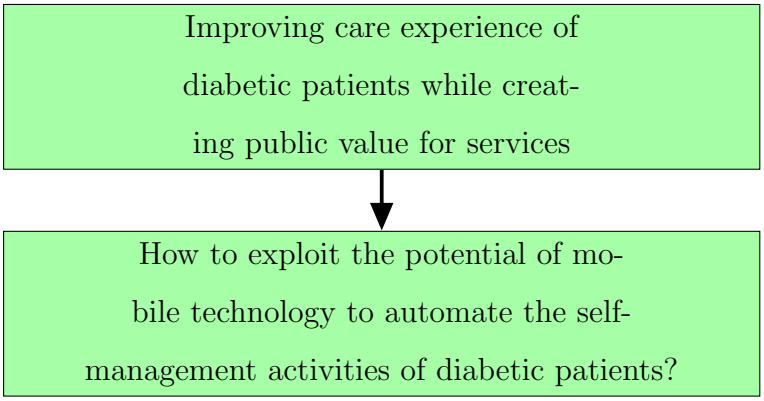

Figure 5.8: Initial steps of the DSR process

review and the field study helped us clearly identify the general problem and define the main objectives of a possible solution. These first two steps of the DSR process are shown in Figure 5.8

To validate the preliminary requirements an initial mobile prototype was developed. The first set of provided functionality covered the basic set of self-care treatments, namely the insulin dosage and ingestion and the calorie count. At this stage of the design lifecycle, the goal was to obtain a first feedback from the users to identify the most confusing or difficult functionality. However, in addition to the feedback about mobile UI, the first DSR iteration provided us with additional valuable information. In particular, the need of a deeper integration of contextual information automatically gathered by using the various sensors available on a modern smartphone with the user-entered data emerged as a fundamental requirement to overcome the limitations of the existing solutions. The benefits of such a combination are twofold.

On one side, a better exchange of contextual information among actors belonging to the same geographic area can dramatically improve several as-

pects of a given territory. For the healthcare domain, in particular, the collected data about patients health condition could be shared with the local physician or the local healthcare institution, thus improving the organization of services and the management of available resources. On the other side, contextual information can help automate a series of repetitive tasks, such as the survey of fitness activity. Adding the support to seamlessly manage, exploit and exchange contextual information led to a deep redesign of the mobile prototype. The first step was the development of a completely new module, the Metadata Collection Framework, responsible to aggregate and manage metadata either directly generated by user-performed activities or coming from the various sensors usually available on a mobile device. Afterward, the functionality offered by the Metadata Collection Framework were used to automate, as much as possible and without the use of CPU-intensive methods, most of the daily activities of a diabetic patient. We developed several additional scenarios to understand the best way to enrich the existing functionality with the collected metadata and address the related changes in the UI design. An example scenario showing how collected metadata can be used to automate the management of fitness activity is provided in Table 5.13. Figure 5.9 and Figure 5.10 instead, show an excerpt of the mobile prototype UI.

The last iteration described here concerns the assessment of the functional requirements elicited in the previous cycles according to the software quality model described by the ISO9126 standard (ISO/IEC 9126-1, 2001).

The ISO9126 standard defines a set of 6 main quality categories that can be used in a quality assurance process, namely Functionality, Reliability, Usability, Efficiency, Maintainability, Portability. As for our investigations, among the quality attributes defined in the standard, we focused on the

Figure 5.9: The mobile prototype UI for the healthcare domain

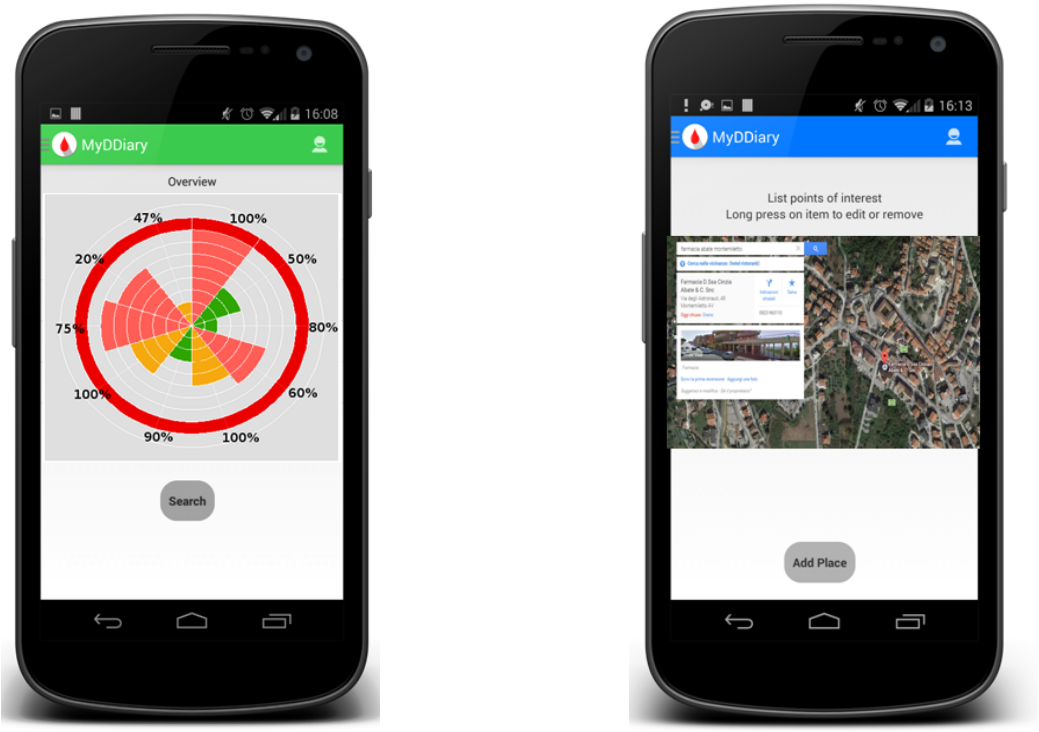

Figure 5.10: The mobile prototype UI for the healthcare domain

Table 5.13: Example scenario of fitness practice

Persona
John is a 30 year old diabetic patient with a family history of the disease.
John has a degree in economics and works as a financial consultant.
He loves sport and is well aware that healthy lifestyles, healthy eating and consistent exercises can dramatically help to deal with his disease.
In addition John is passionate about technology and firmly believes that its advancements can simplify several aspects of people daily activities.
Scenario
John has set on the calendar of his mobile device a recurring fitness plan: Gym from 19.00 to 20.00 from Monday to Wednesday and running on Thursday and Friday.
On the corresponding calendar entry such activities are marked as *Gym* and *Run*. John decides to use the mobile application to monitor his fitness activity and compute the number of times he missed the scheduled training during the last two months. Moreover, he needs to obtain an immediate overview that compares the fitness activity, calorie counts and glycemic values of the last week.
The application checks that the gym activity has been actually performed by verifying whether user's position and the gym location were inside a reasonable convex hull for a certain temporal range. The run activity is instead checked by calculating the speed value within a time range of a calendar entry taken from the personal planning, on the basis of John's position stored as metadata.

security of exchanged information.

When developing an information system dealing with sensitive data, in addition to offer traditional low-level security mechanisms, such as encrypted connections, it is also important to make users aware about the type of information being exchanged and provide clear options to let them choose what to share.

In our specific case, the user evaluation of the refined mobile prototype showed how providing a fine grained control about the information sharing was an element of paramount importance for the social and practical accept-

ability of such a type of applications. The users' main concern was related to the fact that all the information caught by patient's smartphone (e.g., personal calls) could be transmitted to the back end modules. Therefore, we further refined our prototype by providing clear options to control the data that can be actually shared and sent to the back end modules. Among the explicit users' requests we can mention the ability to send their current location only when they are in certain locations, like hospitals and clinics, the ability to share only the phone calls metadata that match some fixed keywords, such as *Doctor's phone number*, and the ability to share only the multimedia files directly captures by the application.

Figure 5.11 shows the device screen for information sharing settings.

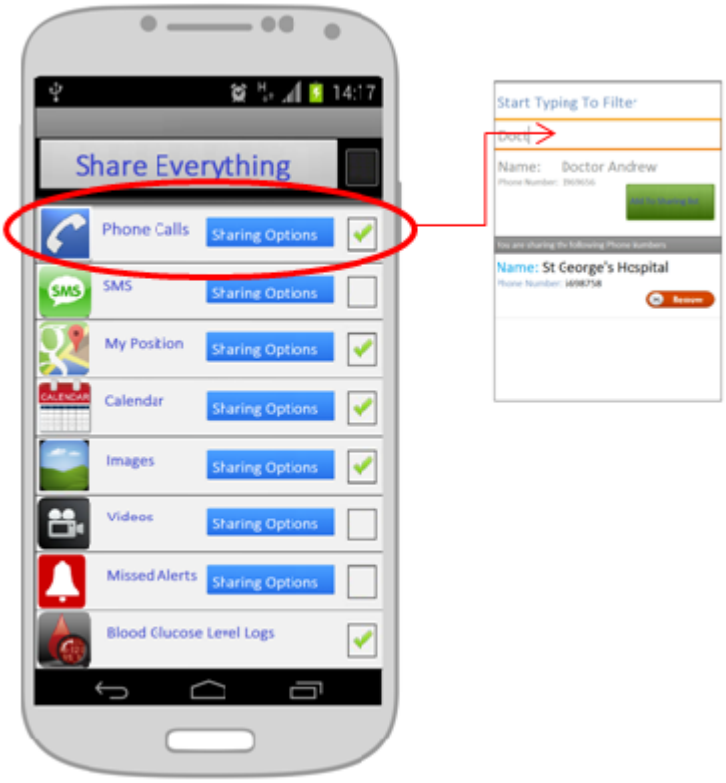

Figure 5.11: Handling privacy issues with the developed application

5.3 An Empirical Evaluation of our Changes to the WSIT Metadata Exchange module

In this Section we firstly discuss the steps performed to verify the correct operation of the changes we made to the client and server packages of the WSIT Metadata Exchange module. A brief theoretical overview about the performance impact of a SOAP-based exchange of geospatial data is also provided.

As explained in Chapter 4, our modifications to the WSIT WS-Metadata implementation are entirely contained into the original *client* and *server* packages. Therefore, the only requirement for their exploitation in a Java-based web application or service is the configuration of the targeted Metro-enabled application server with the modified version of the service stack instead of the original one.

Starting from this assumption, the assessment of the correct operation of the modified APIs has been performed in two different scenarios of use. In the first one we used, as OGC metadata source, a streamlined version of one of the Web Feature Services developed for the SLN4MoP back end. Subsequently, a surrounding wrapper that allows W3C compliant solutions to retrieve the WFS metadata was developed. Finally, the correctness of the retrieval phase was verified by developing a W3C service that queries the wrapper and displays the canonical tree structure of the Capabilities document. The main aim of this first test case was to assess the correct API support for the execution of the following sequence of operations:

1. A SOAP-enabled client sends a GET_METADATA_REQUEST to the wrapper to obtain the whole metadata set for an intended WFS.

2. To serve such a type of request the wrapper incorporates the Capabili-

ties document of the target WFS into a SOAP-based metadata response message and sends it to the requesting client.

3. On receiving the SOAP response, the requesting client can further exploit the classes and methods provided in the *client* package to extract the payload containing the WFS Capabilities document.

Listing 5.1 shows an excerpt of the client source code that deals with the GET_METADATA_REQUEST dispatch, the received payload extraction and the printing of the tree structure on the standard output.

Listing 5.1: Obtaining and exploiting OGC metadata

```
1  import com.sun.xml.stream.buffer.MutableXMLStreamBuffer;
2  import com.sun.xml.ws.mex.client.MetadataClient;
3  import com.sun.xml.ws.mex.client.PortInfo;
4  import com.sun.xml.ws.mex.client.schema.Metadata;
5  import com.sun.xml.ws.mex.client.schema.MetadataSection;
6
7  /..... Other Code ...../
8
9  MetadataClient client = new MetadataClient();
10 Metadata metadataReceiver = client.retrieveMetadataCapabilities
11    ("http://localhost:8081/MEXServer/server_wfs");
12 List<MetadataSection> metadataSection =
      metadataReceiver.getMetadataSection();
13 MetadataSection section = metadataSection.get(0);
```

As we can see, from line 9 to 13, by exploiting the new functionalities of the WSIT APIs, obtaining OGC metadata through the use of the WS-

Metadata protocol becomes a relatively simple task.

Listing 5.2, instead, shows an excerpt of the WFS source code that deals with the Capabilities document creation process in response to a GetCapabilities request. The core Objects of this code snippet are the JAXBContext, the ObjectFactory, the WFSCapabilitiesType and the Marshaller classes. The role of such classes has been already discussed in Section 4.3.4 of Chapter 4.

Listing 5.2: Excerpt of the steps to create the WFS Capabilities Document

```
1  ObjectFactory wfsFactory;
2        if (class instanceof WFSCapabilitiesType) {
3               file = new
                    FileOutputStream("GetCapabilitiesResponse.xml");
4               JAXBContext context =
                    JAXBContext.newInstance("net.opengis.wfs.v_1_1_0");
5               Marshaller jaxbMarshaller =
                    context.createMarshaller();
6               wfsFactory = new ObjectFactory();
7               WFSCapabilitiesType wfsclass =
                    (WFSCapabilitiesType)class;
8               JAXBElement<WFSCapabilitiesType> wfsCapabilities =
                    wfsFactory.createWFSCapabilities(wfsclass);
9               jaxbMarshaller.setProperty
10                (Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
11              jaxbMarshaller.marshal(wfsCapabilities, file);
12    /.... Other Code ..../
```

It is worth noting that, as for the actual retrieval of the Capabilities document from the remote WFS, two main options are available. In the

simplest way, the wrapper invokes the GetCapabilities operation every time a GET_METADATA_REQUEST is received. Alternatively, the wrapper can save a copy of the Capabilities document in a local cache. This step can be performed to minimize the total request-response overhead. In this second case, on receiving a GET_METADATA_REQUEST the following steps are performed:

1. Using a synchronized hash map, the wrapper checks whether a copy of the Capabilities document is available in a local cache. This initial step is performed to minimize, as much as possible, the total request-response overhead.

2. In case a cached copy is not available or the cache validation mechanism states that the stored information is out-dated, a standard GetCapabilities request is sent to the original OGC service.

3. The retrieved document is embedded into a <Metadata> element and a SOAP response is sent to the requesting client.

In our implementation, the second option was chosen.

The main aim of the second testing scenario, instead, was to verify the correct encapsulation of metadata coming from third parties WFS. For this purpose we used the OGC services provided by the National Snow & Ice Data Center of the University of Colorado (http://nsidc.org/) and those provided by the Piedmont region, in Italy (http://www.geoportale.piemonte.it/). In addition, to simulate a request from W3C clients developed using different technologies we manually created an ad-hoc SOAP message containing a GET_METADATA_REQUEST (Listing 5.3) and embedded it into an HTTP POST request (Listing 5.4).

Listing 5.3: Creating a SOAP message with a `GetMetadata` request

```
1  import static
       com.sun.xml.ws.mex.MetadataConstants.GET_MDATA_REQUEST;
2  import static com.sun.xml.ws.mex.MetadataConstants.SOAP_1_1;
3  import static com.sun.xml.ws.mex.MetadataConstants.WSA_ANON;
4  import static com.sun.xml.ws.mex.MetadataConstants.WSA_PREFIX;
5
6  String request = "<" + soapPrefix + ":Envelope " +
7            "xmlns:" + soapPrefix + "='" + soapNamespace + "' " +
8            "xmlns:" + WSA_PREFIX + "='" +
                AddressingVersion.W3C.nsUri + "'>" +
9            "<" + soapPrefix + ":Header>" +
10           "<" + WSA_PREFIX + ":Action>" +
11           GET_MDATA_REQUEST +
12           "</" + WSA_PREFIX + ":Action>" +
13           "<" + WSA_PREFIX + ":To>" + address + "</" + WSA_PREFIX
                + ":To>" +
14           "<" + WSA_PREFIX + ":ReplyTo><" + WSA_PREFIX +
                ":Address>" +
15           WSA_ANON +
16           "</" + WSA_PREFIX + ":Address></" + WSA_PREFIX +
                ":ReplyTo>" +
17           "<" + WSA_PREFIX + ":MessageID>" +
18           "uuid:778b135f-3fdf-44b2-b53e-ebaab7441e40" +
19           "</" + WSA_PREFIX + ":MessageID>" +
20           "</" + soapPrefix + ":Header>" +
21           "<" + soapPrefix + ":Body/>" +
22           "</" + soapPrefix + ":Envelope>";
```

Listing 5.4: Embedding the SOAP message into a HTTP POST request

```
1 HttpURLConnection connection = (HttpURLConnection)
    url.openConnection();
2       connection.setDoOutput(true);
3        connection.setRequestMethod("POST");
4        connection.setRequestProperty("Content-type", "text/xml;
            charset=\"utf-8\"");
5        connection.setRequestProperty
6        ("SOAPAction",
            "\"http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Request\"");
7        connection.setRequestProperty("Host", "127.0.0.1:8081");
8        connection.setRequestProperty("Content-Length", "text/xml;
            charset=\"utf-8\"");
```

Finally, to verify the correctness of the wrapper's response, we exploited the well-known TCPMon utility that allows to view the actual content of the messages exchanged over an HTTP connection. Figure 5.12 and Figure 5.13 show how the wrapper successfully embedded the Capability document into a GET_METADATA_RESPONSE message for both our test sources.

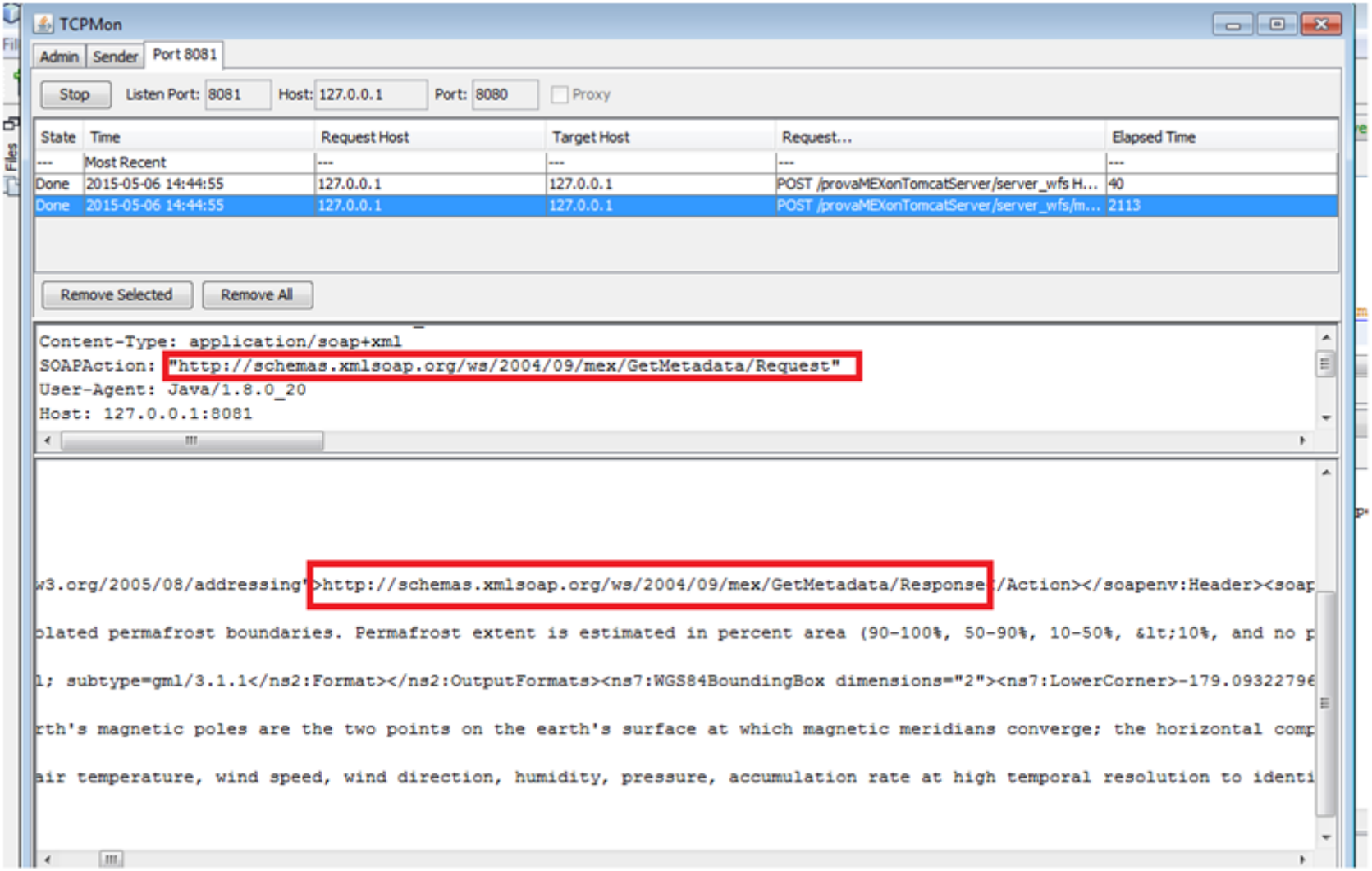

Figure 5.12: Embedding the Capabilities Document for the National Snow & Ice Data Center into a `GetMetadata` request

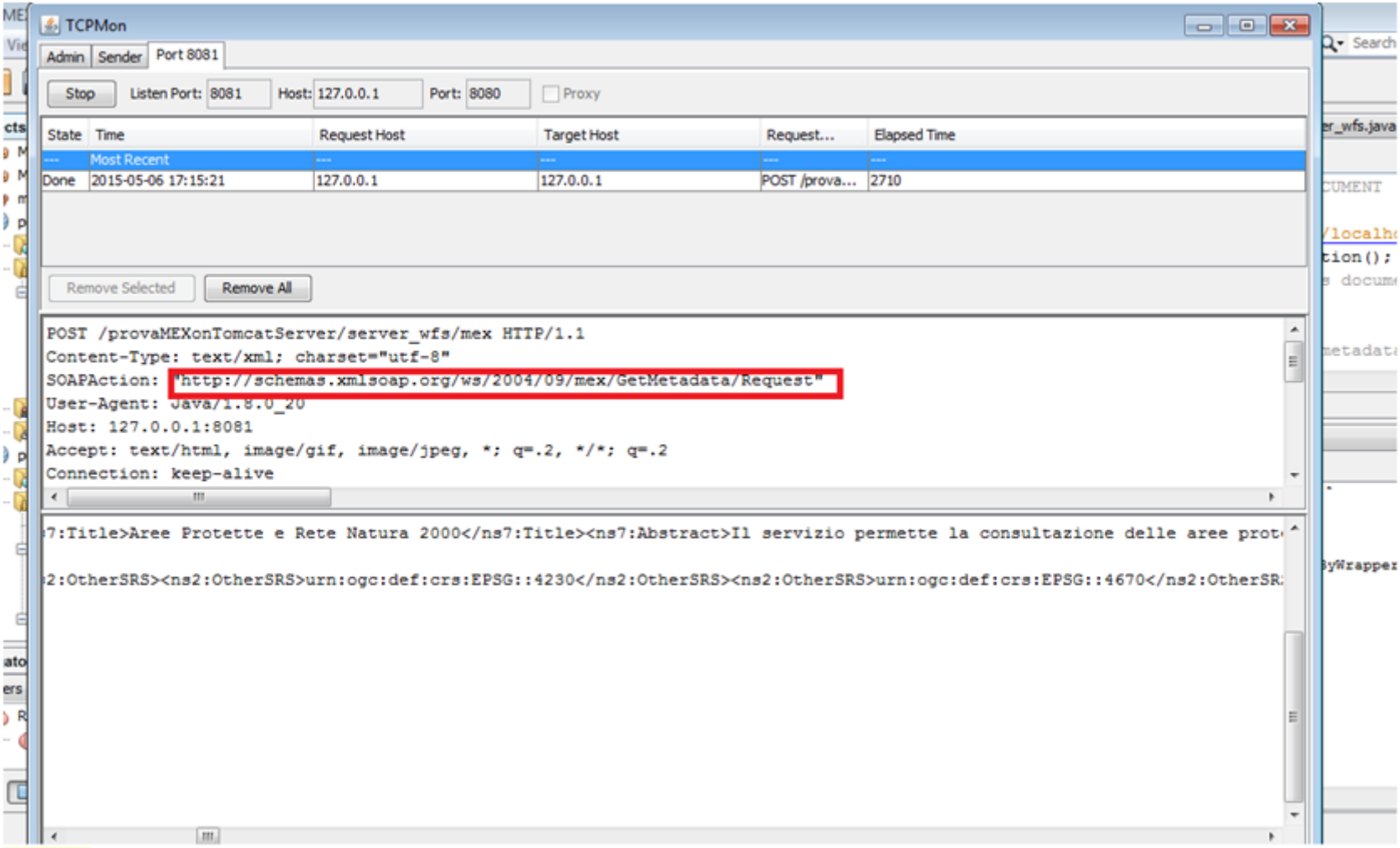

Figure 5.13: Embedding the Capabilities Document for the Piedmont region into a `GetMetadata` request

In the previous two pictures, the *elapsed* time values are related to the wrapper first execution (i.e., when a cached version is not available yet and the Capabilities document is retrieved from the original source). In addition to the network characteristics (e.g., latency time or congestion), a significant part of the entire retrieval time is due to the relevant size of a Capabilities document as compared to the size of traditional SOAP messages. In general, the impact that the specific characteristics of geospatial data have on the wrapper performance cannot be underestimated in a real scenario of use.

The assessment of our proposed approach shows that it is a viable solution to the composition of OGC and W3C services. However, a comprehensive evaluation of the performance of this solution was beyond the scope of this thesis. Indeed the full implementation and integration of the wrapper within the SLN4MoP architecture is being developed by other members of the project research team. Nevertheless, based on the aforementioned preliminary results and existing literature, some general considerations can be deduced.

In a wrapper-based solution geospatial data has to be packaged in the Body element of a SOAP message. Such a task involves the SOAP encoding, decoding and transmission operations and has a direct impact on measurable values like response time or throughput. In addition, it might also influence the behaviour of other aspects (e.g., transaction management or services orchestration) of the entire SOA to which the wrapper belongs. SOAP performance in complex and large-scale distributed application is well-discussed topic in literature. A quite complete description of the SOAP performance issues and a review of the research efforts aimed at SOAP performance enhancement can be found in (Tekli et al., 2012). According to the authors, the performance metrics of service-oriented environments can be grouped into

three main categories: the response time, the throughput and the network traffic. The response time (latency time) can be seen as the time perceived by a client to get a response; the throughput can be seen as the number of requests fulfilled in unit of time, and the network traffic is the total size of messages exchanged during the whole communication. By taking these metrics into account, the use of SOAP to transfer large amounts of geospatial data exhibits several problems mainly due to the XML message encoding and decoding, the verbosity of XML and its redundant textual characteristics. First of all, the latency and the network traffic produced by SOAP are considerably higher compared to technologies like CORBA or Java RMI. These issues can have an even greater performance impact when, for example, the client is a mobile device where the available bandwidth can be low and the latency very high. Moreover, the process of converting a memory object into an XML object and vice versa is a quite expensive computational task, able to consume over 90 percent of the entire end-to-end SOAP processing time. The performance problems are amplified also by difficulties encountered by traditional hardware architectures to simultaneously evaluate multiple conditions, which represents a central issue in XML string and character processing. Finally, the addition of security policies to SOAP messages adds another source of overhead. In fact, the use of the Web Services Security (WS-Security) protocol has a big impact on both the processing and response time and on SOAP message size. In this case, the problem is due to the need of providing for a message level security instead of the traditional channel-level security, where Secure Sockets Layer/Transport Layer Security (SSL/TLS) on the HTTP protocol is used. An empirical proof of the performance problems occurring when using W3C service technologies can be found in (Zhang et al., 2007). In this paper, a prototype system built to

evaluate the performance of composition and invocation of geospatial functionalities by using Web services is discussed. The experimental results agree with the above discussion and show that the trade-off between convenience and overheads are acceptable for small volumes of geospatial data, although the large response time for ample data volumes represents a problem that cannot be underestimated.

Chapter 6

Conclusions

6.1 Thesis Summary

Advances in Information and Communication Technologies enable the development of new software applications, including for mobile devices, to meet a range of requirements. In this context, the problem addressed in this thesis is related to the design of innovative Mobile Information Systems aimed at supporting people in their daily tasks. The peculiar characteristics and computational capabilities of mobile terminals influence not only the way users exploit the provided information but have also a profound impact on several technical aspects related to both the front and the back ends of the proposed solutions.

In this thesis, by using as case study the design of a real Mobile IS to support farming activities in Sri Lanka, we investigated the issues related to the elicitation of requirements, the design of usable mobile UIs and the information exchange among heterogeneous services. The research contributions can be divided into three main parts. Firstly we provided a detailed preliminary background about the areas of interest for our investigation. We examined

the main challenges that must be taken into account during the design phase of a mobile User Interface and the fundamental protocols, technologies and design challenges for the fulfilment of Service Oriented Computing. Subsequently we focused on the critical aspect of user requirements elicitation when dealing with new types of problems and its influence on the design of usable mobile UIs. When developing an information system for a new class of problems, the ad-hoc nature of the involved business processes and user unawareness about the nature of the potential ICT-based solution, makes gathering requirements a difficult task when compared to the development or enhancements of an IS for well-defined processes.

Another major challenge is represented by the proper identification of usability requirements in all those contexts where the developed system should not only meet the functional requirements but also be easy to use by the intended user. To successfully overcome such issues we presented a methodology for incorporating both functional and usability requirements since the early stages of the development process. In particular, we addressed the above-mentioned challenges by blending several techniques in Software Engineering and Human Computer Interaction within a Design Science Research framework. These techniques include traditional surveys and interviews, scenario creation and transformation, use of paper-based and functional prototypes for communicating with users and capturing their feedback, user centered design and incremental development. This approach enabled us to better capture requirements based on usability aspects and guided us to design effective UIs that consider the specific needs of users with little or no experience and facilitate the effective use of the various services available on a mobile device, two factors of the utmost importance for the purposes of our research project. In addition, to demonstrate the wider applicability of

our framework we discussed an example of its application to enhance the usability of healthcare software application self care management of diabetic patients.

The third part of our investigation, instead, concerned the development of advanced service-based infrastructures for mobile solutions. In particular, we focused on solutions for a seamlessly integration of OGC and W3C services. Although the standards that govern their development are based on XML for data exchange and HTTP as transport protocol, some design choices make OGC and W3C services totally incompatible. In light of these important differences, one of the few available options to make W3C and OGC services communicate seamlessly consists of mapping one set of standards to the other at both the syntactic and semantic levels. Although the mapping can either be from W3C to OGC standards or vice versa, the best option is the adaptation of the OGC standards to the W3C ones. Unfortunately such a mapping presents a non-trivial technical complexity. The currently most accepted solution to this aim is represented by the development of a software wrapper, usually a service itself, that translates the requests and responses messages from the W3C services format to a format suitable for the OGC services and vice-versa. The proper management of the fundamental geospatial metadata in a W3C standards-oriented infrastructure represents a key factor during the concrete development of a wrapper. Providing geospatial metadata in a W3C-compliant way is a major challenge in order to support a better interoperability between the two proposals.

Our proposal extends the W3C Web Services Metadata Exchange specification by adding the support for the retrieval of OGC metadata. The main aim of the protocol is to define an XML-based standardized format for encapsulating W3C metadata. In particular, the GetMetadata tag defines an

additional element named Dialect that specifies the format and the version of the involved Metadata. To allow the exchange of OGC metadata using this W3C specification, we added a new set of combinations of values for the Type and Identifier attributes of the Dialect Element. One of the main advantages of our proposal is its total transparency for those services that do not deal with geospatial information and the fact that it does not modify the behaviour and semantics of the WS-Metadata standard. Finally, to show how our contribution can be effectively exploitable in real development contexts, we discussed how the Web services stack of the Java Enterprise Edition platform can be extended to seamlessly support the proposed extensions.

6.2 Limitations of the proposed approach and Future Directions

This Section discusses some limitations of the work presented in this thesis and suggests possible future research directions to overcome them.

The design of usable mobile applications to support users facing emerging problems and the interoperability issues between OGC and W3C services were two of the main topics of our research investigation.

For what concerns the first aspect, we proposed a design methodology that enabled us to derive detailed requirement needs of the user and deal with fundamental usability aspects during the entire design process. We successfully applied such a methodology to develop a real mobile application in a context where factors like the ad-hoc nature of the business process, lack of users exposure to ICT and users unawareness to system requirements, made the identification of the solution a hard task. However, our problem-solving approach strongly relies on the development and continuous assessment of

a concrete artifact. The evaluation of its efficacy to solve a given problem using methods such as experiments, case studies and proofs is of central importance to demonstrate the validity of the proposed solution. To this aim in the future we plan to further validate our results about the design choices made during the development of our prototype for Sri Lankan farmers. As described in Chapter 5, the results of the first field trials show a high appreciation about the design choices for the proposed prototype. However, we intend to perform more detailed studies with a larger sample of users to further validate our findings and insights related to the developed UI. In addition, our long-term purpose is to extend the pilot project to involve also farmers working into developed countries such as Italy in order to offer a complete platform to support everyday activities. Moreover, the differences about the surrounding technological, social and cultural contexts will constitute a different test bed to assess the effectiveness of our design choices.

Another limitation of the proposed approach concerns the necessity for the involved team to have a wide range of expertise in a consistent number of design techniques. In the SLN4MoP project, through the voluntary knowledge sharing occurred across a large research team spread over four continents we had access to required expertise. In the absence of funds, this is an unrealistic attempt if the researchers are not voluntary sharing time and resources. Moreover, in the absence of an active users involvement, the method would also fail if they were not willing to share experiences or to spend their valuable time in evaluations.

For what concerns our contribution to communication among services developed according to different standards, the development of a wrapper has proven a viable solution to overcome the syntactic and semantic dissimilarities between OGC and W3C services. In particular our approach allows the

achievement of one of the key tasks that such a type of software solution should perform, namely the provision of geospatial metadata in a W3C-compliant way. However a wrapper performs usually a one-to-one mapping with an intended OGC service i.e., the translation of request and response messages is explicitly tailored on the specific characteristics of the wrapped service. This strongly limits the capability to easily adapt a solution designed to deal with a certain OGC service to another one. To overcome such a limitation and minimize the required changes, it is necessary to further investigate and enhance two distinguishing features of a wrapper, namely the retrieval phase of a Capabilities Document and the description of the public interface using the WSDL standard. Based on these considerations, our future investigation will focus mainly on improving the wrapper flexibility and reusability. The retrieval of geospatial metadata represents the first operation that a wrapper should perform. As we have discussed, it is possible to exploit a modified version of the WS-Metadata protocol to successfully retrieve the entire capabilities of an OGC service. However, the OGC Common Standard that strictly regulates the implementation of the GetCapabilities operation envisages the ability for an OGC client to request only certain sections of the service metadata document. Although this is an optional functionality, providing support for such a feature would further increase the flexibility of the WS-Metadata protocol thus simplifying the development of a one-to-many wrapper. In relation to the second key characteristic of a wrapper, namely the description of the public interface using the WSDL standard, it is necessary to observe that even though the public interface of a generic OGC service has been strictly standardized by the Consortium, the significant design difference makes the definition of a single WSDL impossible to reuse for every OGC service. Although a general solution to this issue does not seem

possible, in the existing literature several guidelines to tackle this issue have been provided. Such guidelines can be exploited to derive the best solution according to the specific use case. Among them, a well-accepted proposal consists of individually exposing and mapping each available data layer of an OGC service into separate WSDL documents. In this way each wrapper, exposing only a single layer, can replicate the interface originally defined by the OGC for that kind of geospatial service. This is also the approach that we have adopted for the specific purposes of the SLN4MoP project. However, two important drawbacks require further investigation. First of all this solution seems to be applicable only to a small set of operations. Moreover the efficient mapping of the data types for the various layers of an OGC service inside the *Types* element of a WSDL (the container of the data-type definitions used inside a Web service) represents another issue that needs to be investigated. This forms part of our planned future work.

Appendix A

SOC related Protocols and Standards

The Hyper Text Transfer Protocol

The Hyper Text Transfer Protocol (Fielding et al., 1999) represents the manner through which information can be transmitted in the World Wide Web (WWW). It adopts a typical client/server mechanism: a client sends a message containing a request and a server returns a message containing the response. Three main parts that constitute the typical structure of a HTTP request message are the Request line, the Header section and the Message Body. The request line is formed by an HTTP method, the Uniform Resource Identifier (URI) and the protocol version. The methods in HTTP 1.1 can be one among GET, POST, HEAD, PUT, DELETE, TRACE, OPTIONS, CONNECT, while the URI indicates the subject of the request (for example, the desired web page). The Header section carries additional information, such as the client browser version and the server information. Two of the most commonly used HTTP methods are GET and POST. The former is used to get the contents of the URI named resource (such as the content of

an HTML page), while the latter is normally used to send information to the server (for example, form data). In this case, the URI indicates the sending content and the body contains the effective content.

Also the typical structure of a HTTP response message consists of three parts: the Status line, the Section header and the Body. The status line is the first line of a response message and shows a three-digit code, where the first digit of the code identifies the response type (typical examples are the 2xx codes indicating a success and the 4xx codes indicating a Client error). In an HTTP response message, an important header is the Content-Type header that indicates the type of the returned content. The Internet Assigned Number Authority (IANA) manages the encoding of these types, called MIME (Multimedia Internet Mail Extensions). Some common MIME types in an HTTP response are the text / xml for XML documents and the *image / jpeg* for images using the JPEG format.

The Extensible Markup Language

The Extensible Markup Language has been designed *"to bring structured information to the Web, allowing users to define their own set of markup tags relating to the content of their documents, thus delivering both extensibility and potential for validation*(Zisman, 2000). XML thus provides the basic rules for creating markup languages whose content (in textual form) is structured using special delimiters known as tags. XML is completely text-based, platform agnostic and human readable. One of the main goals of XML is a clear separation between the actual data and the presentation of the same (unlike what happens, for example, in HTML). Moreover, being entirely text-based, XML data can be transported, using HTTP, also through corporate firewalls. An XML document consists of XML elements. A start tag and

an end tag delimit each element. The information between the opening and the closing tags, if present, is called the content of the element. The tags in an XML document indicate the data meaning and not the date appearance. In general, an XML element can contain: attributes (that describe additional information about the element), text, other elements or a mix of them. Namespaces are one of the key ideas behind XML strong interoperability. At their simplest, namespaces *"are a way of grouping XML elements and attributes under a common heading in order to differentiate them from similarly named items"* (Fawcett et al., 2012). Therefore they can be seen, for example, as the equivalent of the packages in the Java programming language and, hence, are a useful method to avoid element name conflicts during the common task of mixing different XML documents from different sources. Another problem, during the processing of XML documents, concerns the need to know whether the structure of a document is valid or if all its items follow the original structure intended by the author of the document. This is what XML Schema is used for. The basic idea behind XML Schemas is to describe the legitimate format that an XML document can have and what types of data are allowed. XML Schema also details whether an element can be empty or can contain text or other elements and specifies whether there are fixed or default values for elements or attributes. Moreover, an XML Schema specifies also Data Types for elements and attributes.

SOAP

SOAP is *"a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing in-*

stances of application-defined data types, and a convention for representing remote procedure calls and responses" (Gudgin et al., 2001).

Its main design goals are simplicity and extensibility and, although it can be potentially used in conjunction with any type of transport protocol, currently the only widely used binding is usually the HTTP protocol. In general, SOAP allows different platforms to communicate with each other allowing, for example, Web services implemented with Java to seamlessly communicate with Web services implemented with Microsoft .Net. Being completely XML-based, it heavily relies on XML standards like XML Schema and XML namespaces for data type definitions and messages syntax. SOAP defines four different namespaces and an application generating SOAP messages should always include, for all elements and attributes, the proper SOAP namespaces. Moreover, a SOAP application must be able to use such namespaces during the phases of messages processing and must discard messages with incorrect namespaces. Three basic components characterize a typical SOAP message: an Envelope, a Header and a Body. With the term *SOAP message*, we refer comprehensively to these three elements Figure A.1.

The Envelope is the top-level element of a SOAP message and can be seen as the container of the message itself. An Envelope can contain an optional Header field but has to contain exactly one Body field. A Header, if present, must necessarily appear before the Body.

The Body contains the actual information of the SOAP message intended for the ultimate recipient of the communication. The Body subject can be any well-formed XML content, provided that such content has a namespace and does not contain any processing instruction or Document Type Definition (DTD) reference (Erl, 2005). A SOAP fault is a special type of message used to communicate information about errors that may have occurred during the

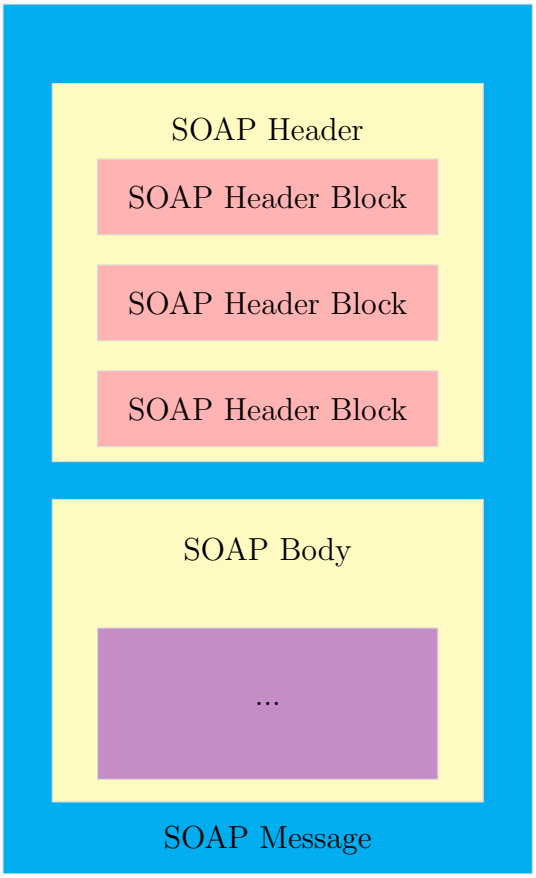

Figure A.1: The general structure of a SOAP message. Source: (`www.w3.org`)

processing of a SOAP message.

When present in a SOAP message, the Header field contains important information about how the message should be processed and provides a generic mechanism for adding features (such as security, transaction support, etc.) in a decentralized manner. It also provides options to indicate whether the information contained in the header is optional or mandatory. Each element in the Header field is called Header block and the namespace of these blocks may also be different from the SOAP namespaces making the general structure of a SOAP Header quite flexible. Such a flexibility and extensibility are the basis of many important features present in contemporary (Web services-based) SOA implementations. In fact, a key feature of the SOAP framework is its emphasis on creating messages that are as self sufficient as possible, a characteristic of the utmost importance in a totally loosely coupled environment such as Web services (Erl, 2005). This independence of messages is achieved by adding the additional information directly in the Header blocks thus avoiding services to store message specific logic. Moreover, since the information is fully contained in the message itself, the parts that communicate with each other do not need a prior agreement. The use of Header blocks has elevated the Web services framework to an extensible and modular enterprise-level computing platform (Erl, 2005). SOAP Intermediaries represent another important aspect related to the SOAP Headers. A SOAP message may move, in fact, from the original sender to the final recipient through a series of nodes called SOAP Intermediaries. Intermediaries are software systems capable either to receive or to forward a SOAP message, and are usually used to provide value added services or to support the scalability of a distributed environment. SOAP provides precise Header blocks for the management of intermediaries that have been taken into account since

the early stages of the protocol design. The set of a SOAP sender, the zero or more SOAP Intermediaries and the ultimate SOAP receiver is known as SOAP message path.

SOAP messages are fundamentally one-way transmissions from a sender to a receiver but (with additional information integrated in the Header blocks) such messages are often combined to implement more complex communication patterns, either synchronous (e.g., Request / Response) or asynchronous (e.g., Fire and Forget).

As for data encoding of the Body content, SOAP supports two main styles: SOAP Remote Procedure Call (RPC) Style and SOAP Document Style. RPC is a style that offers the greatest simplicity and the idea behind it is quite simple: a function available on a remote machine is invoked as if it was a local function and the parameters it needs are sent through the network. The return value of the function is usually the expected result of the computation. SOAP RPC exactly replicates this behaviour: the functionality of a Web service is invoked by a SOAP message containing in its body the parameters needed by the remote method. In a SOAP Document Style, instead, the body content sent to a remote machine contains an entire XML document without even requiring a return value. When compared to RPC style, Document style has several advantages. In the RPC messaging any changes in the signature of the remote function involve changes in all clients that use such a function; in contrast Document style rules are less restrictive. Moreover, RPC style can also require higher parsing times because of the necessity of having, every time, to perform the marshalling of parameters (McCarthy, 2002). Document style also promotes loose coupling between producer and consumer messages and the majority of current SOA solutions use this solution.

Finally, several data types can be used in a SOAP message. SOAP Encoding is an extension of the SOAP framework and defines how a data value should be encoded. The use of an XML Schema document defining the exact data type of a particular element is a possible choice but, since SOAP Envelopes are designed to carry in their Body field arbitrary XML documents it is now common practice to refer also to other types of Schema documents to define the data type of a particular element (Snell et al., 2001).

The Web Services Description Language

The Web Services Description Language (WSDL) is *"an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate"* (Christensen et al., 2001).

In other words, WSDL is an XML-based language for describing Web services and how to access them. It uses XML Schema for the definition of the type system and for the definition of SOAP messages (although their use is not compulsory) and, furthermore, separates the abstract aspects of a service description from the concrete aspects such as the binding with a certain network protocol. Essentially, a WSDL description contains the three fundamental properties of a Web service (What it does, How is accessed, Where is located) using the following elements: Types, Message, Operation, Port Type, Binding, Port, Service Figure A.2.

The former four elements refer to the Abstract Description of a Web service while the latter three refer to its Concrete Description. The Abstract Description specifies the interface characteristics of a Web service with no references to any specific technology platform, while Concrete Description allows to connect the abstract interface of a Web service to a real technology and to transport protocol. One of the main advantages of this distinction is that the *public interface* of the Web service can be preserved by changes in the underlying technology.

WSDL Abstract and Concrete Descriptions

As mentioned before, the Types, Message, Operation and Port Type elements make the Abstract Description of a Web service. The Types element can be seen as the container of the data type definitions used inside the Web service. XML Schema is the preferred data type system but WSDL allows also different notations. Inside the Types element it is possible to incorporate a whole XML Schema definition with either simple or complex types. The Message element represents the data being communicated. In a WSDL document, it comes after the Type element. Each WSDL document can have one or more Message elements. Each Message has a univocal name and contains one or more children referred to as *Parts*. The Parts can be compared to the parameters of a function in a traditional programming language. The data type of a part element can be a simple type or a type defined in the Types element. The Operation element describes, instead, the features that the Web service will expose. Each Operation element is composed of Input and Output elements, which refer to Messages exchanged during the communication.

WSDL defines four basic types of operations:

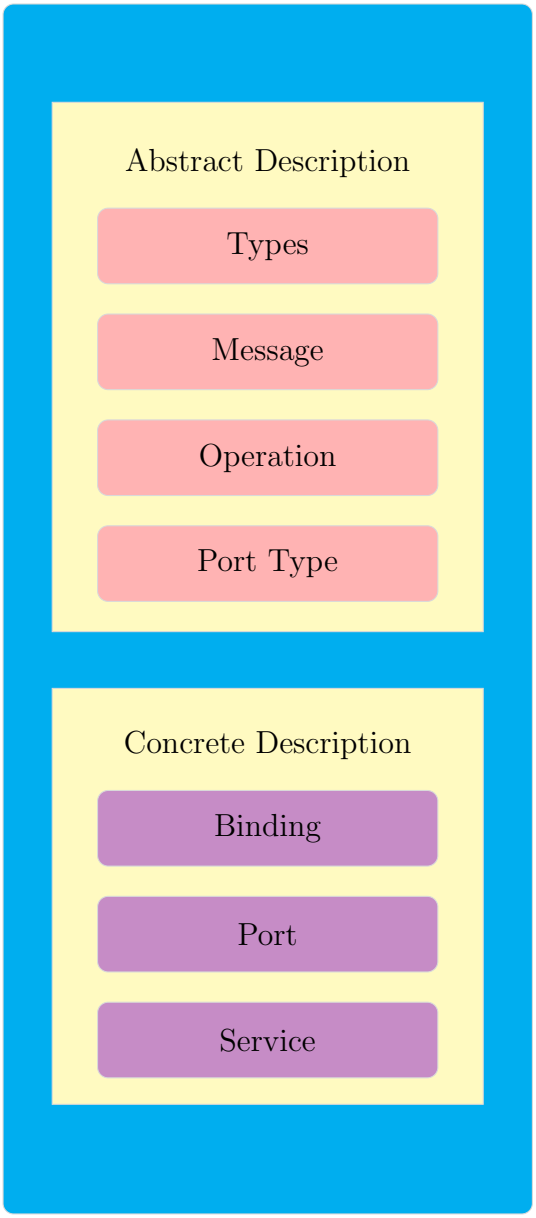

Figure A.2: The structure of a WSDL Document

- One-way: the client sends a message to the service.
- Request-Response: the client sends a message to the service and receives a response.
- Notification: the service sends a message to a client.
- Solicit-Response: the service sends a message to a client and receives a response.

Each of these operations is achieved using different combinations of Input and Output messages. The Port Type element is a named set of abstract Operations and the abstract Messages involved. Every Port Type has a unique name and is made of several Operations.

On the other side, the Binding, Port and Service elements describe how the Abstract Description is mapped into a concrete format. The Binding element, in particular, defines message format and protocols details for the Operations and Messages defined by a particular Port Type.

The WSDL specification allows using various types of bindings: SOAP, HTTP and MIME[1]. The Port element represents an instance of an abstract port (Port Type) obtained as a combination of a binding and a network address and, finally Service, the higher-level element in a WSDL document declares a Web service as a collection of related Ports.

The Geography Markup Language

Geography Markup Language (GML) is *"an XML grammar written in XML Schema for the description of application schemas as well as the transport and storage of geographic information"* (Portele, 2007). In other words, GML is

[1]Multimedia Internet Mail Extensions

an XML language to manage geospatial information. GML Schema describes the document while the instance document contains the actual data. The main goal of GML is to provide a means to exchange and manipulate geographic information in a standard, programming language and source format independent way. In general terms, we can split the role played by GML in the context of geospatial information into three main categories (Lake, 2005):

- An encoding standard for the transport of geographic information from one system to another.
- A storage format for geographic information.
- A modelling language for describing geographic information types.

With respect to the first two aspects GML can be considered as the OGC answer to the need of representing geospatial information in a standard manner in order to facilitate communication and data exchange among a wide variety of autonomous and distributed data sources contributing to the reduction of costs related to the management of spatial information. Like XML, GML represents geospatial information in a textual form, focusing on the content description and relying on other mechanisms for data visualization. Another advantage is that, being XML-based, people can immediately use the plethora of available XML tools to perform all sorts of common XML operations. Additionally, the OGC standard lets users decide whether to store geospatial information directly in GML or use some other storage format and convert it to GML only for data transportation purposes. For what concerns the third aspect above, the only key point is the definition of the feature concept. According to the ISO19101 Reference Model (ISO 19101-1, 2014), a feature *is an abstraction of real world phenomena*; if such abstraction is associated with an Earth location we talk of geographic feature. Features

are fundamental objects in GML (Burggraf, 2006) and are described as a list of properties (in the traditional form of names, types and values) and geometries (composed of basic geometry building block such as points, lines, curves, polygons etc.) In particular, in GML, a feature is represented by an XML element whose individual children elements describe a property. Furthermore a feature can be defined as the result of the composition of other features. Therefore, the use of GML in conjunction with OGC services allows the implementation of infrastructures for sharing geospatial information in a globally accessible manner, independently of the different proprietary formats used (Burggraf, 2006).

Web Map Services

A Web Map Service provides an HTTP-based interface for requesting geo-registered map images from one or more distributed geospatial databases (de La Beaujardiere, 2006). Basically, the requester defines elements such as the area of interest and the response consists of a map image, returned in binary format such as JPEG or PNG. Appropriate MIME types (e.g., *image/png*) usually accompany response objects. According to the standard, a generic WMS can belong to two different types of Basic or Queryable WMS. The basic WMS supports the GetCapabilities operation and the GetMap operation while a Queryable WMS also supports the GetFeatureInfo operation, whose scope is to provide more information about the features contained in the pictures of a map. Textual output (usually XML documents) is also available and can be used to provide errors description or responses to information requests about the features shown on a map. A WMS must support the HTTP GET method and may support the HTTP POST method. The WMS specification provides an example for another difference between an

OGC service and a W3C service. The former can return binary documents (as well as XML documents) while the latter usually relies on pure XML documents (although there are actually several ways to encode binary documents, such as images or PDF documents, in a SOAP message).

Web Feature Services

If a WMS gives users the possibility to retrieve maps from multiple sources, a Web Feature Service, instead, provides interfaces to access and manipulate the previously mentioned geographic features. The guidelines for implementing an OGC compliant WFS can be found in (Vretanos, 2005). According to the standard, besides the mandatory GetCapabilities operation, a WFS must also support operations allowing to Insert, Update, Delete or Discovery geographic features expressed in Geography Markup Language (GML). To accomplish these tasks, five operations are defined:

- DescribeFeatureType: An operation used to describe the structure of any feature that a WSF can service. The only mandatory output is a GML (presently GML version 3) application Schema. Such Schema describes the features encoding (either for input or output operations) expected by the WFS.

- GetFeature: A function used to request and retrieve feature instances. Moreover, a client must be able to specify, for each feature, the desired properties it wants to fetch. A WFS may respond to a Get Feature request in two ways. It can return either a complete document or simply a counter corresponding to the number of features that the GetFeature request would return. The optional resultType attribute in the request message is used to specify how a WFS should respond to a GetFeature request.

- GetGmlObject: A client specifies the identifier (ID) of a GML object and the WFS returns that object.

- Transaction: A transaction request is made of operations (create, update, delete) that may alter the state of one or more features.

- LockFeature: If this operation is supported, a client can ask a WFS to lock one or more instances of a feature type for the duration of a transaction. This operation is fundamental to ensure consistency in scenarios where a client modifies a feature and sends it back to the server (using the above mentioned Transaction operation). Without a locking mechanism there is no guarantee that, while a client is modifying a feature, another client is not allowed to fetch, update and store the same feature.

Depending on the supported functions, a WFS can belong to two major categories: Standard (or read only) or Transactional. Standard WFS must support the DescribeFeatureType and GetFeature operations while transactional WFS would implement the Transaction operation and optionally the GetGmlObject and LockFeature operations. The encoding of the requests, can be done using either KVP values or XML, but the state of geographic features should be encoded using GML.

In a WFS also the use of three normative namespaces is defined, namely:

- `http://www.opengeospatial.net/wfs` for the WFS interface vocabulary,

- `http://www.opengeospatial.net/gml` for the GML vocabulary,

- `http://www.opengeospatial.net/ogc` for the OGC Filter vocabulary.

Regarding the underlying transport protocol, at least one between HTTP GET and HTTP POST methods must be supported and response messages should be accompanied by the appropriate MIME type and by other appropriate HTTP entity headers. Moreover, with the HTTP POST method, the use of SOAP is also possible. In fact a client may send requests using a SOAP message and the WFS may respond with another SOAP message. Nothing is mentioned about the structure of the SOAP Header.

Web Coverage Services

The Web Coverage Service standard (Baumann, 2010a) defines an interface for the exchange of geospatial information representing phenomena that can vary in space and time (known as coverages, a specialized class of features). Like WMS and WFS, in WCS a client has the possibility to specify the desired criteria for its queries. The mandatory operations that a WCS must support are: GetCapabilities, DescribeCoverage (a client submits a list of coverage identifiers and the service returns, for each identifier, the description of such coverage) and GetCoverage (a client requests the processing of a particular coverage from a WCS). The use of the HTTP GET with KVP encoding or HTTP POST with XML encoding are both supported in WCS. In addition, the *OGC WCS XML/SOAP Protocol Binding Extension document* (Baumann, 2010b) specifies how WCS clients and servers can communicate using the SOAP protocol.

The Web Services Business Process Execution Language

The Web Services Business Process Execution Language (WS-BPEL) defines *"a model and a grammar for describing the behaviour of a business process based on interactions between the process and its partners. The interaction*

with each partner occurs through Web service interfaces, and the structure of the relationship at the interface level is encapsulated in what is called a partnerLink" (Alves et al., 2007). From a general point of view, WS-BPEL can be seen as a scripting language to create applications by composing existing Web services. Like other Web services standards, WS-BPEL is expressed by using XML, through XML Schema metadata, and depends on several W3C specifications, the most important of which is WSDL. In particular, the data model used in WS-BPEL processes is specified using either XML Schema or the message element of WSDL documents. There can be two types of WS-BPEL composition (named WS-BPEL process), namely Abstract (useful, for example, to describe a process *template*) and Executable (fully specified and executable). Figure A.3 shows the structure of a typical basic WS-BPEL definition.

The <PROCESS> element represents the root element of the definition. Each <PARTNERLINKS> element contains the <PARTNERLINK> children, a concrete reference to WSDL services (named Partner Services) that will take part in the execution of the business process. In particular, for each <PARTNERLINK>, the partnerLinkType attribute identifies the portType of the referring service. The <VARIABLES> element defines the data variables used during the process workflow and their definition is provided in terms of WSDL Message Types, XML Schema Types or XML Schema elements. The <FAULTHANDLERS> element is used to define what to perform in case of fault (for example, during the invocation of a service). Finally, the <SEQUENCE> element lets process designers organize a series of building blocks (named activities) executed in a sequential and predefined order. Besides the sequential order, the whole process logic can be structured in several other ways, such as conditional branching (<if> element),

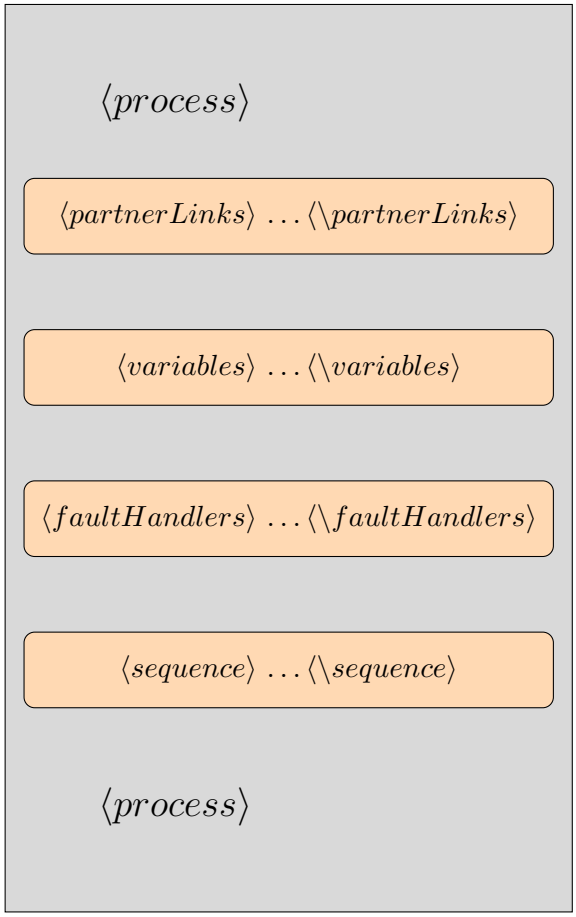

Figure A.3: The definition of a typical WS-BPEL Process

iterations, repetitions (<while> or <repeatUntil> element) and parallel processing (<flow> element).

Appendix B

Data Collection Modules for the Prototype Evaluation

Table B.1: Data Collection Form for SLN4MOP prototype

Date_____	**Participant ID**_____	**Evaluator**_____
Task number_____	**Start time**_____	**Stop time**_____

Comments made by participants:

Errors or problems observed (including assistance offered):

Other relevant observations:

Table B.2: User Reactions Survey

Now that you have completed the tasks, we would like to know some of your reactions, both in general and to specific features of the system.

Name _____

What three things did you like most about the Farming prototype?
What three things did you like least about the Farming prototype?
If the Farming prototype was made available to you, would you use it or not? Why?

Please respond to the following 10 items by circling the opinion that best corresponds to your own.

1. The list of functionalities provided by the application follows crucial steps of the farming process.

[**Strongly Disagree**] [**Disagree**] [**Neutral**] [**Agree**] [**Strongly Agree**]

2. The crop selection functionality allows me to search for crop varieties faster.

[**Strongly Disagree**] [**Disagree**] [**Neutral**] [**Agree**] [**Strongly Agree**]

3. The use of the traffic light metaphor (red/yellow/green buttons) to classify the categories of production levels was clear to me.

[Strongly Disagree] **[Disagree]** **[Neutral]** **[Agree]** **[Strongly Agree]**

4. The purpose of the history button is clear.

[Strongly Disagree] **[Disagree]** **[Neutral]** **[Agree]** **[Strongly Agree]**

5. Gaining information about amounts and prices of certain products in past seasons does help making a decision.

[Strongly Disagree] **[Disagree]** **[Neutral]** **[Agree]** **[Strongly Agree]**

6. The information supplied during the crop selection process does help making a decision.

[Strongly Disagree] **[Disagree]** **[Neutral]** **[Agree]** **[Strongly Agree]**

7. The comparative analysis of different products is a useful alternative to the direct selection of the product to seed.

[Strongly Disagree] **[Disagree]** **[Neutral]** **[Agree]** **[Strongly Agree]**

8. Once I made my decision and selected a crop quantity, the effects of my decision were visible on the interface.

[Strongly Disagree] **[Disagree]** **[Neutral]** **[Agree]** **[Strongly Agree]**

9. It was easy to realize that a product may change the associated color from green to yellow/red and from yellow to red, any time I or other users make a decision on that product.

[**Strongly Disagree**] [**Disagree**] [**Neutral**] [**Agree**] [**Strongly Agree**]

10. Overall, the crop selection functionality enhances the effectiveness of my decision at this stage of the farming process.

[**Strongly Disagree**] [**Disagree**] [**Neutral**] [**Agree**] [**Strongly Agree**]

What do you suggest as changes to the design of the Farming prototype?

Do you have any other comments or reactions?

Thank you for your participation!

Bibliography

Alatalo, T., Heikkinen, T., Kallinen, H., and Pihlajamaa, P. (2001). *Mobile information systems*.

Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guzar, A., Kartha, N., et al. (2007). Web services business process execution language version 2.0. *OASIS*.

Amirian, P., Alesheikh, A. A., and Bassiri, A. (2010). Standards-based, interoperable services for accessing urban services data for the city of tehran. *Computers, Environment and Urban Systems*, 34(4):309–321.

Apple (2012). ios human interface guidelines.

Ballinger, K., Bissett, B., Curbera, F., Ferguson, D., Graham, S., Liu, C., Leymann, F., Lovering, B., McCollum, R., Nadalin, A., Parastatidis, S., von Riegen, C., Schlimmer, J., Shewchuk, J., Smith, B., Truty, G., Vedamuthu, A., Weerawarana, S., Wilson, K., and Yendluri, P. (2008). Web services metadata exchange 1.1 (ws-metadataexchange). *World Wide Web Consortium*.

Ballinger, K., Ehnebuske, D., Ferris, C., Gudgin, M., Liu, C. K., Nottingham, M., and Yendluri, P. (2004). Basic profile version 1.1. *WS-I Specification*, 8:1–1.

Barros, A., Dumas, M., and Oaks, P. (2005). A critical overview of the web services choreography description language. *BPTrends Newsletter*, 3:1–24.

Barton, J. J., Thatte, S., and Nielsen, H. F. (2000). Soap messages with attachments. *World Wide Web Consortium*.

Baumann, P. (2010a). Ogc wcs 2.0 interface standard-core. *Open Geospatial Consortium Inc.*

Baumann, P. (2010b). Ogc web coverage service 2.0 interface standard - xml/soap protocol binding extension. *Open Geospatial Consortium Inc.*

Bertolotto, M., Di Giovanni, P., Sebillo, M., Tortora, G., and Vitiello, G. (2014a). The information technology in support of everyday activities: Challenges and opportunities of the service oriented computing. *Mondo Digitale*, 49(13):1–12.

Bertolotto, M., Di Giovanni, P., Sebillo, M., and Vitiello, G. (2014b). Standard-based integration of w3c and geospatial services: Quality challenges. In *Web Engineering*, pages 460–469. Springer.

Bevan, N. (1995). Human-computer interaction standards. *Advances in Human Factors/Ergonomics*, 20:885–890.

Blackberry Limited (2014). The transformative power of multiplatform mobile app development.

Blythe, M., Hassenzahl, M., Law, E., and Vermeeren, A. (2007). An analysis framework for user experience (ux) studies: A green paper. *Towards a UX Manifesto*.

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. (2004). Web services architecture. *World Wide Web Consortium*.

Bosworth, A., Box, D., Gudgin, M., Nottingham, M., Orchard, D., and Schlimmer, J. (2003). Xml, soap and binary data. *URL: http://www. xml. com/pub/a/2003/02/26/binaryxml. html*.

Brewster, S. (2002). Overcoming the lack of screen space on mobile computers. *Personal and Ubiquitous Computing*, 6(3):188–205.

Burggraf, D. S. (2006). Geography markup language. *Data Science Journal*, 5:178–204.

Bygstad, B., Ghinea, G., and Brevik, E. (2008). Software development methods and usability: Perspectives from a survey in the software industry in norway. *Interacting with computers*, 20(3):375–385.

Chittaro, L. (2010). Distinctive aspects of mobile interaction and their implications for the design of multimodal interfaces. *Journal on Multimodal User Interfaces*, 3(3):157–165.

Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., et al. (2001). Web services description language (wsdl) 1.1. *World Wide Web Consortium*.

Clement, L. (2005). Uddi version 3.0. 2. *OASIS*.

Danis, C. M., Ellis, J. B., Kellogg, W. A., van Beijma, H., Hoefman, B., Daniels, S. D., and Loggers, J.-W. (2010). Mobile phones for health education in the developing world: Sms as a user interface. In *Proceedings*

of the First ACM Symposium on Computing for Development, page 13. ACM.

Davis, D., Malhotra, A., Warr, K., and Chou, W. (2011). Web services metadata exchange (ws-metadataexchange) w3c recommendation. *World Wide Web Consortium*.

Davis, G. B. (1982). Strategies for information requirements determination. *IBM systems journal*, 21(1):4–30.

de La Beaujardiere, J. (2006). Opengis web map server implementation specification. *Open Geospatial Consortium Inc.*

De Silva, L. N., Goonetillake, J., Wikramanayake, G., Ginige, A., Ginige, T., Vitiello, G., Sebillo, M., Di Giovanni, P., Tortora, G., and Tucci, M. (2014). Design science research based blended approach for usability driven requirements gathering and application development. In *Usability and Accessibility Focused Requirements Engineering (UsARE), 2014 IEEE 2nd International Workshop on*, pages 17–24. IEEE.

De Silva, L. N., Goonetillake, J. S., Wikramanayake, G. N., and Ginige, A. (2012). Towards using ict to enhance flow of information to aid farmer sustainability in sri lanka. In *ACIS 2012: Location, location, location: Proceedings of the 23rd Australasian Conference on Information Systems 2012*, pages 1–10. ACIS.

De Silva, L. N., Goonetillake, J. S., Wikramanayake, G. N., and Ginige, A. (2013). Farmer response towards the initial agriculture information dissemination mobile prototype. In *Computational Science and Its Applications–ICCSA 2013*, pages 264–278. Springer.

Di Giovanni, P., Romano, M., Sebillo, M., Tortora, G., Vitiello, G., Ginige, T., De Silva, L., Goonethilaka, J., Wikramanayake, G., and Ginige, A. (2012). User centered scenario based approach for developing mobile interfaces for social life networks. In *Usability and Accessibility Focused Requirements Engineering (UsARE), 2012 First International Workshop on*, pages 18–24. IEEE.

Di Giovanni, P., Romano, M., Sebillo, M., Tortora, G., Vitiello, G., Ginige, T., De Silva, L., Goonethilaka, J., Wikramanayake, G., and Ginige, A. (2013). Building social life networks through mobile interfaces: The case study of sri lanka farmers. In *Organizational Change and Information Systems*, pages 399–408. Springer.

Di Nitto, E., Ghezzi, C., Metzger, A., Papazoglou, M., and Pohl, K. (2008). A journey to highly dynamic, self-adaptive service-based applications. *Automated Software Engineering*, 15(3-4):313–341.

Dix, A., Finlay, J., Abowd, G., and Beale, R. (2009). *Human-computer interaction*. Springer.

Doyle, A. and Reed, C. (2001). Introduction to ogc web services. *Open Geospatial Consortium Inc.*

Dustdar, S. and Schreiner, W. (2005). A survey on web services composition. *International journal of web and grid services*, 1(1):1–30.

El-Gayar, O., Timsina, P., Nawar, N., and Eid, W. (2013). Mobile applications for diabetes self-management: status and potential. *Journal of diabetes science and technology*, 7(1):247–262.

Erl, T. (2005). *Service-oriented architecture (SOA): concepts, technology, and design*. Prentice Hall.

Fang, W., Moreau, L., Ananthakrishnan, R., Wilde, M., and Foster, I. (2006). Exposing uddi service descriptions and their metadata annotations as ws-resources. In *Grid Computing, 7th IEEE/ACM International Conference on*, pages 128–135.

Fawcett, J., Ayers, D., and Quin, L. R. (2012). *Beginning XML*. John Wiley & Sons.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext transfer protocol–http/1.1. *World Wide Web Consortium*.

Friis-Christensen, A., Ostlnder, N., Lutz, M., and Bernard, L. (2007). Designing service architectures for distributed geoprocessing: Challenges and future directions. *Transactions in GIS*, 11(6):799–818.

Frøkjær, E., Hertzum, M., and Hornbæk, K. (2000). Measuring usability: are effectiveness, efficiency, and satisfaction really correlated? In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 345–352. ACM.

Gartmann, R. and Schäffer, B. (2008). Opengis wrapping ogc http -get and -post services with soap. *Open Geospatial Consortium Inc.*

Good, M., Spine, T. M., Whiteside, J., and George, P. (1986). User-derived impact analysis as a tool for usability engineering. 17(4):241–246.

Goodchild, M. F., Johnston, D. M., Maguire, D. J., and Noronha, V. T. (2004). Distributed and mobile computing. In *A Research Agenda for Geographic Information Science*, pages 257–286.

Google (2013). Android design.

Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., Nielsen, H. F., Karmarkar, A., and Lafon, Y. (2001). Soap version 1.2. *World Wide Web Consortium*.

Hassanein, K. and Head, M. (2003). Ubiquitous usability: Exploring mobile interfaces within the context of a theoretical model. In *Proc. UMICS 03 - Ubiquitous Mobile Information and Collaboration Systems*, pages 180–194.

Hassenzahl, M. and Tractinsky, N. (2006). User experience - a research agenda. *Behaviour & Information Technology*, 25(2):91–97.

Hassenzahl, M. and Ullrich, D. (2007). To do or not to do: Differences in user experience and retrospective judgments depending on the presence or absence of instrumental goals. *Interacting with Computers*, 19(4):429 – 437.

Hettiarachchi, S. (2011). Leeks cultivators desperate as price drops to record low. *Sunday Times, ed. Sri Lanka*.

Hettiarachchi, S. (2012). N'eliya carrot farmers in the dumps: Bumper harvest, but prices low. *Sunday Times, ed. Sri Lanka*.

Hevner, A. and Chatterjee, S. (2010). Design science research in information systems. In *Design Research in Information Systems*, pages 9–22. Springer US.

Hevner, R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1):75–105.

Hix, D. and Gabbard, J. L. (2002). Usability engineering of virtual environments. *Handbook of virtual environments*, pages 681–699.

IBM (1996). Ease of use model.

Institute of Medicine (US). Committee on Quality of Health Care in America (2001). *Crossing the quality chasm: a new health system for the 21st century*. National Academy Press.

Ioup, E., Lin, B., Sample, J., Shaw, K., Rabemanansoa, A., and Reimbold, J. (2008). Geospatial web services: Bridging the gap between ogc and web services. In *Geospatial Services and Application for the Internet*, pages 73–93. Springer.

ISO 19101-1 (2014). Geographic information - reference model - part 1: Fundamentals. *International Organization for Standardization*.

ISO 9241-11 (1998). Ergonomic requirements for office work with visual display terminals (vdts). *International Organization for Standardization*.

ISO/IEC 13407 (1999). Human-centred design processes for interactive systems. *International Organization for Standardization*.

ISO/IEC 9126-1 (2001). Software engineering - product quality - part 1: Quality model. *International Organization for Standardization*.

JAX-WS RI Project (2013). https://jax-ws.java.net/.

Jayaweera, P. M. and Senaratne, R. (2011). Mobile service portal for rural fisher community development. In *International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 66–70. IEEE.

Juristo, N., Windl, H., and Constantaine, L. (2001). Special issue on usability engineering in software development. *IEEE Software*, 18(1).

Kirakowski, J. and Corbett, M. (1993). Sumi: The software usability measurement inventory. *British journal of educational technology*, 24(3):210–212.

Korkea-Aho, M. (2000). Context-aware applications survey. *Department of Computer Science, Helsinki University of Technology*.

Kotamraju, J. (2010). The java api for xml-based web services (jax-ws) 2.2.

Lake, R. (2005). The application of geography markup language (gml) to the geological sciences. *Computers & Geosciences*, 31(9):1081–1094.

Law, E. L.-C., Roto, V., Hassenzahl, M., Vermeeren, A. P., and Kort, J. (2009). Understanding, scoping and defining user experience: A survey approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 719–728, New York, NY, USA. ACM.

Lieberman, J., Reich, L., and Vretanos, P. (2003). Ows 1.2 uddi experiment. *Open Geospatial Consortium Inc.*

Lokanathan, S. and Kapugama, N. (2012). Smallholders and micro-enterprises in agriculture: Information needs and communication patterns. Technical report.

Longoria, R. (2004). *Designing software for the mobile context: a practitioner's guide*. Springer Science & Business Media.

Mancini, J. A., Martin, J. A., and Bowen, G. L. (2003). Community capacity. In *Encyclopedia of primary prevention and health promotion*, pages 319–330. Springer.

March, S. T. and Smith, G. F. (1995). Design and natural science research on information technology. *Decision support systems*, 15(4):251–266.

Matera, M., Rizzo, F., and Carughi, G. T. (2006). Web usability: Principles and evaluation methods. In *Web engineering*, pages 143–180. Springer.

Mayhew, D. J. (1999). *The Usability Engineering Lifecycle: A Practitioners Guide to User Interface Design*. Morgan Kaufmann Publishers San Francisco.

McCarthy, J. (2002). Reap the benefits of document style web services.

Medhi, I., Patnaik, S., Brunskill, E., Gautama, S., Thies, W., and Toyama, K. (2011). Designing mobile interfaces for novice and low-literacy users. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 18(1):2.

Microsoft (2013). Design library for windows phone.

Milanovic, N. and Malek, M. (2004). Current solutions for web service composition. *IEEE Internet Computing*, (6):51–59.

Nadalin, A., Kaler, C., Monzillo, R., and Hallam-Baker, P. (2004). Web services security: Soap message security 1.1 (ws-security 2004). *Oasis*.

Nano, O. and Zisman, A. (2007). Guest editors' introduction: Realizing service-centric software systems. *IEEE Software*, (6):28–30.

Nebe, K., Zimmermann, D., and Paelke, V. (2008). Integrating software engineering and usability engineering. In *Advances in Human Computer Interaction*. INTECH Open Access Publisher.

Newcomer, E. (2002). *Understanding Web Services: XML, Wsdl, Soap, and UDDI*. Addison-Wesley Professional.

OASIS (2009). Web services transaction (ws-tx) tc.

OGC Schemas and Tools Project (2009). http://www.ogcnetwork.net/jaxb4ogc.

Ort, E. and Mehta, B. (2003). Java architecture for xml binding (jaxb). *Sun Developer Network*.

Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE 2003)*, pages 3–12. IEEE.

Papazoglou, M. P. and Van Den Heuvel, W.-J. (2007). Service oriented architectures: approaches, technologies and research issues. *The VLDB journal*, 16(3):389–415.

Patnaik, S., Brunskill, E., and Thies, W. (2009). Evaluating the accuracy of data collection on mobile phones: A study of forms, sms, and voice. In *International Conference on Information and Communication Technologies and Development (ICTD)*, pages 74–84. IEEE.

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., and Bragge, J. (2006). The design science research process: a model for producing and presenting information systems research. In *Proceedings of the first international conference on design science research in information systems and technology (DESRIST 2006)*, pages 83–106.

Pernici, B. (2006). *Mobile Information SystemsInfrastructure and Design for Flexibility and Adaptivity*. Springer.

Portele, C. (2007). Opengis geography markup language (gml) encoding standard. *Open Geospatial Consortium Inc.*

Powell, M. (2004). Web services, opaque data, and the attachments problem. *The Microsoft Developer Network Library*.

Preece, J., Sharp, H., and Rogers, Y. (2015). *Interaction Design-beyond human-computer interaction*. John Wiley & Sons.

Pruitt, J. and Adlin, T. (2006). *The persona lifecycle: Keeping people in mind throughout the design process*. San Francisco, CA: Morgan Kaufmann Publishers.

Rautenbach, V., Coetzee, S., and Iwaniak, A. (2013). Orchestrating ogc web services to produce thematic maps in a spatial information infrastructure. *Computers, Environment and Urban Systems*, 37:107–120.

Rautenbach, V., Coetzee, S., Strzelecki, M., and Iwaniak, A. (2012). Results of an evaluation of the orchestration capabilities of the zoo project and the 52 north framework for an intelligent geoportal. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4.

Rosson, M. B. and Carroll, J. M. (2002). Scenario-based design. In *The human-computer interaction handbook*, pages 1032–1050.

Rubin, J. and Chisnell, D. (2008). *Handbook of usability testing: how to plan, design and conduct effective tests*. John Wiley & Sons.

Sancho-Jiménez, G., Béjar, R., Latre, M., and Muro-Medrano, P. R. (2008). A method to derivate soap interfaces and wsdl metadata from the ogc web processing service mandatory interfaces. In *Proceedings of the ER 2008 Workshops (CMLSA, ECDM, FP-UML, M2AS, RIGiM, SeCoGIS, WISM) on Advances in Conceptual Modeling: Challenges and Opportunities*, pages 375–384. Springer.

Schäffer, B. (2008). Ows 5 soap/wsdl common engineering report. *Open Geospatial Consortium Inc.*

Schneidewind, L., Hörold, S., Mayas, C., Krömker, H., Falke, S., and Pucklitsch, T. (2012). How personas support requirements engineering. In *Usability and Accessibility Focused Requirements Engineering (UsARE), 2012 First International Workshop on*, pages 1–5. IEEE Press.

Schut, P. (2007). Web processing service. *Open Geospatial Consortium Inc.*

Sebillo, M., Tortora, G., Tucci, M., Vitiello, G., Ginige, A., and Di Giovanni, P. (2015). Combining personal diaries with territorial intelligence to empower diabetic patients. *Journal of Visual Languages & Computing*, 29:1–14.

Sebillo, M., Tortora, G., Vitiello, G., Di Giovanni, P., and Romano, M. (2013). A framework for community-oriented mobile interaction design in emerging regions. In *Human-Computer Interaction. Users and Contexts of Use*, pages 342–351. Springer.

Seffah, A. and Metzker, E. (2009). *Adoption-centric Usability Engineering*. Springer-Verlag London Limited.

Snell, J., Tidwell, D., and Kulchenko, P. (2001). *Programming Web services with SOAP*. O'Reilly Media, Inc.

Sommerville, I. (2011). *Software Engineering, 9th ed.* Pearson/Addison-Wesley.

Stollberg, B. and Zipf, A. (2007). Ogc web processing service interface for web service orchestration: Aggregating geo-processing services in a bomb threat scenario. In *Proceedings of the 7th International Conference on*

Web and Wireless Geographical Information Systems, W2GIS'07, pages 239–251, Berlin, Heidelberg. Springer-Verlag.

Tekli, J. M., Damiani, E., Chbeir, R., and Gianini, G. (2012). Soap processing performance and enhancement. *Services Computing, IEEE Transactions on*, 5(3):387–403.

Telecenters in Sri Lanka: The Nenasala project (2010).

Telecommunications Regulatory Commission of Sri Lanka, Statistics (2012).

Terrenghi, L., Davies, B., and Eismann, E. (2014). Simplifying payments in emerging markets. *interactions*, 21(2):48–52.

Terrenghi, L., Garcia-Barrio, L., and Oshlyansky, L. (2013). Tablets use in emerging markets: An exploration. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '13, pages 594–599, New York, NY, USA. ACM.

Tsalgatidou, A. and Pilioura, T. (2002). An overview of standards and related technology in web services. *Distributed and Parallel Databases*, 12(2-3):135–162.

Väänänen-Vainio-Mattila, K., Roto, V., and Hassenzahl, M. (2008). Towards practical user experience evaluation methods. In *VUUM2008*, pages 19–22.

Vaishnavi, V. and Kuechler, W. (2004). Design research in information systems.

Van Someren, M. W., Barnard, Y. F., Sandberg, J. A., et al. (1994). *The think aloud method: A practical guide to modelling cognitive processes*. Academic Press London.

Villa, M., Di Matteo, G., Lucchi, M., Millot, M., and Kanellopoulos, I. (2009). Soap primer for inspire discovery and view services. *JRC Scientific and Technical Reports*.

Villa, M., Di Matteo, G., Lucchi, R., Millot, M., and Kanellopoulos, I. (2008a). Inspire network services soap framework. *JRC Scientific and Technical Reports*.

Villa, M., Lucchi, R., Millot, M., and Kanellopoulos, I. (2008b). Soap http binding status, survey on ogc and orchestra specifications relevant for the inspire network services. *JRC Scientific and Technical Reports*.

Vretanos, P. A. (2005). Web feature service implementation specification. *Open Geospatial Consortium Inc.*

Whiteside, A. (2005). Opengis web services architecture description. *Open Geospatial Consortium Inc.*

Whiteside, A. and Greenwood, J. (2010). Ogc web services common standard. *Open Geospatial Consortium Inc.*

Yu, Q., Liu, X., Bouguettaya, A., and Medjahed, B. (2008). Deploying and managing web services: issues, solutions, and directions. *The VLDB Journal*, 17(3):537–572.

Zhang, J., Pennington, D. D., and Michener, W. K. (2007). Performance evaluations of geospatial web services composition and invocation. In *IEEE International Conference on Web Services (ICWS 2007)*, pages 1128–1135. IEEE.

Zisman, A. (2000). An overview of xml. *Computing & Control Engineering Journal*, 11(4):165–167.