Service Oriented Architecture for Constrained Environments

Ph.D. Thesis submitted by

ROHIT VERMA

12110101

Discipline of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY INDORE

July 2018

Service Oriented Architecture for Constrained Environments

A THESIS

Submitted in partial fulfillment of the

requirements for the award of the degree

of

DOCTOR OF PHILOSOPHY

by

ROHIT VERMA

12110101

Discipline of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY INDORE

July 2018

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled "**Service Oriented Architecture for Constrained Environments**" in the partial fulfillment of the requirements for the award of the degree of DOCTOR OF PHILOSOPHY and submitted in the DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING, Indian Institute of Technology Indore, is an authentic record of my own work carried out during the time period from July 2012 to May 2018 under the supervision of Dr. Abhishek Srivastava, Associate Professor, Indian Institute of Technology Indore, India.

The matter presented in this thesis has not been submitted by me for the award of any other degree to this or any other institute.

Signature of the student with date

(**Rohit Verma**)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of Thesis Supervisor with date

(**Dr. Abhishek Srivastava**)

ROHIT VERMA has successfully given his Ph. D. Oral Examination held on

T V Prabhakar

Signature of Chairperson (OEB)	Signature of External Examiner	Signature of Thesis Supervisor
Date:	Date:	Date:
Signature of PSPC Member#1	Signature of PSPC Member#2	Signature of Convener, DPGC
Date:	Date:	Date:

Signature of Head of Discipline

Date:

ACKNOWLEDGEMENTS

I am very grateful to my supervisor Dr. Abhishek Srivastava for his invaluable guidance, encouragement, and direction throughout this work. He gave me the freedom to work on the topic of my choosing and was very supportive throughout the way. Working with him ultimately resulted into a great deal of enjoyment in my dissertation research. His constant encouragement, friendly interactions, and constructive support have enabled this work to achieve its present form. The warm blessings, excellent comments, constructive suggestions, and thoughtful guidance given by him time to time shall carry me a long way in the journey of life, on which I am about to embark.

My PhD Student Progress Committee (PSPC) guided me through all these years. I would like to express my heartfelt gratitude to Dr. Aruna Tiwari and Dr. Prabhat Upadhyay for their valuable time, interesting discussions, and constructive suggestions.

I am thankful to IIT Indore for giving me an opportunity to carry out the research work and providing all the facilities. Very special thanks to Prof. Pradeep Mathur, Director, IIT Indore, for supporting and allowing this work to be represented at various national/international platforms. Also, I express my gratitude to Dr. Kapil Ahuja and Dr. Surya Prakash for their valuable time and support at the institute for the administrative purposes.

I want to thank everyone who have, in one way or another, helped me to conduct this research. I express my appreciation and indebtedness to my friends Dheeraj Rane, Tanveer Ahmed, Pramod Mane, Arpit Bhardwaj, Vipul Kumar, Gyan Praksh who helped me in many ways during my thesis work.

I also take this opportunity to express my humble gratitude towards the members of IIT Indore including the office staff, laboratory technicians, and library staff for their kind cooperation and extended support offered to me by providing the required resources for my work. I further extend my thanks to Mr. Jitendra Gupta, Shailendra Verma, Dheeraj Vijayvargiya, and Lalit Jain.

Lastly, but undoubtedly the most valued, gratitude is expressed for my mother, Mrs. Sunita Verma and my father, Mr. Suresh Verma for their continuous support during the tough phases of my Ph. D. Their suggestions and words motivated me to continue the hard work during the course of this thesis. Special thanks to Dr. Shweta Verma and Dr. Pawan Patidar. I would specially like to thank my wife Khushboo for her support and tolerance during my Ph.D.

Dedicated to my parents

ABSTRACT

KEYWORDS: Service-oriented Architecture; Web Services; Mobile Computing; Service Discovery; Service Description.

The work presented in this dissertation deals with implementing the concepts of Service-Oriented Architecture over constrained environments taking into account the distinct and unique requirements of the same. Over the last decade, Service-Oriented Architecture (SOA) has become the preferred approach to building adaptive distributed information systems. This is owing to the fact that such systems have distinct properties like loose coupling, platform independence, seamless integration within and across organizational boundaries. Such properties enable the system to adopt rapid changes by recombining and scaling existing services. There has been a plethora of work reported on SOA in literature and practice. Most of this work, however, does not consider constrained and mobile environments as distinct. The fact remains that traditional approaches are definitely not directly applicable to mobile environments owing to the latter's dynamic and distinct nature.

This dissertation contributes with a set of approaches and methods to facilitate SOA in constrained environments. First, a service registry framework is proposed that manages dynamic, light weight, and distributed service registries 'solely' over constrained and mobile devices. Second, we propose a dynamic, lightweight, extensible, and detailed service description approach especially designed for constrained mobile environments. The novel approach takes into account crucial description aspects such as isolated data sources, collaborating partners, hardware along with the functional, non-functional, business, and contextual aspects. Third, a model inspired from '*Membrane Computing*', to describe and execute a mobile workflow is proposed. The model orchestrates the mobile service workflow in a decentralized manner. Finally, the novel approaches proposed are validated by engineering a mobile prototype in the form of an android service and evaluating the same in an actual laboratory setting with the intent of assessing the efficacy of the same.

The research presented here has the potential to be applied by practitioners in scenarios where there is little or no preexisting infrastructure. Examples of such scenarios include war-front settings, disaster relief operations and so on. Further, the dissertation would provide researchers with a foundation to work towards security issues, QoS improvement, and other related issues in the field of mobile SOA.

LIST OF PUBLICATIONS

Journals:

J1. Rohit Verma, Abhishek Srivastava. A Dynamic Service Description for Mobile Environments. Springer Computing, pp. 1–29, 2018. (DOI: 10.1007/s00607-018-0611-z).

J2. Rohit Verma, Abhishek Srivastava. A Dynamic Web Service Registry Framework for Mobile Environments. Peer-to-Peer Networking and Applications (PPNA) vol. 11, no. 3, pp. 409—430, 2018. (DOI:10.1007/s12083-016-0540-6).

J3. Rohit Verma, Abhishek Srivastava. Service Discovery in Mobile based Service-Oriented Crowdsourcing. (Submitted to Journal of Internet Services and Applications).

Conferences:

C1. Rohit Verma, Abhishek Srivastava. Towards Service Description for Mobile Environments. In Proceedings of the 12th IEEE International Conference on Services Computing (SCC), New York, USA, 2015. (DOI: 10.1109/SCC.2015.28)

C2. Rohit Verma, Tanveer Ahmed, Abhishek Srivastava. Expressing Workflow and Workflow Enactment using P Systems. In Proceedings of the 15th International Conference on Membrane Computing (CMC), Prague, Czech Republic, 2014.

C3. Tanveer Ahmed, Rohit Verma, Miroojin Bakshi, Abhishek Srivastava. Membrane Computing Inspired Approach for Executing Scientific Workflow in the Cloud. In Proceedings of the 15th International Conference on Membrane Computing (CMC), Prague, Czech Republic, 2014.(DOI: 10.1007/978-3-319-14370-5_4)

C4. Rohit Verma, Abhishek Srivastava. A Novel Web Service Directory Framework for Mobile Environments. In Proceedings of the 21th IEEE International Conference on Web Services (ICWS), Alaska, USA, 2014. (DOI: 10.1109/ICWS.2014.91)

C5. Rohit Verma, Sushmita Ruj. Security Services using crowdsourcing. In Proceedings of International Conference on Ambient Systems, Networks and Technologies (ANT), Hasselt, Belgium, 2014. (DOI: 10.1016/j.procs.2014.05.454)

C6. Rohit Verma, Sushmita Ruj, Abhishek Srivastava. Security Verification using Crowd Sourcing. In Proceedings of the Security and Privacy Symposium (SPS), IIT Kanpur, Kanpur, India, 2013.

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1

Introduction

This dissertation presents our research for facilitating Service-Oriented Architecture in constrained environments, proposing lightweight and dynamic approaches that acknowledge the distinct requirements of these environments. In this chapter, we present the motivation of our study in section 1.1, followed by the research context in section 1.2. Section 1.3 presents the summary of contributions. Finally, section 1.4 describes the organization of the thesis.

1.1 Motivation

> Ubiquitous computing has as its goal the enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user.
>
> -Mark Weiser[1]

Mark Weiser coined the term ubiquitous computing with contemplation of the role of computing devices in day-to-day activities of everyday life. His vision of ubiquitous computing further acknowledges the fact that invention and adaption of modern computing devices in everyday life is just a beginning. The real power of the concept would emerge with the seamless interaction of all the computing devices. Over the last decade, there has been a tremendous growth in constrained computing devices in the form of tablets, mobile phones, smart

phones, PDAs, smart vehicle devices etc. The vision of ubiquitous computing could not be achieved without these constrained and mobile devices. This growth makes the mobile environment a predominant candidate for realizing the vision of ubiquitous computing. Further, to cater to the requirements of ubiquitous computing, there has been a requirement of extensive distribution in software systems. This distribution in software systems requires seamless integration and connection among heterogeneous applications and resources across organizational boundaries. Service-oriented architecture (SOA) is a well proven design paradigm that provides the required conceptual foundations for such integration. This is achieved by creating autonomous, platform-independent, and loosely coupled software entities called services. Service-oriented architecture promotes the idea of development of rapid, low-cost, inter-operable, evolvable, and massively distributed applications [2].

Recent years have seen an emergence of service-oriented systems in mobile environments, where the mobile devices are predominantly considered as service consumers. Several companies like Google, Facebook, Twitter, Dropbox, have (or are in the process of) modernized their existing infrastructure and are repackaging existing applications and services for this new class of mobile consumers. The primary focus of these companies is to consider the mobile device users as consumers. However, the fact remains that the true potential of these hand-held widgets would only be fully utilized when they are also promoted as service providers. One of the many benefits of a mobile service provider would be the ready availability of dispersed data and information via mobile services that are otherwise expensive to collect by a single centralized entity. There has been substantial work towards enabling mobile devices to host and provision web services [3][4][5]. These mobile services are developed and provisioned over a wide variety of platforms and technologies. Therefore, a flexible architecture is required that provides a standardized way of interaction among

such loosely coupled, autonomous, and platform independent mobile services. **Service-Oriented Architecture** (SOA) is an architecture that could make mobile services describable, publishable, and discoverable. Service-orientation is not a new concept and is a well proven web technology in legacy wired environments. This dissertation addresses the challenges and shortcomings of existing SOA technologies when applied in constrained and mobile environments and presents novel solutions for SOA facilitation in the same. One of the motives of the dissertation is to present a Service-Oriented Architecture that is 'solely' driven by constrained devices. One such framework makes use of the computing power dispersed in the crowds' mobile devices and offers security services using them. The security service is a specific example and the same could further be extended to generically provide any service offering over mobile devices. Following is a motivating scenario and demonstrates a possible application domain of our research:

Motivating Scenario

Alice is a high risk cardiovascular patient. Recently, she got an ECG sensor implanted in her body [6] that monitors her cardiovascular health and provides statistics and information as a mobile service via her mobile phone. This service can be consumed by her cardiologist and she can be provided with proper prescriptions as per her current health. One day she had a sudden cardiac arrest on her way to another city. Alarming variations in her ECG signals were observed by the service on her mobile device and the service discovered the nearest ambulance through the latter's exposed mobile service. Further, her mobile service automatically provided access to her latest ECG signals to the ambulance support medical staff and enabled them to prepare well in advance for the patient. The ambulance was able to discover her current location through

another service on her mobile device that provided GPS coordinates. Further, when the ambulance was on its way, the ambulance's mobile service provided the doctors at the nearest hospital with the latest information on the situation. Simultaneously, the hospital was able to make use of Alice's ECG mobile service to gather her ECG history and prior to her arrival the doctors at the hospital had a chance to study her medical profile and case in detail. On its way to the hospital, the ambulance was able to make use of the services exposed by other travelers on their respective mobile devices to avoid the busy route and opt for a path with less traffic. Meanwhile, the insurance company was contacted by Alice's mobile service and her hospital information was provided, so that the financial aspect of the treatment could be taken care of before her arrival. Alice's cardiologist was also able to provide details of his/her prescriptions via his/her mobile service to the doctors in the hospital so that the latter could learn about her medications and allergies if any.

1.2 Research Context

With rapid advancements in computing technologies and mobile/wireless networking, mobile devices have become perhaps the most suitable and economical solutions for the provision of dynamic, transient, contextual, personalized services. These mobile provisioned services can make service access handy and convenient for service consumers. Further, the provisioning of mobile services is an economical solution that requires little or no pre-existing infrastructure. In the discussed scenario, Alice, her cardiologist, the ambulance, hospital staff can make use of each other's mobile services in critical situations and can provide assistance to Alice. This explains the importance of mobile services. Subsequently, the scenario also raises a question, leading us to formulate the following

thesis question:

> "How are the mobile services described, published, discovered, and utilized in an uncertain constrained environment?"

Analyzing the question in more detail, we discern a number of sub-problems that need to be solved before the envisioned mobile SOA and the discussed motivating scenario becomes a reality. We identify three main research challenges and consider certain sub-issues out of these challenges:

Mobile Service Registry. A major concern when realizing service-oriented architecture over mobile environments is service discovery. This has received considerable attention over the past [7][8][9], but work catering specifically to mobile environments is still missing. Several challenges specific to hosting web services over mobile devices need to be taken into account in such service discovery mechanisms. These include, but are not limited to battery and network constraints, limited computational power of mobile devices. Moreover, such dynamic mobile services are prone to uncertainty (owing to network outage, battery issues, physical damage) and frequent changes in functionality (primarily owing to the change of context), and hence make frequent service updates a necessity to effectively function as web-services.

The role of the *service registry* therefore, becomes one of prominence to properly manage such dynamism. Traditional service registry solutions for web-services such as UDDI [10], ebXML [11], can not be directly utilised in such environments that require frequent updates. What contributes to this is the exhaustive data model of such registry offerings that is hard to analyze and parse for mobile devices at run-time.

Mobile Service Description. A key challenge that is overlooked in mobile environments is *"service description"*. Service description is crucial for

the consumers of services to get a sense and better understanding of the offered services and operations. This is of further importance in mobile environments, where the service invocation requires a great deal of service understanding owing to the dynamic nature of transient services. Traditionally, WSDL (Web-Service Description Language) [12] is used to describe and publish the functional description of web-services. Well written WSDL documents provide binding implementation information, detailed description of input-output messages, information on how messages are sent through the network, in addition to other information. WSDL documents, however, only provide the functional information of a web-service. They do not provide information on the *non-functional aspects*, *contextual aspects*, and the *business aspects* of a web-service. This information is crucial and of utmost importance in selection and proper usage of available services especially in the context of providing such services over mobile devices.

Mobile Service Workflow. One of the major concerns in realizing and making the mobile SOA adaptable to the needs of the real world is effective workflow management. When adopting a mobile oriented architecture a decentralized workflow is required. Fulfillment of this requirement is imperative because in a mobile SOA environment each executing unit i.e. mobile provider is presented with only a partial view of the entire system and therefore a decentralized workflow description is in order. In addition to this, decentralization benefits through the context of Elastic Computing. Consider a compute intensive work-item that is processing huge volumes of data. If a centralized orchestrator is handling this task, then it is incumbent upon the orchestrator to gather resources, provision them in the existing workflow, and perform migrations at remote locations etc. All this resulting in an incredible and unnecessary burden on the orchestrator. In a decentralized scenario, each mobile node is made responsible for provisioning of extra processing capabilities on its own. This results

in proper distribution of the burden across nodes. A scenario that bodes well for constrained mobile environments. Furthermore, a centralized engine fosters substantial infrastructure, development, maintenance and operational costs. For most requirements, therefore, a decentralized architecture is an ideal candidate.

1.3 Contribution

The primary contribution of this dissertation is to devise novel ways and means to facilitate SOA over mobile environments. This directly implies facilitating SOA in a environment devoid of wired and/or high end systems. We further acknowledge, however, that a novel system would have limited adaptability if it were to focus on replacing all existing technologies with new ones. Therefore, our presented contribution provides alternative for mobile environments in a manner that it complements existing technologies. With regard to the same, the contribution of this dissertation are summarized as:

- A novel approach to manage service registry systems is proposed 'solely' over mobile devices, and thus realizes a decentralized service registry system for SOA without the need for high-end computing systems. The approach manages a dynamic service registry system in the form of light weight and distributed registries.
- A dynamic, lightweight, extensible, and detailed service description especially designed for mobile environments is proposed that considers crucial aspects such as isolated data source, collaborating partners, and hardware aspects along with the functional, non-functional, business, and contextual aspects. The description has been partitioned along these lines and various parts of the description are distributed between service registries and the mobile service providers. An up-to-date and light weight description has been achieved by this, without compromising on the overall consistency of the description.
- A model inspired from '*Membrane Computing*', to execute a mobile workflow in a decentralized manner. The benefits of this paradigm come from

the natural process of autonomy, where each cell provisions resources and executes the workitems on its own. The approach is devised keeping in mind the feasibility of deployment of mobile web services in a decentralized and distributed manner.

These contributions result in an architecture with reduced Total Cost of Ownership (TCO). Further, the architecture is driven by constrained devices that provide services to other constrained devices. Such architecture helps in seamless service interactions in real world constrained environments such as smart homes, smart health care domains, smart city environments. We would be using the terms constrained environments and mobile environments interchangeably in the text .

1.4 Organization of the Thesis

The organization of thesis follows:

- In Chapter 2, we provide a background for various SOA and web services related technologies. We provide a brief introduction to services, web services, mobile services, SOA, WSDL, UDDI, REST, and SOAP as these are the basic technologies supporting the work presented in the thesis.
- In Chapter 3, we present in detail an effective means for providing a dynamic service registry for constrained environments. The chapter discusses the detailed architecture of the proposed dynamic service registry.
- In Chapter 4, a dynamic and rich service description approach is presented for web services in constrained mobile environments.
- In Chapter 5, we present a decentralized workflow description and enactment framework for mobile and constrained environments.
- Chapter 6 concludes our work and directions for future work are reported.

Chapter 2

Background

This chapter constitutes the technical foundation of this dissertation. Therefore, a wide range of topics are presented that contribute to the overall motivation of this work towards facilitation of mobile service-oriented architecture.

2.1 Mobile Web Services

Before discussing mobile web services, it is important to comprehend services in general. A service as defined by Kotler et al.[13] "activities or benefits offered for sale that are essentially intangible and do not result in the ownership of anything". Another definition by Katzan [14] suggests service as "a provider/client interaction that creates and captures value". We find the following definition by Wirtz et al. [15] most suitable for our understanding.

> Services are economic activities offered by one party to another. Often, time-based performances are used to bring about desired results in recipients themselves or in objects or other assets for which purchasers have responsibility. In exchange for their money, time and effort, customers expect to obtain value from access to goods, labor, professional skills, facilities, networks, and systems. However, they do not normally take ownership of any of the physical elements involved.

Now let us understand what is meant by web services. There are a variety of definitions that can be found in literature. A generic definition is given by Alonso [16] "Web services are applications accessible to other applications over the web". W3C [1] defines a web service as:

> 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 (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

From the above definition, a web service may be described as machine-to-machine interaction over HTTP. When such web services are implemented over mobile devices these are called mobile web services or simply mobile services. In this dissertation, we will use both the terms interchangeably.

We use the term mobile services to imply self-contained and reusable services that are provided by *mobile devices* or sometimes by *human users* via mobile devices. Such mobile services are application components that facilitate device to device communication over mobile environments. They provide a means to communicate between various software applications hosted over mobile devices. Such services may be utilized for commercial and/or non-commercial purposes. A few prospective applications of mobile services are:

- Partial or total replacement of physical day-to-day items: Physical credit cards, debit cards, identification cards, access keys can be provisioned as mobile services hosted over mobile devices of users.

[1] https://www.w3.org/TR/ws-gloss/#webservice

- Gateway to sensor provided information: Mobile devices can provision services that offer information provided by general purpose sensors (e.g. Location sensor) or special purpose sensors (e.g. Medical sensors (ECG sensors, Body glucose level sensors)), Environmental sensors (Fire sensors, Barometer sensor).

- Personal information provider: Services that offer information about a person can act as a dynamic and digital visiting card. This can help him/her socialize without the need of introducing himself/herself again and again. Further, this can be used by applications that record attendees' information in a particular meeting.

- Service in Infrastructure-less environment: Mobile services are particularly useful in scenarios where there is little or no preexisting infrastructure by functioning through mobile ad-hoc networks. Examples of such scenarios are a war-front, post-disaster relief operations.

Though such services are already provided over wired networks and are widely used, the concept of services hosted and provided over mobile devices is relatively new and is at an inception phase. Services offered over mobile devices of users has the potential to substantially reduce the total cost of ownership for hosting a service. Further, such mobile services are able to provide personalized and contextual services/information in a more effective manner. One may also suggest running a web service over a wired infrastructure and making a mobile device a proxy for the same rather than actually hosting it over the mobile device. Though running mobile services as a proxy can work effectively in certain situations, there are several applications and scenarios (as discussed above) that require services to be present on the mobile device. Further, firewalls sometimes block access between mobile services and the proxy. Also, provisioning, configuring, and maintaining a proxy is more tedious for a common user than

maintaining a device itself. Hence, hosting web services over mobile devices is a better approach in most scenarios. [17] provides an interesting discussion on the same.

2.2 Service-oriented Architecture

The growing popularity of web services popularized the vision of Service-oriented Architecture [18] (SOA). OASIS [2] defines service-oriented architecture as

> A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

There are a few important principles [19] of service-oriented architecture that include: loose coupling, abstraction, statelessness, re-usability, autonomy, standardized service contracts, discoverability, composibility, interoperability and service-orientation. SOA has three primary components that form the iconic SOA triangle (shown in Figure 2.1): Service Requester, Service Registry, and Service Provider. Services are offered by the provider and the service requester is the client of the service. Communication between the provider and requester are facilitate by the service registry. The offered services are described by WSDL (Web Service Description Language). Further, with the standardization of web services and the seamless use of the World Wide Web (WWW) as a transport medium, HTTP and WWW have become integral components of SOA. As we progress through this thesis we will understand that this enables dynamic

[2] https://www.oasis-open.org/

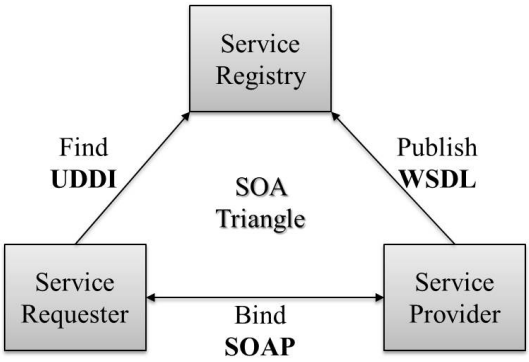

Figure 2.1: Service-oriented Architecture

binding of reusable web services in a loosely coupled manner across enterprises. We will now elucidate the idea of SOA on the basis of the 'Find-Bind-Publish' terminology.

2.2.1 Find : UDDI

In traditional SOA, the find operation is performed over the service registry. Service registries are generally UDDI [10] based. UDDI stands for Universal Description, Discovery and Integration. UDDI is the name used for the group of web based service repositories that expose information about a business and its technical interfaces commonly known as API's or Application Programming Interfaces. These UDDI based service registries are operated over multiple sites and can be used by any service provider to register its business services and by clients that want to search information about web services made available by or on behalf of business entities. The information that a business entity can register about its web service includes all information that would enable a service requester to answer the questions: "who, what, how, and where" about a web service. Figure 2.2 shows the information stored in a typical UDDI registry. The information is provided by the web service provider to the UDDI registry

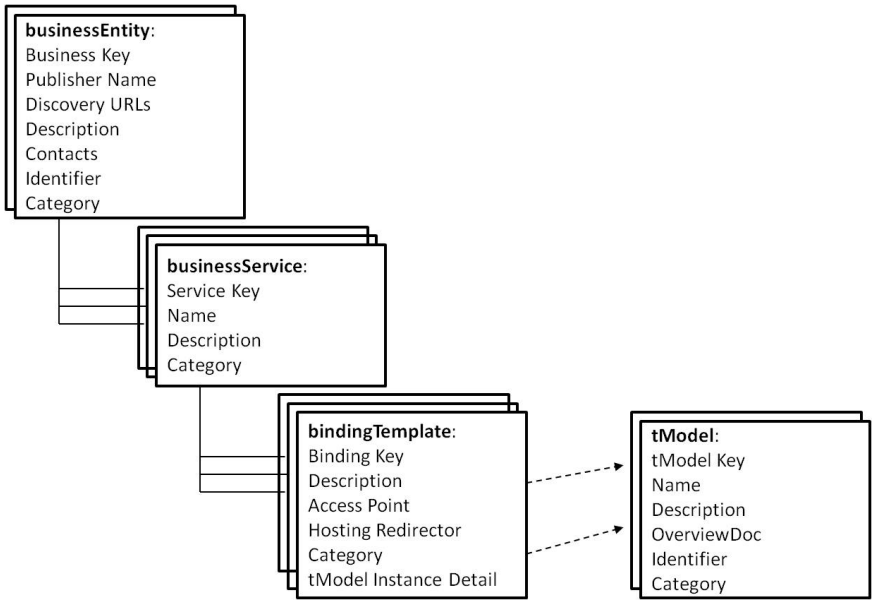

Figure 2.2: UDDI Registry Structure

through publisher APIs (The information is transported as XML tags, we do not show the XML for the UDDI structure here for simplicity and space constraints. Interested readers may refer to: http://goo.gl/cn8VaP).

A UDDI registry comprises a set of three components: **White Pages** comprise the basic information about the web service provider (e.g. contact, name, address), **Yellow Pages** comprise business category listing information about the related service provider and follow a standard taxonomy. **Green Pages** provide detailed technical information about the interface of a web service. Usually, WSDL is a part of green pages. The primary purpose of these components is to ease service discovery for the service requester.

2.2.2 Publish : WSDL

In traditional SOA, web services are published using WSDL [12]. WSDL stands for Web Service Description Language. WSDL is an XML based language used

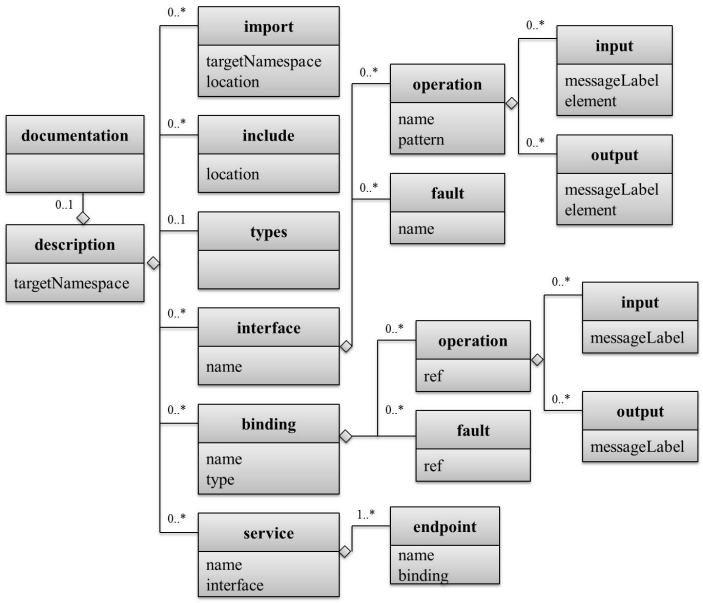

Figure 2.3: WSDL 2.0 Structure

to describe a web service interface. It is the most commonly used technology for web service description. A WSDL document describes through XML the types of inputs and outputs of the service. WSDL is currently used in two different versions. The most recent version of WSDL is WSDL2.0. This supports binding with all HTTP request methods and this facilitates improved support for RESTful web services as well. We will be describing RESTful web services subsequently in this chapter. Figure 2.3 shows the structure of a WSDL2.0 document.

A WSDL document is a concrete description of a web service. It is usually retrieved via a URL in the SOA framework. The WSDL description of a web service can be either statically generated in advance or may be generated dynamically upon request.

There have been few of the other solutions to describe web services that are well discussed and compared in chapter 4 of this dissertation, to name a few WADL (Web Application Desscription Language), Hydra [20] (a linked

data based semantic service description for RESTful services), SA-REST [21](a mashup based semantic service description for RESTful services). Several other service descriptions have been taken into account for comapritive analysis that have been presented in tabular form and in related work section of chapter 4 4.

2.2.3 Bind : SOAP and REST

Binding means connecting a web service function to another application that drives the execution of that function. In other words, binding is where a web service is connected (its functionality) to an endpoint location (where it is executed). In traditional SOA, web services are bound via SOAP binding [3]. REST is the other prominent alternative for web services that has become popular over the years.

SOAP: Originally, SOAP stood for Simple Object Access Protocol but with version 1.2 of SOAP this acronym was dropped. It is a lightweight, XML based communication protocol for exchanging information between distributed applications. It specifies the format of messages and their processing rules. SOAP relies on XML for representing messages and the format of messages is specified in term of encoding rules for application-defined data-types. Initially, SOAP was designed to work over Hypertext Transfer Protocol (HTTP) only. Subsequently, it was improved upon to be used over other transport protocols as well. SOAP is stateless and supports fundamentally one-way transmissions from senders to receivers. However, several SOAP based messages are combined to implement various other message exchange patterns e.g. request/response. A typical SOAP message consists of three parts: 1. *SOAP envelop* that encloses the entire SOAP message and describes what is in the message and

[3]https://www.w3.org/TR/soap/

how to process it. 2. *SOAP Header* that comprises information on security, session and routing. 3. *SOAP body* is a mandatory element of a SOAP message and contains the message itself. This includes information on the RPC (Remote Procedure Call), information on which operation to invoke with the supplied parameters (SOAP requests), and also data that is returned during the RPC (SOAP responses).

REST: REST stands for REpresentational State Transfer. The REST architectural style was introduced by Roy Fielding [22] as part of his doctoral thesis in the year 2000. REST aims to realize the architectural aspects that make the web, particularly HTTP, a scalable network based hypermedia system. RESTful web services, as the name suggests, are resources on the web that can be used to get specific information. A client requests a resource from a server and the server sends back the response (if there are no errors). The response is itself a representation of the resource that is present on the server. For example, it could be a JSON, XML, PDF , DOC etc. This clarifies why the service is representational. Another important aspect of REST is that it is stateless. This means that all the information that the server requires to respond is supplied with the request itself and each request is considered as new without having any presumption whatsoever. Requests are usually made using an HTTP connection. Requests are generally in the form of URIs (Uniform Resource Identifier). URIs are used to locate the path to a resource on the web server. REST defines certain 'verbs' in order to interact with the resources. A few of these are: *GET*: to receive the resource representation, *POST*: to add some information to the resource, *PUT*: modify the resources, *DELETE*: delete the resources.

2.3 Crowd-sourcing based Service Architecture

Crowd-sourcing has emerged from being just a buzzword to an important paradigm for solving a bunch of problems for mankind. The term was coined by Jeff Howe and Mark Robinson [23] as a web-based business model that utilizes the wisdom of the crowd. Crowd-sourcing represents the act of a company or institution taking out a task once performed by its employees and outsourcing it to an undefined network of people in the form of an open call. The concept of crowd-sourcing can refer to the aggregation or selection of information provided by the crowd connected to the Internet. There are several examples of successful adoption of this idea, such as information sharing systems (wikipedia, online question forums), voting/rating systems (survey websites, product/movie review websites), creative systems using crowd intelligence (Threadless, Amazon Mechanical Turk) and several others [24]. Although crowd-sourcing has been used by a variety of industries for addressing various issues, adoption of crowd-sourcing in the field of services is still in at an emerging phase [25]. In service-oriented system, provision and hosting services could be crowd-sourced as this would considerably reduce the total cost of ownership of resources involved. Moreover, Service-Oriented Crowd-sourcing [26] is driven by people and hence has the potential to provide a considerable range of services which involve creativity, quality and innovation at incredibly reduced costs. In many contexts the crowd proves to be an excellent provider of services rather than just being consumers, these include creation of local street maps, rating quality of products of local vendors, provision of information on the latest offerings in local stores, real time traffic updates and so on. These and others are examples where the quality of service provided by the crowd can far outperforms that provided by a single industry.

One part of the work of this thesis proposes a Service-Oriented Architecture

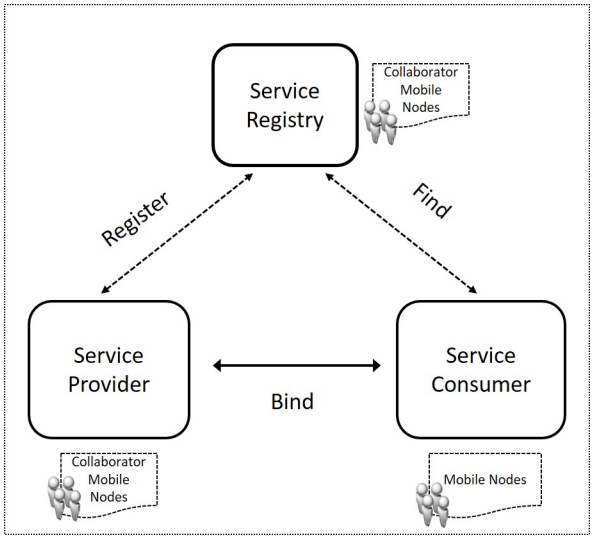

Figure 2.4: Crowdsourced SOA Environment

that effectively utilizes crowd-sourcing and concepts of volunteer computing. An important motivating factor of this work is the increasing processing power of general purpose computers and hand-held devices. With such devices increasingly available with the common man, a group of people (crowd) is able to perform computationally intensive tasks without the need for high end servers. The pictorial representation of the same is presented in the figure 2.4. In the figure, all the operations of SOA are realized by the unanimous crowd, where service provider, service comsumer, and service registry are the crowd. The issue though is that, in spite of such computation capability available with the crowd, most of the devices remain idle for large periods of time. We propose a unified, subscriber-centric, and subscriber-driven service model, where the actual service computations are performed by a crowd of constrained devices (called computational nodes). The service provider (one of the computational nodes) performs 'journaling' of crowd-sourced computation, available crowd resources, and scheduling decisions. This service provider acts as an interface between the service subscriber and the computing nodes.

Figure 2.5: SOA and various technologies

Prior to realizing such an architecture, however, there are quite a few design challenges that need to be addressed. These include and are not limited to:

- Computational Node Management: How to maintain a dynamic registry of available computational nodes?
- Service Description Management: How to maintain a unified service description for subscribers?
- Decentralized Service Workflow: How to describe and execute the service workflows involving such a large number of services and/or computational nodes?

In this thesis, we focus on addressing these challenges in the context of a constrained environment.

2.4 Summary

In this dissertation, we present solutions and alternatives for various SOA technologies customized for constrained mobile environments. Traditional web ser-

vice service technologies uch as WSDL, UDDI, WS-*, REST, SOAP are available and have proven to be robust in conventional environments. These technologies are, however, not directly adaptable for mobile environments. Figure 2.5 present a quick snapshot of SOA and the various associated solution technologies. We have primarily worked towards providing alternate solutions to the top three layers that would work well in constrained mobile environments. In subsequent chapters, we look at each of these layers in detail one at a time and present alternatives for mobile environments.

Chapter 3

Dynamic Web Service Registry Framework

3.1 Background

Continued evolution in technology has made computing devices an integral part of one's life. The most common manifestation of this is the 'Mobile Phone'. Modern technology has transformed the mobile phone from a mere communication device to a versatile computing device. These hand-held devices have enabled us not only to access information, but also to provide information to others on the move. Modern mobile phones, equipped with powerful sensors, have endowed capabilities to provide and create near real-time information. This real-time information is useful for oneself and for others. An established approach for sharing and provision of information and creating useful applications in a distributed environment is Service Oriented Architecture (SOA) [18]. Realizing SOA over mobile devices has the potential to convert mobile phones devices by common people from mere *information subscribers* to *information providers* and beyond.

The major advantage of this is that it can be used in scenarios where there is little or no preexisting infrastructure. Examples of such scenarios include War-front, Post-disaster relief management. In such scenarios, mobile based SOA has the potential to enable ground teams to provide runtime information to commanding units, help teams at disaster sites to exchange data, analyse damage and examine various statistics using mobile devices. In such systems of SOA over mobile devices all three elements of the SOA triangle: service

providers, service consumers, and service registries are realised over mobile devices. Moreover, service provisioning would be done in peer-to-peer manner over the mobile devices.

Web services are the proven way towards implementation of a "Service Oriented Architecture". Advancement in mobile device technology has motivated researchers to explore the possibilities of effectively hosting web services over mobile devices, and thereby trying to realize service oriented systems in mobile environments. There has been substantial work towards enabling mobile devices to host web services [3][4][5]. An important aspect of service oriented systems, "service discovery", however, remains a challenge in mobile environments. There is literature available on service discovery for distributed environments [7][8][9], but one catering specifically to mobile environments is still lacking. Several challenges specific to hosting web services over mobile devices need to be taken into account in such service discovery mechanisms. These include, but are not limited to battery and network constraints, limited computational power of mobile devices. Moreover, such dynamic mobile services are prone to uncertainty (owing to network outage, battery issues, physical damage) and frequent changes in functionality (primarily owing to the change of context), and hence make frequent service updates a necessity to effectively function as web-services. The role of the *service registry* therefore, becomes one of prominence to properly manage such dynamism. Traditional service registry solutions for web-services such as UDDI [10], ebXML [11], can not be directly utilised in such environments that require frequent updates. What contributes to this is the exhaustive data model of such registry offerings that is hard to analyze and parse for mobile devices at run-time. To the best of our knowledge, the current work is the first attempt to comprehensively investigate these issues and design a dynamic service registry that facilitates service discoveries in mobile environments.

As mentioned earlier, the ultimate aim is to realize a service oriented architecture over mobile devices without involving high end servers. Hence, the proposed architecture provides all registry related information and operations using mobile devices itself, without requiring high-end computers or high management costs. Further, in order to support scalability, fault tolerance, and fault localization, we propose a distributed and category based service registry.

To demonstrate the feasibility of the approach, we have engineered a prototype deployment. This includes heterogeneous and loosely coupled mobile devices deployed in a collaborative manner to manage the service registry along with native hosted services. We also compare the proposed approach with the traditional UDDI system for managing service registry from the perspective of mobile devices. The evaluation shows propitious results in favour of our approach wherein the latter is shown to have acceptable battery requirements, low data communication costs, promising scalability, and little or no hindrance to the working of native applications of mobile devices.

In the presented work, we provide a holistic service registry framework that makes use of XMPP based service registry framework at the core. We defined the roles for mobile devices involved in the service registry framework to provide a scalable mobile registry solution. We have further extended the service registry operations to cater the specific needs of the mobile environment. We further provide detailed descriptions of the various registry operations that facilitate the realisation of a dynamic mobile service registry. We further evaluated the proposed approach by realizing it through a working prototype and deployed it over mobile devices of volunteers. We further present a detailed literature survey that covers various categories of service registries.

3.2 Motivation

Kotler et al. [13] suggested services as "activities or benefits offered for sale that are essentially intangible and do not result in the ownership of anything". A mobile service defined in this work is a service that is offered from mobile phones of providers; this may also include information provided by the mobile sensors, third party software, or human users. This allows different machines to exchange information with each other over a network, without necessarily requiring a user interface. In general, the service may be a component or sub-part of the web application that is usually used by human users. For example, a chatting web application provides GUI to human users to communicate with another human. While a presence service embedded in the web application detects the presence of other machines, this presence service does not require any human intervention.

3.2.1 Motivating Scenario

This motivating is similar to the one presented in the first chapter introducing the thesis.

Alice is a high risk cardiovascular patient. Recently, she got an ECG sensor implanted in her body [6] that monitors her cardiovascular health and provides statistics and information as a mobile service via her mobile phone. This service can be consumed by her cardiologist and she can be provided with proper prescriptions as per her current health. One day she had a sudden cardiac arrest on her way to another city. Alarming variations in her ECG signals were observed by the service on her mobile device and the service discovered the nearest ambulance through the latter's exposed mobile service. Further, her mobile service

automatically provided access to her latest ECG signals to the ambulance support medical staff and enabled them to prepare well in advance for the patient. The ambulance was able to discover her current location through another service on her mobile device that provided GPS coordinates. Further, when the ambulance was on its way, the ambulance's mobile service provided the doctors at the nearest hospital with the latest information on the situation. Simultaneously, the hospital was able to make use of Alice's ECG mobile service to gather her ECG history and prior to her arrival the doctors at the hospital had a chance to study her medical profile and case in detail. On its way to the hospital, the ambulance was able to make use of the services exposed by other travelers on their respective mobile devices to avoid the busy route and opt for the path with less traffic. Meanwhile, the insurance company was contacted by Alice's mobile service and her hospital information was provided, so that the financial aspect could be taken care of even before her arrival. Alice's cardiologist was also able to provide details of his/her prescriptions via his/her mobile service to the doctors in the hospital so that the latter could learn about her medications and allergies if any.

With rapid advancements in mobile technologies and wireless networking, mobile devices have become perhaps the most suitable and economical solutions for the provision of dynamic, transient, contextual, personalized services. These mobile provisioned services can make the service access handy and convenient for service consumers. Further, the provisioning of mobile services is an economical solution that requires little or no pre-existing infrastructure. In the discussed scenario, Alice, her cardiologist, the ambulance, hospital staff can make use of each other's mobile services in critical situations and can provide assistance to Alice. This explains the importance of mobile services; subsequently, however, the above scenario also raises a question: "How is the mobile service consumer able to discover the appropriate service among such large number of

services and that too in an uncertain environment as a mobile environment?".

3.2.2 Need for a Novel Mobile Service Registry

Web services hosted on mobile devices are mainly useful for sharing contextual, personal, proximal information. Mobile devices in such environments are mostly distributed arbitrarily and make service discovery and management of service registry a cumbersome process. In this section, we discuss the need for a novel service registry architecture for mobile environments.

In order to provide an effective service registry for mobile environments, two approaches are possible. The first is a classical centralized service registry approach where all information on the available mobile services is maintained at one place and this is usually over a powerful computing device; The second approach is a decentralized service registry approach. Here, the registries are maintained by a system of distributed nodes in such a way that each node caters to a fraction of the services and there is a large degree of redundancy. Between these two approaches, the decentralised service registry approach appears to be more appropriate for mobile environments. There are several reasons for this such as the issue of a single-point-of-failure in the case of a centralised system, the lack of a definite guarantee of continuous reliable connections between mobile devices and the central server, the difficulties of rapid and regular updates in large centralised registries thus giving rise to obsolete information and so on. There are, of course, drawbacks in the decentralised system as well. It is these drawbacks that we will discuss and attempt to overcome in the rest of this chapter. Cloud offloading is another approach that is often used for facilitating services over mobile devices. In cloud offloading, the service logic, usually, resides on the cloud and the mobile devices may work as the proxy for these

services. The associated concerns of cloud offloading [27] (significant network delay and latency, rigid SLA requirements etc.) however do not make it a potential candidate for dynamic mobile service registry. Sanaei et.al. [27] discuss these challenges in detail.

A decentralised service registry may be realised using either traditional service registry approaches that are commonly used in legacy wired systems (such as UDDI [10], ebXML [11]) or a new approach especially catering to the vagaries of mobile environments may be adopted.

Though possible, adopting traditional registry approaches such as UDDI [10] from W3C is ill suited to dynamic mobile environments. The traditional registry architecture comprises UDDI data entities (businessEntity, businessService, bindingTemplate, tModel, publish-erAssertion, subscription), various UDDI services and API sets, UDDI Nodes for supporting node API set, UDDI Registries. Such a base architecture is quite 'heavyweight' and makes it difficult to host UDDI over mobile and resource constrained devices. Further, services offered over mobile devices tend to behave in an anarchic manner; as the changes in the functional, non-functional, other aspect of the services may be quite frequent owing to regular change in context and networking environment of the device. This requires frequent updates to the service registry. UDDI, on the other hand, is designed around concepts of SOAP/WSDL, heavyweight technologies that make frequent updates a cumbersome process.As a consequence, the information on the UDDI registry quickly becomes obsolete. A new approach, therefore, is imperative for maintaining an effective registry system for mobile environments.

3.2.3 Mobile Service Registry Requirements

Before we get into detailed discussions on the proposed approach, here is a quick point-wise summary of the requirements for effective service registries for mobile environments. This is along the lines of Dustdar et al. [28] who did something similar for articulating general requirements for web service registries.

R1: Management of transient web services: The very nature of mobile devices makes hosted web services repeatedly and randomly enter and leave the network. A service registry should be such that it supports such dynamic and frequent arrival and departure of service providers.

R2: Lightweight: A service registry designed for mobile environments should be lightweight. A lightweight service registry would complement the power (i.e. battery) and computational constraints of mobile devices. Furthermore, a lightweight service registry is agile and is easier to integrate with diversified mobile environments.

R3: Minimum communication overhead: Given the battery and network constraints in mobile devices, emphasis should be towards a registry system with minimum communication overhead.

R4: Distributed service registry: As the number of mobile devices (and therefore potential web services over these) are increasing exponentially, a centralized service registry system has limited utility and gets outdated very quickly. Hence, a distributed service registry system is required to support scalability.

R5: Enabling run time search: An important enabler of mobile based service oriented systems is support for run time search. This is necessary owing to the frequent arrival of new and often more competent services and/or failure of

existing services.

Conforming to the above points could potentially ensure a service registry suitable for mobile environments.

3.3 Proposed Approach

In mobile based SOA environments, each mobile device can perform the functionalities of both a service provider and a service consumer. As a service consumer, a mobile device discovers web-services and invokes them after negotiating with the providers. As a service provider, a mobile device hosts services and publishes hosted services with service registries. However, as stated earlier, an effective mobile registry to publish and discover such mobile services in dynamic environments is still lacking. The mobile services are provided by mobile devices and may be consumed by another mobile device in a peer-to-peer manner.

3.3.1 Registry Details

Our approach suggests a service registry system that comprises a light-weight registry server at each participating mobile device. The registry at each mobile device contains minimal information that is *just* sufficient to uniquely identify the registered entity. A registry server (at each mobile device) manages either of two types of registries:

1. Service Registry: Registered mobile services are managed in the service registry. The service registry contains an entry for each service as: *service name, service access point, service ID, service description, service groups, availability, service location, service provider, other service information.*

2. Group Registry: Registered services are categorized into service groups. These service groups are managed in a group registry. The group registry contains the following information for a service group: *group name, group domain, group description, registrant, groupid, group access point, other group information*.

The organization of registries, as proposed, is shown in Figure 3.1. The service information that is just enough to identify a registered service and that is less likely to change is kept in the registries and service information that is more likely to change but does not affect the discovery process of the service is kept in the vicinity of the provider. The service binding description, and contextual descriptions are provider specific and are likely to change in the mobile environment. Therefore, these descriptions of the service are kept in close vicinity of the mobile service provider. Close vicinity here implies that the description is hosted on the same mobile device as the service or a third party repository, where these descriptions can be updated rapidly.

We define several registry related operations in Section 3.4 that are performed using XML streams. These XML streams are inspired by XMPP [29] (eXtensible Messaging and Presence Protocol), a well known and established communication protocol. XMPP is already in wide use in mobile environments in several instant messaging applications.

The proposed approach provides service registry operations that facilitate effective discovery of a service. Details like the non-functional descriptions and quality of service values of the services have deliberately been kept out of the proposed system to make it as 'lightweight' as possible and hence suitable for mobile environments.

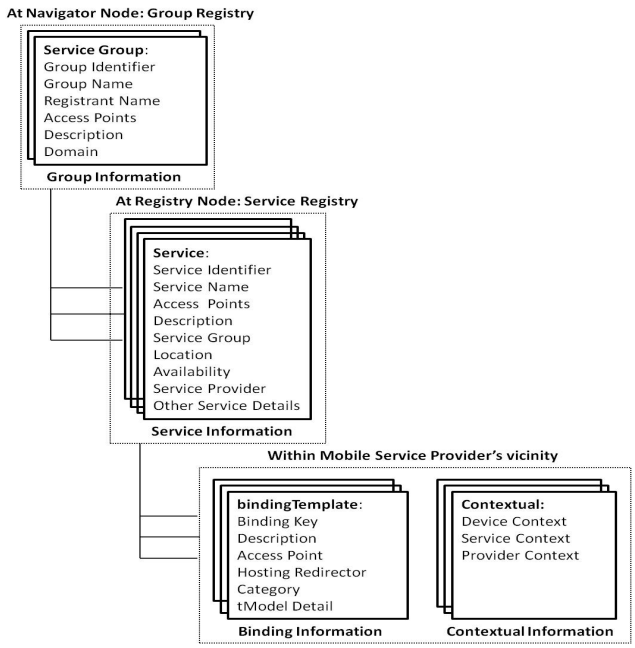

Figure 3.1: Mobile Registry Entries

3.3.2 Design Concept

There are four primitive steps in the proposed mobile service registry approach that are presented in Figure 3.2:

1) *Mobile service registry access point is retrieved:* As shown in Figure 3.2.a, a registry requester (represented by a mobile icon with **M**) accesses the public registry to retrieve the access point details of mobile service registry. 2) *Mobile service registry is accessed via Navigator Nodes:* As shown in Figure 3.2.b, the navigator nodes (represented by mobile icons with **N**) are contacted for the "Group Registry". This Group Registry contains the list of service groups. Service group of the required service provider is discovered in the group registry. 3) *Service Group is contacted via Registry Nodes:* As shown in Figure 3.2.c, the registry nodes (represented by mobile icon with **R**) are accessed via groupid for retrieving the service provider's information. "Service Registry" is traversed for the required service provider. 4) *The service provider is contacted:* As shown

in the Figure 2.d, finally, the required service provider is discovered and it is contacted for service negotiation and service binding.

These steps are discussed in detail in the next subsection. We first start with the roles performed by the mobile devices. A mobile device potentially performs the following roles (as shown in Figure 3.2.a): Navigator Node and Registry Node.

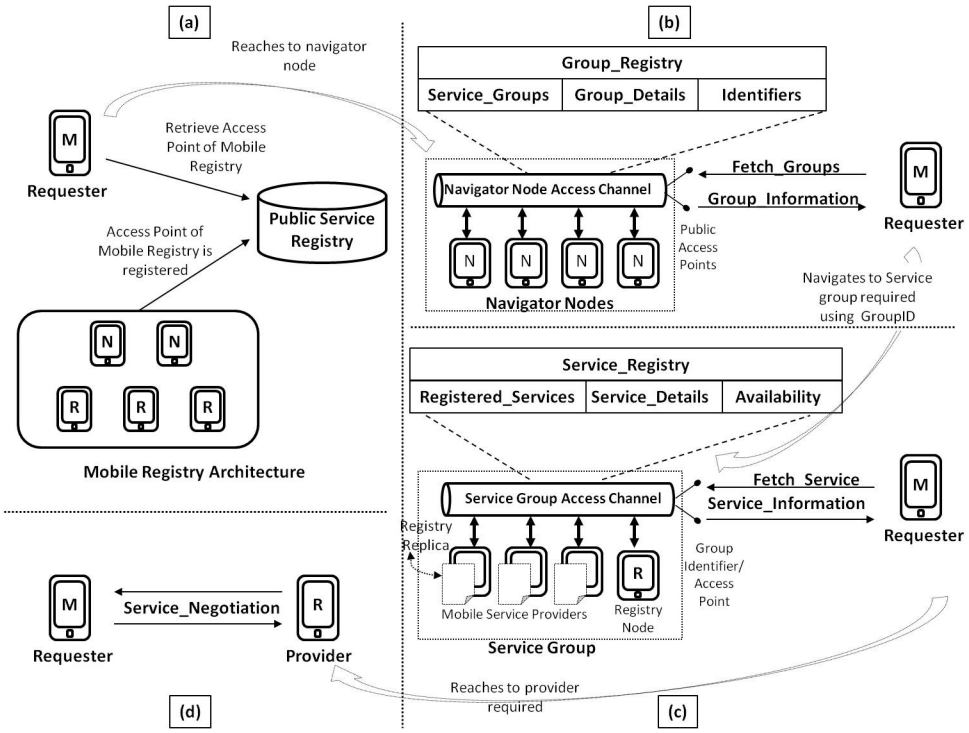

Figure 3.2: Mobile Service Registry Approach

Navigator Nodes

Navigator nodes are the entry points for the mobile registry architecture as shown in Figure 3.2. These navigator nodes are accessible by service consumers via public access points. We have devised the mobile registry architecture as a service itself. The mobile registry and its public access points can be regis-

tered with any global public registries just to make them globally discoverable (as shown in Figure 3.2.a). The motive to use global public registry is to provide the access point details of the mobile registry architecture; mobile devices would need to use the global public registry just once to retrieve the access point details of the architecture. (These global public registries could be any existing service registries as discussed in section 4.5. These registries are assumed to be well in place, hence is not discussed in detail. The global public registry is not suitable for mobile services, the reasons are already discussed in section 3.2.2.) There can be multiple navigator nodes connected to the public access points via a common access channel, as shown in Figure 3.2.b. The common access channel can be viewed as a communication bus that enables various mobile devices to communicate. The common access channel gives mobile devices the liberty to join and leave the network at any time without disturbing other navigator nodes. Whenever a new mobile device joins as a navigator node, the group registry is updated/downloaded via the access channel and the shared domain ontology becomes accessible. The idea of a common access channel can be realized using existing networking technologies as suggested in RFC1112 and RFC5771.

Navigator nodes are the mobile devices that manage the group registry (refer Figure 3.1). As discussed in the earlier section, the group registry manages service groups. These service groups are uniquely identified by group identifiers or groupid. The group registry comprises the list of service groups present in the network along with the respective group identifiers and other group details. The navigator nodes are further responsible for categorizing the registered services into the various service groups and to navigate the service providers to the assigned service groups.

Navigator nodes rely on existing ontological approaches to categorize services on the basis of their domains of offering (or offered service type). All

navigator nodes have access to the shared domain ontologies [30] for categorization of the offered services. Whenever a service provider needs to register its offered service, the group registry is referred first for matching the service with its service group. In case the service offered by the provider does not belong to any of the existing service groups, a new service group is created by referring the domain ontology and updating the group registry with a new group entry.

We have used an existing classification method for classifying services into groups [31]. The motivation behind adopting this method is that the method does not require a training set for classification and dynamic changes to the classification parameters is possible without having to retrain the classifier. This is of particular importance in mobile environments as it provides run time updates to the domain ontology without disturbing the classifiers.

The service group classification method follows generic steps. First, the mapping criteria are parsed from user supplied service description which is in the textual format. Then, the domain ontology is mapped to the mapping criteria. The process calculates a matching categorization score of the service description with the ontological context as defined by Allahyari et al. [31].

For example, the mobile service providers hosting services for: a.) Doctor's rating and b.) Hospital building floor map, share the same service group *Hospital*, hence they are identified by the same *groupid*. However a new mobile service provider offering contact information of pizza outlets would fall in a separate service group. We do not dwell upon the classification approach in this work. The interested reader is referred to [31] for more details on this.

Registry Nodes

Registry nodes are the mobile devices that manage the service registries (refer Figure 3.1). As discussed in the previous section, a service registry manages the registered services that are uniquely identified by service identifiers. This enables a service provider to provide multiple services over a single mobile device. The service registry comprises the list of registered services in a service group, their availability information along with the service details that are just sufficient to manage and identify the registered services. Of these details, the real time availability information of the registered services is what mainly contributes to overcoming the uncertainty of mobile environments. This availability information managed at the registry node gives the much needed reliability to the services hosted on mobile devices. Registry nodes are responsible for managing the up-to-date service registry, responding to service registry related queries, and performing registry related operations.

The service group can be seen as an overlay group of registered mobile service providers and registry nodes that are identified by the groupid. We have devised this group identifier as a multi-cast address for the service group members. The requests sent to the service group are received by all the member mobile service providers, however, only the group member acting as the registry node responds to the requests (as shown in Figure 3.2.c).

To improve query response time, a replica of the service registry that is retrieved from the registry node is managed at all the mobile devices hosting services in the service group. Selective updates are performed to keep the local replica updated. During service discovery, the local replica is first referred to, in case discovery fails at the local replica then the registry node is contacted. Maintaining replicas of this kind do add a little overhead to the architecture but

in the larger context the local replicas reduce traffic meant for discovery over the network substantially. Further, the local replicas ease the transition of service providers into full fledged registry nodes in the eventuality of a registry node failure (details on this in the subsequent subsection).

Failure Management:

Mobile devices acting as navigator nodes or registry nodes can also depict uncertain behavior and are prone to failure. Hence, in the case of existing registry node failure, a new registry node can be elected from the member mobile service providers, without compromising on the consistency of other service groups or navigators. (The same approach is also applied to navigator nodes.) Heartbeat operations are used to detect registry node failure. Any mobile service provider can become a registry node by participating in an election and declaring its candidature. Our method of registry node election is inspired by the leader election problem of distributed computing. The algorithm is discussed in more detail in Section 3.4.2.

Use-Case Scenario:

Potential service consumers use the proposed mobile registry architecture for discovering their desired services. This service discovery is a three step process. In the first step as shown in Figure 3.2.b, the service consumer sends the discovery request to the mobile registry architecture via a well known access point or URI (Uniform Resource Identifier) for the desired service. This request for service discovery is first handled by the navigator node. The Navigator node searches its group registry and provides the matching service group details along with the groupid for the requested service. This *groupid* acts as a multi-casting

address for all the service providers that belong to the group. In the second step as shown in Figure 3.2.c, the service consumer contacts the service group using the groupid of the required service. The Registry node of the service group responds with the available matching services and their corresponding service details. In the third step as shown in Figure 3.2.d, service consumer contacts the mobile service provider offering the required service to retrieve the technical description and performs the service negotiation for service access.

Mobile service providers use the proposed mobile registry architecture for registering their offered services. The service registration process primarily comprises two steps. First, the mobile service provider sends a service registration request to the mobile registry architecture via the well known access points. The Navigator nodes first handle the registration request and respond with the matching service group along with the groupid and other group details registered in the group registry. Second, the mobile service provider contacts the service group and registers its service with the service registry of the registry node. Alternatively, if there is no matching service group in the group registry, the navigator node creates a new service group. In this case, the registrant mobile service provider becomes the registry node of the newly formed service group.

3.4 Mobile Registry Operations

In this section, we describe the operations and functionalities that provide registry operations such as registration, discovery, service updates, and service binding in the proposed architecture. An inline comparison with UDDI is also presented.

3.4.1 Basic Registry Operations

Registration:

In the proposed approach, we have two types of registrations: 1) Group registration (at navigator node) 2) Service registration (at the registry node). These registrations are shown in the Figure 3.3.

Group Registration : The service group is registered in the group registry at the navigator node. A mobile service provider (registry client) contacts the mobile registry framework via the navigator node to register the service. However, if a matching service group is not yet registered in the group registry, a new service group registration request is initiated. The mobile service provider sends a group registration request to the navigator node along with the service details. Hereafter, the navigator refers the domain ontology and based on the service details, a matching group is mapped. This matched service group is updated in the group registry along with its groupid. The registry is then shared among all the navigator nodes. Subsequent to the successful registration process, a "groupid" is sent to the newly registered mobile device (in a 'result' type IQ stanza). At this point the registrant is the registry node of the newly formed service group. Figure 3.3 shows the group registration process.

Service Registration : Services are registered in the service registry at the registry node. The mobile service provider fetches the matching service group at the navigator node and contacts the registry node of the matching service group via the groupid. Hereafter, the registry node receives service related information from the mobile service provider and generates a serviceid for the new service. The registration process is completed when the information of the new service is updated at the service registry of the registry node. This updated registry is made available to the service providers in the service group for

provider initiated updates (pull based updates). Upon successful registration a "serviceid" is sent to the newly registered mobile service provider (in a 'result' type IQ stanza). Figure 3.3 shows the service registration process.

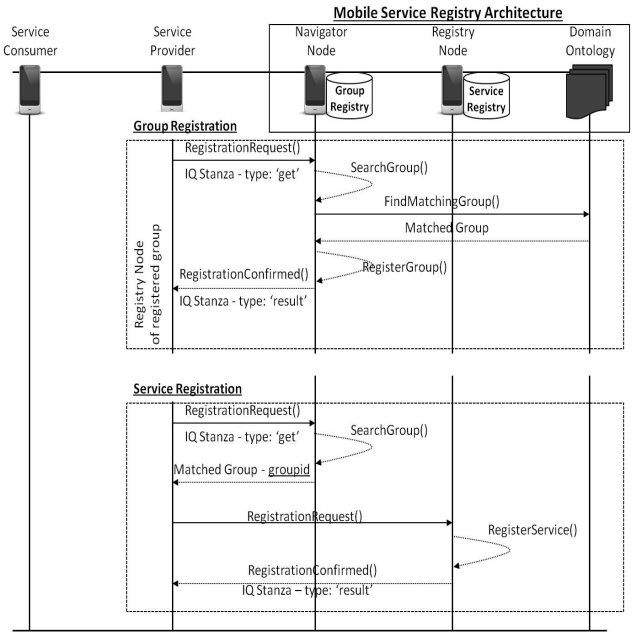

Figure 3.3: Group and Service Registration

Web Service Registration in UDDI:

Here we quickly discuss the registration process in the traditional UDDI registry system so that one can appreciate the significance of the proposed approach. In UDDI, registration is done using the *publisher APIs set* exposed by the UDDI, such as save_service, save_business, save_binding, save_t_model. These APIs are used to save detailed information on the web service, which may not be necessary in case of the mobile based web services. Moreover this information would tend to become heavy for mobile devices to process or transport.

A typical UDDI registry [16] primarily consists of the following informa-

tion: businessEntity, businessService, bindingTemplate and tModels for a registered service. The information is passed on by the web-service provider to the UDDI registry through publisher APIs (The information is transported as XML tags, we are not showing the XML for the UDDI structure owing to space constraints here. For the benefit of interested readers, we have uploaded details on this at: http://goo.gl/ cn8VaP). Deploying such a UDDI registry over a mobile device would tend to become heavy owing to the limited computational power and network constraints in mobile devices. The UDDI registry would also significantly lag behind in managing the dynamic nature of mobile devices.

Service Discovery:

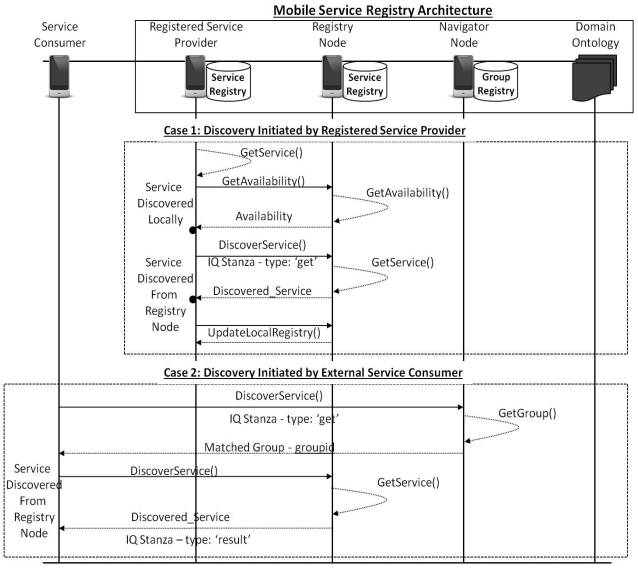

Figure 3.4: Service Discovery

Two possible cases are included in the prototype: 1. Discovery initiated by the registered service provider, 2. Discovery initiated by an external mobile service consumer.

Discovery by registered service provider: The registered service provider

first matches the service group of the required service with the local replica of the service registry. If the group matches then it fetches the required service locally. In case the service is found locally, a selective update is performed from the registry node to obtain information on the latest availability status of the service. In case the service is not found locally, the query is propagated to the registry node and subsequently the local replica is updated with the latest service registry status.

Discovery by external service consumer: The service consumer first contacts the navigator node and retrieves the matching service group information. Subsequent to this the service discovery is forwarded to the matching service group. Afterwards, the registry node of the service group responds with the matching service information. The registered service provider from the other service group also follows this process. Figure 3.4 shows the service discovery process.

Service Discovery in UDDI:

Web service discovery in a UDDI registry is done via public inquiry APIs of the UDDI, such as find_service, find_binding, find_business, find_tModel. The service discovery is performed centrally by the UDDI registry server, which requires high computational capability. This is because the consumer requests the UDDI registry server which in turn does the query search centrally and responds to the consumer with the results. The complexity and structured nature of the UDDI data structure would makes searching tedious were it adopted in a mobile environment. Though traditional UDDIs enable consumers to query the registry and are effective in a centralized system, they are ill suited to the mobile environments that are mostly distributed.

Service Binding:

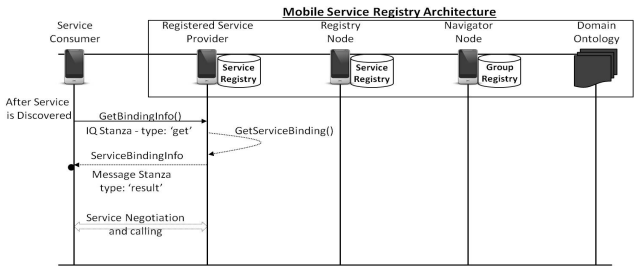

Figure 3.5: Service Binding

Web service binding information is necessary to call a particular web service. It includes the technical information on a web service, such as the access endpoint, required parameter values, return type etc. In the proposed architecture, binding information is exchanged directly between the service consumer and the service provider (as shown in the Figure 3.2.d). The service provider can provide the WSDL/WADL document or it's global URL in the binding information as well. The functional description of the service is kept in the close vicinity of the service provider. The reason being that the mobile services tend to change frequently and this might result in a change in the technical description of the same. Therefore, a proximal location of the functional description facilitates mobile service providers to readily change the service operations dynamically without violating the service registry information (Figure 3.5 shows the service binding process). Furthermore, other types of descriptions viz. non-functional, contextual, business descriptions are usually present on the same device as the service provider to keep descriptions up-to-date without increasing traffic over the mobile registry architecture.

Service Binding in UDDI :

Service consumers can retrieve the service binding information of the registered service providers from the UDDI registry using t_model and WSDL

Table 3.1: Summary of Operations Performed in Proposed Approach and UDDI.

Operations	Our Approach	UDDI
Service Registration	IQ Stanza	APIs: save_service, save_business, save_binding, save_t_model
Service Discovery	IQ Stanza	APIs: find_service, find_binding, find_business, find_tModel
Service Binding	Message Stanza	t_model and WSDL documents

documents. The t_model is an exhaustive technical description of the service binding. However, due to its inherent complexity service providers often do not update the binding information. In fact, some services do not even register themselves owing to this complexity. This has ultimately translated to the unavailability of a working and updated global UDDI based registry. Nowadays, as a general practice service consumers use web search engines to fetch the binding information. This is done by querying search engines for *filetype* as *wsdl* (for SOAP based web services) or *wadl* (for REST based web services). The results of this searching mechanism leads to all sorts of bias arising out of the indexing and page ranking algorithms of the web search engine. Furthermore, the selection would require human intelligence and analysis.

Table 3.1 summarizes the discussed operations performed with the proposed architecture and with UDDI. The operations discussed in the following subsection viz. presence notification, registry sharing, and registry update are specific to our approach and ones in which no equivalent UDDI operations exist.

3.4.2 Mobile Specific Registry Operations

Presence Notification:

This is one of the novel features of the proposed architecture. The presence information of a mobile service is of utmost importance for providing any sort of certainty in the volatile mobile environment. The presence notification provides the current availability information on the mobile service. This helps in avoiding as much as possible the failed access of offline services. Here the terms presence information and availability information are used interchangeably.

The service registry manages presence information for each service. This helps to uniquely manage the presence information of multiple offered services of a service provider. The presence information is dynamically updatable and provides availability of a service at a particular instance of time. The presence information is similar to the availability information in an instant messaging application. We managed the presence information as 'Available' or 'Unavailable' for a service in the presence tag (please refer Figure 3.7). However, this presence tag is a placeholder that can further be advanced to incorporate other presence related information.

The proposed approach also incorporates event triggered presence notification that is generated on the occurrences of events that cause the service to become unavailable. For example low battery level, dropping network strength, critical overload at provider. These events can be easily detected programmatically using API's exposed by modern operating systems of mobile devices.

Registry Sharing:

Registry sharing is required to :

1. Manage the latest information about services in registries of various registry nodes.
2. Keep a local service registry replica at the service providers in service group.

Registry sharing facilitates sharing the registry system over several nodes to form a distributed service registry structure. Such a distributed registry system becomes particularly useful when a new mobile service provider joins the service group. Registry sharing enables the newly joined mobile service provider to retrieve the registry from the group and manage a local replica. Further, the joining of a new mobile service in the service group triggers an update in the registry. That needs to be shared with the members of the group in a distributed structure. This functionality is extensible and can be adopted for timely updates or event driven updates to manage synchronization in the service registry of the various registry node.

Registry Update:

As our approach is mobile service provider-centric and the mobile environment is dynamic, hence a service provider tends to change its configuration on the run. The registry is required to be updated whenever information on a service gets changed, such as location, access point, service descriptions.

Unregister: Unregister is a registry update performed when a mobile service provider discontinues its service offering. Whenever a service provider does not want to provide a hosted service, it unregisters itself from the service registry using the unregister action.

Heartbeat Operation:

Heartbeats are used to probe various nodes in the proposed registry architecture and to keep the service registry up-to-date. A registry node periodically probes other service providers in the service group and retrieves their latest availability information to keep the service registry up-to-date. Similarly, navigator nodes probe the service group to know if the group is alive and accordingly update the group registry. Further, heartbeat operations are used for registry node election. New registry nodes can notify the service group members about their presence.

Dynamic Registry Node Election:

Dynamic registry node election is performed when a registered service provider tries to get promoted to a registry node or when the registry node fails. A registry node election is called, when periodic heartbeat signals are not received by the registered providers of the service group.

The registry node election problem is analogous to the leader election problem of distributed computing. Several solutions have been proposed for leader election in distributed computing. We have adopted an approach for the registry node election that is inspired from Luby's Algorithm [32]; a similar approach is described in [33]. The election approach can be implemented in a distributed manner without involving a central authority and produces less message traffic. In the proposed framework, each service provider announces itself as a candidate for becoming a registry node by sending its device details that include battery information, network details, device hardware details, uptime etc. to the existing registry nodes. The service provider with the maximum capability in terms of the mentioned paramenters is selected as the new registry nodes. This new registry node sends a heartbeat signal to other registry nodes to make them

aware about its new role.

3.5 Implementation

We developed a prototype web service implementation based on the proposed approach for mobile devices, using android SDK (ADT 23.0.2) and Oracle Java (version 1.7.0_72). The prototype was built to realize the architecture as presented in section 3.3. It is worthy to note that *the deployment of the prototype neither required any modification to the device nor did it require root permissions to run*. The prototype developed is independent of the native applications of the mobile device and hosted the web services. Our approach is applicable to all mobile operating systems, however we chose android for our prototype implementation as it is open source and commands much wider community support.

Our experimental setup comprised seven mobile devices (including Samsung Galaxy S Duos with Android 4.3, Sony Xperia M with Android 4.3, Google Nexus 7 with Android 4.4, Motorola G2 with Android 5.0, three Asus Zenfone 5 with Android 4.4), one laptop (Intel i3 2.13 GHz with 3GB of RAM) and a few running instances of the prototype running on virtual instances of Android devices running on the laptop. These devices comprised the engineered prototype. The setup also had multiple services and service groups. Two experimental wireless networks were setup for the validation. All our experiments were carried for varied network sizes with multiple service providers joining and leaving the network.

Prototye Design: The prototype performs all the roles as mentioned in the Section 3.3 - Service Navigator Node, Service Registry Node, and service provider. We have devised registry consumers external and also embedded them with ser-

vice provider. We have designed a parser to parse and generate the XML streams (Figure 3.7 presents the used XML infoset). Figure 3.6 presents the architecture used for prototype.

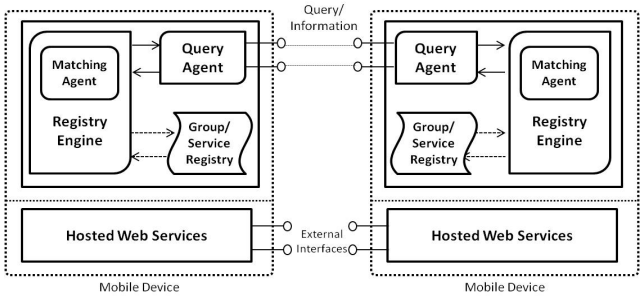

Figure 3.6: Architecture of Registry and Navigator Nodes

The presented architecture is deployed on both the nodes: Service Navigator Node, Service Registry Node. The various parts of this architecture are: 1. Query Agent: This component accepts, generates, and processes the queries from/for other mobile devices. This has external interfaces that can be contacted by any other mobile device. It can be viewed as the XMPP parser for registry management. 2. Registry Engine: All the mobile registry related operations as explained in the Section 3.3 are handled by this component. 3. Matching Agent: This component parses the result of the service query and evaluates them against the required parameter. Further, it has access to the ontology for service group formation. 4. Group/Service Registry: This is the local replica of the service registry or group registry depending on the nature of the node (whether registry node or navigator node). The presented approach works even in presence of the hosted mobile web service.

Prototype Specification

This subsection presents identifier and communication related specifications of the prototype for the architecture presented in Section 3.3.

Identifiers:

Each hosted service over a mobile device was addressed by an identifier. This enabled co-existence of multiple services over a single mobile service provider. Further, this gave flexibility to the service providers to remove a service without disturbing other services. We used two types of primitive identifiers: group identifier and service identifier. An identifier, as discussed earlier, is similar to an email address and is uniquely addressable. The structure of identifier is inspired from XMPP. The format of an identifier is:

```
[GroupID]@[NetID]/[ServiceID]
```

`NetID` specifies the domain or network of a service registry, as there can be multiple service registries that are collocated globally. For private registries, *local* is used as the NetID. `GroupID` is the identifier of the service group. `ServiceID` identifies the service offered by the mobile service provider. Therefore, there could be multiple unique ServiceIDs for a mobile device offering multiple unique services.

A service group could be “TrafficInfo" where the offered services are related to traffic information. A service registry could be for the network: Acme City. This depicts the service group identifier as:
`trafficinfo@acmeCity`. This service group is shared by all the service providers of service group trafficinfo. A unique identifier for a mobile service provider offering service related to the “Traffic Information of Main street" could be `trafficinfo@acmeCity/mainstreet`.

The identifiers are managed distinctly in a domain by the registry engines and query agents of the mobile devices. Further naming conflicts are resolved by incorporating the service provider‘s participation. In the move to enable net-

working and realize the proposed architecture over a real network, the identifiers are mapped to the physical network using several well accepted networking concepts and technologies. These technologies/concepts are available in ad-hoc networking technology, such as *multi-cast domain name system* suggested in RFC 6762, dynamic host auto configuration IP range suggested in RFC 5735 and 3927, multicast networking methods suggested in RFC 1112 and RFC 5771. The libraries for this implementation are available for the Android OS.

Communication: In the move to provide interoperability over heterogeneous mobile devices, we make use of XML streams for the service registry related communications. The mobile devices communicate and send queries/information in the form of XML stanzas. These XML stanzas are inspired by [29, 34]. An XML stanza is the basic unit of communication in XMPP. This discrete semantic unit of structured information or XML stanza is sent from one mobile device to another over an XML stream.

There are mainly three stanza types used in the architecture: `<message />`, `<presence />`, `<iq />`. A stanza is a first-level element (at depth=1 of the stream) whose element name is “message”, “presence”, or “iq” The three stanzas are briefly described below.

Message stanza is primarily used for storing, editing, and sharing service/-group information at the service/group registry. The main purpose that this is managing and updating the service/group registries. The Message stanza works in two ways: “push” and “pull”. In push, the information exchange is initiated by the sender e.g. sharing the registry at the registry node with the mobile service providers. Whereas in pull, the receiver initiates the information exchange e.g. selective update of the registry at the mobile service providers. Further, the message stanza is primarily used for service binding operations, registry shar-

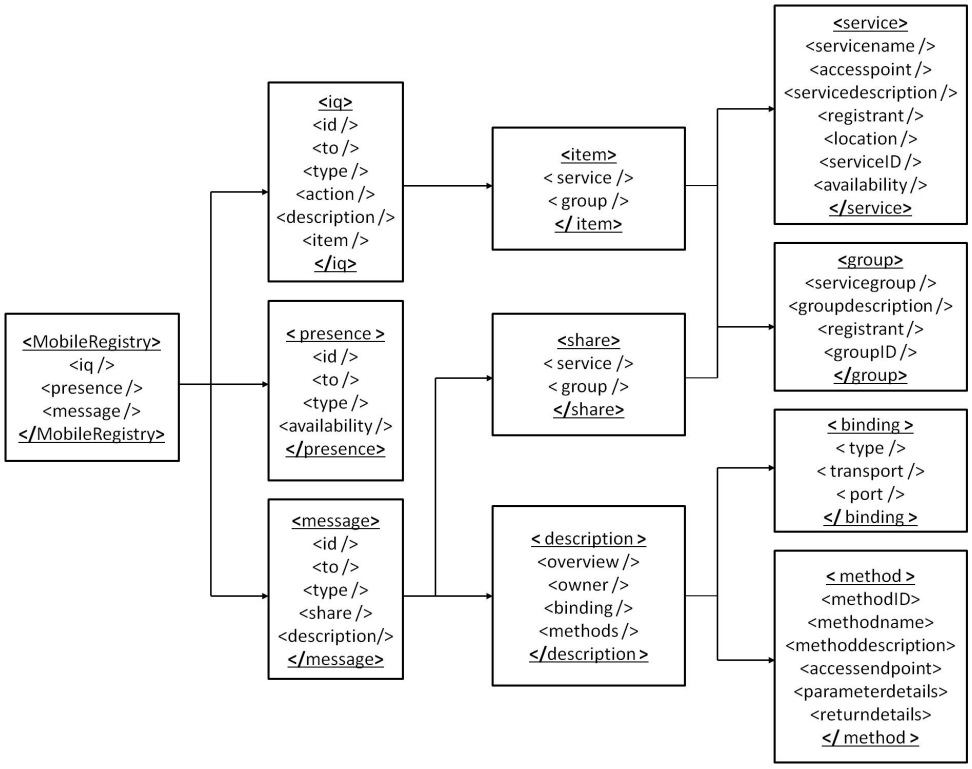

Figure 3.7: XML Streams Used in Proposed Approach

ing, registry node elections.

Presence stanza is used to update the availability information of services hosted on mobile devices. Each presence stanza includes a brief description and service identifier of the hosted service, along with its availability information. We use 'available' and 'unavailable' as the primitive presence types in the proposed approach. The presence status of a web service hosted on a mobile device is reflected on the service registry. Presence notification, heartbeat operations are implemented using this stanza.

IQ stanza is short for Information/Query stanza. It is based on a request-response mechanism and guarantees a response to a query. The nature of request in the IQ stanza is represented by `type`. Registry information request is represented by `get` and it is similar to the HTTP `GET` method. Any com-

munication involving queries from other mobile devices primarily makes use of the IQ stanza. These play an important role in getting information from other mobile devices that host the service registry i.e. registry nodes.

These stanzas are uniquely identified by the “id” element. The “type” element represent the type of registry operation that is performed by the mobile devices. Whereas “to” element contains the access point of the recipient (individual mobile device or service group). Figure 3.7 shows the detailed XML structure used in the architecture. Group and service registration operations, discovery operations, registry update operations are implemented primarily using IQ stanzas.

3.5.1 Evaluation

To evaluate our approach, we deployed the architecture (presented in Figure 3.6) over real mobile devices. We solicited volunteer participation to host our prototype over their personal mobile devices. This enabled us to analyze the feasibility of our approach in a practical scenario. We established two experimental wireless networks within our institute building to connect the volunteers' mobile devices, laptop, and virtual instances (refer Figure 3.8). During the experiment, volunteers were doing their routine work and hence the mobility of the devices followed random patterns. We repeated this experiment with varying numbers of service providers, service registration requests, service discovery queries, with the intent of emulating uncertain situations in practical scenarios. In this section, we present the results of the experiment with the motive to show the feasibility of the proposed approach.

The first experiment evaluated the effect that hosting a registry server had on the battery of the mobile devices. We analyzed the effect of various registry

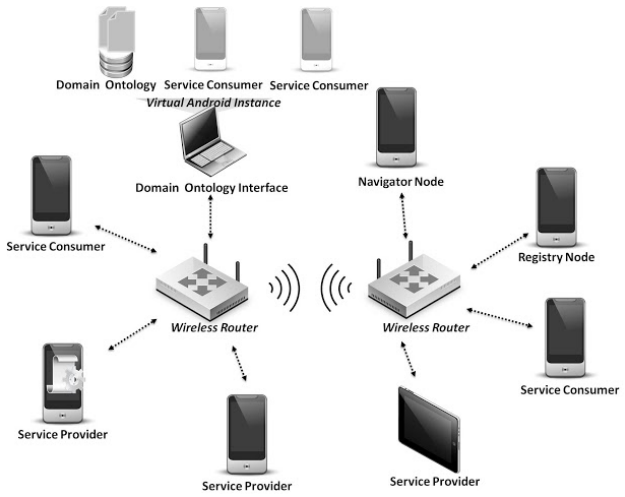

Figure 3.8: Experimental Setup

operations on the battery. For this purpose, we made use of the power model and solutions suggested by Zhang et al. [35]. This power model considers the power consumption by a mobile application and also takes into account the application's effect on Wi-Fi, CPU, Cellular interface etc. Table 3.2 shows the initial battery power consumption by the registry operations. We sent 50 requests of service/group registration, service/group discovery each on the navigator node and the registry node. We observed the power consumption at the navigator nodes and registry nodes by these requests. The values presented in the table are the average battery power consumption. For a quick reference, gmail android app had 567mW, facebook android app had 1297.5 mW, GSM call had 511 mW, and Airplane Mode had 6.4 mW power consumption (depends on build/model of mobile phone). The idea of including these is to emphasize upon the point that the implemention of the proposed mobile service registry solution seems to have acceptable power consumption. (Interested readers are referred to [36] for detailed analysis of power consumption in smartphone).

In our second experiment, we further observed the data bytes being exchanged during these requests. Table 3.3 shows the total number of bytes ex-

Table 3.2: Power Consumption by Various Registry Operations

Registry Operation	Power Consumption
Service Registration (For 50 requests)	44 mW
Service Discovery (For 50 requests)	63 mW
Group Registration (For 50 requests)	120 mW
Group Discovery (For 50 requests)	52 mW
Heartbeat	73 mW

changed for various registry operations. We concluded that less than a 1KB of data is exchanged for service/group registration. For service/group discovery the number of data bytes exchanged depends on the number of matching groups or services. However, for the purpose of the experiment we had a single matching group or service. That limited the data exchange for discovery requests to under 1 KB for each matching service/group. (Size of an average compressed image sent over a messaging app like WhatsApp is 20-25 KB, and 2.52 MB was the background data usage for the Instagram app (with a few account following) for a day. Therefore, the proposed approach has acceptable data exchange.)

Table 3.3: Data Exchanged by Various Registry Operations

Registry Operation	Total Data (Received + Transmitted)
Service Registration (For 50 requests)	48510 bytes
Service Discovery (For 50 requests)	41231 bytes
Group Registration (For 50 requests)	42161 bytes
Group Discovery (For 50 requests)	31487 bytes

In our third experiment, we sent (50*4=) 200 new service registration requests from other mobile devices and virtual instances to the registry architecture. Figure 3.9 depicts the total response time behavior for a service provider. Also noteworthy here is the fact that the mobile devices were continuously moving with the volunteers, hence the devices were randomly joining and leaving the network. Therefore, we observed a few outliers in the response time behavior. We concluded that the average service registration time (including outliers)

is near 5 seconds which seems acceptable for practical purposes.

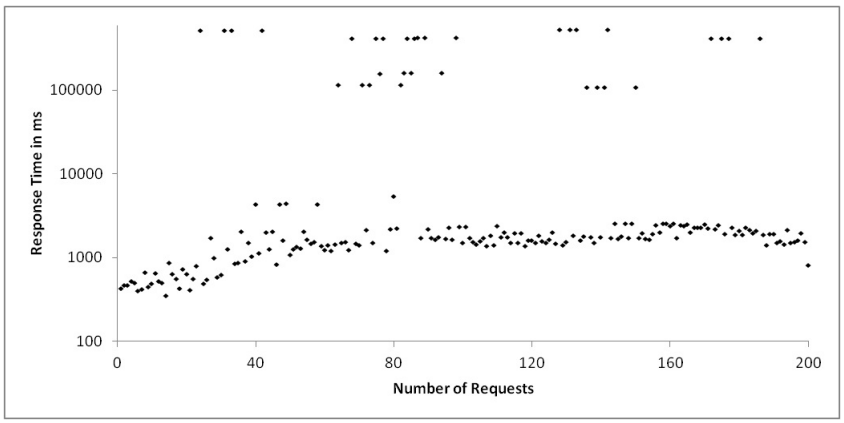

Figure 3.9: Total time taken for new service registrations

The fourth experiment evaluated the effect of directory size on the discovery time. We registered multiple services in a service group on the registry node. In order to test the scalability of our prototype, we ran service discovery operations for various directory sizes: 500, 1000, 5000, 10000, 50000, 100000. We discovered that the discovery time increases as the size of the directory increases. However, in spite of this even with 100000 registered services the query response time was under 1 second which is acceptable for all practical purposes.

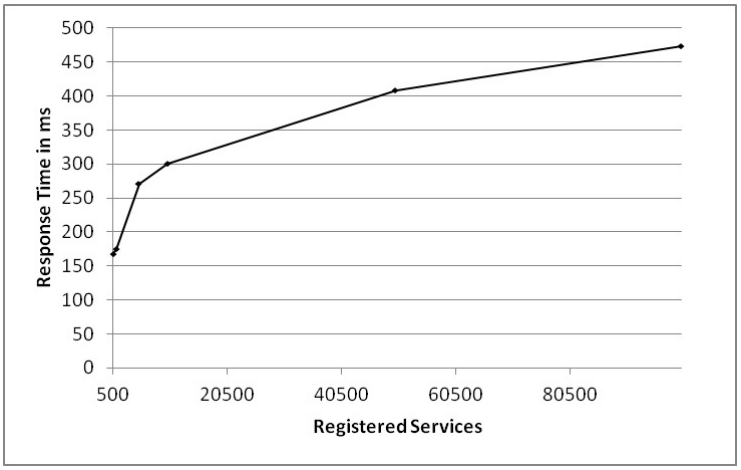

Figure 3.10: Service Discovery Time for varied registered services

We sent service discovery requests from four mobile devices and virtual

instances to varied numbers of registered services and the response time was calculated for these requests. Figure 3.10 shows the average response time for these discovery requests for various registry sizes. Through the whole experiment, the mobile device acting as the registry node was moving continuously within the network. Further, the size of the registry increased linearly with the increasing number of registered services. Even with thousands of services registered, the registry size was under 10 MB (shown in Figure 3.11). It should be noted here that we made use of SQLite for implementing the service and group registries over android mobile devices.

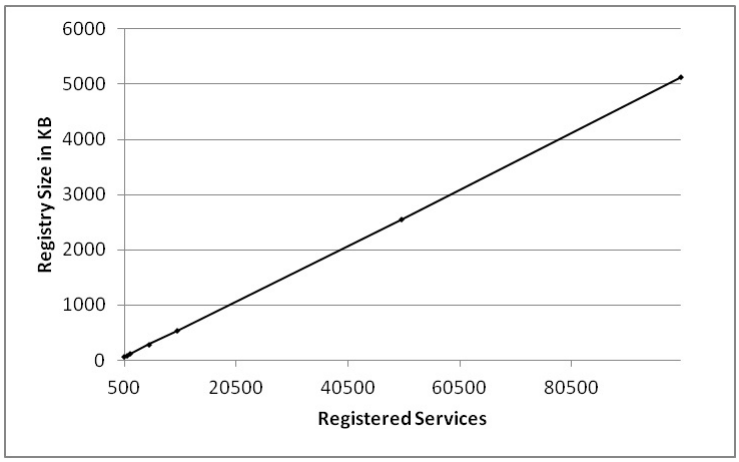

Figure 3.11: Registry size for varied registered services

Our fifth experiment aimed at evaluating the feasibility of the proposed approach when running simultaneously with other phone activities. In this experiment, we conducted a comparative study on the difference in response time for the service discovery requests a) when a volunteer was answering a phone call and, b) when the device was idle. For this experiment, we sent 50 requests for group discovery to the volunteer's mobile device. First, we observed the response behavior when the device was idle in the volunteer's pocket. During this phase the volunteer was randomly moving within the network. The response time behavior during this period is shown in Figure 3.12. Next, we made a

phone call on the volunteer's mobile phone and observed the response time during the call. The results demonstrate that there is a change in response time, but this change is well within acceptable limits. Response time behavior during the call is shown in Figure 3.13. There is an initial peak in the response time when the call is made. From the initial results, we conclude that the mobile device takes a little time to respond to the first discovery request. This is due to the fact that some processing time is required to *awaken* the sleeping mobile application. We feel, therefore, that *piggybacking of incoming requests* could be a good approach to reduce the energy overhead.

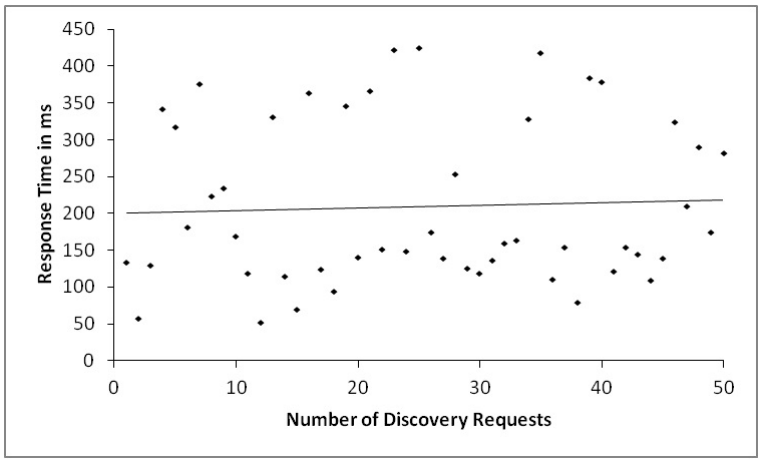

Figure 3.12: Response Time behavior without an active call

The sixth experiment involved an evaluation of the reliability of the approach. We toggled the availability status of the service provider from *available* to *unavailable* and back to *available* with a time difference of 10 seconds. These toggles were repeated 120 times at the registry node. During this time, we continuously probed the registry node from the service consumers to get the status of the service provider. The initial results shown by the experiment have less than 1% false negative, where false negative implies - registry node returns availability information as *available* when the service provider has updated it to *unavailable* and vice versa.

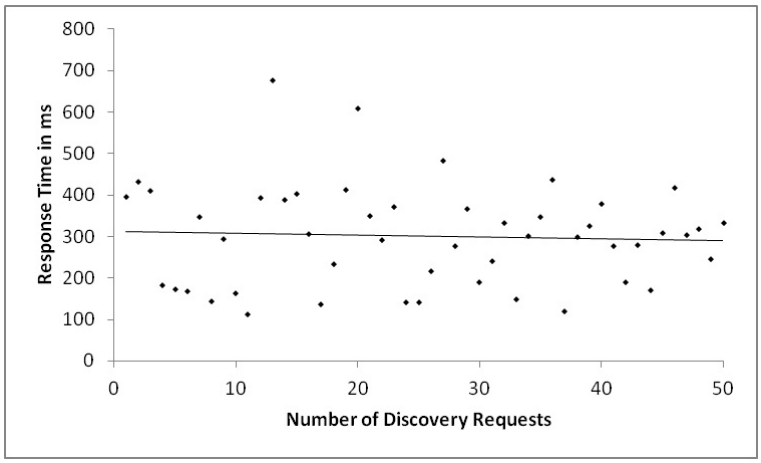

Figure 3.13: Response Time behavior during an active call

Although the scale of these experiments was limited, the results were promising. It will be interesting to explore the performance of this approach for a much larger number of service providers, and registry clients and for much longer durations (a few days). (Interested readers may refer to https://goo.gl/4VG895 for further details about the architecture and experimental setup.)

Nonetheless, we present the speculated trend of our experiments with a large number of devices:

1. Effects on battery: With increasing number of devices there would be more numbers of registry requests and responses; hence more battery power would be needed. However, in such scenarios the service groups and navigator nodes would play a crucial role. More specialized service groups would be formed as suggested in section 4.3.1, this would keep the number of devices in a service group within limits in turn keeping the battery usage in check (irrespective of any number of registered services).

2. Effects on data exchange: With increasing number of devices there would be more exchange of data. However, the split service groups would keep a check on the data exchange. Further, keeping in view the current data usage by modern smartphones, the data usage by large number of devices should be in acceptable limits.

3. Effects on service registration time: We believe that with increasing number of devices and service registrations, the response time would increase

but would be well within the accepted range. Also, this is a onetime process for a service, it would be acceptable for all practical purposes.

4. Effects on service discovery time: Figure 3.10 shows the discovery time for a large number of registered services varying from one thousand to a few hundreds thousand. This is still within acceptable limits.

5. Effects on registry size: The registry size will increase linearly for increasing number of register services. Trend is shown in the Figure 3.11. While other experiments are dependent on the individual mobile device without depending on the other device's behavior. Further, the effects of other contextual parameters on increasing number of devices (e.g. network, ISP, terrain, carrier type etc.) would be interesting to look into but unfortunately these are not within the scope of current focus.

Discussion

The proposed architecture caters to the issues of providing a dynamic service registry in mobile environments. It manage to incorporate the requirements outlined in Section 3.2.3.

R1: Management of transient web services: The approach effectively manages the availability information on each registered web service and is capable of dynamically updating it. This helps in satisfying requirement *R1* and keeping the service registry up-to-date irrespective of random entry and exit of transient web services.

R2: Lightweight: The architecture seeks information that is just enough to uniquely identify and manage a web service from the registrant mobile device for registry related operations. The registries manage information that is less likely to change and whatever change does happen is updatable via a watchdog process. This keeps the registry architecture as lightweight as possible and thus satisfies requirement *R2*.

R3: Minimum communication overhead: We have made use of just three

XML stanzas in order to minimize communication overhead, satisfying requirement *R3*. During our experiments, we observed the exchange of just a few kilobytes of data for hundreds of request transfers. This small overhead also helped us minimize battery utilization, as reflected in Table 3.2 and 3.3. Furthermore, the approach uses just one stanza "IQ Stanza" for easier service registration and de-registration.

R4: Distributed Service Registry: The proposed approach manages the service registry over dispersed registry nodes and navigator nodes. Thus satisfying requirement *R4* and improving fault localization. These features, contribute to a light weight and autonomous service registry effective for mobile environments.

R5: Run time search: The dynamic availability information, minimal data transfer, and faster response time helped in performing effective run time searches and in the process satisfied requirement *R5*.

Further, the proposed dynamic mobile service registry is compatible with existing UDDI and other registries. The dynamic registry can register external UDDI and other service registries as one of the registered services along with their access points. Contrary the proposed registry can be registered with other registries and UDDI as any of the registered service.

Assumptions and Limitations

There are certain important assumptions that have been made in the proposed service registry framework. These are listed as follows: 1) The load balancing of the incoming registry requests is assumed to be handled at the device level. There could be a threshold capacity limit, at each device depending on its hardware capacity. On exceeding this capacity limit, the incoming requests are not entertained and these surplus requests are handled by other registry nodes that

have access to the common access channel. 2) There is no reward system suggested in the current state of the work. Hence, it is assumed that the mobile device owners are self-motivated to provide their respective devices for serving as the registry. We may look into a system of rewards/incentive as part of future work. 3) The updates are performed to keep the local registry replica updated (at the navigator/registry nodes) in an automated manner without manual intervention.

A few important limitations of the current work are listed as follows: a) The current design of the framework does not deal with the privacy issues of the mobile phone users associated with service registry and service provisioning. b) The current design of the mobile registry does not handle the QoS (Quality of Service) aspect of the mobile web service. The QoS may be handled by providing a link to the data server (external to the framework) that could have QoS and other details on the service description. c) Though we have a system in place to detect the unavailability of registry nodes through heartbeat signals, the behavior of the mobile device owner, poor network connectivity, and physical damage to the mobile device could result in the abrupt unavailability of the registry/navigator node. These have not been dealt with in this work. d) Finally, the current evaluation of the approach was conducted within a supervised lab environment. The registry power and data requirements, service discovery performance, and other results were therefore well within acceptable range. There could be slight variations in these if the experiments were carried out at a much larger scale including thousands of mobile devices.

3.6 Related Work

Service discovery is an important aspect of service-oriented architecture. Two types of approaches are primarily adopted for service discovery: Registry based Approach and Registry-less Approach. The Registry-less approach usually makes use of overlay networks, hash tables, and other broadcasting/multi-casting techniques. One of our works, proposes a registry-less service discovery approach for crowdsourced SOA environments. However, we believe that registry based approaches are more aligned with the requirements of the constrained SOA environments and SOA principles. Therefore, in the proposed work, we perform service discovery using the former i.e. "Registry based Approach".

We surveyed existing literature from the late 90's. We classified the registry based approach into two broad categories: the Centralized Registry Approach and the Distributed Registry Approach. These two can further be classified into those for mobile environments and those for non-mobile environment. Our survey includes works from the areas of SOA, peer-to-peer networking, mobile ad-hoc networking.

3.6.1 Centralized Service Registry:

The centralized service registry approach is used in several popular technologies: Service Location Protocols [37], Sun's Jini architecture [38], Service Discovery Services [39], Microsoft's Universal Plug and Play (UPnP) [40]. These service discovery infrastructures rely on a central registry for discovering capable services. The service information is stored at a centralized registry. All registry related operations are performed by a single entity.

One well known example of centralized service registry architecture for

web services is UDDI (Universal Description Discovery and Integration) [10]. UDDI is not the service registry itself. However, UDDI is the specification of a framework for describing web-services, registering web-service, and discovering web-service. Several data structures and APIs have been published for describing, registering and querying web-services through the UDDI. The ebXML (electronic business XML) [11] standard is another example of a centralized web service registry architecture. Hoschek [41] presented a grid based hyper registry for web services in peer-to-peer networks. The registry is an XQuery based centralized database that manages dynamic distributed contents.

Juric et al.[42] proposed an extension to the UDDI for incorporating version support for services. They presented modifications to the category tag of business service and tModel of the UDDI infoset with the intent to introduce service interface versions in UDDI and WSDL. Bernstein and Vij [43] proposed the use of XMPP for intercloud topology, security, authentication and service invocations. In some ways this work is similar to ours. However, our work focuses on registry management in mobile based service oriented architecture. We focus on managing the service registry in a distributed manner over resource constrained mobile devices. Seto et al. [44] proposed a service registry for ubiquitous networks to dynamically discover service resources. Their registry divides the service operation into the source, transformation, and sink, specifies physical meta-data to manage devices, and associate a keyword with it.

Feng et al. [45] proposed a registry framework to include interoperability among various semantic web service models. For this, they made use of registry meta model for interoperability-7 and several mapping rules for handling semantic mismatch between services. Feng et al. [46] proposed a service evolution registry where providers can register their service evolution information

and consumers can be sent an alert regarding the service evolution. Their work manages service versions along with their dependencies and discrepancies.

More recently [47] presents the idea of using object relational databases in the service registry. The registry extends the search to include search on the basis of service commitments and service expectations.

3.6.2 Centralized Mobile Service Registry:

Some existing work [48][49] discusses the possibility of centralised service registries for mobile environments. Diehl et al. [48] talk about centralized service registries that store the service domain, service types, location and access rights to manage service mobility and adoption of services in wireless networks. Beck et al. [49] propose an adaptable service framework for mobile devices that relies on a central service registry for dynamic service registration and discovery.

Doulkeridis et al. [50] discuss the idea of managing contextual information in service registries. The main focus of the work is to ease service discovery in mobile environments by maintaining context aware service registries. Deepa and Swamynathan [51] talk about a directory based architecture that make use of two integrated architectures: backbone-based and cluster based. Their work was intended to facilitate service discovery in mobile ad-hoc networks and to achieve improved network traffic, response time, and hit ratio.

3.6.3 Distributed Service Registry:

Chen et al. [52] and Sivashanmugam et al.
[7] discuss a few initial approaches to maintain UDDI in a federated environment. One approach [52] supports QoS based discovery from requesters

and provides an aggregated result from the federated registries. The other approach [7] suggests the use of various metadata and ontologies to manage UDDI in a federated environment. Verma et al.[8] propose METEOR-S WSDI, that focusses on providing registries in distributed and federated environments. Extended Registries Ontology was used to provide access to these distributed registries and organizing them in domain based categorization.

The approach discussed in [53] presents a distributed service registry for grid application. The approach utilises Xpath queries and ontological trees for domain based service discovery. Baresi and Miraz [54] talk about an approach to enable heterogeneous federated registries to exchange service information. The approach is based on the publish and subscribe model. Ad-UDDI [9] is a distributed registry architecture that adopts an active monitoring mechanism. The approach extends UDDI to incorporate automated service updates in a federated registry environment.

Treiber and Dustdar [55] propose an active web service registry that make use of atom news formats. RSS software is used in the approach to form an active distributed registry. Shah et al. [56] also propose an RSS-based distributed service registry in the move to achieve global SOA. The proposed registry is intended to provide dynamic discovery using RSS and tries to resolve synchronization issues in RSS. Jaiswal et al. [57] introduce a decentralized registry using the Chord protocol for peer-to-peer environments. The registry comprises distributed hash tables of web-service names and web-service IPs. Their method claims to cater to demand driven web-service provisioning.

Another direction in distributed service registry systems is one meant for cloud environments [58] and [59]. Lin et al. [58] present a hadoop-based service registry for the cloud environment. The work proposes geographical knowledge service registries that are designed to simplify service registration, improve dis-

covery and other registry operations for cloud services. Elgazzar et al. [59] propose to manage a local service registry at the provider's site for the offered services. This local service registry is managed in a distributed manner and has two types of services: local and remote. The paper proposes Discovery-as-a-Service in the cloud environment.

Das Gupta et al. [60] in a more recent paper, discuss about the possibility of a federated registry system for P2P networks. The work makes use of multi-agent based distributed service discovery for non-deterministic and dynamic environments. They propose that super peer nodes manage the distributed service registry and other peers register their services with these registries. Zhang et al. [61] discuss the integration of peer to peer technology with SOA. They talk about self-organizing, semi-structured P2P frameworks for support and propose to use a private service registry at each peer for discovering the manufacturing services. The local private registry of the peer is traversed first to discover a service.

3.6.4 Distributed Mobile Service Registry:

Handorean and Roman [62] propose probably the first work that discusses the possibility of distributed service registries in mobile environments. In their approach, the availability of services is shown in the registry along with an atomic update facility to maintain consistency. Konark [63] is a distributed XML based registry that has a tree structure. A top-down approach is used for the tree, with generic classification of services at the top and specific classification at the bottom. Every node maintains a service registry, where it stores information about its own services and also about services that other nodes provide. The approach provides a semantic service registry and enables servers as well as clients to

register and discover services in a distributed manner.

Schmidt and Parashar [64] present a distributed hash table based approach for distributed registries in peer to peer networks. The approach supports service discovery based on keywords, wild cards on an Internet scale. Indexing of keywords associated with the web service description document is managed at the peers. Tyan and Mahmoud [65] discuss an approach for service discovery in mobile ad-hoc environments. The work considers a registry as a tree and makes this registry available to every node in the network. The approach makes use of a location aware routing protocol and divides the network into hexagonal grids with a gateway for each that have the service registry. Golzadeh and Niamanesh [66] discuss an approach for a service registry system for mobile ad-hoc networks. The approach divides the MANET into clusters with one head for each cluster that acts as a directory for the cluster. The head node has two types of registries: the service provider's registry and the other head node registry.

A decentralized service registry for location based web services is discussed in [67]. The approach is relied on cellular network system and base transceiver station for retrieving the local registry address. These addresses are broadcasted by base station and interested mobile devices download the registry address for location based services. One of the latest works by Jo et al. [68] makes use of bloom filter to manage distributed service registry for mobile IoT environments. Proposed work uses hierarchical bloom filters for reducing message exchanges among registries in move to find available services.

Although there is a good amount of work that has been done towards developing effective service registries, an architecture that enables mobile devices to host mobile service registries and that takes the distinct features of mobile environment into account such as intermittent connectivity, dynamic nature, frequent service description changes is still lacking. Our work focuses on registry man-

agement in mobile based service oriented architecture. We focus on managing the service registry in a distributed manner on resource constrained mobile devices. This service registry contains minimal information about the registered services in a manner that is just enough to uniquely discover the services.

3.7 Summary

In this chapter we looked into, perhaps, the most challenging aspect of implementing an SOA in mobile environments: effective service registries. Our studies show that traditional approaches for implementing service registries (such as UDDI) cannot be directly adopted in mobile environments, given the dynamic, volatile and uncertain nature of such environments. A novel approach to manage service registries 'solely' over mobile devices was proposed that effectively addressess issues specific to mobile environments and enables run time service discoveries in peer-to-peer manner.

We evaluated the approach by developing a prototype and deploying it over real mobile devices. To emulate real world usage as closely as possible we requested volunteers to deploy our prototype on their personal mobile devices and continue doing their routine tasks. The experimental results indicate that the proposed solution is an effective enabler for SOA in mobile environments. We performed several experiments to confirm the efficacy of the prototype across several parameters such as: timely performance, battery consumption, effect of the random/nomadic behaviour of people carrying mobile devices, conflict with native mobile apps, reliability.

Future work in this direction would be towards mobile service registries that focus on QoS factors unique to mobile environments. Future work will also tackle security related issues, dynamic service group splitting in mobile service

registries. Further, the availability of the service registries will be a prime focus.

Chapter 4

Dynamic Web Service Description

Over the past two decades, mobile technology has gained widespread popularity and has become a part of day-to-day life. In particular, smart phones and mobile devices strongly impact the way human beings communicate and deal with digital information. The modern era has witnessed rapid advancements in the field of mobile technology and wireless networking. As a response to this advancement and growth, a large number of services are emerging in the market that can provide digital information over hand held mobile devices. With such dramatic growth, smart phones and mobile devices have the potential to become *"service providers"* from merely being *"service consumers"*.

The materialization of the vision to host and provision web services in peer-to-peer manner over mobile devices can bring a new level of usability to mobile users. The mobile web services will allow the mobile user agent to directly interact with other mobile user agents. This reduction of human intervention in service provisioning will speed up service execution, limit the chances of error, automate redundant tasks, and most importantly reduce the annoyance of human users. A few prospective applications of mobile web services are: 1) Credit cards, debit cards, visiting cards can be provided as web services from mobile devices without the need of having the user search for them or even carry them physically. 2) Localization of personal information can be done seamlessly through services over mobile devices. 3) Modern mobile devices are equipped with powerful sensors. Mobile devices laden with such sensors play the role of a “gateway” facilitating proper access to the capabilities of the sensors. 4)

Mobile services are particularly useful in scenarios where there is little or no preexisting infrastructure by functioning through ad-hoc networks. Examples of these scenarios are war-front, post-disaster relief.

The realization of web services over mobile devices has gained attention in the community. Several approaches have been proposed to provide web services over modern mobile devices [69] [70] [17] [71] [72] [73]. However, a key challenge that is overlooked in mobile environment is *"service description"*. Service description is crucial for the consumers of services to get a sense and better understanding of the offered services and operations. This is of further importance in mobile environments, where the service invocation requires a great deal of service understanding owing to the dynamic nature of transient services. Traditionally, WSDL (Web-Service Description Language) [12] is used to describe and publish the functional description of web-services. Well written WSDL documents provide binding implementation information, detailed description of input-output messages, information on how messages are sent through the network, amongst others. Further, WSDL documents facilitate discovery of the intended web-services over standard service registries such as UDDI - Universal Description Discovery and Integration [10]. On the flip side, however, WSDL documents only provide the functional information of a web-service. They do not provide information on the *non-functional aspects*, *contextual aspects*, and the *business aspects* of a web-service. This information is crucial and of utmost importance in selection and proper usage of available services especially in the context of providing such services over mobile devices.

In this chapter, we present the work that incorporates functional, non-functional, contextual, and business aspects of services along with service collaborators, data source details, hardware aspects, and consumer base to service descriptions for mobile devices. Mobile devices may sometimes act as the "gateway"

to information provided by data sources such as embedded sensors, third party applications, or other mobile services. In such scenarios, the mobile service provider and the data source can be viewed as two separate entities. Both the entities are operated autonomously, with their own unique characteristics and further both entities are prone to failure independently. In such scenarios, traditional service description solutions do not suffice as they consider data source and service provider as indistinguishable entities. An important perspective covered by the proposed mobile service description is: "data source" and "mobile service provider" are looked upon as two disjoint and independent entities. Further, we acknowledge the fact that mobile web services are usually light weight and provide limited functionality. Mobile services can be combined and aggregated among themselves to build and compose more complex and useful services. Hence, this collaboration can lead to provide services that can be readily useful in a real world scenario. In the proposed approach, we further incorporate details on the collaborative partners. The goal of this work is to provide various service descriptions and information handy to the service consumer. Access to such information along with functional descriptions at the time of service discovery speeds up the service selection process considerably. Further, this can facilitate the service consumers to shortlist and select the most suitable and optimum service provider beforehand without the need to communicate with individual service providers. The trade-off though in making such additional information available is that it proportionately increases the size of the description document. The mobile service consumers need to perform heavier processing to handle such lengthy descriptions. Furthermore, such description information (which could include availability, location, response time, latency, price, service scope) is subject to frequent changes owing to the very nature of mobile environments. Management of such detailed service description documents at the service registry therefore could easily suffer from consistency issues, lack of

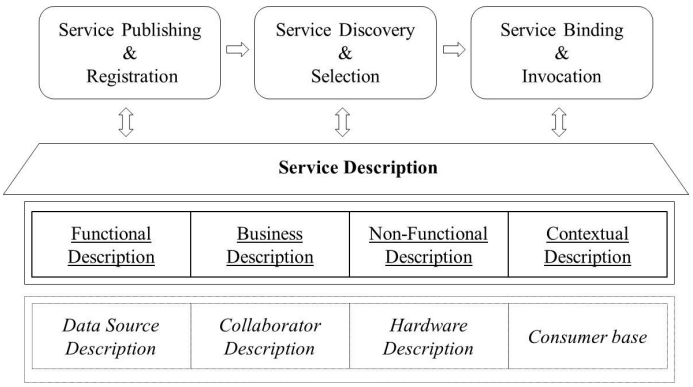

Figure 4.1: Role of Service Description

up-to-date information, and increased network traffic. We tackled these issues and provide a feasible solution for mobile service descriptions.

The aim is to provide a lightweight solution that is dynamically update-able and facilitates rich service descriptions in mobile environments. We emphasize, however, that the proposed solution is not a *replacement* for existing technologies but one that complements it. It acknowledges the heterogeneity of the environment that supports a co-existence of wired, wireless, and mobile devices. The idea is to extend the WSDL 2.0 [12], to incorporate non-functional, contextual, business, data source, collaborator information. The extension takes into account the constraints and issues of mobile environments. To the best of our knowledge, this is the first attempt at providing a lightweight yet exhaustive service description solution that facilitates dynamic updates in mobile environments.

4.1 Background

An effective way to make the most of mobile services and easing service consumers interaction with mobile services is a well defined “Service Description”.

This enables service consumers to effectively discover and use the offered mobile service in peer-to-peer mobile environment. Figure 4.1 depicts the role of service description in service systems. In a service system, the usual stages are: Service Publishing, Service Discovery, Service Selection, Service Binding, and Service Invocation. Service description is an integral part in most of these stages and therefore the role of service descriptions can not be underestimated. A service description is a means to express the characteristics of the offered service to prospective consumers. Well expressed descriptions are important for service consumers to comprehend of the offered services and operations. Service descriptions provide clear and structured instructions on how to invoke a service which is particularly important to first time service consumers. Moreover, an exhaustive service description eases device-to-device communication by automating various stages of service systems (as shown in Figure 4.1). Hence, the our prime focus in this chapter is service description for mobile services.

In the proposed work, we propose a solution to complement the existing technology particularly WSDL (being one of the widely used languages for service description). We work towards extending the features of the WSDL document to accommodate the needs of the mobile environment. WSDL already provides a concrete fundamental support to the web service technology that can be adapted for mobile environments to describe and publish functional parts of the mobile services. In the current context where wired legacy systems and modern hand-held mobile devices coexist, a service description framework that works well in the heterogeneous mixture of diverse technologies and provides a platform for interoperability is required.

To the best of our knowledge WSDL 2.0 is best suited for such requirements. WSDL 2.0 is capable of describing both the major web service technologies: *SOAP based and REST (Representational State Transfer) based*; this is possi-

ble as WSDL 2.0 has good support for describing HTTP bindings. WSDL 2.0 further provides a generic mechanism to define service operations using Message Exchange Patterns (MEP) [12]. This feature encourages message-oriented operations and supports arbitrary message exchanges that are pertinent for heterogeneous mobile environments. Although several other description languages have been proposed since the inception of WS-* technology, most do not take into account the distinct nature of mobile environments. This, along with the wide acceptance of WSDL has made us rely on it for the functional description of services in mobile environments. Furthermore, continuing with WSDL would require least tinkering with existing protocols and technologies.

Being XML based, WSDL 2.0 has very convenient in-built extension capabilities that can sufficiently cater to our requirements. Our approach is to extend WSDL 2.0 to accommodate various other description aspects in addition to the functional description that it already takes care of. For this, we have utilized the "import" statement of WSDL 2.0 for linking physically separate description documents. These partitioned descriptions enable lightweight, dynamic, and consistent management of the overall service description.

4.2 Problem Statement

Several heterogeneous *mobile devices* constitute the mobile environment. These devices could be of varying processing capabilities, power requirements, memory, transmission protocols. Further, these devices are prone to uncertain behavior and dynamic changes as they usually are in continuous motion and they can randomly join/leave the network. Therefore, a holistic service description mechanism is required for services hosted by such mobile devices that comprehensively covers the various unique aspects of the mobile environment. Merely

a functional service description does not provide enough information to the service consumer for service selection in such environments. Service selection made on the basis of only functional service description may lead to the invocation of an obsolete service or an off-line service provider.

A novel approach is required that takes into account the distinct nature of mobile environments and that considers service aspects in addition to the functional description, such as the non-functional, contextual, data source description, hardware, and business descriptions of a mobile service. In the current scenario *legacy wired systems* and *modern wireless mobile systems* coexist. Therefore, a completely novel architecture that intends to replace the existing solutions is undesirable. A service description solution that complements the current solutions and also has added features catering to mobile environments is the need. As elaborated in the earlier sections, in mobile environments the service descriptions require frequent updates owing to regular context changes, non-functional and/or business related changes. Moreover, mobile environments are prone to failures reasons being frequent network outages, battery limitations and are usually constrained in terms of processing power. This leads to the added requirement of a 'lightweight approach' for such service descriptions. Further, keeping in mind the distinct nature of mobile environments, information about the data source, collaborator, and hardware becomes an important parameter during service selection. Hence, a service description approach that considers the issues associated with mobile environments and at the same time blends with the existing technological solutions is required.

We summarise the requirements of mobile service description as: detailed (including non-functional, contextual, business, collaborator, data source, hardware aspects in addition to functional); run-time update-able (i.e. dynamic); and lightweight.

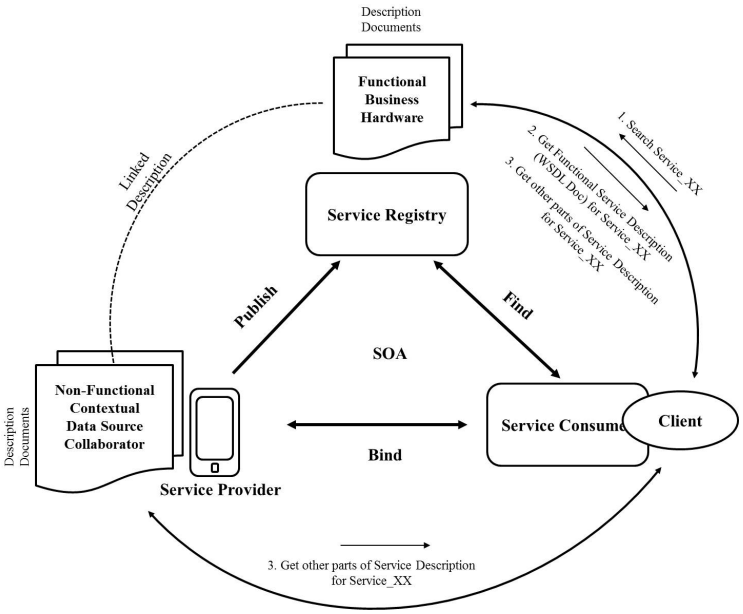

Figure 4.2: Mobile Service Description

4.3 Proposed Approach

As discussed, we intend to use and extend the existing technologies and tools for service description in mobile environments. The mobile services are provided by mobile devices and may be consumed by another mobile device in peer-to-peer mobile environment. Therefore, we propose to extend WSDL for service description to accommodate the distinguished requirements and features of services in mobile environments. In the subsequent subsections, we provide details about the proposed approach:

4.3.1 Design Concept:

We propose to use the following service description documents for mobile services:

1. Functional Description Document.

2. Non-functional Description Document.
3. Contextual Description Document.
4. Business Description Document.
5. Data Source Description Document.
6. Collaborator Description Document.
7. Hardware Description Document.

WSDL documents are widely used for service descriptions. These documents provide detailed functional description of services. Hence, we rely on WSDL documents for functional description of the services provided over mobile devices. In this work, we propose to link the WSDL document with other description documents mentioned above using the "import" statement defined in WSDL. The "import" mechanism allows referral to other WSDL documents defined elsewhere. We use this to connect descriptions that are split across multiple documents for the mobile service.

Figure 4.2 shows an abstract view of the proposed approach. There are three primitive entities: Service Provider (mobile device hosting a service), Service Consumer (mobile or non-mobile device), and Service Registry (mobile registry or traditional non-mobile registry). These entities have the Publish /Find /Bind relationship between them as shown in Figure 4.2. The service descriptions are split into multiple documents and placed at the Service Registry and the Service Provider (i.e. the mobile device in this case). The motive behind this splitting and placing of multiple parts of the description at different locations is:

- to facilitate faster, independent, and dynamic updates related to service provider in the descriptions and keeping the description up-to-date.
- to maintain the overall consistency of the description in case of simultaneous updates.

- to provide a lightweight and detailed service description.

The descriptions of a mobile service may be dynamic and subject to updates regularly. This includes descriptions related to the current network (Wi-Fi or GSM), location, current availability status of the mobile service provider hosting the service. These updates are managed by various independent entities or authors and, therefore, the mutual independence of separate description documents saves the hassle of inconsistent updates. For instance, the non-functional description of a mobile service can be managed and updated by a third party auditor or a broker, whereas the contextual description can be managed by a simple mobile application residing on the same device. This demonstrates the efficacy of splitting and delocalising service description documents in the mobile environment.

Registration of services hosted over mobile devices with a service registry is assumed to follow the same process as in traditional systems. In case of UDDI, the service registration involves the bindingTemplate, businessEntity, businessService, publisherAssertion, tModels [16]. Our approach extends and makes use of WSDL that primarily affects the tModels. We are not proposing any change to the fundamental structure of the WSDL. Hence, the existing process for service registration and publishing is adapted. The proposed approach, therefore, blends well with wired systems and existing SOA standards. The main differences, however, lie in the service description retrieval process. Following are the steps used in description retrieval in the proposed approach (as shown in the Figure 4.2):

1. Search a service at the service registry.
2. Retrieve the functional service description from the service registry.
3. Retrieve the rest of the service descriptions from the mobile service provider.

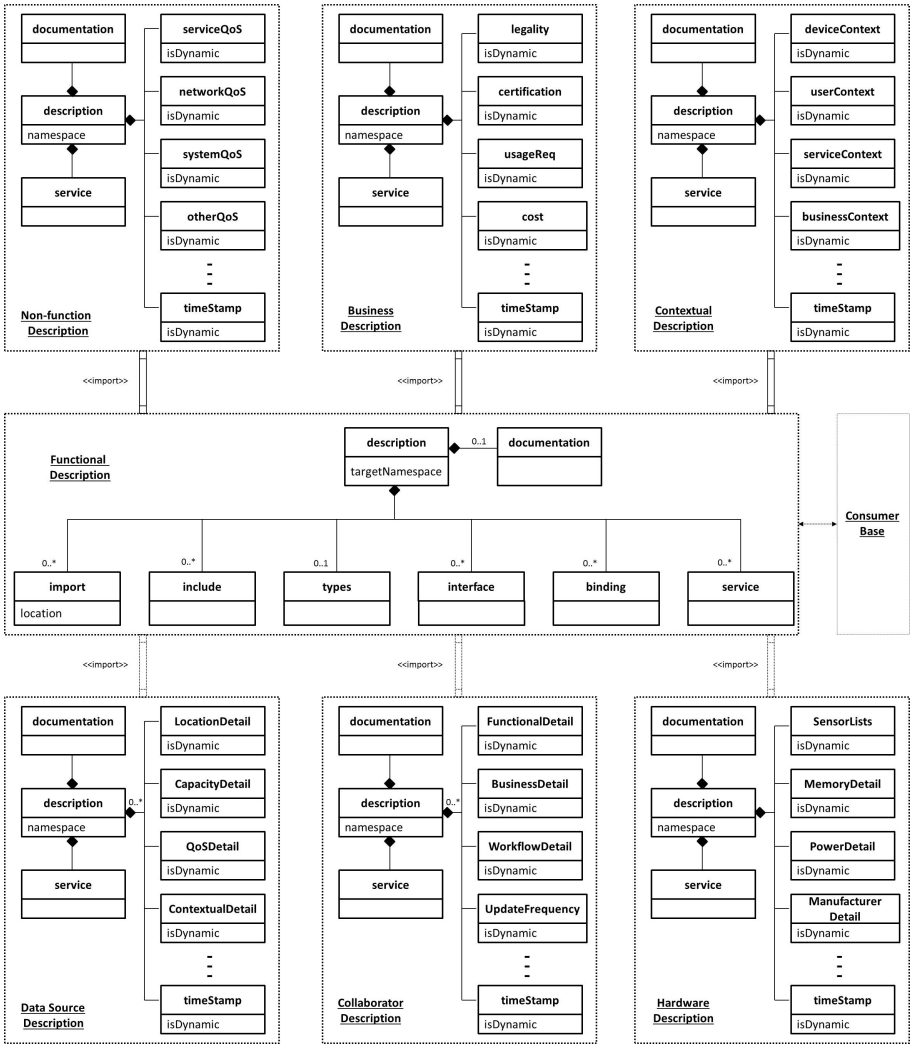

Figure 4.3: Service Description Infoset for Mobile Services

4.3.2 Description Components:

As discussed in Subsection 4.3.1, we propose to use various service description documents that provide a holistic understanding of a mobile service and that describe the mobile provisioned services in a distributed manner. As shown in Figure 4.3, seven service descriptions are interlinked by the *import* statement and an information base about the service consumers. Each element of the descriptions has an attribute *"isDynamic"*, that indicates whether the element is dynamic or not i.e. if the element requires regular updates. This particularly helps in cases where a third party broker or application is responsible for updating the description. We included several description metamodels for mobile services, however, defining all the elements associated with each description metamodel is out of the scope of our work.

A brief overview of each service description is as follows:

Functional Description

As discussed in the earlier subsection, we rely on WSDL 2.0 for a detailed functional description of the service (refer to Figure 4.3). A functional description should describe "What", "How", and "Where" of a service:

- *What* - The function of a service or the operations that a service provides is described by the functional description of a service. The *interface* component of the functional description (refer to Figure 4.3) helps to achieve this.
- *How* - The mode of invoking the service, the message formats, and transmission protocols used by the service constitute an important part of the function description. The *binding* component of the functional description (refer to Figure 4.3) helps to achieve this.
- *Where* - The location of a service or the URI of the service endpoint conveys to the service consumer the address of the service. The *service* com-

ponent of the functional description (refer to Figure 4.3) and the endpoint element in it defines the URI of a service deployment.

This description is usually the first criterion during the service selection process. First of all, the services are shortlisted on the basis of the functionality they provide, subsequent filtering of the services is done through further criteria. In the proposed approach, therefore, we link the remaining descriptions of the service with the functional description using the "import" statement. A typical import statement comprises a namespace and the location of the importing description document: `<import namespace="anyURI" location="anyURI"><documentation/></import>`

Non-Functional Description

The non-functional properties or the quality of service implies the overall performance of the service experienced by the service consumers. We propose to associate a "timeStamp" attribute with the non-functional description document that indicates the time-stamp of the last update. In the uncertain mobile environment, the non-functional properties change with a change of context of service (e.g. location, network, battery etc.). Hence, the time-stamp brings in a degree of certainty to the non-functional description. This helps in better management of dynamically varying description elements. We propose to use the "timeStamp" attribute with the business and contextual descriptions as well.

We propose four 'Quality of Service' (QoS) groups in mobile environments:

1. *serviceQoS:* Service QoS are the quality attributes of a service as experienced by service consumers. A few important QoS attributes include: Availability, Capacity, Latency, Throughput, Performance, Reliability.
2. *networkQoS:* Network QoS include the quality attributes associated with the underlying network used by the service. This network varies from

Wireless LAN, GSM, WiMAX. A few attributes of this group are: Packet Loss, Network Delay, Delay Variation, Bandwidth Capability.

3. *systemQoS:* System QoS are the quality attributes that characterize the whole system instead of just the service or network or third party application. A few attributes in this category are: Accessibility, Security, Usability, Scalability, Interoperability, Robustness (Failure-Management), Extensibility.

4. *otherQoS:* This group or placeholder is proposed to categorize the quality attributes that do not fall in any of the groups mentioned above. This group is extensible and can further categorize service attributes. A few examples of this group are: Testability, Modifiability, audit-ability.

This description is not only limited to these four types and can be extended to include other QoS attribute categories as well. A few pointers to work on non-functional properties of services are [74][75][76][77]. We consider the QoS as the totality of the features and characteristics of the service that are based on its ability to satisfy the implied needs (as per ISO 9000). We only focus on the description of the claimed or known QoS values of the offered services. Determination/Estimation of the varying QoS attribute values is beyond the scope of this work.

Business Description

The business related information of a service is expressed in the business description document. This description mainly comprises:

- *Legality*: The legal obligations or conditions associated with the service are represented by this placeholder. For instance, a service is not available in a specific country, the service is making use of proprietary applications, disclaimer notifications. The legality description is particularly important in case of mobile services as they can easily migrate from one legal boundary to another.

- *Certification*: The certification placeholder specifies the business related certifications or licenses associated with the service. For example ISO certification, SSL security certificates.
- *Usage Requirement*: The preconditions for service usage (if any) and other service usage requirements are described by the usage requirement placeholder. This may include the minimum version of software agents or device capabilities.
- *Cost*: The Cost or pricing placeholder specifies the price for use of the service. The cost placeholder could further be extended to cover discounts, special offers, group pricing for a set of services.

Apart from this, the business description document could include information related to referred service choreography, service offering background (e.g. in-house development, third party applications), service version[78], service scope (what a service covers and what it does not), service type (human provided or manual, automated, semi-automated). The business description information is necessary in mobile environments as it provides greater exposure to the business related offerings of mobile services.

Contextual Description

The often varying context of mobile devices makes the contextual information of utmost importance for services provisioned over mobile devices. A good definition of the term context is given by Bazire and Brézillon [79]: "Context is any information that can be used to characterize the situation of an entity, where the entity is a person, place, or object that is considered relevant to the interaction between a user and its application, including the user and the application themselves".

The context information includes constraints, nature, and attributes that influence the behavior of the service. The description chiefly has placeholders or category for the following elements:

1. *deviceContext:* This represents the context of the mobile device that hosts and provisions the service. This includes device related information, its operating conditions.
2. *userContext:* This placeholder depicts the user (mobile device owner or human service provider) related information. User context is worth considering in mobile environments as the mobile device user's (or human service provider's) activities, behavior can directly influence the service experienced by the service consumers. This may include the user's routine, availability of the service, the user's background (e.g. profession), user's situation (walking, running, driving), location (address, GPS coordinates, time zone).
3. *serviceContext:* Service related contextual information is depicted by this placeholder. A few examples of service context are service domain, service connection preference, service specialisations.
4. *businessContext:* Business contextual information includes information such as preferred business scenario (e.g. combination of user's and device's context), preferred service partners, compositions.

Service and *Documentation* are common attributes in all descriptions. Service specifies the associated service name and its URI for which the description has been provided, while documentation specifies the human readable descriptions of the attributes and service description. These two attributes are borrowed from WSDL 2.0.

Data Source Description

In mobile services, mobile service providers (or mobile devices) often serve as the "gateway" to information provided by the data source which is itself an independent entity. The data source can be anything that provides the mobile service with data. The data source and mobile service provider can be viewed as two independent entities that have independent failures, context, capacity. Examples of such data sources are internal sensors that are physically located

within mobile devices (GPS sensor, digital compass, barometer, pedometer) or external sensors (smart home sensors, body area network sensors etc.) or third party mobile software applications.

As technology progresses towards the Internet of Things (IoT), there is expected to be rapid increase in the number of such data sources. The mobile service provider will then more commonly provide an abstraction for data obtained from such 'things'. The abstraction would be such that the data is made available in formats that are standard and facilitates seamless usage. While for the service consumers, the description of these data sources would provide a better understanding and a holistic view of the service. That, consequently, might become an important service selection criteria. The data source description primarily contains placeholder for the following elements:

1. *LocationDetail:* This comprises the location details of the data source. This includes the GPS coordinates and other the location information.
2. *CapacityDetail:* This comprises the technical details on the data source. This mainly includes the operating capacity of the data source. In certain scenarios, this may include battery information and computation capacity as well.
3. *QoSDetail:* This provides non-functional information on the data source. This may include availability, throughput, reliability, network delay, security information.
4. *ContextualDetail:* Contextual information (as discussed in the earlier point) provides information about the constraints, nature, and structure of the factors that influence the behavior of the data source.

The data source descriptions are provided and managed by the mobile service provider. This document could further be viewed as: Dynamic Part (These are the pointers to the inforamtion located at data source and are likely to change) and Static Part (This information is placed at the service provider e.g. Capacity-

Detail, QoSDetail). This description is necessary as it provides greater exposure to the important constituents of the mobile service.

Collaborator Description

Mobile devices are powerful enough to provide services on their own, yet their capabilities can be improved manifold through mobile service collaboration. Description and information on collaborators helps prospective service consumer to take a decision on a service provider.

Collaborator description provides information and conformity to the service consumer about the service being offered. We provide the following placeholders for collaborator description:

1. *FunctionalDetail:* This placeholder is proposed to provide information on what, how, and where of the service collaborator. The functional description of the collaborator (as discussed in section 4.3.2) may be reused.
2. *BusinessDetail:* This placeholder is proposed to provide business related information on the collaborator. This includes legality, usage requirements, and other related aspects.
3. *WorkflowDetail:* This placeholder provides information on the particiapating workflows of a collaborator. This is mainly to provide the information about the collaborator's work assessment.
4. *UpdateFrequency:* This placeholder specifically works towards prevention of out dated information in a large workflow. This enables the service consumer to have an updated service all the time.

The collaborators description could already present as the mobile service description. Therefore, instead of managing all the information at the service provider, it manages a short summary of the functionality offered by the collaborator and provides the pointer to the functional description of the collaborator. WorkflowDetail provides active workflows with a collaborator as few of the workflow may become outdated due to collaborator's service update.

Hardware Description

Hardware details of the service provider are provided in this placeholder. This description is introduced to minimize the need of service negotiation from non-potential providers. Service consumers can assess the service claim and the hardware that is used to provide the service before actually using the service. The following placeholders are introduced for a detailed hardware description:

1. *SensorLists:* Modern mobile devices are equipped with several modern sensors. This placeholder provides a detailed list of the equipped sensors and their functionality.
2. *MemoryDetail:* This briefs the service consumer about the memory details of the mobile device providing the service. Memory details include information on the primary memory and secondary memory of the mobile device. In certain scenarios, this placeholder also includes details about external memory locations (in case cloud storage is used).
3. *PowerDetail:* Power or battery plays an important role in the selection of mobile services. This placeholder provides the runtime power profile (device battery over a specific time period) of the mobile device.
4. *ManufacturerDetail:* This placeholder provides information on the manufacturer, kernel versions, and other device related information. This could further provide information on the manufacturer of the mobile processor, WiFi and bluetooth adapters.

Mobile service provider may fetch the hardware related details at runtime using APIs exposed by the modern mobile operating systems. For example android provides detailed and sophisticated libraries that access the hardware information efficiently. This description document could viewed as two parts: Static Part (comprises the unchanging elements - SensorList, ManufacturerDetail and is placed at the service registry) and Dynamic Part (comprises the changing elements - MemoryDetails, PowerDetail and is placed in the vicinity of the service provider).

Consumer Base Details

The consumer base provides details about the earlier consumers of the service such as number of consumer accessed, location partitioning of consumers. This helps prospective service consumers better assess a service provider. We can further extend this placeholder to include feedback and rank the service providers. These can be used to propose a recommendation system for mobile services. Detailed discussion on such recommendation systems is beyond the scope of our study.

4.4 Evaluation

We have evaluated the proposed approach with the rationale of demonstrating its usability, feasibility in practical scenarios, and efficacy for mobile service description. We have used the following evaluation techniques (as discussed in [80]): 1) Feature Comparison, 2) Empirical Evaluation, and 3) Conceptual Evaluation. Section 4.4.1 provides a detailed feature comparison, Section 4.4.2 discusses the empirical evaluation, Section 4.4.3 makes use of case studies for theoretical evaluation.

4.4.1 Feature Comparison

In this section, we provide a thorough comparison of the proposed approach with existing service description methods available in literature. For this comparison, we have examined 22 related approaches and compared the proposed approach on the following criteria: applicable domain, ability to dynamically update the description, representation style, aspect of description covered, tech-

niques used for description, validation approach for the method, and other salient features. Table 4.1 and Table 4.2 present a detailed comparison of the proposed approach and existing works chronologically in literature .

Table 4.1: Comparison of Proposed Approach with Existing Approaches in Literature

Work	**Domain**	**Dynamic Update**	**Representation**	**Technique Used**	**Validation Approach**	**Description Aspect**
Web Service Description Language [81]	Wired	No	Syntactic	XML	W3C Specification	Functional
DAML-S [82]	Wired	No	Semantic	Darpa Agent Markup Language(DAML+ OIL) ontology	Example	Functional
WSDL extension for Security description of Web services [83]	Wired	No	Syntactic	XML	Not Mentioned	Functional
Security Description Framework [84]	Wired, Wireless	Yes	Syntactic	XML	Example	Functional and Security Description
OWL-based framework of the Semantic Web [85]	Wired	No	Semantic	RDF	W3C Specification	Functional
Formal Service Description Language [86]	Wired	No	Syntactic and Semantic	ForSeL Calculus	Case Study	Functional
Situation Aware Service based Systems [87]	Wired	No	Syntactic and Semantic	Extention of OWL-S with situation ontology (SAW-OWL-S)	Example	Functional and Contextual
Model driven WSDL extension [76]	Wired	No	Syntactic	XML	Example	Functional and Non-Functional
Semantics for service description [88]	Wired	No	Semantic	Distributed Semantic Tree (DST)	Not Mentioned	Functional
WSDL-Lite [89]	Wired	No	Syntactic and Semantic	RDF Schema	Example	Functional, Non-Function and Behavioral
SOAP Service Description Language (SSDL) [90]	Wired	No	Semantic	XML	Not Mentioned	Functional
WSMO-Lite [91]	Wired	No	Semantic	SAWSDL annotations	Not Mentioned	Functional
Web Application Description Language [92]	Wired, Wireless	No	Syntactic	XML	W3C Specification	Functional
WSDL extension for version support [42]	Wired	No	Syntactic	XML	Prototype	Functional
Unified Service Description Language [93]	Wired	No	Syntactic	MOF-based metamodel	W3C Specification	Functional
WSDL extension for non-functional attributes [94]	Wired	No	Syntactic	XML	Case Study	Functional and Non-Functional

Intentional approach to service description [95]	Wired	No	Intentional (From business Perspective)	Intentional Service Modeling for Service Description	Example	Functional
WSDL extension for criteria support [96]	Wired	No	Syntactic	XML	Case Study	Functional
WSDL-temporal [78]	Wired	No	Syntactic	XML	Case Study	Functional
Service description with extended semantic and commercial attributes [97]	Wired	No	Semantic	XML	Prototype	Functional and Business
Context Aware Mobile Cloud Services [98]	Wired, Wireless	No	Syntactic	JSON and HTTP	Prototype	Functional
WSDL extension for mobile environment [99]	Wired, Wireless	Yes	Syntactic	XML	Prototype	Functional, Business, Non-Functional, Contextual
WSDL extension for holistic mobile service description (Presented Approach)	Wired, Wireless	Yes	Syntactic	XML	Prototype, Case Study	Functional, Business, Non-Functional, Contextual, Data Source, Collaborator, Hardware

4.4.2 Empirical Evaluation: Prototype

We put together a working prototype for assessing the feasibility of the proposed approach. Actual mobile devices were used to deploy the prototype. The prototype is capable of managing dynamic service descriptions in a mobile environment. For this, an android application was developed that is capable of communicating with service registries, retrieving the description documents, extracting relevant information from these descriptions, and updating the descriptions dynamically. The Android operating system was chosen to implement the prototype because it is open source, has wide market share and availability. The proposed approach is generic and can be extended for use on any mobile platform. Our experimental setup comprised four mobile devices (including two Samsung Galaxy S Duos with Android 4.0, Google Nexus 7 with android 5.0, and Asus Zenfone 5 with Android 4.4), one laptop (Intel i3 2.13 GHz with 3GB of RAM) and a few instances of the prototype running on virtual instances of android devices on the laptop.

Table 4.2: Salient Features of Existing Approaches in Literature

Work	Salient Features
Web Service Description Language [81]	Web Service Description Language is a W3C recommended XML based interface definition language. WSDL provides machine readable description of the functionality that are offered by a web service. The current version is WSDL 2.0.
DAML-S [82]	This work provides semantics to the web service by the use of DAML+OIL ontology. The objective being making web services computer interpret-able and enabling service discovery, invocation, inter-operation, composition, verification, and other operations semantically enriched.
WSDL extension for Security description of Web services [83]	This work extends WSDL and UDDI to incorporate security features. The extension facilitate both public-key and trust policy. The description supports publication of various security parameters such as provider encryption, public key signatures, access control policies, and data usage policy.
Security Description Framework [84]	The paper presents a scalable security description framework for mobile web services. WSDL is extended such that change in the service context changes the channel security level and AAA service flow.
OWL-based framework of the Semantic Web [85]	OWL-S provides semantic web description for web services and enables semantic based automated service discovery, invocation, and composition. OWL-S was formerly known as DAML-S.
Formal Service Description Language [86]	The work presents a formal service description to represent functional aspects of a service. The services are considered to be "re"-action system that is activated when input is triggered and some precondition holds.
Situation Aware Service based Systems [87]	This work presents a situation based service aware system. An extension to OWL-S has been presented for situation ontology that is incorporated in service specification.
Model driven WSDL extension [76]	The WSDL extension is proposed to incorporate QoS characteristics of a web service in the description. Model driven architecture recommendations were used to carry out meta-model transformation.
Semantics for service description [88]	This work presents semantic service description. A light-weight semantic service description is proposed using distributed semantics trees. These trees are hierarchical representation for service effect descriptions.
WSDL-Lite [89]	The work presents an extended web service description stack that adds a semantic layer to the service description. Web service modeling language is used to express the service description semantics.
SOAP Service Description Language (SSDL) [90]	The work presents an alternative web service description language SSDL. SSDL provides a lightweight solution for SOAP based web services, it is a message-centric approach that fits with SOA systems.
WSMO-Lite [91]	A minimal lightweight ontology for semantic web services is presented. SAWSDL was used to define WSMO-lite for arbitrary semantic description.
Web Application Description Language [92]	Web Application Description Language is designed to provide machine readable, XML based descriptions to the HTTP based web applications and RESTful services.
WSDL extension for version support [42]	This work presents extension of WSDL to support versioning of web service. Service-level and operation-level versioning is handled in the proposed work.
Unified Service Description Language [93]	Universal Service Description Language is a proposed for describing business, operational, and technical aspects of the universal services. It is a general purpose, domain independent language for Internet of Services.
WSDL extension for non-functional attributes [94]	A flexible extension of WSDL is proposed to incorporate non-functional attributes of a web service. Model driven architecture is used to extend the WSDL meta-model. Further few of the requirements of IOT have also been addressed.
Intentional approach to service description [95]	The work presents intentional level service description. Intentional service model is presented to describe the intentional services and register them with the service registry.
WSDL extension for criteria support [96]	WSDL is extended to X-WSDL that included 'criteria' as the non functional property of a web service. 'criteriadefinition' and 'criteriaservice' keyword has been included in the extended WSDL.
WSDL-temporal [78]	WSDL-temporal is proposed as an extension of WSDL to manage the issues related to the change management in the web services. The proposed method allows to multiple version of the interfaces within a single web service.
Service description with extended semantic and commercial attributes [97]	A model to include multiple attributes for semantic and commercial information in service description is proposed. A holistic XML-based description language along with primitive prototype is discussed.
Context Aware Mobile Cloud Services [98]	A user oriented service description is proposed to allow simple user interaction without any technical details. Provision for both REST and SOAP is discussed. The primary focus of the work is to use cloud services from mobile phones using a in-build cloud assistant.
WSDL extension for mobile environment [99]	This work proposes a light weight and extensible approach for service description, that is designed for mobile environments. Dynamic update to the service description is proposed to keep the description up-to-date.
Presented Approach	WSDL 2.0 is extended to provide description for mobile hosted services. Further, descriptions are distributed to the service registry and service providers on the basis of their update frequency. Functional, Business, Non Functional, Contextual, Data Source, Collaborator, Hardware descriptions are proposed to provide a holistic description for mobile services

Table 4.3: Proposed Features and its Prototype Realization

Proposed Features	**Realization in Prototype**
Lightweight	Service descriptions are pulled from the service registry and the service providers on requirement. XML parsing is done by the android toolkit's in-built DOM parser. Used android "service"[1] majorly for development.
Runtime Update	Several Android API's[2] are available to get the battery, network, location, sensor details. A watchdog process is created to update the description with the latest information.
Detailed Description	Managed various descriptions on various documents located at the service provider. Descriptions placed at the service provider provide detailed updated business, contextual, and claimed non-functional information (Figure 4.3).
Extensible	Considered description meta-models/infoset in the description documents, with an option to introduce new parameters in these meta-models based on service description requirements.

[1]http://developer.android.com/guide/components/services.html
[2]http://developer.android.com/reference/android/package-summary.html

The proposed framework was evaluated in a practical setting. We requested four volunteers to deploy the prototype over mobile devices and roam around within the institute campus. We established an experimental wireless network within the institute's building to connect the volunteers' mobile devices, laptop, and its running virtual instances. This is shown in Figure 4.4. During the experiment, the mobile devices followed a random pattern of mobility as the volunteers did not follow any predefined roaming pattern. The battery usage by the prototype on one of the android devices is shown Figure 4.5. This shows minimal battery usage and our prototype also had a small memory footprint of 1.14MB. This clearly indicates the feasibility of the prototype for use in the real world. A brief overview of how the various features of the proposed approach were realized in our prototype implementation is given in Table 4.3.

Two mobile devices and a virtual android instance played the part of the service provider and hosted services along with the description documents as

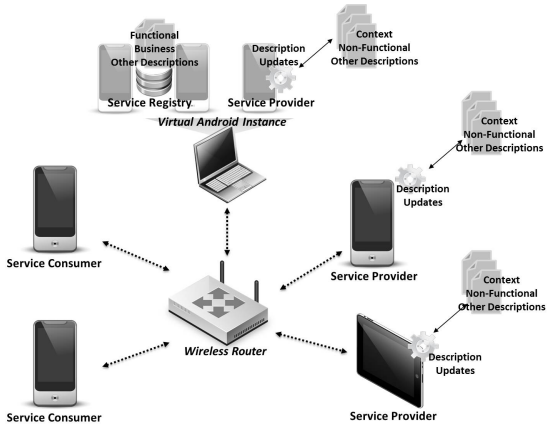

Figure 4.4: Mobile Service Description Prototype

discussed earlier. Further, we engineered a 'watchdog' application to sense the changes in the service provider's contextual, business, and non-functional information and accordingly kept the description documents updated. We made use of a mobile based service registry [100] and hosted the functional description documents over it. The mobile devices acting as service consumers retrieved the functional description document (i.e. WSDL document) from the service registry. Subsequently, the consumers extracted the location information on the other description documents (viz. business, contextual, non-function) from the same WSDL and these were retrieved. (We have conducted the experiment in supervised lab environment and over the real mobile devices, hence the results/values of experiments are likely to change in unsupervised real life scenario depending on the make/model of mobile device, usage habits of mobile owner, movement of mobile owner, network connections, existing apps on the mobile device. Therefore, instead of providing a long list of experimental results (that are bound to change), we provide a detailed feature comparison and case study for the validation purpose.)

We also analysed the UDDI registry to identify if any tweaking was required to accommodate the requirements of the proposed approach. For this, we de-

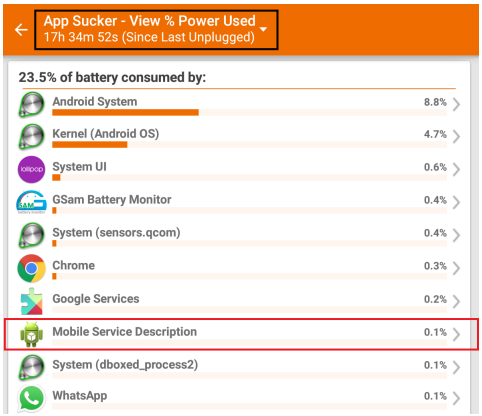

Figure 4.5: Prototype Battery Usage on the Android Device

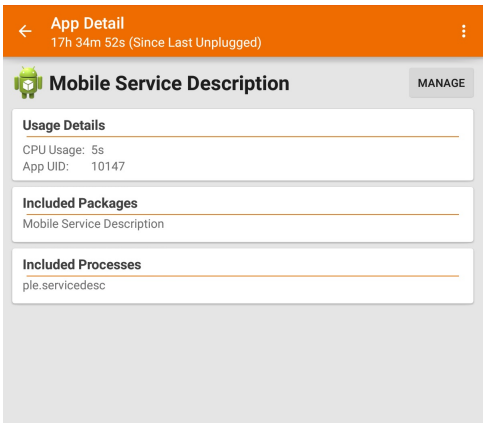

Figure 4.6: Prototype CPU Usage on the Android Device

ployed Apache jUDDI version 3.1.5 on CentOS Linux server 6.2 and SoapUI version 4.6.3 on a Windows 7 machine. One of the concerns was related to the fact that we were using WSDL 2.0 and some of the existing UDDI registries are meant for WSDL 1.1. However, this turned out to be a non-issue as most WSDL 2.0 documents can be mapped to WSDL 1.1 documents. We conclude that given the fact that we made use of WSDL as the base description and that UDDI already maps the WSDL through tModel (as discussed in Subsection 4.3.1), there is no requirement for major change to the UDDI registry in the move to accommodate the proposed framework.

4.4.3 Conceptual Evaluation: Case Study

In Section 4.4.2, we described a real world deployment of the proposed approach by developing a working prototype and deploying it over real mobile devices. In order to further validate the proposed service description, we apply the proposed approach on the case studies and analyse the same in this section.

We acknowledge that the conventional wisdom about case study research has several prejudices. These prejudices have been discussed in detail in an interesting article "Five Misunderstandings About Case-Study Research" by Bent Flyvbjerg [101]. Going along the points mentioned in this article, we present three different service case studies that may exist in the service ecosystem. First, we discuss the case of a shopping mall where the services provide information on the latest offerings: **MallLatestOffer**. Second, we discuss the case of a **SalesmanTracking** mobile service. And third, we take the example of a **CarPoolingMate** service. Table 4.4 discusses these cases briefly. The motive behind discussing these three examples is to assess our description approach for three types of primitive mobile services: 1) Automated Mobile Services: Services that are offered by mobile device itself and do not involve the human. Example- Mobile services that offer sensor provided information, Mobile services that offer personal information viz digital visiting card. 2) Semi-Automated Mobile Services: Services that are provisioned over mobile devices that sometimes requires human intervention. Example- Mobile service that offers meeting availability for a person along with its GPS location. 3) Manual Mobile Services: Services that are offered by human and mobile devices act as a gateway or interface for their services. Example- Mobile service interface for human provided services [102].

We perform requirement coverage analysis of the proposed description ap-

Table 4.4: Mobile Services Case Study Details

Service Name	Service Details
MallLatestOffer	**Type**: Semi-Automated Mobile Service **Dependencies**: Other services from Mall **Functions**: Provides latest offers from various brands of the Mall. Make use of existing services of brands that provides offer details and provides the offer information manually if offer service is not available.
SalesmanTracking	**Type**: Automated Mobile Service **Dependencies**: GPS sensor, Mapping Service **Functions**: Provides location information of the salesman that helps the manager to track the salesman's location and plan their next visit. This make use of mobile phone's GPS sensor and mapping service.
CarPoolingMate	**Type**: Manual Mobile Service **Dependencies**: None **Functions**: Provides the carpooling information. This helps the traveler to fetch the car pooling mate may be in a meeting or a remote public function. This requires to provide the information manually at the provider's end.

proach for these case studies in Table 4.5. This coverage analysis helps us assess whether the proposed mobile description is required for various types of mobile services and whether the proposed description meets the unique description requirements of various classes of mobile services i.e. Automated mobile services that make use of the device's sensor or other services, semi-automated services that may use other services and sensors and also make use of manual information/data supply, and manual services that requires manual supply of information/data (human-automation continuum).

Based on our analysis with these case studies, all services definitely require a functional description. Most other descriptions discussed in the approach are usually also required by most services. The exceptions to this are description about data-source and the collaborator which are not necessary for manual mobile services as there is no other collaborator involved.

Table 4.5: Mobile Service Description Requirement Coverage for Case Studies

Service Description		**Case Study** MLO[1]	ST[2]	CPM[3]
Functional Description	Include	✓	✓	✓
	Types	✓	✓	✓
	Interface	✓	✓	✓
	Binding	✓	✓	✓
	Service	✓	✓	✓
Non-functional Description	ServiceQoS	✓	✓	✓
	NetworkQoS	✓	✓	✓
	SystemQoS	✓	✓	✓
	OtherQoS	✓	✓	x
Business Description	Legality	x	✓	✓
	Certification	✓	✓	x
	UsageRequirement	✓	✓	✓
	Cost/Price	✓	✓	✓
Contextual Description	DeviceContext	✓	✓	✓
	UserContext	✓	✓	✓
	ServiceContext	✓	✓	✓
	BusinessContext	✓	✓	✓
Data Source Description	LocationDetail	✓	✓	x
	CapacityDetail	✓	✓	x
	QoSDetail	✓	✓	x
	ContextualDetail	✓	✓	x
Collaborator Description	FunctionalDetail	✓	✓	x
	BusinessDetail	✓	✓	x
	ReputationDetail	✓	✓	x
	UpdateFrequency	✓	✓	x
Hardware Description	SensorList	x	✓	x
	MemoryDetail	✓	✓	✓
	PowerDetail	✓	✓	✓
	ManufacturerDetail	✓	✓	✓

[1]MLO - MallLatestOffer [2]ST - SalesmanTracking [3]CPL - CarPoolingMate

4.5 Related Work

Service description is an important mean that provides service specification to the prospective service consumers. Although literature emphasize the necessity of service descriptions for mobile web services, the unique service description requirements for mobile services is often overlooked. Existing literature rely on the traditional approaches of service description for mobile services, however, these approaches fall short to cater the specific needs of the mobile environment.

One of the most prominent and widely used description language is WSDL [81]. It has been used traditionally to describe wired web services. WSDL describes various functional perspective of web services including service, interface, operations, endpoint, binding, and type definition. Despite the fact WSDL being effective and popular, it does not cover various other aspects of the service specifications (non-functional, contextual, business, data-source) that are pertinent to the mobile environments.

As WSDL is capable of providing functional information, some of the works focused on extending WSDL to incorporate unavailable properties while relying on WSDL to describe functional aspect. One of the earlier work from Adams and Boeyen[83] extended WSDL that introduced security of web services in the description itself. They added optional security parameters to the WSDL and UDDI in order to provide a secure web service transaction. D'Ambrogio [76] introduced QoS characteristics of web-services in description by proposing lightweight extension of WSDL. In this inspirational work, author made use of metamodel transformation and model driven architecture (MDA). Versioning of web service interfaces was introduced in description by Juric et al.[42], WSDL extension was proposed to support versioning of service interfaces at development-time and run-time. Agarwal and Jalote [103] proposed extensions to WSDL and

suggested end-to-end support for non-functional properties description, measurement, and update. Parimala and Saini[96] proposed an extended WSDL and specified the criteria as non-functional properties of web-services. Change management was focused in the work of Banati et al.[78]. WSDL extension (WSDL-Temporal) was proposed to handle issues related to change management. Their approach suggested management of multiple accessible versions of a web-service. Some other work [94, 96] also suggests incorporation of non-functional attributes to the WSDL as well. Amongst recent work [104] extend WSDL for describing complex geodata in GIS services.

O'Sullivan [75] presents a domain independent taxonomy for conventional services and web services, that is capable of describing non-functional properties. There work provides the ability to communicate non-functional properties along with the service descriptions. Scheithauer et al.[105] presents various perspectives and service properties to specify service description. Zachman framework [106] was used for specifying service properties and their relationship from a service provider's viewpoint for service descriptions depending on the relative perspective. Kritikos et al. [107] presents an extensible ontological specification OWL-Q that provides semantic QoS based web service description. Cardoso et al.[108] and Charfi et al.[93] proposed a new service description language named Unified Service Description Language (USDL). USDL supports human and IT supported services and provides a domain independent description. However, we feel that an entirely new technology is constrained owing to lack of support for legacy systems.

Recently there has been several approaches proposed for cloud services. Galán et al.[109] proposed a service specification language based on OVF (Open Virtualization Format) standard for cloud computing platforms. Sun et al.[110] presents a description for cloud resources of cloud service provider, thus en-

abling the cross-cloud implementations. Liu and Zic [111] proposed cloud# to provide the service delivery transparency and enhance the trust of cloud service users. This cloud service specification focused on describing how services are delivered inside a cloud. Sun et al.[112] presents an interesting survey of service description languages from the point of view of cloud computing.

A few other related endeavours include: an ontology related to the context explained in [113], Dustdar's survey on context aware web-service systems [114], [115] context ontology for mobile environments. [116] is an inspiring work by Dorn and Dustdar that suggests three types of context for mobile web-services: User-related Context, Service-related Context, Task-related Context. [117] is a patent that makes use of WSDL and proposes multi-parted description. Although it only describes functional description in multiple documents. A recent work [118] presents linked USDL that aims to provide automated service trading over the web and provides a vocabulary based on linked data for this purpose. The work [119] takes linked USDL to the next level and adds semantic model with the intent to provide shared service level agreement over the web in automated manner. However, none of these works deal with the description of mobile hosted services specifically.

As stated earlier, a detailed discussion and comparison of the proposed approach and existing methods is already presented in Table 4.1 and Table 4.2. We have studied the existing approaches for service description from the point of view of mobile devices.

Though most of these works do not provide generic specification to fit functional, non-functional, contextual and business aspects of services. Moreover, these works do not cover tackle the specific requirements of dynamic update, separate specifications for data sources, and collaborators. To the best of our knowledge, no existing work is especially dedicated to incorporate the intri-

cacies of the mobile environment. Our work is the first attempt to propose a service description mechanism for such environments.

4.6 Summary

In this chapter, we present a novel, lightweight, dynamic, and extensible mechanism for service description especially designed for services hosted over mobile devices. The proposed service description facilitates automated service discovery, selection, and composition. The approach is designed around WSDL 2.0 with the intent of making it useful across both wired and wireless environments.

The mobile environment is very dynamic and it is normal for service description attributes to change frequently over time. We proposed the partitioning of the mobile service description into multiple parts: Functional Description, Non-functional Description, Business Description, and Contextual Description. Further, we added descriptions to facilitate better assessment of mobile services by service consumers such as data source information, collaborator description, hardware details. The parts of the description that tend to change regularly are made local to the mobile device hosting the service. The motive is to enable seamless dynamic updates in service descriptions without compromising on the overall consistency of the description. The proposed solution has the potential to further ease the service selection for prospective service consumers in peer-to-peer manner. Further, the solution has the potential to help consumers confine service shortlisting and would avoid the obsolete and irrelevant services.

Chapter 5

Dynamic Web Service Workflow

Workflow is an abstract term used to describe the tasks or steps required to be executed with an intent to achieve a common goal. In an SOA environment, a workflow involves the execution of tasks or steps (commonly called work-items) that are distributed and/or decentralized in nature.The workflow therefore mainly involves the flow of information between various executable units i.e. services offered by organizations or people. During execution, the computing requirement by the workflow may change owing to unanticipated requests, user interventions, change in execution environments. To cater to such a dynamic environment, a flexible and elastic resource provisioning mechanism is in order. In this chapter,therefore, we present a dynamic service workflow description and execution mechanism for constrained environments.

5.1 Background

It is common today for business entities and communities to design products and services that can (or are meant to) be used together in collaboration to achieve a larger common goal. Workflow technologies enable collaboration between such distributed services in a flexible and efficient manner [120]. Service-Oriented Architecture (SOA), as discussed earlier, is a multipurpose paradigm executing business processes and offering services for various day-to-day applications [121].SOA paradigms have made it possible for loosely coupled, distributed,

and decentralized services to work in parallel and provider simultaneous offerings catering to the same functionality. The availability of a multitude of such service offerings by business and scientific entities has led to the new concept of *Elastic Computing* in services and workflow execution. Recent advancements in technology [122, 123] have enabled such elastic resource provisioning in workflows (the resources imply other web services available in constrained environments). In addition to this, the autonomy of the available services have made it possible to realize workflows involving such elastic resources in a decentralized manner. Conventionally workflow execution largely relies on a central entity to orchestrate a business process or data intensive applications (such as scientific applications). The orchestrator, owing to its inherent centralized nature is a possible single point of failure and suffers from issues of scalability, reliability, fault tolerance, security, privacy [124].

It is widely acknowledged by the research community that the Internet today is evolving towards what is colloquially termed the '*Future Internet*'. The Future Internet is a broad term comprising endeavours towards making the Internet faster and reducing traffic congestion through various means. In conformance with this, adopting a decentralized architecture achieves a cutback in traffic, congestion, and network load (e.g. RedShift). Considering scalability issues of scientific applications and the Future Internet, "*centralized servers make less sense when dealing with data centric workflows (GBs/TBs)*" [125]. Furthermore, we believe that by specifying a decentralized execution methodology, each 'workitem' is presented with only a partial view, thereby achieving an increased level of privacy.

The benefits of decentralization are further manifested through the phenomenon of Elastic Computing. This attains further significance in constrained environment, where a large number of temporal services exist. Consider a work-

flow in a constrained environment that involves several mobile services. If a centralized orchestrator is handling this workflow execution, it is incumbent upon the orchestrator to provision services/resources to the work-item, enable passage of data to various executing entities etc. thus resulting in unnecessary burden on the controlling authority. In a decentralized scenario, on the other hand, each node is responsible for provisioning extra processing capabilities on its own. Hence, there is very effective distribution of work among all involved entities. Moreover, a centralized engine fosters substantial infrastructure, development, maintenance, and operational costs. For a constrained environment limited by resource and budget constraints, therefore, a decentralized architecture that also acknowledges the dynamic nature of the environment is ideal.

Bell *et. al* [126] states that "*the rapidity with which any given discipline advances is likely to depend on how well the community acquires the necessary expertise in database,* ***'workflow management'****, visualization, and cloud computing technologies*". Taking this line of reasoning as a benchmark for the proposed work, we propose a decentralized solution to execute a workflow in constrained environments using a novel membrane computing approach. We take inspiration from biology to design a workflow management model that executes services, realizing the *process-steps* or *workitems* in a decentralized manner. We use the elementary principles of membrane computing to model the execution of a workflow. We consider each membrane as a service capable of executing a scientific workitem. The services are self capable of discovering other resources (or services) on their own. Each service is provided with *evolutionary rules* (Membrane terminology) [127] [128]), it is through these rules that the services evolve and execute their respective tasks.

Membrane computing paradigm is widely appreciated and accepted in the computer science community as an alternative to solve NP-Hard problems. How-

ever, in this work we focus to use membrane systems for elastic and decentralized workflow description and execution in constrained environments. The benefit of membrane computing comes with a fact that it provides a natural method to model workflows, pass parameters, and elastically increase or decrease resources at runtime. Membrane computing is a branch of computing that takes inspiration from the functioning of living cells. The architecture is called P-Systems. Membrane computing is a branch of computer science which has started to receive much attention in the research community due to its inherent parallel, autonomous, and decentralized nature.

A lot of work in literature, for example [121], [125], [129], [130], [131] etc., focus on achieving scientific workflow execution in the cloud. However, to the best of our knowledge most of them rely on a central orchestrator to elastically increase or decrease resources at runtime, or don't focus on elasticity at all in the constrained environments. Therefore, our objective is to introduce '*autonomy*' in resource provisioning by the workitems themselves.

In this work, we explore the possibility of describing and executing a workflow via Membrane Computing. To demonstrate the viability in actual deployment, we have developed a prototype with real services. We execute real scientific workflows collected from myexperiment[1], and deploy the prototype on a virtualizated platform to test the validity of the proposed work. Moreover, the same workflows are executed within the Institue's Intranet on decentralized nodes. During validation the services exchange data and parameters via a decentralized stable storage space. The stable storage itself was offered as a service, thereby affirming to the standards of the service oriented architecture. Moreover, using virtualized networking, we study the effects of constrained bandwidth capabilities during execution.

[1] http://myexperiment.org

The contribution of this work is two-fold:

1) A novel decentralized and distributed workflow description, with autonomous and elastic provisioning of computing resources.

2) A novel membrane inspired approach for decentralizing workflow execution, with autonomous provisioning of computing resources

For the purpose of clarification, a service based workflow is called a workflow throughout this chapter. A resource is analogous to a service instantiated on a constrained device.

5.2 Membrane Computing Paradigm

Before beginning the discussion of Membrane Inspired workflow management, we present a *'small'* discussion on the Membrane Computing paradigm. It should be pointed out here that the discussion is brief, to the point, and limited to the functionality used in the model. For a detailed description and understanding, we refer the interested reader to [132].

Membrane computing takes its inspiration from a living cell. A living cell is encapsulated by a membrane that separate its internals from the external environment. The cell encloses multiple natural artifacts, e.g. nucleus, golgi apparatus, molecules, vesicles etc. The cytoplasm hold charged ions, whose movement (either inwards or outwards) is controlled by the presence of certain type of proteins. Using chemical receptors, the membrane allows a selective passage of molecules and achieves cell-to-cell signaling.

The pioneering work in the area of membrane computing was proposed in [132]. The author proposed, the basic structure of a membrane consists of sev-

eral separate sub-membranes. The membranes consist of the delimiting region, called multiset, where several different objects are placed. The evolution, manipulation, and transformation of objects is accomplished via evolutionary rules. The objects are transferred from one membrane to another membrane, causing transitions and carrying out the intended tasks. The execution rules are chosen in a non-deterministic manner, thereby presenting an illusion of having infinite parallel computing capability. The application of these rules is conditional i.e. a rule is invoked if certain reaction conditions are applicable. The rules as explained in [132] are of the form $a \longrightarrow b$, where a and b represents multisets of objects. Since, the data and objects are transferred from one membrane to another, the author proposed the notion of 'target indications'. Using target indications, the objects are retained, transferred and consumed. It can be deduced that using these rules the multiset can be written very easily. An example of a rule applied towards object evolution is demonstrated below.

Consider a rule of the form *(ij)* $\longrightarrow$ *(i,here) (i,out) (j,in) (j,out)*. In this example, a copy of i and j is consumed and two copies of i and j are produced. One copy of i is retained within the same membrane (the 'here' indicator), while the other one moves out to the surrounding environment (the 'out' indicator). Out of the two copies of j produced, one goes to the surrounding environment and the other moves inwards toward the inner membrane(s). There exists catalytic rules demonstrating the applicability only in the presence of a certain type of an object, e.g. $cb \longrightarrow cv$, where c is the catalyst. Also, there are non co-operating rules, e.g. $a \longrightarrow b$, membrane dissolving rules, e.g. $j \longrightarrow o\delta$, where δ denotes the membrane dissolving action. It should noted here, the author [132] deliberately points out that the membrane dissolving rule cannot be applied to the skin membrane (for obvious reasons). Further, there are communication rules, symport and antiport, demonstrating how membranes communicate. As outlined earlier, in the real world the membranes communicate via protein channels.

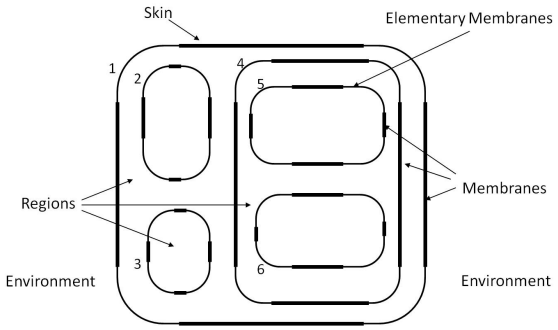

Figure 5.1: Membrane Structure

Therefore, the protein channel and the molecules are the agents of communication in membrane computing. The *'symport rules'* allows for the passage of molecules in one way. On the other hand, the *'antiport rules'* allow for a two way communication via molecules.

There also exists membrane division and merging rules. A membrane division rule is of the form $[_1a]_1 \longrightarrow [_2b]_2[_3c]_3$ (a membrane is denoted as '[ ]', [132]), while a membrane merging rule is of the form $[_2e]_2[_3f]_3 \longrightarrow [_1d]_1$. Further, there exists endocytosis rules, exocytosis rules, gemmation rules etc. The rules are applied locally in each membrane in a non-determinstic, maximally parallel manner, thereby causing *transitions* in the system.

A membrane structure is graphically represented by a Venn diagram as shown in Figure 5.1 [128], where a membrane may contain other membranes. The membrane structure is a hierarchical arrangement of membranes that are embedded in the *skin membrane* separating the inner membranes from external *environment*. *Elementary Membranes* are the membranes that do not have any other membrane inside. The membranes and the regions delimited by them are labeled with positive integers to address them distinctly. Furthermore, each region contains a *multi-set of objects* and a *set of evolution rules*.

At this point, it is worth describing the formal *basic* (there are multiple definitions) definition of Membrane Computing. A membrane structure (as shown in Figure 5.1) can be represented by a string of matching parenthesis as in:

$$[_1[_2]_2[_3]_3[_4[_5]_5[_6]_6]_4]_1$$

Definition 1: The basic (there are multiple definitions) definition of Membrane system (Π) according to Gheorghe Păun [127], is as follows:

$$\Pi = (O, T, C, \mu, w_1, w_2....w_m, R_1, R_2....R_m, i_0)$$

where:

i) O is the set of all objects.

ii) T is the output alphabet and $T \subseteq O$.

iii) C is set of catalysts and $C \cap T = \emptyset$.

iv) μ is the membrane structure consisting of m membranes.

v) $w_1, w_2....w_m$ are the multisets of objects over O with the regions 1,2,....m of μ.

vi) $R_1, R_2....R_m$ are the evolutionary rules over O for each of the m membranes.

vii) i_0, $m \in i_0$ represent the label of the output region.

In this work, we try to use these elementary concepts to achieve a decentralized workflow execution. Based on the discussion so far, it is understood that this paradigm has a natural orientation, and can execute with any type of computation problem (e.g. SAT [133], TSP [134] etc). It is due to this feature that it has received substantial attention in literature from the time of its inception. It can be deduced that the membrane computing paradigm allows a natural

metaphor and an intuitive mechanism to model the complex behavior of workflows. As discussed earlier the paradigm allows communication rules (symport and antiport), membrane dissolving rules, membrane division and merging rules etc. Using these rules, workflow constructs and functionality can be very easily managed. Further, applying the evolutionary rules, a workflow itself can be modified dynamically (via endocytosis, exocytosis, gemmation etc.).

5.3 Membrane Inspired Dynamic Workflow Description

Workflow can be defined as the computerised facilitation or automation of a business process, in whole or part [135]. Workflow is often associated with the business processes of an organization. However, workflow is also useful in areas such as personal computing, scientific computations, where various computational tasks are processed in a specifed order to achieve a goal. To proceed with workflow execution, there are several constructs available in literature. In a broad sense, workflow specifications may have varied perspectives [136]:

- *Control flow perspective* describes the flow of execution order among different tasks and work-items of a workflow. The control flow can be specified following certain basic patterns, such as: Sequence, Parallel Split, Synchronization (AND-join), Exclusive Choice (Decision), Merge.
- *Data flow perspective* deals with the flow of processing data needed to execute a workflow. This may include a business document or an object supplied to the workflow, local variables generated at the time of execution of individual work-items.
- *Resource flow perspective* provides the necessary infrastructure requirements needed for an efficient workflow execution. The requirement may range from human provided services to machine based physical resources.

Furthermore these workflow specifications can be sub-categorized into: *Abstract Workflow* and *Concrete Workflow* [137]. An abstract workflow specifies the solution of the problem along with the input data, but without containing any means for actual execution. In contrast, a concrete workflow specifies the mapping between physical resources responsible for executing (such as services provided by machines or human) the abstract work-items.

Definition 2: A workflow can be specified as:

$$W = (F, D, T, R)$$

where:

i) F is the set of functions or work-items in the workflow.

ii) D is the working dataset, where data $d_{i,j}$ is intermediate input for f_j produced by f_i, where $d_{i,j} \in D$ and $f_i, f_j \in F$.

iii) T is the set of transformation rules depicting data and control flow dependencies.

iv) R is the set of physical resources required to realize work-items F.

A workflow W is represented as the set of work-items $f_1, f_2, f_3, ...f_k \in F$ having working dataset $d_{i,j}$ where $i, j \leq k$. When f_i does not have a predecessor, it is called as the initiating work-item f_{init} and when f_i does not have any successor, it is called a terminating work-item f_{term}. $d_{init,i}$ is initial data set given as input for starting the workflow and $d_{j,term}$ is the output produced as the result of the workflow enactment. $d_{(1,2,3,...k-1),k}$ shows merging of various data input at f_k. In a special case, $d_{i,j} \mid j = 0$, intermediate result is not consumed by any other work-item $f \in F$. $t_{i,j} \in T$ is the transformation rule applied in order to progress the execution from f_i to f_j. $r_{(l,m,n)} \in R$ is the resource executing f_l work-item with n instances of the resource with id m.

Advancements in the field of distributed computing (such as cloud computing, peer-to-peer computing, mobile computing etc.) have enabled several providers to offer services for the realization of workflow in distributed, decentralized and automated manner. These providers are also equipped with technologies to provide the services elastically. The term "elastic" here stands for the dynamic provisioning of resources. Our first aim is to propose a generic framework for dynamic workflow description, using P-systems for depicting the decentralized, autonomous, and elastic enactment of the workflows. We focus on defining and executing the workflows using P systems. Our definition and execution relies on the membrane vision of every *"work-item"* or *"process step"* involved in a workflow. In our approach, the membrane represents the service executing a work-item along with the data they process and possess, control-flow information, resource allotment information in the form of evolution rules of membranes. In this work, the following types of evolution rules are considered:

i) Data-flow rules: Rules describing the flow of data for executing the workflow.

ii) Control-flow rules: Rules describing the flow of execution control for coordination among the various services executing the work-items of the workflow.

iii) Resource-provision rules: Rules describing the computing resource allotment during the execution of a workflow.

The proposed P system is able to depict the "elasticity", "parallelism", and "decentralization" in the workflow enactment :

- "Elasticity" of the computing resources is achieved using evolution rules *membrane division* and *membrane merging*. The new membrane 'forked' due to membrane division will inherit the dataset and evolution rules of its parent. While the membrane merging will take place only for the membranes depicting the same behavior in terms of dataset and evolution rules.

- "Parallelism" of various work-items can be easily depicted by any of the P systems. As membranes in the P system are autonomous and execute in parallel. Hence, P systems are most suitable for describing workflows and their execution.
- "Decentralization" in the workflow enactment is achieved by using an *object*. We use objects in the proposed P system for communication. These communication exercises carry intermediate data and global data needed for executing a work-item. We assume that there are multiple copies of objects available at the membranes.

5.3.1 Workflow Definition

From the membrane computing perspective, we propose a novel and generic definition for a workflow:

Definition 3: A workflow with n work-items is a construct Π:

$$\Pi = (V, C, \mu, (w_1, R_1, R'_1, t_1)...(w_i, R_i, R'_i, t_i), i_0)$$

where:

- V is a multiset of all objects.
- C is the set of catalyst and catalyst do not occur in t_i.
- μ is the membrane structure consisting of i membranes: $[_n[_{n-1}...]_{n-1}]_n$.
- $w_i \in V$ is the multiset of initial contents of region i of μ.
- R_i is the set of data-flow rules.
- R'_i is the set of resources provisioned and and control-flow.
- t_i is the multiset of final contents of region i of μ.
- i_0 is the label of regions.

In the above definition, we have introduced the notation of a specialized operator, the dependency operator, denoted as $\xrightarrow[Dependency]{(Details)}$. This operator gives a sense of determinism in the P system. It is utilized while provisioning resources or resolving control dependencies.

With regard to control flow, the operator specifies the dependency among membranes. This control dependency is shown as $\xrightarrow[Control]{<MembraneList>}$, where $< MembraneList >$ are the list of membranes that must be executed (or dissolved) first for the execution of the present evolution rule. The proposed dependency operator separates the control rules from the rest of the evolution rules and provides synchronization in workflow execution. For example:

$$[_1[_{S_1}a]_{S_1}[_{S_2}b]_{S_2}[_{S_3}c]_{S_3}]_1 \xrightarrow[Control]{S_2,S_3} [_1[_{S_1}a]_{S_1} \ \ e \ \ f \ \ ]_1\delta \tag{5.1}$$

The above rule states that S_2, S_3 should be executed prior to the S_1, though all the three membranes S_1, S_2, S_3 are at the same level. It is a known fact that all the membranes at the same level (part of the same parent membrane 1) get executed in parallel and in a non-deterministic manner. However, while accomplishing a workflow few membranes are required to be restricted from execution (in case of synchronization). In such scenarios, the proposed symbol gives a certain level of determinism and restricts a few of the membranes from execution.

In case of resource provision rules, the dependency operator states the dependency of resources for enacting a work-item. Resource dependency is of the form $\xrightarrow[Resource]{<Dependencyparameter>}$. The proposed operator enables the specification of the resource requirements for a work-item, along with the workflow definition. The dependency operator enables the elastic workflow execution under predefined resource requirements (as in the case of most of the modern pay-as-you-go

technologies e.g. cloud computing)

The present technology enables a client to specify the process steps as well as resource dependencies at the same time. As pay-as-you-go models, such as cloud computing, are becoming more popular, the constrained client has the ability to borrow the resources from outside. The type of resources, the exact specifications, the requirement can be specified that are needed to successfully execute a workflow. These resource requirements are depicted by resource provision rules R_i, (as shown in the definition 3). We propose to use the symbol $\xrightarrow[Resource]{(S,r,o)}$ for specifying the need of the requester:

Resource Dependency (S): Any work-item requires a physical resource such as service, physical devices etc. for execution. In our case, a membrane may depend on other membranes for the execution. For example, requester might need to execute the work-item on a specific membrane/resource. The symbol S shows the dependency on S for executing an evolution rule.

Resources Requirement (r): The symbol r states two types of requirements: Qualitative and Quantitative. Qualitative requirements are the non-functional requirements for the resource S, such as reliability requirement and other QoS (Quality of Service) parameters. While the quantitative requirements are the functional requirements (r) such as memory requirements, processor requirement, number of parallel threads of execution.

Resource Ownership (o): The symbol o shows the ownership requirement of the resource. In case of shared resources, the requirement depicts the dedicated need for a resource. This specifies the requirement for the ownership change of

the shared resource.

Suppose a sequential workflow comprises of three work-items S_1, S_2, S_3. Workflow will only be realized when the second work-item S_2 satisfy additional qualitative requirements (x_1, x_2, x_3) (e.g. x_1 = OS: Linux, x_2 = RAM: 2GB, x_3 = MEM: 50GB). Moreover, for the execution purpose ownership of the resource should be given to first work-item S_1. This can be shown in rule (5.2):

$$\begin{aligned} [_1[_{S_3}[_{S_2}[_{S_1}a]_{S_1}]_{S_2}]_{S_3}]_1 &\xrightarrow[Resource]{(S,r,o)} [_1[_{S_3}[_{S_2}\ \ b\ \ ]_{S_2}]_{S_3}]_1\delta \\ &\Big| \ S \in \{S_2\}, r \in \{x_1, x_2, x_3\}, o \in \{S_1\} \end{aligned} \tag{5.2}$$

Few of the features (of the proposed formal notation) apart from the discussed *elasticity, parallelism, and decentralization* are:

- Reliable Data Exchange: This is usually achieved by stable communication. In proposed definition, this stable communication and data exchange is shown by inter-membrane communication via object passing.
- Heterogeneous Environment: Our approach of workflow states the workflow definition and workflow enactment at the abstract layer. The execution level details for the enactment, such as executing resources are open ended. These execution resources could be web services, computing devices, mobile devices, any constrained device or even human.
- Fault Tolerance: Our approach is fault tolerant for workflow enactment. A failed membrane may produce erroneous result that could be rejected by the next membrane in workflow. Hence, resulting in the redundant execution of the membrane. However, a more sophisticated function can be introduced *Rollback(*Π*,*C_v*)*. Rollback function is able to trace back to last valid configuration C_v of the P system Π.

Before proceeding any further, first we must justify the proposed definition for a workflow. In proposed definition, V is the multiset of all the objects including initial input to the workflow, intermediate processing dataset etc. C also

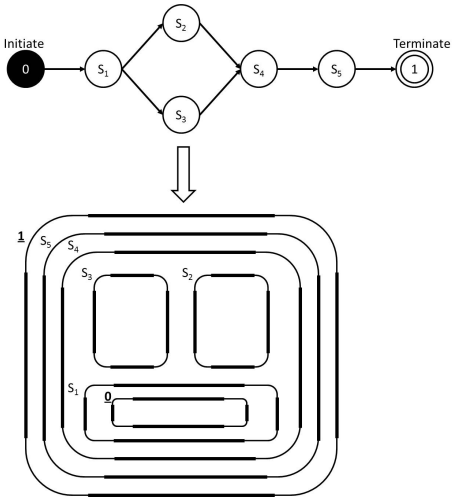

Figure 5.2: A Simple Workflow and its Membrane Representation

termed as the catalyst is the set of all the dataset and conditions that remain in the membrane after the execution. The catalyst does not add to the final contents or result, they help in faster execution of the work-item. μ is the set of all the membranes or work-items with the resources in the workflow. w_n is the initial state of the membranes with which they start executing. t_n is the final content of the membrane once its execution is completed. Hence, for defining the workflow enactment in terms of P system, these are the minimal required elements. In the move to execute membranes or work-items on the resources, evolution rules are required. The definition shows two types of evolution rules: R_n - Data Rules and R'_n - Resource Provision and control flow Rules. Expressive power of P systems enables them to specify both data-flow and control-flow rules in a single evolution rule.

Figure 5.2 represents a simple workflow and its corresponding membrane representation in a graphical manner. The same workflow can also be represented as:

$$[_1[_{S_5}[_{S_4}[_{S_3}]_{S_3}[_{S_2}]_{S_2}[_{S_1}[_0]_0]_{S_1}]_{S_4}]_{S_5}]_1 \tag{5.3}$$

5.3.2 Workflow Pattern using Membrane Computing

There are various workflow patterns [138] which we have solved using the P systems, mainly utilizing membrane evolution rules. Few of the basic patterns are:

Sequence: Any work-item in the workflow is executed only after the completion of its predecessor work-item. Evolution rules depicting the sequence pattern involve dissolution operation after each successful enactment of a work-item/membrane (as shown in rule (5.4), (5.5), (5.6)).

$$[_1[_{s_3}[_{s_2}[_{s_1}a]_{s_1}]_{s_2}]_{s_3}]_1 \rightarrow [_1[_{s_3}[_{s_2}b]_{s_2}]_{s_3}]_1\delta \tag{5.4}$$

$$[_1[_{s_3}[_{s_2}b]_{s_2}]_{s_3}]_1 \rightarrow [_1[_{s_3}c]_{s_3}]_1\delta \tag{5.5}$$

$$[_1[_{s_3}c]_{s_3}]_1 \rightarrow [_1d]_1\delta \tag{5.6}$$

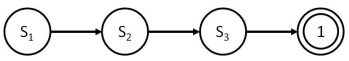

Figure 5.3: Sequential Execution of Work-items

Parallel Split: A parallel split is the point in a workflow where, a single work-item feeds the data/control flow to multiple work-items. These multiple work-items execute in parallel. Rule (5.7), (5.8) shows the parallel split:

$$[_1[_{s_1}a]_{s_1}[_{s_2}]_{s_2}[_{s_3}]_{s_3}]_1 \rightarrow [_1[_{s_2}b]_{s_2}[_{s_3}b]_{s_3}]_1\delta \tag{5.7}$$

$$[_1[_{s_2}b]_{s_2}[_{s_3}b]_{s_3}]_1 \rightarrow [_1\ c\ \ d\ ]_1 \tag{5.8}$$

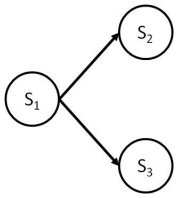

Figure 5.4: Parallel Split of Work-items

Synchronization: Synchronization in the workflow is the point when multiple parallel work-items converge onto a single work-item. A condition involved here is that all the parallel multiple work-items (that are converging) should have been executed only once by following a certain order. Rules (5.9), (5.10), (5.11), (5.12) shows the synchronization pattern:

$$[_1[_{S_1}a]_{S_1}[_{S_2}b]_{S_2}[_{S_3}c]_{S_3}[_{S_4}]_{S_4}]_1 \xrightarrow[Control]{S_1} [_1[_{S_2}b]_{S_2}[_{S_3}c]_{S_3}[_{S_4}\ d\ ]_{S_4}]_1\delta \tag{5.9}$$

$$[_1[_{S_2}b]_{S_2}[_{S_3}c]_{S_3}[_{S_4}\ d\ ]_{S_4}]_1 \xrightarrow[Control]{S_2} [_1[_{S_3}c]_{S_3}[_{S_4}\ d\ \ e\ ]_{S_4}]_1\delta \tag{5.10}$$

$$[_1[_{S_3}c]_{S_3}[_{S_4}\ d\ \ e\ ]_{S_4}]_1 \rightarrow [_1[_{S_4}\ d\ \ e\ \ f\ ]_{S_4}]_1\delta \tag{5.11}$$

$$[_1[_{S_4}\ d\ \ e\ \ f\ ]_{S_4}]_1 \rightarrow [_1\ \ g\ \ ]_1\delta \tag{5.12}$$

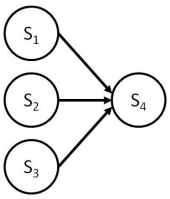

Figure 5.5: Synchronization of Work-items

Exclusive Choice: This workflow pattern is to specify the decision condition, where one of the several incoming work-items is chosen based on some condi-

tion. Rule (5.13) states that service S_4 accepts d from the predecessor services S_1, S_2, S_3, where d is satisfying certain condition, which is not satisfied by e and f. Here $k1 \geq d \geq k2$ where $k1$ and $k2$ are integers.

$$\begin{aligned} &[_1[_{S_1} a]_{S_1}[_{S_2} b]_{S_2}[_{S_3} c]_{S_3}[_{S_4}]_{S_4}]_1 \rightarrow [_1[_{S_4}\ d\ ]_{S_4}\ \ e\ \ f\ \ ]_1\delta \\ &\Big| \ d := \{k1 \geq x \geq k2 \mid x \in \{a, b, c\}\}\ and\ e \neq d\ and\ f \neq d \end{aligned} \tag{5.13}$$

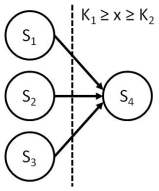

Figure 5.6: Exclusive Choice from Work-items

Simple Merge: Merge pattern specifies that two or more work-items are converging onto a single work-item without any synchronization.

$$[_1[_{S_1} a]_{S_1}[_{S_2} b]_{S_2}[_{S_3}]_{S_3}]_1 \rightarrow [_1[_{S_2} b]_{S_2}[_{S_3}\ d\ ]_{S_3}]_1\delta \tag{5.14}$$

$$[_1[_{S_2} b]_{S_2}[_{S_3}\ d\ ]_{S_3}]_1 \rightarrow [_1[_{S_{3\star}}\ f\ ]_{S_{3\star}}\ \ e\ \ ]_1\delta \tag{5.15}$$

$$[_1[_{S_{3\star}}\ f\ ]_{S_{3\star}}\ \ e\ \ ]_1 \rightarrow [_1\ \ g\ \ e\ \ ]_1\delta \tag{5.16}$$

Rules (5.14), (5.15) and (5.16) shows the merging of work-items S_1 and S_2 in work-item S_3. Rule (5.15) has the dissolution and division of S_3, this states the non-parallel execution of work-items S_2 and S_1. (Symbol $\star$ is used to show the membrane left after division and dissolution).

Elastic workflow enactment is the unique feature of our model. There has been several work on workflow execution in the literature, however, to the best

of our knowledge we are the first to investigate this through membrane computing paradigm. Elastic execution enables the workflow for dynamic procurement of computing resources. This enables the pay-as-you-go models of current technologies on the workflow enactment as well as the load balancing by dynamic provisioning of computing resources. We propose to use membrane division rule for depicting elasticity in the formal notation of workflow.

Consider a membrane is executing a resource intensive task. In the middle of the execution, the load-indicator sensed that new computing resource must be provisioned in order to balance the load. Such a scenario can be shown by evolution rule comprising of membrane devision. Furthermore, the advancements in technology has enabled these computing resources to procure extra resources as per requirement. The membrane is divided into multiple membrane depicting the behavior of dynamic provisioning of computing resource. All the child membranes, resulting from membrane division, have the same set of objects and evolution rules. Hence, children membrane mimics the behavior of the parent membrane.

Suppose a workflow comprises of three work-items S_1, S_2, S_3. Work-item S_2 is resource intensive and require two extra replica of it. This can be shown using the resource dependency operator with quantitative resource requirements.

$$\left[_1\left[_{S_3}\left[_{S_2}\left[_{S_1} a\right]_{S_1}\right]_{S_2}\right]_{S_3}\right]_1 \xrightarrow[Resource]{(S,r,o)} \left[_1\left[_{S_3}\left[_{S_2}\ b\ \right]_{S_2}\left[_{S_2'}\ b\ \right]_{S_2'}\left[_{S_2''}\ b\ \right]_{S_2''}\right]_{S_3}\right]_1 \delta \quad \Big| \; S \in \{S_2\}, r \in \{3\}, o \in \{S_1\} \tag{5.17}$$

5.4 Membrane Inspired Workflow Execution

As discussed, a membrane is considered as a service capable of realizing a single 'workitem'. Each membrane has its fluid (local memory) capable of storing the contextual information and local data (or molecules). The contextual information includes the load-indicator parameters, the inner membranes (the successor workitems), the outer membranes (the predecessor workitems), the resource pool etc (discussed below). The membranes communicate via the 'symport' rules to pass control to the subsequent membrane. The objects and data are passed via the multiset. In the Future Internet, we envision both software and human services to become tradable and executable, therefore the membrane can be a Machine-Based Computing Element, a Human-Based Computing Element [139] or a Network Provisioned Entity [140], thereby a membrane is capable of virtualizing both humans and machines, and offer workflow as a service. The data is equivalent to the proteins capable of penetrating the membrane structure. The membranes do not pass data directly, but rather direct the subsequent membranes to read from the stable storage location (the distributed shared memory). The outermost membrane represents the skin membrane, the inner membranes demonstrates the individual workitems. All membranes pass data and parameters to the global multiset (implemented as a distributed shared memory). Each membrane operate on the objects available in the multiset locally. After completing an execution, the membrane dissolved and left the transformed objects in the multiset. This procedure is followed till the objects are pushed out to the surrounding i.e. the execution of a workflow has completed. It should be noted that when new child membranes are provisioned (resource elasticity property), they (new membranes) also read and wrote to the same multiset. In this case, it is the responsibility of the parent membrane to direct the newly *'born'* child membranes to the appropriate location.

It is understood that if the load of a application is huge, therefore the division process will take some time. However, in the results section we show that the division process, rather than slowing things down, actually speeds up the process. While conducting the experiments, the location of the extra resources were specified via the resource pool (Listing 1), this could be viewed as a dynamic registry (as discussed in Chapter 3). We believe, keeping a resource pool does not involve any financial transactions, only invoking the resources requires monetary arrangements. It should be noted here, the resource pool consisted of the location of the resources to be provisioned, the resources were never used. They were provisioned elastically, under duress only.

To dynamically divide a membrane, a *load-indicator* is instantiated with each membrane. When a threshold value is reached, for either the response time, the throughput, the queue size etc., the membrane division rule is invoked and the parent membrane is divided into multiple child membranes. Whenever, the load-indicator sensed the load has been lightened up, then the membrane merging rule was invoked and the extra provisioned resources were released. It is worth pointing out, during provisioning extra resources, the factors of cost, quality and resource elasticity take a major toll on the application. Finding an optimal balance between these factors is non-trivial and is out of the scope of present work. In the experiments we conducted, we assumed the infrastructure and financial aids are available in plenty.

The multiset is considered as a stable, semantically distributed shared memory offered as a service. Semantic space allow an inherent capability for parallel processing and a distributed stable data management platform. This type of platform allows huge volumes of data to be stored in a semantic format, with event-driven and asynchronous mode of communication. Moreover, the data stored in such a storage location can be retrieved via current access protocols.

Since, the discussion of semantic spaces is out of scope for this work, we direct the interested reader to [141], [142].

As outlined earlier, while elastically increasing resources, the newly provisioned resources have to be directed to the read and write locations. To accomplish this functionality, symport rules were utilized. In this case, the parent membrane directly communicated and sent the location of the stable storage to the child membranes. The motivation to include this feature (not direct messaging) is explained below.

Consider a membrane is executing a data intensive task. In the middle of a transaction, the load-indicator sensed that new resources must be provisioned. In this scenario, since the processing of data has already begun, therefore reinitiating the entire cycle from ground zero does not make sense. Hence, the parent membrane sent whatever data has been processed so far to the stable storage, and directed the new resources to the location. Consider the partially processed data is of size 2 GBs. Writing the data of this size to a single child membrane requires a total transfer of 4GBs (via the stable storage). On the other hand, direct communication requires only 2 GBs of data. Though, a reduction of 2GBs is achieved, but it is noteworthy that if multiple child membranes are provisioned, then every single time the parent membrane has to initiate the transfer. Using a stable storage and directing the children to read from the location reduces the load on the parent. Furthermore, since the storage is stable, therefore if the data is lost or corrupted during transmission, especially in case of mobile membranes with availability issues, a copy will always remain unharmed. Therefore, following this line of reasoning, directing the child membranes to a storage location was considered to be a practical, an efficient and a viable option.

In the membrane computing model, each membrane is assigned a specific

role. Moreover, the membranes are assigned a unique name and an identifier. Membranes assigned to the same role can execute the same functionality. In the Future Internet, the issue of reliability is inevitable, therefore redundant membranes should be kept as back-up in case one fails in the middle of an execution. Next, while executing a workflow there are certain input and output dependencies that must be resolved before proceeding. In the proposed work, these dependencies are specified in an XML format thereby providing a straightforward mapping to a machine readable format. Since, a lot of work [125], [143], [144] etc., has been done to resolve dependencies and automate the execution of traditional workflows, therefore we rely on those procedures to proceed with the execution.

Listing 5.1: ResourcePool

```
<ResourcePool>
<Resource>
<Address>
http://10.200.40.139/Traffic/Diverte/node1
</Address>
<Endpoint>
.
.
</Endpoint>
</Resource>
<Resource>
<Address>
http://10.200.40.132/Traffic/Diverte/node2
</Address>
<Endpoint>
.
.
</Endpoint>
</Resource>
```

```
<Resource>
<Address>
http://10.200.40.131/Traffic/Diverte/node4
</Address>
<Endpoint>
.
.
</Endpoint>
</Resource>

</ResourcePool>
```

Now, to begin with the execution, a workflow is specified to the multiset. Every participating membrane read its corresponding dependencies (a low level locking mechanism). In the experiments, we used an XML schema. The schema is not limited and was constructed using the principles of domain specific languages. Therefore, any type of workflow can be mapped to a machine readable and executable format, thereby presenting a language independent interface. In that case, each membrane must be equipped to handle any type of description. A question arises here: How do membranes understand these specifications? To interpret these constructs, each membrane is equipped with a local interpreter. Hence, an extra layer is added to the membranes to correctly interpret the workflow description (either XML or normal rules).

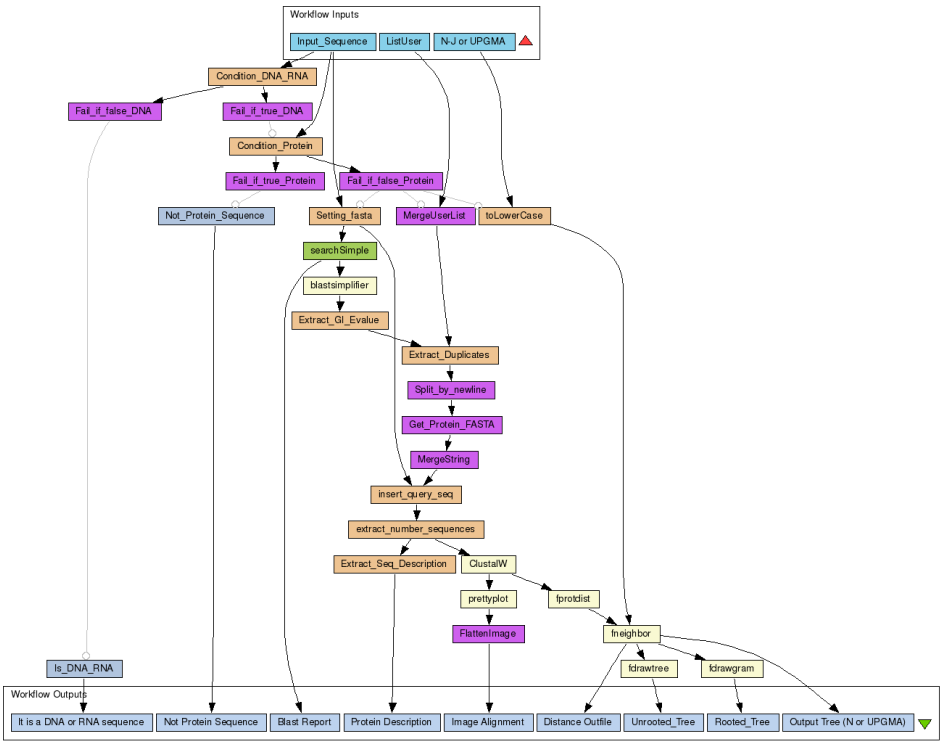

Figure 5.7: One of the Workflows for Experimentation

5.5 Results

5.5.1 Experimental Setup

In order to evaluate the efficiency of the proposed work and the viability in actual deployment scenarios, to execute different scientific workflows, we have conducted experiments with multiple workflows collected from myexperiment.org. The execution of these workflows was achieved in 1) Inside the Institute's Intranet 2)Virtual Machines within the Computing Lab of the Institute. The distributed shared memory was deployed as a RESTful service. In this work, the only a DSM[2], MozartSpaces, was deployed (not semantic) on multiple nodes, and after a specific interval of time, the sub-DSMs synchronized their databases with each other. The application container for the services was Apache Tomcat v7.0.41. However, any other application container for constrained environment such as i-jetty, Tiny Java Web Server (TJWS) could be used.

To Study the effect of a low bandwidth environment, the networking capabilities of each Virtual Machine (VM) was constrained. In this work, one of our motive was to test the model's performance and feasibility to execute a decentralized scientific workflow under duress with limited bandwith capabilities. Under these conditions, we discuss the behavior of the model in the following subsection.

5.5.2 Execution Efficiency

As outlined previously, we have chosen workflows from myexperiment. The workflows are uploaded by the people in the research community, and spans different fields viz. Bio-Informatics, Protein Sequence Analysis, nucleotide and

[2]http://www.mozartspaces.org/

protein sequence analysis. Each workflow was executed multiple times. A total of 13, 28, and 18 services were developed for workflow I[3], II[4], and III[5].

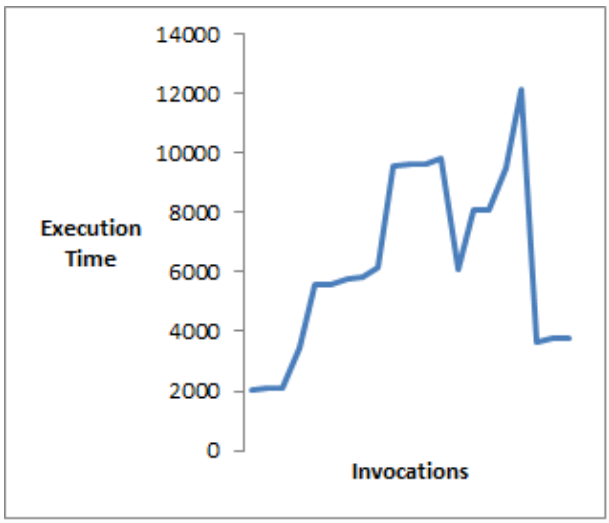

Figure 5.8: Execution Time WF-I No Constraints

During the experiments, it was assumed that each of the discrete workitems are realized by a web service. In Fig 5.8, we have shown the execution time of the workflow I, with no constraints on the bandwidth capabilities. As visible, the execution time of the workflow started at a normal pace. But, when the invocations increased linearly, the execution time followed. The moment, the load-indicator (*of each individual membrane*) sensed duress, a new resource or a VM (each VM was assigned 1 GB of RAM) was provisioned from the resource pool. The newly provisioned resource was made aware of the stable storage location and access methodology.

After provisioning resources, the execution time experienced a sudden drop. This is clearly visible in Fig 5.8. It should be noted here that in the experiments, each workitem provisioned resources on its own, without a central orchestrator.

[3]http://www.myexperiment.org/workflows/244.html
[4]http://www.myexperiment.org/workflows/124.html
[5]http://www.myexperiment.org/workflows/1477.html

The first reduction in the execution time is due to the fact that only a few membranes provisioned an extra resource. Therefore, the drop is not that much steep. However, a sudden decrement in the execution time at the end of the graph indicate multiple VMs (or resources) were provisioned to complete the workflow. We invoked the same workflow multiple times (in regular intervals) so as to test the behavior of autonomously provisioning resources. It should be noted here, when the membranes provisioned extra resources, it happened when the load indicator sensed duress for the *new* incoming request. The already existing requests were not dynamically *migrated* (live migration). In business terminology, the SLAs were violated for the new requests only, there is no need to provision resources for non-SLA violating requests (*principles of cost elasticity*).

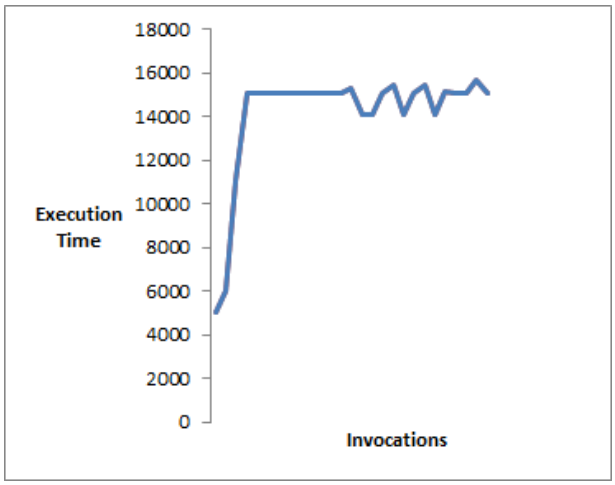

Figure 5.9: Execution Time WF-III No Constraints

The same procedure was followed to execute workflows II and III. The result demonstrating the execution time for workflow II is shown in Figure 5.9 (for workflow III the execution behavior is similar to the one shown in 5.8). As visible, the execution time dropped the moment the load indicator sensed an increment in the load of the individual membrane. It can be seen from this Figure, the execution does not experienced a steep drop, the drop is marginal. It was also observed, the membranes provisioned only one or two (max) VMs.

We believe, this is due to the configuration settings, and the structural design of the workflow (a lot of waiting time to resolve the control dependencies). In the experiments conducted so far, the bandwidth of each VM was not limited.

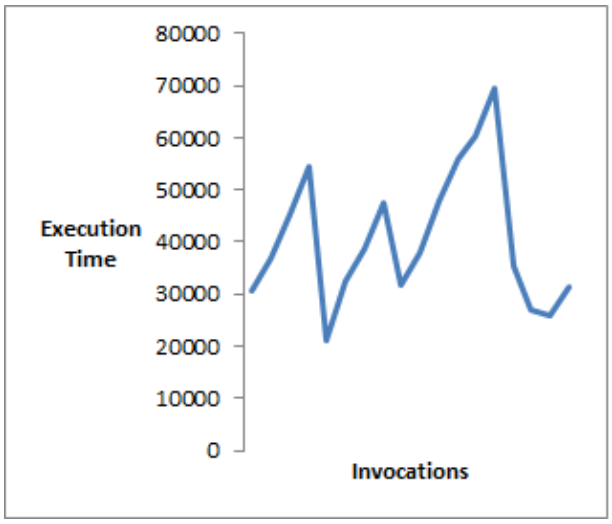

Figure 5.10: Execution Time WF-I Limited Bandwidth

Next, we limited the bandwidth of each VM deployed to 2KBytes/s. The resulting graph for the execution time is shown in Fig 5.10 (workflow I). As demonstrated in the Figure, the provisioning of extra resources resulted the same sudden drop in the execution time. However, in this case the execution time increased. This effect is due to the fact that the bandwidth is limited and each individual membrane required some time to receive the data dependencies. Moreover, it was observed that there were instances when the request was dropped due to severe congestion.

Figure 5.11: Network Performance Limited Bandwidth

In Figures 5.11 and 5.12, we have shown the snapshots of the network perfromance of two VMs chosen at random. It can be seen from the Figures that

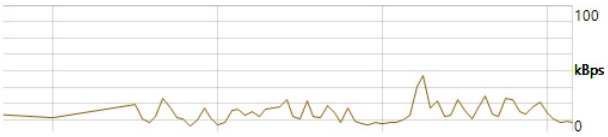

Figure 5.12: Network Performance Limited Bandwidth

the network capabilities witnessed its peak during execution. It is at that instant, the new incoming requests were dropped. Further, the performace showed that whenever the load increased beyond the threshold limit, new VMs were provisioned to balance the load.

5.6 Related Work

There are numerous techniques available in the literature on workflow management and enactment. However, decentralized workflow enactment and dynamic resource provisioning in workflow is rarely targeted. In related work section, we present a brief discussion on these techniques and models.

Fernandez et al [145] propose executing a workflow using the chemical paradigm. Similar to our work, the authors used a decentralized environment, however, they used a centralized shared memory (hence, the authors suffered from scalability issues). Moreover, they kept the services fixed to execute the work-items, with no provision of dynamic adaptations. Further, the issues of elasticity is not addressed in this work. Another work by Caeiro et al [146] discuss about the dynamic workflow enactment using chemical analogy. The authors presents a generalized notion of workflow representation using HOCL (Higher Order Chemical Language). Another work by Németh et al [147] present modern workflow execution on large scale computing devices and propose an enactment model using gamma calculus. In [148] Weske discuss about the different aspects of flexible workflow management and presents workflow

schema using object-oriented approach. There are several literature available on workflow specifications by Leymann et. al. [120], Hollingsworth [135], Aalst et.al. [136], Weske [148].

In literature, there are lot of techniques available to execute a workflow, either centrally or decentrally. In the decentralized scenario, the services share data and parameters either by passing messages directly or indirectly. However, only a few are in the context of dynamically provisioning resources for scientific workflows with actual deployment.

Nature inspired metaphors have caught some attention lately. Based on these approaches we found two interesting metaphors 1) Chemistry 2) Physics. A decentralized framework to execute workitems, is proposed in [144], [121]. Fernandez *et al* [121] propose executing a workflow using the chemical paradigm. Similar to our work, the authors used a decentralized environment, however, they used a centralized shared memory (hence, the authors suffered from scalability issues). Moreover, they kept the services fixed to execute the workitems, with no provision of dynamic adaptations. Further, the issues of elasticity is not addressed. A similar method to achieve orcehstration and choreography is proposed in [144]. The author used the same chemistry model to achieve orchestration and choreography. Next, the work in physics, focus achieving *motion co-ordination*, using the notion of '*Computational Fields*' [149]. However, the focus is distributed motion co-ordination, not scientific workflow execution. A similar to technique to synchronize the motion of 'Unmanned aerial system (UAS)' is proposed in [150]. The author has used the notion of physics inspired co-fields to accomplish this functionality.

A cloud based middleware is proposed in [130]. It is a platform proposed to elastically provision resources at run time. However, the main focus of [130] is not scientific workflow execution in a decentralized environment. In [129],

a comparison between the resource consumption between a high performance computing machine and a cloud based environment is presented. The cloud experiments are conducted on Amazon's EC2. It was found that the resources provisioned from the cloud were not as powerful when compared with a traditional HPC machine. It was further found that though executing scientific workflow was acceptable, the data transfer costs were high. This is one of the factors we will be focusing on in future work. How to find an optimal balance between resource and cost elasticity? [131] introduces the concept of scheduling data intensive scientific workflows in a cloud based environment with virtual clusters. The scheduling is based on the 'iterative ordinal optimization' algorithm. The application of the algorithm produces a significantly lower processing overhead involved in scheduling the workflows. In this work, we also achieved a decentralized workflow execution based on real virtual clusters.

5.7 Summary

In this chapter, we introduced a membrane computing paradigm towards defining and realizing a workflow. The presented workflow model works well for dynamic workflows in constrained environments. The model presented, however, is quite generic in nature and may also be used in other contexts such as workflows involving business, scientific, mobile, cloud applications. The model especially fits well for distributed, decentralized and elastic execution of the workflow. The membranes act independently, following a global behavior, and provision resources autonomously. It was demonstrated that using the presented methodology the resources can be provisioned autonomously at run-time, thereby validating the technique for an actual deployment.

Chapter 6

Conclusion and Future Work

> We can only see a short distance ahead, but we can see plenty there that needs to be done.
>
> -Alan Turing

This dissertation studies the gaps in the current Service-Oriented Architecture paradigm for mobile environments. Subsequently, the dissertation provides approaches to facilitate SOA over modern mobile environments. The proposed approaches address the various challenges of mobile environments to provide dynamic, decentralized, and light weight solutions. Further, our study emphasizes on the fact that the novels solution provided here should complement existing technologies instead of replacing them and thus actuate their applicability. This dissertation is an endeavor to put together the entire triad of effective models for SOA implementation including Registering, Publishing, and Binding, in a constrained environment. Further, efforts have been made at utilizing the approach used for conventional SOA implementations but with the customization to make them as lightweight as possible to suit constrained environments. This has been inspired from the ideology that a newer solution should be complementing the existing solutions rather than replacing them for a wider acceptability.

Chapter 3 deals with the service registry of the SOA triad in the constrained environment. For registering services, a novel approach employing XMPP is presented. Such an approach is very simple, straightforward, and effective. The

idea has widely been used in instant messaging applications and hence its efficacy is widely validated. We presented the approach to facilitate a decentralized and dynamic mobile service registry. The approach makes use of XMPP for managing a dynamic registry along with the availability information of mobile services. A dedicated model suitable for web-services in constrained environments has been put together and demonstrated through a prototype implementation. The prototype demonstrated the feasibility of the approach in a practical setting. The validation of its effectiveness is also done through comparison with UDDI.

In chapter 4, a detailed, dynamic, and lightweight service description framework is presented. Publishing of web services in constrained environments is done through incremental addition of modules to WSDL2.0. The contents included in description are especially designed for mobile environments, considering crucial aspects of mobile services such as isolated data source, collaborating partners, and hardware aspects along with the functional, non-functional, business, and contextual aspects. The description has been partitioned along these lines and various parts of the description are distributed between service registries and the mobile service providers. An up-to-date and light weight description has been achieved by this, without compromising on the overall consistency of the description. The idea is to introduce a WSDL that is seamlessly able to connect and compose web services dynamically irrespective of whether they are available in the wired, conventional, wireless, and constrained domains. The modified WSDL is also made light weight by eliminating certain content/-modules that are not relevant for constrained environments. A prototype of the proposed system has been implemented with the intent of validating the feasibility and efficacy of the approach. The working prototype has been deployed and tested over a small network of mobile devices.

Finally in chapter 5, the binding part of the model for constrained environments is presented. The use of ”Elastic Computing” through membrane computing is presented. The approach provides the much required flexibility of provisioning resources dynamically at runtime binding. The solution provides a novel membrane inspired approach for decentralized workflow execution, with autonomous provisioning of computing resources for dynamic environments. This approach presents a novel mobile workflow description and thus can help in decentralized orchestration of mobile web services. To demonstrate the viability of the same in an actual deployment, we have developed a prototype with real services. We execute real scientific workflows collected from *myexperiment* and deploy the prototype to test the validity of the work. Moreover, the same workflows are executed within the Institue's Intranet on decentralized mobile nodes. During validation, the services exchange data and parameters via a decentralized stable storage space.

The presented works in dissertation are well suited for the dynamic constrained environments and addresses the unique requirements and features of the environment. The proposed novel approaches facilitate the SOA in distributed constrained environment without the need of wired and/or high end systems. The presented works are lightweight, dynamic, extensible in nature. The presented contributions have resulted in an architecture with reduced Total Cost of Ownership (TCO). The present contributions are the initial efforts in the field and may interest the researchers to further explore the fields. The dissertation have few of the open challenges that are yet to be explored such as security and privacy issues in the constrained environment, reward mechanism for the collaborators in crowd-sourced SOA, QoS aspects of mobile services. These challenges are further discussed in the next subsection.

To summarize, the work presented in the dissertation handles three primary

challenges in mobile SOA: Dynamic Service Registry, Mobile Service Description, Decentralized Workflow description and enactment.

Future Work

The work presented in this thesis address the the challenge of effective implementation of SOA in mobile environments. There is, however, substantial room for further research. We provide an overview of these open research areas that we hope to get a chance to work on in future.

In the current work, we have evaluated our approach in a constrained laboratory environment. It would be fascinating to evaluate the same approach for several millions of real service and with real service consumers-providers.

QoS aware mobile web service composition is an open problem in the field as the the nature of QoS expectations in mobile environments is very different from conventional ones.

Security aspects in constrained environments is another open research issue. Mobile services often involve personal data (location, personal information) that is shared between multiple parties. Assessing the trustworthiness of these parties is a challenging issue. Further, the assumption of the thesis is that all the participating nodes (service provider and service consumers) are transparent. It would be an interesting research challenge to incorporate varied levels of privacy in the offered services.

The application of the work of this thesis to a wider SOA scenario over a sustained period of time is expected to throw up unexpected challenges and would also provide data to further refine our findings. In-depth analysis of various SOA environments and subsequent incorporation of the presented dynamic

registry, description, and workflow into such environments would again provide an opportunity to enhance this study.

The employment of semantic techniques in the proposed research would further facilitate the wider adaption of SOA in constrained environments.

Bibliography

[1] M. Weiser, "Hot topics - ubiquitous computing," *Computer*, vol. 26, no. 10, pp. 71–72, Oct 1993.

[2] M. P. Papazoglou and W.-J. van den Heuvel, "Service-oriented computing: State-of-the-art and open research issues," *IEEE Computer*, vol. 40, no. 11, 2003.

[3] S. N. Srirama, M. Jarke, and W. Prinz, "Mobile web service provisioning," in *Advanced International Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*. IEEE, 2006, pp. 120–120.

[4] R. Tergujeff, J. Haajanen, J. Leppanen, and S. Toivonen, "Mobile soa: service orientation on lightweight mobile devices," in *IEEE International Conference on Web Services*. IEEE, 2007, pp. 1224–1225.

[5] F. AlShahwan and K. Moessner, "Providing soap web services and restful web services from mobile hosts," in *International Conference on Internet and Web Applications and Services (ICIW)*. IEEE, 2010, pp. 174–179.

[6] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour, "Wireless body area networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1658–1686, Third 2014.

[7] K. Sivashanmugam, K. Verma, and A. Sheth, "Discovery of web services in a federated registry environment," in *IEEE International Conference on Web Services*, July 2004, pp. 270–278.

[8] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, "Meteor-s wsdi: A scalable p2p infrastructure of registries for semantic publication and discovery of web services," *Information Technology and Management*, vol. 6, no. 1, pp. 17–39, 2005.

[9] Z. Du, J. Huai, and Y. Liu, "Ad-uddi: An active and distributed service registry," in *Technologies for E-Services*, ser. Lecture Notes in Computer Science, C. Bussler and M.-C. Shan, Eds. Springer Berlin Heidelberg, 2006, vol. 3811, pp. 58–71.

[10] Oasis, "Uddi version 3.0.2 spec technical committee draft," http://uddi.org/pubs/uddi-v3.0.2-20041019.pdf, 2004.

[11] F. Najmi, N. S. RosettaNet, I. Bedini, F. Telecom, K. Breininger, J. Chiusano, P. Kacandes, M. MacKenzie, M. Martin, and D. Nickull, "ebXML registry information model," May 2005. [Online]. Available: https://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.pdf

[12] W3C Recommendation, "Web services description language (WSDL) version 2.0 part 1: Core language," June 2007, [Accessed on August 10, 2016]. [Online]. Available: http://www.w3.org/TR/wsdl20/

[13] P. Kotler, G. Armstrong, J. Saunders, and V. Wong, *Principles of Marketing*. Pearson Education, Inc., Upper Saddle River, New Jersey, 1996.

[14] J. Harry Katzan, *Foundations of Service Science: A Pragmatic Approach*. iUniverse, Inc., New York Bloomington, 2008.

[15] J. Wirtz, P. Chew, and C. Lovelock, *Essentials of services marketing*. Pearson Education, Singapore, 2012.

[16] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services: Concepts, Architectures and Applications*, 1st ed., M. J. Carey and S. Ceri, Eds. Springer Publishing Company, Incorporated, 2010.

[17] S. Berger, S. McFaddin, C. Narayanaswami, and M. Raghunath, "Web services on mobile devices-implementation and experience," in *IEEE Workshop on Mobile Computing Systems and Applications*, Oct 2003, pp. 100–109.

[18] M. P. Papazoglou, "Service-oriented computing: Concepts, characteristics and directions," in *International Conference on Web Information Systems Engineering*. IEEE, 2003, pp. 3–12.

[19] T. Erl, *Soa: principles of service design*. Prentice Hall Upper Saddle River, 2008, vol. 1.

[20] M. Lanthaler and C. Gütl, "Hydra: A vocabulary for hypermedia-driven web apis." in *Proceedings of the Workshop on Linked Data on the Web (WWW2013)*, vol. 996. ACM, 2013.

[21] A. P. Sheth, K. Gomadam, and J. Lathem, "Sa-rest: Semantically interoperable and easier-to-use services and mashups," *IEEE Internet Computing*, vol. 11, no. 6, pp. 91–94, 2007.

[22] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, 2000.

[23] J. Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006.

[24] M.-C. Yuen, I. King, and K.-S. Leung, "A survey of crowdsourcing systems," in *IEEE third International Conference on Social Computing (socialcom) Privacy, Security, Risk and Trust (passat)*. IEEE, 2011, pp. 766–773.

[25] D. Schall, H. Psaier, M. Treiber, and F. Skopik, "Engineering service-oriented crowdsourcing for enterprise environments," 2010.

[26] D. Schall, *Service-Oriented Crowdsourcing: Architecture, Protocols and Algorithms*. Springer Publishing Company, Incorporated, 2012.

[27] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 369–392, First 2014.

[28] S. Dustdar and M. Treiber, "A view based analysis on web service registries," *Distributed and Parallel Databases*, vol. 18, no. 2, pp. 147–171, 2005.

[29] P. Saint-Andre, "Rfc 3920: Extensible messaging and presence protocol (xmpp): Core," Internet Engineering Task Force (IETF) proposed standard, Tech. Rep., 2004.

[30] M. Sabou, C. Wroe, C. Goble, and H. Stuckenschmidt, "Learning domain ontologies for semantic web service descriptions," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, no. 4, pp. 340–365, 2005.

[31] M. Allahyari, K. Kochut, and M. Janik, "Ontology-based text classification into dynamically defined topics," in *IEEE International Conference on Semantic Computing (ICSC)*, June 2014, pp. 273–278.

[32] M. Luby, "A simple parallel algorithm for the maximal independent set problem," *SIAM Journal on Computing*, vol. 15, no. 4, pp. 1036–1053, 1986.

[33] J.-S. Han, K.-J. Lee, J.-W. Song, and S.-B. Yang, "Mobile peer-to-peer systems using super peers for mobile environments," in *International Conference on Information Networking*, Jan 2008, pp. 1–4.

[34] P. Saint-Andre, "Rfc 3921: Extensible messaging and presence protocol (xmpp): Instant messaging and presence, october 2004," Internet Engineering Task Force (IETF) proposed standard, Tech. Rep., 2004.

[35] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES/ISSS '10. New York, NY, USA: ACM, 2010, pp. 105–114. [Online]. Available: http://doi.acm.org/10.1145/1878961.1878982

[36] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone." in *USENIX annual technical conference*, vol. 14. Boston, MA, 2010.

[37] E. Guttman, "Service location protocol: Automatic discovery of ip network services," *IEEE Internet Computing*, vol. 3, no. 4, pp. 71–80, 1999.

[38] J. Waldo, "The jini architecture for network-centric computing," *Communications of the ACM*, vol. 42, no. 7, pp. 76–82, 1999.

[39] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz, "An architecture for a secure service discovery service," in *ACM/IEEE*

international conference on Mobile computing and networking. ACM, 1999, pp. 24–35.

[40] B. A. Miller, T. Nixon, C. Tai, and M. D. Wood, "Home networking with universal plug and play," *IEEE Communications Magazine*, vol. 39, no. 12, pp. 104–109, 2001.

[41] W. Hoschek, *Peer-To-Peer Grid Databases for Web Service Discovery*. John Wiley & Sons, Ltd, 2003, pp. 491–539.

[42] M. B. Juric, A. Sasa, B. Brumen, and I. Rozman, "WSDL and UDDI extensions for version support in web services," *Journal of Systems and Software*, vol. 82, no. 8, pp. 1326 – 1343, Aug 2009.

[43] D. Bernstein and D. Vij, "Intercloud directory and exchange protocol detail using xmpp and rdf," in *World Congress on Services (SERVICES)*. IEEE, 2010, pp. 431–438.

[44] H. Seto, S. Matsumoto, and M. Nakamura, "Ubi-regi: Service registry for discovering service resources in ubiquitous network," in *International Conference on Information Integration and Web-based Applications and Services*, ser. iiWAS '11. New York, NY, USA: ACM, 2011, pp. 395–398.

[45] Z. Feng, R. Peng, B. Li, K. He, C. Wang, J. Wang, and C. Zeng, "A service registry meta-model framework for interoperability," in *International Symposium on Autonomous Decentralized Systems (ISADS)*, March 2011, pp. 389–398.

[46] Z. Feng, D. Chiu, and K. He, "A service evolution registry with alert-based management," in *International Conference on Service Science and Innovation (ICSSI)*, May 2013, pp. 123–130.

[47] H. Lakshmi and H. Mohanty, "Extended service registry to support i/o parameter-based service search," pp. 145–155, 2015.

[48] N. Diehl, D. Grill, A. Held, R. Kroh, T. Reigber, and T. Ziegert, "System integration for mobile computing and service mobility," in *IEEE International Conference on Distributed Platforms: Client/Server and Beyond: DCE, CORBA, ODP and Advanced Distributed Applications*, 1996, pp. 44–56.

[49] J. Beck, A. Gefflaut, and N. Islam, "Moca: A service framework for mobile computing devices," in *ACM International Workshop on Data Engineering for Wireless and Mobile Access*, ser. MobiDe '99. New York, NY, USA: ACM, 1999, pp. 62–68.

[50] C. Doulkeridis, E. Valavanis, and M. Vazirgiannis, "Towards a context-aware service directory," in *Technologies for E-Services*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, vol. 2819, pp. 54–65.

[51] R. Deepa and S. Swamynathan, "A service discovery model for mobile ad hoc networks," in *International Conference on Recent Trends in Information, Telecommunication and Computing (ITC)*, March 2010, pp. 135–139.

[52] Z. Chen, C. Liang-Tien, B. Silverajan, and L. Bu-Sung, "Ux-an architecture providing qos-aware and federated support for uddi," in *IEEE International Conference on Web Services*, 2003.

[53] M. Bubak, T. Gubala, M. Kapalka, M. Malawski, and K. Rycerz, "Workflow composer and service registry for grid applications," *Future Generation Computer Systems*, vol. 21, no. 1, pp. 79 – 86, 2005.

[54] L. Baresi and M. Miraz, "A distributed approach for the federation of heterogeneous registries," in *International Conference on Service-Oriented Computing (ICSOC)*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 4294, pp. 240–251.

[55] M. Treiber and S. Dustdar, "Active web service registries," *IEEE Internet Computing*, vol. 11, no. 5, pp. 66–71, Sept 2007.

[56] D. Shah, M. Agarwal, M. Mehra, and A. Mangal, "Global soa: Rss-based web services repository and ranking," in *International Conference on Internet and Web Applications and Services (ICIW)*, May 2010, pp. 256–261.

[57] D. Jaiswal, S. Mistry, A. Mukherjee, and N. Mukherjee, "Efficient dynamic service provisioning over distributed resources using chord," in *International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, Dec 2013, pp. 257–264.

[58] J. Lin, X. Wu, C. Chen, and Y. Liu, "Hadoop-based service registry for geographical knowledge service cloud: Design and implementation," in *International Conference on Information Science and Technology (ICIST)*, March 2013, pp. 961–966.

[59] K. Elgazzar, H. S. Hassanein, and P. Martin, "Daas: Cloud-based mobile web service discovery," *Pervasive and Mobile Computing*, vol. 13, no. 0, pp. 67 – 84, 2014.

[60] S. DasGupta, A. Aroor, F. Shen, and Y. Lee, "Smartspace: Multiagent based distributed platform for semantic service discovery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 7, pp. 805–821, July 2014.

[61] W. Zhang, S. Zhang, F. Qi, and M. Cai, "Self-organized p2p approach to manufacturing service discovery for cross-enterprise collaboration," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 3, pp. 263–276, March 2014.

[62] R. Handorean and G. C. Roman, "Service provision in ad hoc networks," in *International Conference on Coordination Models and Languages*, ser. COORDINATION '02. London, UK, UK: Springer-Verlag, 2002, pp. 207–219.

[63] S. Helal, N. Desai, V. Verma, and C. Lee, "Konark - a service discovery and delivery protocol for ad-hoc networks," in *IEEE Wireless Communications and Networking Conference*, vol. 3, March 2003, pp. 2107–2113 vol.3.

[64] C. Schmidt and M. Parashar, "A peer-to-peer approach to web service discovery," *World Wide Web*, vol. 7, no. 2, pp. 211–229, 2004.

[65] J. Tyan and Q. H. Mahmoud, "A comprehensive service discovery solution for mobile ad hoc networks," *Mobile Networks and Applications*, vol. 10, no. 4, pp. 423–434, Aug. 2005.

[66] A. Golzadeh and M. Niamanesh, "Dsdst - a distributed service discovery approach with service type for mobile ad hoc networks," in *International Conference on Networking and Distributed Computing (ICNDC)*, Sept 2011, pp. 267–271.

[67] M. D'Souza and V. S. Ananthanarayana, "Enhanced lbs discovery in a decentralized registry based web services environment," *Journal of Web Engineering*, vol. 13, no. 1&2, pp. 1–23, Mar. 2014.

[68] H. J. Jo, J. H. Kwon, and I. Y. Ko, "Distributed service discovery in mobile iot environments using hierarchical bloom filters," in *Engineering the Web in the Big Data Era*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2015, vol. 9114, pp. 498–514.

[69] S. N. Srirama, M. Jarke, and W. Prinz, "Mobile web service provisioning," in *Advanced International Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*, Feb 2006, pp. 120–120.

[70] S.-T. Cheng, J.-P. Liu, J.-L. Kao, and C.-M. Chen, "A new framework for mobile web services," in *Symposium on Applications and the Internet (SAINT)*, 2002, pp. 218–222.

[71] M. Adacal and A. Bener, "Mobile web services: a new agent-based framework," *IEEE Internet Computing,*, vol. 10, no. 3, pp. 58–65, May 2006.

[72] M. Hassan, W. Zhao, and J. Yang, "Provisioning web services from resource constrained mobile devices," in *IEEE International Conference on Cloud Computing (CLOUD)*, July 2010, pp. 490–497.

[73] F. AlShahwan, K. Moessner, and F. Carrez, "Distribute provision strategies of restful-based mobile web services," in *IEEE Global Telecommunications Conference (GLOBECOM)*, Dec 2011, pp. 1–6.

[74] L. O'Brien, L. Bass, and P. Merson, "Quality attributes and service-oriented architectures," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-2005-TN-014, 2005, [Accessed on August 10, 2016]. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7405

[75] J. O'Sullivan, "Towards a precise understanding of service properties," Ph.D. dissertation, Queensland University of Technology, 2006.

[76] A. D'Ambrogio, "A model-driven WSDL extension for describing the QoS of web services," in *International Conference on Web Services*, Sept 2006, pp. 789–796.

[77] K. Kritikos and D. Plexousakis, "Requirements for qos-based web service description and discovery," *IEEE Transactions on Services Computing*, vol. 2, no. 4, pp. 320–337, Oct 2009.

[78] H. Banati, P. Bedi, and P. Marwaha, "Wsdl-temporal: An approach for change management in web services," in *International Conference on Uncertainty Reasoning and Knowledge Engineering (URKE)*, Aug 2012, pp. 44–49.

[79] M. Bazire and P. Brézillon, "Understanding context before using it," in *Modeling and Using Context*, ser. Lecture Notes in Computer Science, A. Dey, B. Kokinov, D. Leake, and R. Turner, Eds. Springer Berlin Heidelberg, 2005, vol. 3554, pp. 29–40.

[80] K. Siau and M. Rossi, "Evaluation techniques for systems analysis and design modelling methods : a review and comparative analysis," *Information Systems Journal*, vol. 21, no. 3, pp. 249–268, 2011.

[81] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. (2001, March) Web Services Description Language (WSDL) 1.1. W3C. [Accessed on August 10, 2016]. [Online]. Available: http://www.w3.org/TR/wsdl

[82] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and

K. Sycara, "Daml-s: Web service description for the semantic web," in *International Semantic Web Conference – The Semantic Web ISWC*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 348–363.

[83] C. Adams and S. Boeyen, "UDDI and WSDL extensions for web service: A security framework," in *ACM Workshop on XML Security*, ser. XMLSEC '02. New York, NY, USA: ACM, 2002, pp. 30–35.

[84] M. Morioka, Y. Yonemoto, T. Suzuki, and M. Etoh, "Scalable security description framework for mobile web services," in *IEEE International Conference on Communications*, vol. 2, May 2003, pp. 804–808 vol.2.

[85] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne *et al.*, "OWL-S: Semantic markup for web services," pp. 2007–04, November 2004, [Accessed on August 10, 2016]. [Online]. Available: https://www.w3.org/Submission/OWL-S/

[86] J. Hartmann, S. Rittmann, D. Wild, and P. Scholz, "Formal incremental requirements specification of service-oriented automotive software systems," in *IEEE International Symposium on Service-Oriented System Engineering*, ser. SOSE '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 130–133.

[87] S. S. Yau and J. Liu, "Incorporating situation awareness in service specifications," in *IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, April 2006, pp. 8 pp.–.

[88] H. Pfeffer, D. Linner, C. Jacob, I. Radusch, and S. Steglich, "Towards light-weight semantic descriptions for decentralized service-oriented systems," in *International Conference on Semantic Computing*, Sept 2007, pp. 295–303.

[89] T. Vitvar, J. Kopecky, M. Zaremba, and D. Fensel, "Wsmo-lite: lightweight semantic descriptions for services on the web," in *European Conference on Web Services*, Nov 2007, pp. 77–86.

[90] P. Fornasier, J. Webber, and I. Gorton, *Component-Based Software Engineering: 10th International Symposium, CBSE 2007, Medford, MA, USA, July 9-11, 2007. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, ch. Soya: A Programming Model and Runtime Environment for Component Composition Using SSDL, pp. 227–241.

[91] T. Vitvar, J. Kopecký, J. Viskova, and D. Fensel, "The semantic web: Research and applications," Berlin, Heidelberg, pp. 674–689, 2008.

[92] M. Hadley, "Web application description language," pp. 2007–04, August 2009, [Accessed on August 10, 2016]. [Online]. Available: http://www.w3.org/Submission/wadl/

[93] A. Charfi, B. Schmeling, F. Novelli, H. Witteborg, and U. Kylau, "An overview of the unified service description language," in *IEEE European Conference on Web Services (ECOWS)*, Dec 2010, pp. 173–180.

[94] C. Dai and Z. Wang, "A flexible extension of WSDL to describe non-functional attributes," in *International Conference on e-Business and Information System Security (EBISS)*, May 2010, pp. 1–4.

[95] C. Rolland, M. Kirsch-Pinheiro, and C. Souveyet, "An intentional approach to service engineering," *IEEE Transactions on Services Computing*, vol. 3, no. 4, pp. 292–305, Oct 2010.

[96] N. Parimala and A. Saini, "Web service with criteria: Extending WSDL," in *International Conference on Digital Information Management (ICDIM)*, Sept 2011, pp. 205–210.

[97] J. Keppeler, P. Brune, and H. Gewald, "A description and retrieval model for web services including extended semantic and commercial attributes," in *IEEE International Symposium on Service Oriented System Engineering (SOSE)*, April 2014, pp. 258–265.

[98] M. J. O'Sullivan and D. Grigoras, "Delivering mobile cloud services to the user: Description, discovery, and consumption," in *IEEE International Conference on Mobile Services*, June 2015, pp. 49–56.

[99] R. Verma and A. Srivastava, "Towards service description for mobile environments," in *IEEE International Conference on Services Computing (SCC)*, June 2015, pp. 138–145.

[100] ——, "A novel web service directory framework for mobile environments," in *IEEE International Conference on Web Services (ICWS)*. IEEE, June 2014, pp. 614–621.

[101] B. Flyvbjerg, "Five misunderstandings about case-study research," *Qualitative inquiry*, vol. 12, no. 2, pp. 219–245, 2006.

[102] D. Schall, *Human-Provided Services*. New York, NY: Springer New York, 2012, pp. 31–58.

[103] V. Agarwal and P. Jalote, "Enabling end-to-end support for non-functional properties in web services," in *IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, Jan 2009, pp. 1–8.

[104] S. Gao, L. You, Z. Gui, and H. Wu, "Extending WSDL for describing complex geodata in gis service," in *International Conference on Agro-geoinformatics*, Aug 2014, pp. 1–6.

[105] G. Scheithauer, S. Augustin, and G. Wirtz, *Describing Services for Service Ecosystems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 242–255.

[106] J. A. Zachman, "A framework for information systems architecture," *IBM Systems Journal*, vol. 26, no. 3, pp. 276–292, 1987.

[107] K. Kritikos and D. Plexousakis, "Owl-q for semantic qos-based web service description and discovery," in *International Conference on Service Matchmaking and Resource Retrieval in the Semantic Web*, ser. SMRR'07, vol. 243. Aachen, Germany, Germany: CEUR-WS.org, 2007, pp. 114–128. [Online]. Available: http://dl.acm.org/citation.cfm?id=2889955.2889964

[108] J. Cardoso, A. Barros, N. May, and U. Kylau, "Towards a unified service description language for the internet of services: Requirements and first developments," in *IEEE International Conference on Services Computing (SCC)*, July 2010, pp. 602–609.

[109] F. Galán, A. Sampaio, L. Rodero-Merino, I. Loy, V. Gil, and L. M. Vaquero, "Service specification in cloud environments based on extensions to open standards," in *International ICST Conference on COMmunication System softWAre and middlewaRE*, ser. COMSWARE '09. New York, NY, USA: ACM, 2009, pp. 19:1–19:12.

[110] Y. L. Sun, T. Harmer, A. Stewart, and P. Wright, "Mapping application requirements to cloud resources," in *International Conference on Parallel Processing*, ser. Euro-Par'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 104–112.

[111] D. Liu and J. Zic, "Cloud#: A specification language for modeling cloud," in *IEEE International Conference on Cloud Computing (CLOUD)*, July 2011, pp. 533–540.

[112] L. Sun, H. Dong, and J. Ashraf, "Survey of service description languages and their issues in cloud computing," in *International Conference on Semantics, Knowledge and Grids (SKG)*, Oct 2012, pp. 128–135.

[113] D. Preuveneers, J. Van den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, and K. De Bosschere, "Towards an extensible context ontology for ambient intelligence," in *Ambient Intelligence*, ser. Lecture Notes in Computer Science, P. Markopoulos, B. Eggen, E. Aarts, and J. Crowley, Eds. Springer Berlin Heidelberg, 2004, vol. 3295, pp. 148–159.

[114] H. Truong and S. Dustdar, "A survey on context aware web service systems," *International Journal of Web Information Systems*, vol. 5, no. 1, pp. 5–31, 2009.

[115] M. Poveda Villalon, M. C. Suárez-Figueroa, R. García-Castro, and A. Gómez-Pérez, "A context ontology for mobile environments," *Proceedings of Workshop on Context, Information and Ontologies*, vol. 626, Oct 2010.

[116] C. Dorn and S. Dustdar, "Sharing hierarchical context for mobile web services," *Distributed and Parallel Databases*, vol. 21, no. 1, pp. 85–111, Feb 2007.

[117] J. Knutson, G. Truty, and P. Wang, "Publishing multipart wsdl files to url," July 2005, US Patent App. 10/762,085 [Accessed on August 10, 2016]. [Online]. Available: http://www.google.com/patents/US20050160153

[118] C. Pedrinaci, J. Cardoso, and T. Leidig, *Linked USDL: A Vocabulary for Web-Scale Service Trading*. Cham: Springer International Publishing, 2014, pp. 68–82. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-07443-6_6

[119] J. M. Garcia, P. Fernandez, C. Pedrinaci, M. Resinas, J. Cardoso, and A. Ruiz-Cortes, "Modeling service level agreements with linked usdl agreement," *IEEE Transactions on Services Computing*, vol. 10, no. 1, pp. 52–65, Jan 2017.

[120] F. Leymann and D. Roller, "Production workflow: concepts and techniques." Prentice Hall, 2000.

[121] H. Fernandez, C. Tedeschi, and T. Priol, "A chemistry-inspired workflow management system for a decentralized workflow execution," in *IEEE Transactions on Services Computing*, vol. PP, no. 99, March 2013, pp. 1–1.

[122] T. Dornemann, E. Juhnke, and B. Freisleben, "On-demand resource provisioning for BPEL workflows using amazon's elastic compute cloud," in *IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE, 2009, pp. 140–147.

[123] C. Lin and S. Lu, "SCPOR: An elastic workflow scheduling algorithm for services computing," in *IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE, 2011, pp. 1–8.

[124] G. Alonso, D. Agrawal, A. E. Abbadi, and C. Mohan, "Functionality and limitations of current workflow management systems," *IEEE Expert*, vol. 12, 1997.

[125] A. Barker, C. D. Walton, and D. Robertson, "Choreographing web services," *IEEE Transactions on Services Computing*, vol. 2, no. 2, pp. 152–166, April 2009.

[126] G. Bell, T. Hey, and A. Szalay, "Beyond the data deluge," *Science*, vol. 323, no. 5919, pp. 1297–1298, 2009. [Online]. Available: http://science.sciencemag.org/content/323/5919/1297

[127] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[128] G. Păun and M. J. Pérez-Jiménez, "Membrane computing: Brief introduction, recent results and applications," *BioSystems*, vol. 85, no. 1, pp. 11–22, 2006.

[129] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling, "Scientific workflow applications on amazon ec2," in *IEEE International Conference on E-Science Workshops*, Dec 2009, pp. 59–66.

[130] R. N. Calheiros, C. Vecchiola, D. Karunamoorthy, and R. Buyya, "The aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds," *Future Generation Computer Systems*, vol. 28, no. 6, pp. 861 – 870, 2012, including Special sections SS: Volunteer Computing and Desktop Grids and SS: Mobile Ubiquitous Computing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X11001397

[131] F. Zhang, J. Cao, K. Hwang, and C. Wu, "Ordinal optimized scheduling of scientific workflows in elastic compute clouds," in *IEEE Third International Conference on Cloud Computing Technology and Science*, Nov 2011, pp. 9–17.

[132] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108 – 143, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000099916938

[133] J. M. Cecilia, J. M. García, G. D. Guerrero, M. A. M. del Amor, I. Pérez-Hurtado, and M. J. Pérez-Jiménez, "Simulating a p system based efficient solution to sat by using gpus," *The Journal of Logic and Algebraic Programming*, vol. 79, no. 6, pp. 317 – 325, 2010, membrane computing and programming. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1567832610000123

[134] T. Y. Nishida, *Membrane Algorithms: Approximate Algorithms for NP-Complete Optimization Problems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 303–314. [Online]. Available: https://doi.org/10.1007/3-540-29937-8_11

[135] D. Hollingsworth, "The workflow reference model," *Workflow Management Coalition*, vol. Document Number TC00-1003, 1995.

[136] W. M. Van Der Aalst and A. H. Ter Hofstede, "Yawl: yet another workflow language," *Information systems*, vol. 30, no. 4, pp. 245–275, 2005.

[137] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh *et al.*, "Mapping abstract complex workflows onto grid environments," *Journal of Grid Computing*, vol. 1, no. 1, pp. 25–39, 2003.

[138] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distributed and parallel databases*, vol. 14, no. 1, pp. 5–51, 2003.

[139] S. Dustdar and H. L. Truong, "Virtualizing software and humans for elastic processes in multiple clouds- a service management perspective," *International Journal of Next-Generation Computing (IJNGC)*, vol. 3, 2012.

[140] J. Cardoso, A. Barros, N. May, and U. Kylau, "Towards a unified service description language for the internet of services: Requirements and first developments," in *IEEE International Conference on Services Computing*, July 2010, pp. 602–609.

[141] X. Wang, J. S. Dong, C. Y. Chin, S. R. Hettiarachchi, and D. Zhang, "Semantic space: an infrastructure for smart spaces," *IEEE Pervasive Computing*, vol. 3, no. 3, pp. 32–39, July 2004.

[142] H. Zhuge, "Semantic grid: Scientific issues, infrastructure, and methodology," *Communications of ACM*, vol. 48, no. 4, pp. 117–119, Apr. 2005. [Online]. Available: http://doi.acm.org/10.1145/1053291.1053325

[143] J. M. Zaha, A. Barros, M. Dumas, and A. ter Hofstede, "Let's dance: A language for service behavior modeling," in *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, R. Meersman and Z. Tari, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 145–162.

[144] C. Wang and J. L. Pazat, "A chemistry-inspired middleware for self-adaptive service orchestration and choreography," in *IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, May 2013, pp. 426–433.

[145] H. Fernandez, C. Tedeschi, and T. Priol, "A chemistry-inspired workflow management system for a decentralized workflow execution," in *IEEE Transactions on Services Computing*, vol. PP, no. 99, 2013, pp. 1–1.

[146] M. Caeiro, Z. Németh, and T. Priol, "A chemical model for dynamic workflow coordination," in *Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. IEEE, 2011, pp. 215–222.

[147] Z. Németh, C. Pérez, and T. Priol, "Distributed workflow coordination: molecules and reactions," in *International Parallel and Distributed Processing Symposium*. IEEE, 2006, pp. 8–pp.

[148] M. Weske, "Formal foundation and conceptual design of dynamic adaptations in a workflow management system," in *Annual Hawaii International Conference on System Sciences*. IEEE, 2001, pp. 10–pp.

[149] M. Mamei, F. Zambonelli, and L. Leonardi, "Distributed motion coordination with co-fields: a case study in urban traffic management," in *International Symposium on Autonomous Decentralized Systems*, April 2003, pp. 63–70.

[150] H. Reza and K. Ogaard, "Modeling uas swarm system using conceptual and dynamic architectural modeling concepts," in *Conceptual Structures for Discovering Knowledge*, S. Andrews, S. Polovina, R. Hill, and B. Akhgar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 331–338.