University of Leicester

Department of Computer Science

Research Report no CS-10-001

Context-aware Automatic Service Selection

Hong Qing Yu

May 2009

Context-aware Automatic Service Selection

Thesis Submitted for the degree of

Doctor of Philosophy

at the University of Leicester.

by

Hong Qing Yu

Department of Computer Science

University of Leicester.

May 2009

Abstract

Service-Oriented Architecture (SOA) is a paradigm for developing next generation distributed systems. SOA introduces an opportunity to build dynamically configurable distributed systems by invoking suitable services at runtime, which makes the systems being more flexible to be integrated and easily to be reused. With fast growing numbers of offered services, automatically identifying suitable services becomes a crucial issue. A new and interesting research direction is to select a service which is not only suitable in general but also suitable towards a particular requester's needs and services context at runtime.

This dissertation proposes an approach for supporting automatic context-aware service selection and composition in a dynamic environment. The main challenges are: (1) specifying context information in a machine usable form; (2) developing a service selection method which can choose the adequate services by use the context information; (3) introducing context-awareness into the service composition process. To address the challenges, we employ Semantic Web technology for modelling context information and service capabilities to automatically generate service selection criteria at runtime. Meanwhile, a Type-based Logic Scoring Preference Extended (TLE) service selection method is developed to adequately and dynamically evaluate and aggregate the context-aware criteria. In addition, we introduce the composition context and a Backward Composition Context based Service Selection algorithm (BCCbSS) for composing suitable services on the fly in a fault-tolerant manner.

Furthermore, this dissertation describes the design and implementation of the method and algorithm. Experimental evaluation results demonstrate that the TLE method and BCCbSS algorithm provide an efficient and scalable solution to deal with the context-aware service selection problem both in single service selection and composition scenarios. Our research results make a further step to develop highly automated and dynamically adaptive systems in the future.

Declaration

The studies outlined in this dissertation were undertaken in the Department of Computer Science, University of Leicester and supervised by Dr. Stephan Reiff-Marganiec. I declare that this dissertation is my own account of my research and contains as its main content work which has not previously been submitted for a degree at any tertiary institution. All of the work submitted for assessment is my own and expressed in my own words. Any use made within it of works of other authors in any form are properly acknowledged at their point of use.

Research work presented in some sections has been previously published, in particular: the idea of context-awareness process (Chapter 3) has been published in [YRM09a], the specifications for non-functional property based service selection (Chapter 2) has been published in [YRM08b], the Type-based Logic Scoring Preference extended service selection method is published in [YRM08a], and the idea of composition context is published in [YRMT08]. There are some further publications of the author which are referenced in the appropriate chapters.

Hong Qing Yu
Leicester, UK May 2009

Acknowledgements

I would like to express my profound gratitude to my supervisor: Dr. Stephan Reiff-Marganiec for his valuable suggestions, encouragement, and fantastic supervision throughout my PhD research work. His moral support and continuous guidance enabled me to complete my research work successfully. I also highly thank Dr. Emilio Tuosto, Dr. Fer-Jan de Vries, Prof. Jose Fiadeiro and Prof. Reiko Heckel, for their second opinions on my work and opportunities working with them on European projects. Thanks are also made to all my colleagues in the Department of Computer Science in Leicester for their support and help, which makes the department being my second home away from China.

Two European Union projects: inContext (IST-2006-034718) and SENSORIA (IST-2005-16004) are thanked for funding this research work, which provided my financial support over the past two and half years. In the framework of these projects, I had some very valuable discussions with project partners and thanks to all of you for the time spending.

I am as ever, especially indebted to my parents Mr Yu Baomin and Mrs Liu Meizhen for their love and support throughout every aspects of my life.

Finally, I wish to express my thanks from my heart to my wife Xia for her patience and support throughout my last six years overseas studying and working.

Thank you!

Hong Qing Yu

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Using a mobile phone to read the English version of Chinese news in Beijing, finding a wonderful Italian Restaurant using a PDA, or accessing an important document that is related to your presentation via a laptop's wireless connection are imaginable visions of pervasive computing. The most recent essential developments of pervasive computing are the widespread deployment of inexpensive context-aware devices and the innovations of the Service-Oriented Architecture (SOA) paradigm. "A great number of devices and software components collaborating unobtrusively in a smart space to provide people with required services is becoming a reality with the advance of technology developed on internet and network" [ESB07]. On the one hand, the devices can provide plenty of personal information that combined with other dynamic and static data supplies context which in turn can be used to provide appropriate services. On the other hand, SOA makes software components available as services on demand through the Internet. Services can be invoked directly for a simple task or can collaborate with other services for completing a complex task.

One opportunity is to achieve context-aware automatic service provision by combining context-aware computing with service selection. While each of the

two aspects comes with its own research challenges, the combination is a new research area to investigate.

Context-aware computing utilises the context information to characterize the situation of an entity. In general, context information refers to user's dynamic environments. For example, a user's location can change during him/her traveling; the available devices for a user might be different for different tasks. In a service selection scenario, a user can be an active or passive client, e.g. a person sending a message is a active client of a notification service and the person receiving the message is a passive client. However, in a service composition scenario, the user's context is not sufficient to select adequate services for the composite task, because composition related context (e.g. collaboration time, cost and errors) is also required to evaluate the collaboration efficiency. Overall, context information from both user and services are very important to provide the data for making the best decision when selecting services.

Automatic services provision primarily includes automatic service discovery, selection and invocation. However, if no suitable services are available, the additional issue of automatic services composition becomes relevant. In the last few years, most service provision research work focused on service discovery protocols based on the functional requirements of Input, Output, Postcondition and Effects (IOPE) [ZMN05] and service invocation. Other research has focused on service composition challenges. However, there is less research work aiming to directly tackle the service selection problem, and even less considering automatic selection based on changing context. With the increasing availability of services, many services provide similar or identical functionality. Therefore, service selection becomes a crucial issue by addressing the need to provide the users with not only "a" service, but the most suitable one. The selection process, therefore, needs to compare the services identified as functionally suitable based on user preferences of non-functional properties (NFPs). Thus, service selection is one step beyond discovery, and it is also an essential

step to enhance the service composition. However, there are many research challenges ahead to achieve context-aware automatic service selection.

The remainder of this chapter is organized as follows. Some context-aware service selection examples are studied to identify research challenges in Section 1.1. Then, the research challenges will be discussed in Section 1.2. The overall process of context-aware automatic service selection will be introduced in Section 1.3. The research aims, objectives and statement of this dissertation will be given in Section 1.4. The key assumptions and brief overview of the research approach are discussed in Section 1.5. The contributions of this work are highlighted in Section 1.6. Finally, an outline of the dissertation and a summary of this chapter are provided in Section 1.7.

1.1 Motivating Scenarios for Context-aware Service Selection

Service selection scenarios can be divided into two groups namely, single service selection and composite service selection. Only one kind of service is requested from the user to perform an autonomic task in single service selection scenarios. All functionally appropriate services will be ranked, returning to the user a list of suggested by services ordered ranking scores. In contrast, several different kinds of services are demanded to be composed together for completing a complex task is the composite service selection scenario. In the composite service selection process, the most adequate service will be selected and invoked directly and automatically without interaction with users.

In this section, we consider two cases for each of the scenarios. The case studies are provided by the inContext [UF06] project.

1.1.1 Single service selection scenarios

Notification service selection

Consider a scenario where a meeting has been scheduled for the next day. To ensure all related people are notified on time, a notification service is required to send a reminder message. Different participants may be in different context situations including differences in locations, time zones, availability via devices, contact preferences, online status and economic constraints of the sender, hence different kinds of notification services are likely to be required.

Making this example more concrete, we lists 3 people to be invited to the meeting in scenario. The following are summaries of their context and preferences.

1. Bob is on holiday with only his mobile phone and PDA. The holiday location is in Spain. On holidays, Bob prefers to be contacted by PDA rather than mobile phone message.

2. Alice has switched off her mobile phone to conserve battery power, but she is online using IM: Instant Massager (IM is also her preferred method of contact).

3. John is working in his UK office with access to a wide variety of communication devices (mobile phone, email, IM and PDA). However, he prefers to be contacted by email message.

Considering notification services, we are aware that there is a wide choice. However, different notification services may have different NFPs (Non-Functional properties). For example, one service has lower cost, but can use only one communication method and covers a small area. One service may have higher cost but good quality, the ability to use multiple communication methods and world wide usability. Another service may be good in one or more of these features,

or may introduce additional properties. The NFPs may include more aspects of reputation, security and network permissions. Therefore, a suitable service selection method is demanded to provide the most suitable notification service according to certain individual's runtime context and service NFPs.

By analysing the scenario, we see that the notification services involve two kinds of users (a sender and a receiver). However, the context information is related to both sides. In particular, the receiver's context information should be given higher priority as the receiver is the passive user of the service. Here 'passive' means, the receiver cannot specify the service selection criteria as she/he does not know of the sending message event. The following research issues can be identified from this service selection scenario: (1) automatically reasoning about user's runtime context information to constrain selection criteria; (2) evaluating possible services against different types of criteria; (3) dynamically and adequately aggregating the different criteria evaluation results into overall ranking scores.

Medical support service selection

The second case study comes from a medical support service selection for the Wolverhampton fair. The fair will be held in a big park. A large number of people are attending the fair. Different levels of incidents are expected in such a situation. Two medical tents are prepared for providing injury aid.

Based on past experience, one tent (service 1) has more medical staff and also has mobile staff who can provide help away from the tent. The tent is equipped to deal with minor injuries only. The other tent (service 2) has fewer staff but more advanced medical equipment to deal with serious incidents as well as normal injuries. The two tents are located in different locations to the left and right of the lake as shown in Figure 1.1. The two tents wait for the incidents to be reported to their aid systems. Once they receive the report

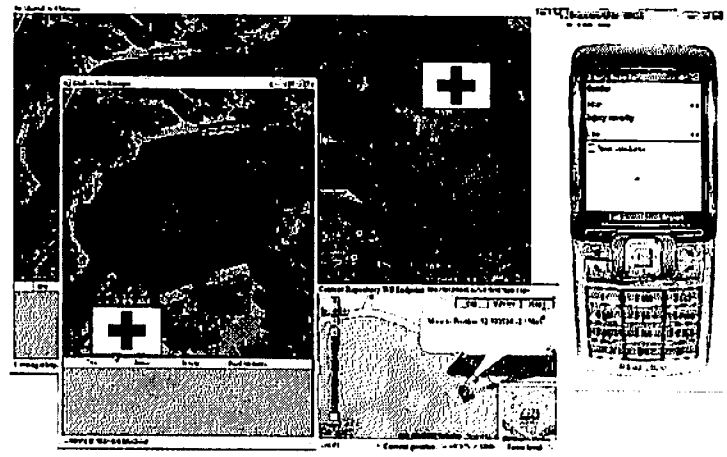

Figure 1.1: Medical support service selection scenario

from the fair help team, the tents will prepare for providing the medical aid. The members of the help team are responsible to locate the incidents and use their communication systems to report. The communication systems enable to identify and contact the most suitable medical support service according to the injury level, location of the incident, availability of the tent's staff and response time. Each service here is deployed inside a tent for receiving the injury reports and sending back the confirmations. The service can be invoked from the communication systems in the devices.

There are 3 typical scenarios:

1. The injury has been reported close to service 1 and it is a minor injury.
2. The injury has been reported close to service 2 and it is a minor injury.
3. The injury has been reported close to service 1 but it is a severe injury (severe injuries raise an emergency status).

Service selection is required for the communication systems to invoke the most suitable medical support service at runtime based on the injury context information. In this case, the context information only refers to the active user.

The context information is also more directly related to the service capabilities (NFPs). The medical support service selection scenarios bring further research issues: (1) making a direct connection between user's context and service NFPs, although they may be represented in different forms; (2) enabling service selection criteria to be affected by the emergency situation where all normal selection preferences can be broken.

1.1.2 Composite service selection scenarios

Organising a meeting

A meeting is required to be held for discussing the detailed planning of a particular event [TRMY07]. Organising a meeting involves a series of tasks. The tasks include searching for suitable participants, finding a suitable date, booking a meeting room and sending invitation notifications to the participants. The meeting organiser integrates these tasks as a workflow template (see Figure 1.2). Each task can be performed by a service.

1. The participant search task can be performed by a people-search service. There are two available people-search services offered by different providers. Both services have the same function of taking people requirement attributes, such as skills, experiences and positions to produce a list of people as output. However, these two services have different NFPs. One service can find the people who are in the organiser's organisation and is more accurate by having access to more information about people. The other can search people who are both inside and outside the organisation, but it is less accurate. Also, the first service's response speed is slower than the second one.

2. The date finding task can be completed by the meeting scheduling service. Again, there are two scheduling services available offered by dif-

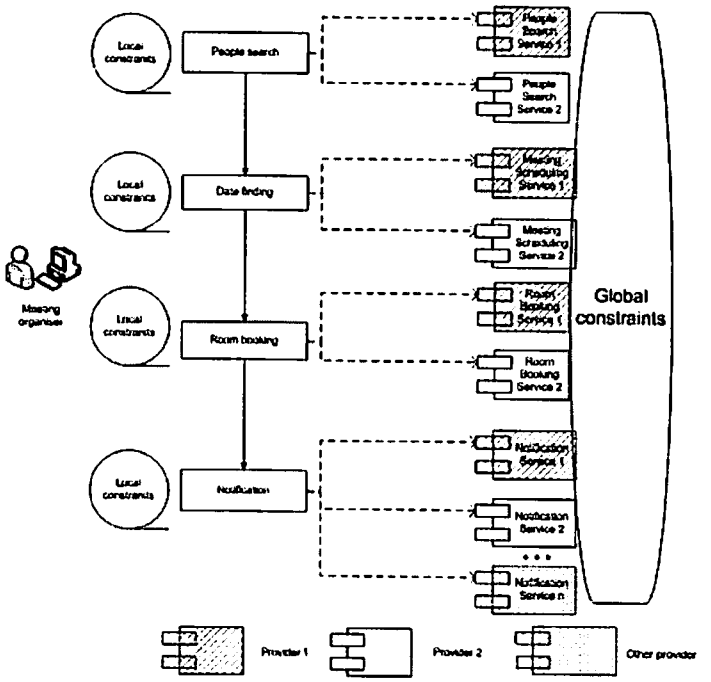

Figure 1.2: Workflow of organizing a meeting

ferent providers. They both use people's calendars' URL addresses as input and return the most suitable date for all involved people as output. One scheduling service only has ability to check Google and MSN online calendar systems and supports around 90% optimal dates (e.g. 9 people out of 10 are available on the scheduled date). The other service has ability to check all kinds of current existing online calendar systems and supports around 70% optimal date.

3. The room booking task can be executed by room booking services. The booking service takes the date and facility requirements as input and produces the place address and room information as output. There are two booking services available. One service supports booking rooms with normal meeting facilities. The other service supports booking rooms with both normal facilities and advanced equipment.

4. The notification task can be performed by the notification services. There are many services available. We already discussed the notification ser-

vices as the first case study of the single service selection scenario.

In single service selection scenarios, the user's context and preferences are the only determining factors. These factors will called *local context*. In contrast, service selection in composition scenarios also needs to consider the extra *composition context*. The composition context captures, amongst others, the collaboration relations among the different kinds of services, invocation policies among them and different integration costs. Therefore, the service selection problem becomes more complicated by having to consider both local context and composition context.

The research questions coming from this case study are: (1) where does the composition context come from? (2) what are the elements of the composition context? (3) how to define the composition context? (4) how to evaluate the composition context? (5) How to balance the local context constraints and composition context constraints.

Planning a trip

Let us consider another typical workflow example: planning a trip. Generally, the planning activity requires three tasks of booking transports, purchasing travel insurance and booking hotels as shown in Figure 1.3. Moreover, purchasing insurance and booking the hotel are two independent tasks but both rely on the transport date and time.

Because many travel related services are available, the competition is tight.

1. There are many different transport services available, the local constraints are faster speed and cheaper price (price refers to the service fee, not ticket price or other buying price throughout this dissertation).

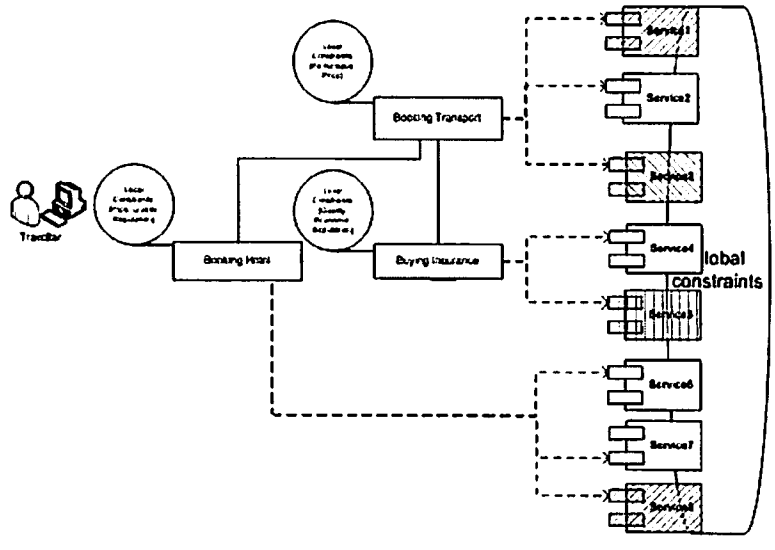

Figure 1.3: Workflow of planning a travel

2. There are many insurance services as well, the local constraints are cheaper price and better reputation.

3. There are also many hotel booking services, the local constraints are the place is covered, between 3 stars and 4 stars hotel, economic and good reputation.

On the one hand, this workflow example needs to be completed by invoking different services, similar to the previous case study. On the other hand, unlike the sequential workflow of the previous one, planning a trip is a parallel workflow. Therefore, the composite service selection problem has one more important research issue: designing or finding a suitable service composition mechanism which can efficiently cope with more complex composition workflow rather than sequential workflow only.

1.2 Research Challenges

Considering the questions arising from the scenarios, we derive 3 research challenges which we consider in more detail next.

Context-awareness. [SAW94] highlighted the challenge to distributed computation caused by changing context information and reaction to these changes. The difficulty of addressing this problem is to cover the gap between context information and the reaction resources. In SOA, suitable reaction resources are the service properties (functional and non-functional) and selection methods in the service selection domain. For example, in the notification service selection scenario, the context of an invited person being online with his/her IM account. To correctly react to this context for service selection, we need to understand relations between "online", "IM account" and the service properties. Normally, the user context is modelled by client context experts while service NFPs are defined by the service providers. Therefore, there is a gap between the user's context information and services' NFPs. The research issues are:

- What is the bridge between user's context and services' NFPs?

- How to build the bridge?

- How can the bridge affect service selection?

Service selection method. Finding a selection method is not difficult, however, finding a selection method which can deliver the best selection result in a dynamic, context-aware environment is difficult. Context is runtime information which demands a runtime method, which in turn needs to be automatic and reasonably fast. There are two difficulties: On concerns evaluating individual criteria, the often aggregating individual scores into a comprehensive whole. From an evaluation viewpoint, context information differs from other data, e.g. numerical data, using different expression types, e.g. text data, to allow people and machine to understand its meaning. Therefore, it causes the

difficulties for automatic evaluation. From the viewpoint of aggregation, different services may concern different selecting constraints with different importance levels based on user context and service domain features. In particular, the constraints are not isolated from each other but have dependent relations. Therefore, automatically choosing a suitable aggregation function to aggregate different aspects of the evaluation results is very difficult. [Duj07] explained 3 groups of aggregation properties: replaceability, simultaneity and mandatory.

- replaceability: if the insufficient satisfaction of one criterion in a group can be compensated by increasing satisfaction of any other member of the group, then such an aggregation is a model of replaceability. For example, cost and quality are the desired criteria for the notification service selection. If the requirement states that higher cost for a higher quality service is fair, then this is a replaceability aggregation scenario because a high score for quality can compensate a low score for cost.

- simultaneity: simultaneously satisfying two or more requirements is the most frequent criterion in practical system evaluation. For example, a car buyer typically wants a car that simultaneously satisfies criteria of performance and safety. This means that a dangerous car must get lower overall score even if the score for performance is outstanding.

- mandatoriness: some of the evaluation criteria are minimum requirements which must be satisfied. For example, if the cost of using a service must be lower than £50, it means that any service which is more expensive than £50 must be discarded, independent of how other features compare.

Different aggregation properties require different aggregation functions. Therefore, the service selection raises two research issues:

- How to evaluate the individual criterion for services in a context-aware environment?

- How to dynamically applying the adequate aggregation function?

Composition context The context information related to service composition has not been researched and modelled. In fact, the composition context is an important factor for ensuring an adequate service selection for service composition. In the travel planning scenario, assuming service A and service B have the price of £3 and £2 for adminstration fees. If we only consider the price criteria, then service B is obviously the better choice. However, service A supports free insurance collaboration with any other insurance services, while service B will charge extra £2 for the collaboration. Hence overall service B is not a good choice any more. There are many more composition context aspects that need to be considered and they all influence adequacy of service selection from a global point of view. Thus, the research issues are:

- What information is related to the composition context?

- How to define the composition context?

- How to use the composition context for service selection in the composition scenarios?

The research presented in this dissertation addresses these 3 challenges. In particular, we present a novel context-aware automatic service selection method. Based on this method, we develop a composition context focused service composition mechanism.

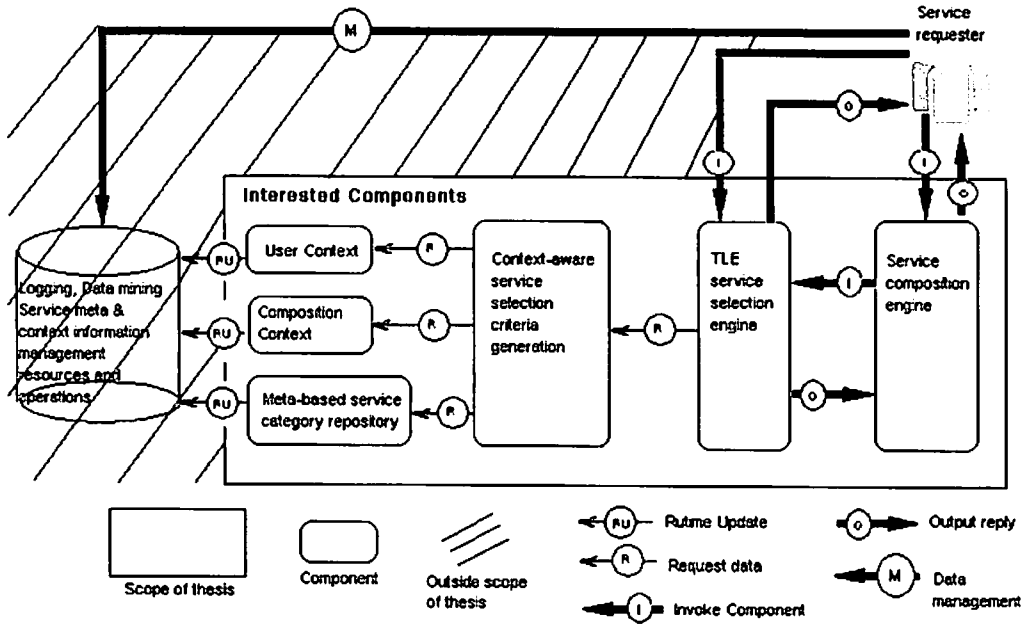

Figure 1.4: The service selection process

1.3 Overview of the Service Selection Process

The complete service selection process is represented in Figure 1.4. The process addresses the discussed research challenges by concentrating on 4 related aspects.

The first aspect is concerned with context information and service capability information and the category repository. The context information includes user context (Chapter 3) and the service composition context (Chapter 5) which are updated through data management and run-time monitoring. The monitoring may include system logging activities, inserting and updating the context and service information. However, the details of how to maintain the data resources is not part of the research in this dissertation. The service capability information includes both functional and nonfunctional properties that can be compared to context information and service selection criteria (Chapter 3). A categorising service repository is built based on the capability meta in-

formation for enhancing the service discovery and selection. The second aspect is concerned with dynamically generating the context-aware service selection criteria by querying the context information and service capability meta information (Chapter 3). The third aspect, one of two major contributions, is concerned with an extended Logic Scoring Preference based service selection method which can use the context-aware criteria to complete the service selection task (Chapter 4). This has been implemented as service referred to as relevance engine. The final aspect is the service composition mechanism, which including the composition context, forms the second major contribution. The mechanism takes care about the workflow requirements and fosters the selection method to achieve the service composition task (Chapter 5).

1.4 Research Aims and Statement

The overall aims of this dissertation are developing a context-aware automatic service selection method and a composition context based service composition mechanism. It is useful to further divide the aims into more specific problems, objectives and questions.

Aims

1. Generating context-aware service selection criteria.
This aim includes the modelling of user context concepts (related to service ranking), modelling the service profile (especially including non-functional properties) and bridging the gap between the two. Furthermore, adaptation of selection criteria to context has to be considered.

2. Developing an efficient and automated service selection method.
This includes the development of a suitable method which allows to rank services based on their NFPs, taking aggregation properties and preferences relations into account.

3. Developing an efficient service composition mechanism.
This aim includes modelling the relation between tasks to carry the data relevant for selection from one to the other and adapting the method from Aim 2 to also work in workflows.

There are a number of objectives that need to be met by the approach. When we speak of "automatic", we mean no human attention at run-time.

Objectives

1. The selection method shall be capable of dealing with a large number of services and selection criteria (scalability in two dimensions).

2. The selection method shall automatically produce adequate final service ranking scores by aggregating different criteria considering preferences (different levels of importance) and preference dependency relations (aggregation properties).

3. The selection results shall reflect the user's context.

4. The selection method includes evaluation functions that allow automatic computation of evaluation scores for criteria. Automatically give the evaluation scores of services for different criteria.

5. The selection method can easily be expanded to handle extra aspects, like the composition context representing a global point of view to select services for service composition.

6. The composition mechanism shall have low complexity to deal with large scale composition problems.

To address the first two objectives, we need to answer the following questions.

- How many service selection methods have been proposed to deal with similar problems?
- What are the pros and cons of these existing methods?
- Have these existing methods met our criteria and important aggregation

properties? If they do not, which improvements or new selection theories are required?

- How can we measure scalability?
- How can we measure the adequacy?
- What does adequacy mean here?

To fulfil the third objective, three particular questions should be answered.

- How is the context information modelled, stored and retrieved?
- What types of user context relates to which service selection criteria?
- How can the context related criteria be dynamically affected?

Regarding the fourth objective of combining automatic evaluation functions for different types of criteria, two research questions should be considered:

- How are the context/property values of service represented in real world scenarios?
- How can the evaluation functions be dynamically mapped to the different types of services' context representations?

The last two objectives are related to the following research questions.

- What kinds of composition context criteria should be considered during service composition?
- How can this context information be detected and used?

- What are the current strategies or mechanism for services composition? Are these suitable for our purpose? If not, how can we define a new service composition mechanism?

In this dissertation, we address all these questions, however, some aspects are considered in greater depths.

Thesis Statement

Providing the right service at the right time is a major challenge in Service-Oriented Computing. We show that a context-aware automatic service selection method and a composition context-aware service composition mechanism are feasible and practical.

1.5 Key Terminology and Assumptions

To ensure a focus on the outlined research aims, objectives and questions, some crucial terminology and assumptions have to be made. Some of these assumptions may already have efficient solutions, some are under active investigation by others. Some of these assumptions may not very realistic at the moment, but they have already been proposed as future research focuses. Furthermore, in the young field of SOA different researchers use some terms in slightly different ways. To avoid ambiguity we present our use of the relevant terms.

1. In the current implementation, a service might provide several operations which perform different tasks, however, we consider operation as the core concept and hence "service" will mean a single operation which only performs one task at a time.

2. Service is used in a quite general way, it not only refers to Web Service – recall the example of the medical tent.

3. Non-functional property descriptions for all services are available using OWL-S (Web Ontology Language for Services) [MBH+04].

4. The terminologies of ontology variables are unified, which means, there are no conflicting or unclear concepts and variables.

5. We assume all services are registered in a centralized repository and categorized by functional keywords.

6. Service discovery is a simple process such as key word based or ontology based discovery focused on functional property. In this dissertation service selection and composition are steps considered beyond service discovery, and hence we assume a suitable discovery mechanism to be in place.

7. Context is modelled and stored using OWL/RDF [Org04b, PS06] technology. We assume that context information is centrally stored as our focus is not on retrieving context, but rather using it.

8. "Service composition" refers to instantiating abstract workflows. The service composition workflows are predefined using an orchestration language (e.g. WS-BPEL [Org07a] or Windows Workflow Foundations [Cor07]) and deployed in the composition task template store [TYRM+08].

1.6 Research Methodology and Contributions

We use the case study based research methodology. Our research began with analysing the real world service selection and composition scenarios. The scenarios were proposed by the industry research partners of the "inContext"

project. Based on the case studies, the selection and composition requirements are represented and defined. For successfully achieving the requirements, we investigate currently proposed selection and composition methods from literature and we develop an automatic Type-based Logic Scoring Preference Extended (TLE) service selection method and a Backward Composition Context-based Service Selection (BCCbSS) algorithm.

The main contributions of this dissertation are:

- A technique for automated generating context-aware service selection criteria as the bridge for linking user context information to the service non-functional properties (Chapter 3).
- A service selection method based on type-based evaluation functions and automatic Logic Scoring Preference [Duj73] functions. The type-based functions can be automatically applied to evaluate the different types of evaluation criteria. The automatic Logic Scoring Preference functions solves the user's preferences-based selection issue considering both preferences and aggregation properties (Chapter 4).
- Based on the context-aware composition scenarios, the composition context concept is defined. Additionally, we define an extensible composition context information category (Chapter 5).
- The BCCbSC (Backward Composition Context based Service Selection) algorithm using the TLE service selection method is developed. It is efficient and suitable for the context-aware environment, in particular addressing the issues of using composition context (Chapter 5).
- A test simulation system is implemented. The system implements the proposed method and algorithm to evaluate adequacy and scalability (Chapter 6).

1.7 Dissertation Outline and Summary

This chapter discussed the purposes of the dissertation - *addressing context-aware service selection and composition problem.* We explained the crucial challenges of combining context awareness and service selection through both single service and composite service selection scenarios. In order to draw our research border, we listed the interesting research aims, objectives with their related questions and provided focus with a clear thesis statement. The key assumptions and terminology were explained. In addition, we highlighted our research contributions in the previous section.

The reminder of this dissertation is organised as follows:

- In Chapter 2, we give a more detailed description of the research background. Specifically, we are going to discuss the related work on service selection and the "inContext" research project which gave a wider frame to this research. We are also going to introduce some basic concepts which form the foundations for this research such as SOA and context-aware computing.

- In Chapter 3, we illustrate a way to automatically generate context-aware service selection criteria. The Chapter includes a discussion of user context model, service categorization, criteria generation and implementation.

- In Chapter 4, we focus on the service selection method. More precisely, the mathematic foundations of the LSP method and type-based evaluation function are going to be explained. The strengths of our selection method is illustrated with a worked example.

- In Chapter 5, we extend our context-aware selection method by defining and adding the service composition context for enhancing service selec-

tion in workflows. Furthermore, a Backward Composition Context-based Service Composition mechanism is introduced.

- In Chapter 6, we introduce our prototype implementation and discuss system evaluation results for service selection adequacy and scalability.
- In Chapter 7, we provide a concluding discussion and identify potential future research directions.

Chapter 2

Research Background and Related Work

In chapter 1, we discussed the research motivations, scenarios, aims and objectives of context-aware automatic service selection. Our work fits into a wider context set by SOA, Semantic Web and Context-aware computing. It addresses the important aspect of service selection which has been considered only marginally in these fields as we have indicated. The author was involved in the inContext project during the research. In this project automatic context-aware service selection is highly demanded and thus it has been a testbed for many of our ideas. This chapter will discuss the research background and related work, which is divided into 6 sections.

The literature on Service-Oriented Architecture, Semantic Web and context-awareness and context modeling are introduced in Section 2.1, 2.2 and 2.3 respectively. Section 2.4 discusses the service selection issue and its requirements. Section 2.5 discusses current related work on context-aware service selection approaches and compares them to the requirements. Section 2.6 provides details about the inContext project. Section 2.7 draws the conclusion for the chapter.

2.1 Overview of Service-Oriented Architecture

The first Service-Oriented Architecture for many people in the past was with the usages of DCOM [Red97] or Object Request Brokers (ORBs) based on the CORBA [Gro04] specification.

One of the SOA definitions given by the SOA Open Working Group is [oTOG06]: "A paradigm for organising and utilising 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."

SOA includes at least three elements: Application Frontends (also called service requester or client), Services (also known as providers) and Services Repositories (sometime seen as service broker) [KBS04].

Application Frontends initialize and control all activities of the enterprise system.

Services are software components with distinctive functional meaning that typically encapsulates a high-level business concept.

Service Repositories provide facilities to discover services and acquire all information necessary to use the services.

Web services is one of the most widely used implementation and standard nowadays to implement SOA. The Web services architecture mainly includes three parties mapping to SOA as shown in Figure 2.1.

Enabling services to be discovered, service providers must firstly publish the services to the UDDI (Universal Description, Discovery and Integration) repository [Org04a]. The service requesters is then able to find services from the

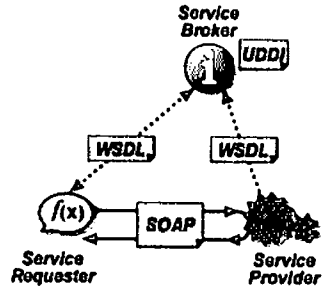

Figure 2.1: Web services architecture [ACKM04]

UDDI descriptions. To invoke a service the requester needs to know the interaction interface of the service. This invocation information is provided by the WSDL (Web Service Description Language) [Org07c] service description which forms part of the UDDI contents. The communication between requester and service is accomplished through SOAP (Simple Object Access Protocol)[Org07b] messages conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Web services, however, is only one implementation of SOA. Any paradigm that performs all the features of SOA can be a SOA implementation. In this dissertation, the term "Service" is not limited to Web services.

"SOA is a flexible, standardised model with a deeply rooted concept of encapsulating application logic within services to better support the integration of various applications and the sharing of data" [Erl04]. The goal of SOA is to allow fairly large chunks of functionality to be composed together to form ad hoc applications that are built almost entirely from existing software services. However, to achieve the goal, SOA still faces many challenges. Service selection is one of these challenges. In this dissertation, we contribute to a possible solution for service selection is combination with context-awareness and service composition.

2.2 Overview of Semantic Web

The Semantic Web is envisioned as an extension of the current web where, in addition to be human-readable using WWW browser, documents can be annotated with meta-information which defines what the information is about in a machine processable way [DFvH03]. Currently, the crucial part of the Semantic Web technology is Description Logic (DL) based ontology language such as OWL (Web Ontology Language) [Org04b] or WSMO (Web Service Modeling Language) [Gro05]. These ontology languages include two layers based on the Description Logic namely, TBox and ABox. The TBox specifies the conceptional presentation model of a description domain and ABox is the container of the individual information according to TBox specifications [BCM+08]. For example, TBox is defined as ontology syntax and ABox is defined as RDF syntax in OWL language. Both syntaxes are build on the top of XML language, therefore Semantic Web technology gives us a capability for describing, storing the information or entities. The semantic information enables machine to access, read and compute. There are three major advantages to use Semantic Web technology:

Efficient reasoning: because Semantic Web is based on DL, information reasoning is every efficient by applying DL inference theories through open world assumptions [HPSMW08]. We can get more facts from the reasoned results, which is the most differences from other information modelling and storing technologies.

Structure independent query language: currently, the query language for OWL is the SPARQL [PS06] query language. Unlike other query languages (e.g. SQL) which are tightly coupled to the information structure, SPARQL is independent from information structure. For example, if we query an age value of a given name in database system, we need to know the database name and table name in order to query the data. However,

SPARQL does not require this information to perform a query, which is done only by specifying the RDF triple statement. Therefore, the query codes do not need to be changed when the information structure is modified which often happens in the dynamic environment.

Distributed information sharing mechanism: as the name implies, the information modelled and stored by the Semantic Web is distributed over the Internet. The sharing mechanism is the same as Web technology identifying the URL for information resources. As a result, it becomes more simple and efficient to access the information decentralised than traditional database data sharing mechanisms.

In this dissertation, user context and service properties are modelled and stored as OWL and OWL-S technologies. The detail information about the models and the way to use it will be discussed in Chapter 3.

2.3 Overview of Context-awareness and Context Modelling

Context-aware Computing is concerned with enriching computing environments with concepts that can improve interaction and ultimately increase productivity and reduce the burden on users. The key concepts in this area are context modelling and context-awareness. **Context** is formally defined by [DA99] as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves." Meanwhile, **Context-awareness** is defined as "a property of a system that uses context to provide relevant information and/or service to the user, where relevance depends on the user's task." [DA99]

Context is a powerful and longstanding concept in human-computer interaction. As human beings, we can more efficiently interact with each other by fully understanding the context in which the interactions take place. It is difficult to enable a machine to understand and use the context of human beings.

Nowadays, the notion of context is much more widely appreciated. The term context-aware is generally defined by those working in ubiquitous/pervasive computing, where "it is a key to the effort of dispersing and enmeshing computation into our lives. Consequently, context also refers more to the physical and social situation in which computational devices are embedded" [MD01]. One major task in context-aware computing is to acquire and utilise information about the context of a user in order to provide the most adequate services. The service should be appropriate to the particular person, place, time, event, etc where it is required. A user may be a person who uses the service or who is the target object of a service. For example, a cell phone could always vibrate and never ring in a concert, if the system knows the location of the cell phone and the concert schedule. However, this is more than a simple question of gathering more and more information about complex situations. More information is not necessarily more helpful [MD01]. What makes the context information more useful is to relate it to the tasks and activities, to organize it well and to increase its utility.

In this dissertation we use context information as one of the important factors for service selection. In order to achieve this aim, we establish a context model which is efficiently to be used for service selection process.

Context modelling is demanded for the wide range of heterogeneous context information in context-aware computing. It helps application designers and developers to uncover the possible context and simplify the context manipulation [RDN05]. The model can be very helpful in providing quality context information. For example, conflicts can be resolved by favoring the classes of

context that are most reliable over those that are more often subject to error (sensed and derived).

The conceptual viewpoints of context models summarised by [RDN05] include: who, where, what occurs, when, what can be used and what can be obtained. Most researchers follow the model categorization from a conceptual viewpoint. For instance, [NP05] proposed to put context information into five semantic dimensions (categories): identity (who), location (where), time (when), activity (what) and devices profile (how). The classes in their context model are put into five sub-packages: ActorSet (who are the actors for the interaction?), LocationSet (where does it happen?), TimeSet (when does it start or finish?), ActivitySet (what are they intend to do?) and ProfileSet (how do actors do that?). A similar proposal is provided by [RPC+04] as "People","Places and Location", "Devices" and "Services".

The ultimate purpose of context modeling, however, is not only to categorise the context information but also to introduce a way to use context information efficiently. In our dissertation, the context model is implemented using Semantic Web technology of OWL/RDF and its query language SPARQL for information retrieval. OWL can be considered as the conceptual model and RDF is the instant context data of the model.

2.4 Service Selection and Selection Requirements

Service selection is a process of allowing a prospective user to choose services which best suit his/her functional and non-functional requirements [DLP]. With the rapidly growing number of available services, users are presented with a choice of functionally similar (or even identical) competitive services. Therefore, this choice strongly depends on the NFPs that provide differences

among competitive services. Service selection can be performed by two types of methodologies:

1. Static service selection. The service(s) discovery and selection are manually conducted through the service discovery protocols (e.g. UDDI) and human readable NFP presentations. Once the service is selected, it will hardly be changed in the future.

2. Automatic service selection. The service(s) are dynamically discovered and selected based on machine understandable representations (e.g. OWL-S or WSMO) and run-time collected service requirement constraints. The selected service is only used once. When new requirements come next time, a different service may be used.

Automatic service selection is one of the important goals of the SOA. However, it is a very complex task because of various different types of non-functional properties and uncertainties of the runtime environment. Apart from context-awareness, the major requirements for service selection methods based on NFPs are as follows [YRM08b].

Model for non-functional properties: Service requesters or passive users need to objectively distinguish services based on their non-functional properties to make the most appropriate choice amongst a number of services with equal or similar functionality. In the light of that a model for non-functional properties is required, which can be used in service descriptions as well as service requests. Due to the versatility of non-functional properties (and the fact that new ones might be required at any time), it is unlikely that a complete standard set can be identified. Non-functional properties should be considered differently depending on the specific service domain. For instance, the printing service domain

should consider print speed, color-options, location, quality and price properties. Financial service, in contrast, should consider security, privacy and performance properties. Furthermore, the selection method build on top of the non-functional property model must be generic enough to be able to work with additional properties (possibly using additional information provided through the model).

Properties' preferences: Service requesters usually have various preferences for the non-functional properties depending on the situation they find themselves in, and of course different situation will mostly have different preferences. A good mechanism should not only express values for each property, but preferably also represent the relations among the preferences. For example, in the emergency situation, a financial service may require to consider the security property as more important than privacy. Hence, the selection approach needs to provide mechanism to specify preferences for different situations and service domains.

Evaluation of properties: As we discussed earlier, it is difficult to predict how many non-functional properties will be available, as well as the types of these properties. For example, the evaluation function to compute the speed criteria will be very different from the function to calculate the location criteria. It is imposible to define a universal evaluation function for all kinds of non-functional properties. Hence, the evaluation framework must not only adapt to various numbers of non-functional properties, but also automatically identify the measurement methods that should be used to evaluate each non-functional property.

Dynamic aggregation: When all desired non-functional properties can be evaluated, the next important step is to aggregate individual scores to gain a final score for the service. In this step a suitable aggregation method needs to be selected. Intuitively, arithmetic or geometric means based on weighted sums or products might appear to be an efficient and

understandable choice. Unfortunately, sticking to one of them all time is not the best choice for complex situations with tens or even hundreds of evaluation criteria. For example, the aggregation may consider both speed and price to be mandatory criteria, the aggregation results should reflect the preference to score 0 for not satisfying any one of these criteria. This aggregation feature can not be gained through summation, but a product aggregation function would solve this issue. Meanwhile, in reality there are more complicated issues than just mandatory and optional criteria. With large numbers of criteria, it is easy that extremely high value measured selection criteria with a low weight can overshadow or replace values of other factors with higher weights (and hence higher importance). One way to completely solve above two issues is to use a adequate aggregation function for different aggregation requirements. We have discussed it as one of the major challenges in Chapter 1. We define such kind of requirements as aggregation properties with three different dimensions of replaceability, simultaneity and mandatory. The aggregation function should enable to consider both the criteria preferences and the aggregation properties among different criteria to control the replacement/overshadow level. Correctly choosing the aggregation function can be done by analysing the service selection scenarios manually. However, manual aggregation function selection will heavily decrease automatic level service selection, particularly with context-aware requirements. Therefore, it is important that the aggregation functions are chosen automatically to best match the aggregation requirements.

Automation: Service selection can be performed by a human to look up suitable services in a registry and make decisions as to which one to choose (as a matter of fact, this is currently often common practice). However, the ultimate goal of service selection research, and especially service selection based on non-functional properties, is to provide fully automatic

processes. A service designer would still specify data for the service when making it available, and a user would still be able to specify requirements, but the selection would be performed without human intervention. Such automatic selection methods are essential when, for example, considering context-aware service selection (where requirements are automatically generated, and change rapidly) or selection of services within workflow contexts (essentially allowing for the execution of "abstract" processes where specific service endpoints are not predefined). We have discussed two points of automation in previous requirements. One is the selection of evaluation functions for specific non-functional criteria, the other is the selection of adequate aggregation function. For context-awareness, automatically using context information to affect the service selection is also one aspect of automation.

Scalability and accuracy: Scalability here not only means that the approach can consider large numbers of properties, but also means many selection processes are taking place simultaneously. Scalability can not be measured independently from selection accuracy and automation level. We have just discussed the automation requirements, there is also a question on how accurate the result is. While one would aim for perfect accuracy (that is one has proveably chosen the best service), it is often sufficient to choose a good enough service if the decision can be made quickly.

2.5 Related Work

In recent years there have been many efforts dedicated to develop approaches for service selection based on NFPs. Some approaches touched the concern of context information. It is clear that much progress has been made, and by considering the individual approaches there is some overlap in functionality, but obviously some divergence. In this section, we will discuss and compare

some proposed selection approaches against the requirements that was made in the previous section.

2.5.1 Non-context aware selection approaches

We firstly discuss service selection approaches that do not consider context and distinguish them into 3 dimensions and 6 categories.

Policy vs reputation

Policy based service selection approaches allow to specify the non-functional requirements by coding them in a QoS (Quality of Service) policy model or policy language. [LNZ04] and [JS07] are typical examples of policy based selection approaches.

The QoS policy model in [LNZ04] is designed as a textual document. It offers two types of non-functional properties: generic and domain specific. The domain specific properties are extensions of the generic ones for different kinds of services. The content of the policy model represents the service requester's non-functional constraints and preferences. It also defines two universal evaluation functions (one is used for the case where a lower value benefits the requester and the other is for the opposite case) to evaluate the service's non-functional properties against the policy model. The relations between the non-functional criteria are expressed in a matrix, which is also used for their aggregation.

There are some disadvantages of this approach. Firstly, it is difficult to formalize all the non-functional criteria in order to allow overall score computation. Secondly, all the non-functional properties have to be presented as numbers or be converted into that format. Thirdly, while it captures how to express requirements, there is no mention of where and how the properties' value is stored and expressed. Fourthly, the matrix aggregation function is difficult to

be understood by users and does not show the relation of user's preferences at all. Finally, the final ranking scores do not reflect the satisfaction level that can be expected from the service as the overall range of values is not specified.

A similar approach is introduced in [JS07]. The improvement is that it formalises the NFPs into a conditional policy language. The other differences is that the service properties are dynamically detected by hardware sensors that monitor whether the selected service breaks the requester's requirements. However, this feature limits the number of properties for practical reasons (it is only possible to monitor a small range of properties). This technique could, however, be very useful combined with a selection strategy that uses service execution history.

In contrast to the policy based approaches, there are a number of approaches based on trust and reputation presented in [WV07] and [GGD07].

The web service selection criteria presented in [WV07] is statically defined for all kinds of services and each criterion is linked to a trust and reputation typology. The values of the criteria for different services are collected through the typology based feedback from communities or agencies. The selection processes are different based on the classification of trust and reputation systems which might be centralised or decentralised. For example, a centralized reputation system may use a PageRank [PBMW98] selection function. A similar idea is proposed in [GGD07], but using IRS-III [DCG+06] selection methodology based on ontology mapping technology to calculate the ranking scores. All of these approaches focus on evaluating the selection criteria based on trust – that is whoever provides the values for the services is a trusted party. However, there has not been any uptake of these approaches for real world problems because of the complexity and time consuming manner for establishing the trust/reputation community – a system similar to certificate agencies might be required. Furthermore, the proposals do not present service evaluation and

aggregation functions or consider the requester's preferences.

Although the approaches in this category introduce two interesting aspects, namely capturing user requirements by policies and relying on observations of the services before making decisions, there are some common shortcomings.

- They do not define a model of expressing service properties – they assume that the values of service properties are simply available somewhere.
- They do not consider the evaluation functions to different criteria – they all define a unified function for all kinds of non-functional properties (which is not practical in general as the properties vary widely).
- They do not consider aggregation functions in detail, simply assuming that this is not an issue (but we have mentioned earlier that this is a complex matter in itself if large numbers of criteria are considered to have complex aggregation properties).

UDDI-extensions vs Semantic Web Services

In order to address the issues of modelling and using service's properties, some research projects have investigated extensions to UDDI and Semantic Web services technologies.

[SJS05] and [AMM07] proposed two similar types of UDDI extensions for service selection. [SJS05] adds an extra component in the SOA called *Quality broker* which sits between the service requester and UDDI repository. The Quality broker randomly invokes the services which are registered in the UDDI repository through the WSDL endpoint and in this way monitors the performance (response time and throughout), safety (availability and reliability) and cost. In this approach all kinds of service selection problems only consider these three non-functional properties and hence do not allow additional properties,

for example domain specific ones. Likewise, the approach uses a simple three value approach for representing the match to the required properties: gold, silver and bronze. Two utility functions representing the gains for service requester and provider are composed by linear programming following equations 2.1 to 2.3. In the functions, x_i is the monitored value of the i-th criterion and v_i is a universal evaluation function, which means all x_i are calculated by the same function and are assumed to be numerical values.

Moreover, w_i^r and w_i^p are possibly different as they represent importance considerations from requester and provider. Consequently, the *LinearFunction* (equation 2.3) matches the result between requirements and offers. It is not obvious why the service with the highest score is the best one to be selected. It represents the best overall score, but is almost certainly not requester optimal.

$$RequesterUtility = \sum_{i=1}^{n} w_i^r v_i^r(x_i), 0 \leq RequesterUtility \leq 1. \quad (2.1)$$

$$ProviderUtility = \sum_{i=1}^{n} w_i^p v_i^p(x_i), 0 \leq ProviderUtility \leq 1. \quad (2.2)$$

$$LinearFunction = RequesterUtility + ProviderUtility. \quad (2.3)$$

A similar system has been proposed in [AMM07]. The differences are (1) the *Quality broker* is a database which can be queried by giving the names of desired functional and non-functional properties; (2) the aggregation function is a simple function summing all desired properties; and (3) the criterion evaluation function is a universal function designed to measure a weighted distance of each value from the maximum value for the criterion.

The two basic disadvantages of the UDDI based approaches are:

- The service data and the quality information are separated. Therefore, the service provider needs to register details of their service in more than one place or the quality broker has to dynamically monitor the registered services. These limit the number of criteria as monitoring is expensive and impractical and the broker would somehow need to be aware of any new service added.

- There is no extensible service quality model, which means that the approaches are restricted for selections based on a few predefined, generic criteria.

With these disadvantages in mind, some work has been conducted to define nonfunctional models for web services using Semantic Web Service (SWS) technology. [WVKT] introduced a WSMO (Web service Modeling Ontology) based approach. The non-functional properties are organized as QoS ontology and vocabulary in WSMO. However, its evaluation functions do not make use of the full power of the SWS technology because all values have to be numerical number without considering the semantics of vocabulary at all. [MT05] enhances on this by introducing a DAML-S based service selection approach. In [MT05], the matching algorithm uses the semantics of the vocabulary by introducing concepts of $\{Exact, Plugin, Subsumption, Container, PartOf, Misjoint\}$ matching.

Graphic vs ontology-based preference modelling

It is also important to consider how users can best express their needs. However, previously discussed approaches address the providers perspective service selection in some sense, little attention is paid to the requester. It is important to express non-functional properties from both provider and requester.

To that end, a graphical preference modelling and service selection approach

has been discussed in [SBS+07], where the preferences are modelled as TCP network graph [BDS06] or UCP network graph [BBB01]. These network graphs not only present simple importance relations among different non-functional properties but also model the dependency relations between them. For example, "colour=true" might be the most important criterion for selecting a printing service outweighing price and quality. However, if there is a colour printing service available, then price is more important than quality, otherwise, quality is more important. Although this might sound trivial, it is a situation that one would naturally consider in many decisions. Hence, being able to capture it in a aggregarion way for service selection is a huge achievement.

The graph based modelling approach has a big disadvantage when graphs become very complex and difficult to understand for an average user. Moreover, the selection algorithm is based on simple textual matching without making use of a model for NFPs. It is less extensible and cannot deal with hierarchically structured properties.

[MS04] and [LASG07] present two approaches which use ontology modelling techniques to model both the requester's requirements and service properties. In [MS04], the selection algorithm is quite simple by only selecting a service which fully match the requester's requirements using the *Exact* match concept. In contrast, [LASG07] uses a price based evaluation method. The method is proven to be a NP-complete.

The ontology modelling approaches only solve the first part of capturing the requester's preference by formally specifying the considered selection criteria with semantic vocabulary and a classification structure. Unlike the graph modelling approach, they do not model the aggregation properties.

2.5.2 Context-aware selection approaches

The selection approaches introduced in section 2.4.1 do not consider contextual information at all to select the most appropriate service for the user. Some other research work has been conducted to support context-aware service selection.

Location, which was introduced in the Cooltown project [Pac04. RT06] and Jini [KEKW04] present the earlier context information used to utilize the service selection. The work can discover and select the service nearest to the user. Nevertheless, the context information is only the location context.

The improvements have been illustrated later by [CKL05] and [LH03]. They extended the context information by adding so called *Dynamic and Static Service Attributes*. The dynamic service attributes are those characteristics of a service whose values change over time. Otherwise the attribute is said to be static. Since there are more than one context constraints, they also introduced the Weighted Vector based aggregation functions for ranking the services and returning the top matches to the user. However, there are two main drawbacks:

1. Their work relies on a syntactic representation of contextual information of services. Consequently, it is impossible to apply more advanced semantic level searching, matching and reasoning.

2. They all only focus on modelling services' attributes/context information without specifying the user's context information. Thus, context-awareness in this sense means service attributes awareness.

[ESB07] realises the first issue and tackles the issue by utilising concepts from the Semantic Web. However, it does not address the second problem of modelling user context constraints at all.

From a totally opposite research approach, [SVC+03] makes a lot of effort on defining user's context information in detail and identifies nine categories: User information, Personal Information, Activity Information, Social Information, User Defined Rules, Environment Information, Application Information, Terminal Information and Network information. However, their architecture is concerned with providing the information to service developers to build suitable new services in order to satisfy these context constraints.

In conclusion, current context-aware service selection approaches do not build the bridge to fill the gap between user's context and service's NFPs. Few of them give the clear picture of using user's context information for generating the service selection criteria/constraints.

Having presented a number of methods for service selection, we will now provide an overall comparison between these by considering whether they match the requirements (see Table 2.1). We will use two types of comparsion values: *yes/no* to show whether the approach achieved the requirement, and *low, average, high* to indicate at what level the approach reaches the requirement.

The table shows that most of the approaches lack flexible and automatic mapping methods for evaluating properties. Moreover, the level of expressing meaningful preferences is low. However, the usage of Semantic Web or ontology technologies has a huge advantage for addressing preference modelling and services non-functional properties. The other significant downside is short of dynamic aggregation methodology and automation process. By analyzing the current service selection approaches, we found that most of them are designed to address just one or a few aspects of the overall service selection problem.

2.6 The inContext Project Architecture

The "inContext" project is designated to support a context-aware environment for team collaboration. The major module in the project is the PCSA (Pervasive Collaboration Services Architecture) platform . The PCSA acts as a server for *User Agent* to consume collaboration services. It is a key enabler of SOA, and can be seen as a management and execution environment which enables discovery, registration, selection, and invocation of services, in a context-aware way. Five high level components are included in the platform (see Figure 2.2): *Access Layer*, *User, Team and Role Manager*, *Service Management*, *Context provider*, and *Data and Patterns Mining*.

- The access Layer is the single access point of inContext Platform exposed to *User Agent* and other PCSAs. It handles user authentication and authorization as part of the login procedure. It provides unified APIs for User Agent to lookup and invoke services as well as service registration. The unified APIs make it easier for User Agent to interact with the inContext platform. Only the *Access Layer* is visible from *User Agent* point of view.

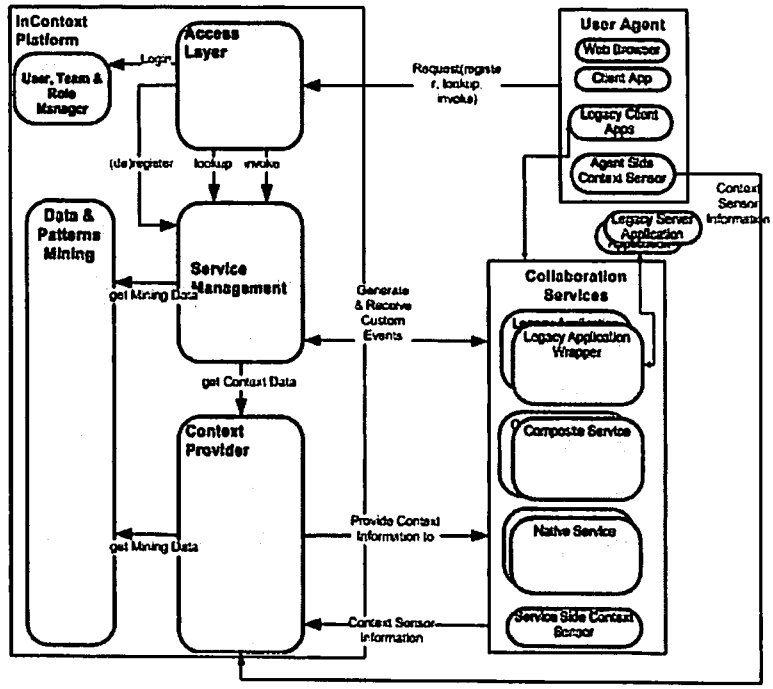

Figure 2.2: inContext architecture

- The User, Team and Role Manager manages information regarding user account, team members, etc. This information is vital for the PCSA. How a user uses the collaboration services (authentication/authorisation); what type of collaboration environment he/she is in; how he/she is supported in his/her collaboration activities. To answer all these questions, where the *User and Group Management* needs to be involved. Typically, the Access Layer interacts with the *User and Group Management* to process user authentication and authorization. *Service Management* interacts with it to provide relevance-based service discovery and composition. *Context Management* interacts with it to acquire user and team related context. Collaboration services interact with it to act in a more adaptive way.

- The Service Management is the component to manage all the operational aspects related to Collaboration Services. It supports service registration, discovery, composition and execution. Furthermore, with support

from *Context Management* and *User and Group Management*, it provides relevant and a proactive interface to *Service Management*. The relevance mechanism aims to dynamically provide collaboration services to user which are considered relevant in his/her current context. Proactive mechanism aims to anticipate user's future requirements and to automatically provide the most relevant services to him/her.

- Context provider manages collaboration related context information, including user context, team context, device context, environment context, etc. Context information plays a key role in the PCSA. The context is modelled using OWL/RDF language.

- Data and Pattern Mining keeps the PCSA continuously self improving by data mining and machine learning. It provides functions such as management of pattern storage, collection of log data from all kinds of sources, support for pattern retrieval for the use of other services.

The most relevant parts of the inContext project to the research presented in this thesis are service selection and composition approaches in the *Service Management* component and modelling and using the context information from the *Context provider* component.

The Service management component uses

- The meta data related service category repository concept which is introduced in this thesis to organise the services for requesting.

- The automatic context-aware criteria generation process which is illustrated in this thesis to set the service selection constraints at runtime.

- Our proposed TLE method and BCCbSS algorithm to provide the suitable service to the Access layer.

The Context provider uses the user context model, service NFPs model and service composition context model based on our simplified model presented in chapter 3 to represent the context information which is gained by the Data and Pattern Mining component.

However, our research work is not tightly coupled to this particular project. Our research aim is to develop a generic context-aware service selection methodology. The inContext project gives us a testbed and useful resources to deliver an evaluation of our research developments and results. Similar architectures have been proposed for multiple and large scala agent based systems, e.g. the Cougaar architecture. The Cougaar architecture "loads and manages software units called components that connect to and interact with one another through abstract interfaces" [HTW04]. Because the Cougaar architecture does not adopt the Web service concept, the underlining message transport uses traditional RMI or UDP based protocols. The Cougaar architecture can publish abstract interfaces via a blackboard mechanism which allows dynamic interface discovery by using traditional string exactly matching techniques and it does not have any selection method. As in the inContext project, the Cougaar architecture requires the additional methods to enable dynamic selection interfaces beyond discovery. In the light of this, Cougaar could reuse our research contributions.

2.7 Summary

SOA provides a new paradigm for organising and utilising distributed components to offer services, discover services and interact with end-users. In order to enrich computing environments with concepts that can improve interaction and ultimately increase productivity and reduce burdens on users, context-aware computing gives a concept to use context to provide relevant

information and/or service to the user. Context-aware service selection is an important development by combining SOA and context-ware computing.

We identified 6 requirements of *modelling for non-functional properties, properties preferences, automatic evaluation, dynamic aggregation, high automation, high scalability and accuracy* to develop a quality service selection method. Additionally, these requirements are based on context-awareness.

By analysing current related research work against the requirements, we find they lack flexible and automatic mapping methods for evaluating properties. Moreover, the level of expressing meaningful preferences is still low and lack of dynamic aggregation methods and automation processes. Most of the work are designed by focusing on one or few aspects of the overall service selection problem.

The involvement in the inContext project assists to develop a generic context-aware service selection method to cover current research issues. From the next chapter, we will start to illustrate the solution being contributed to the context-aware automatic service selection and composition field.

Chapter 3

Generation of Context-aware Service Selection Criteria

We listed three major challenges to achieve context-aware service selection in the Chapter 1. The first one is to build the connections between context information and the service NFPs. However, user context information normally is quite different from service NFPs and they are defined by different groups of people. For example, the user context may specify that a user has a mobile phone; but the service NFPs does not have the same description because services may only describe what types of message they can send. Therefore, we need a way to link "mobile phone" to "type of message". The link is the bridge we are looking for to develop the context-awareness. For us, the context information consists of active and passive user context and service composition context. In this chapter, we address this challenge by only considering the connections between user context and service non-functional properties. We will discuss composition context of services in the composition approach in Chapter 5. More precisely, the context-aware criteria generation process includes three major components:

OWL/RDF based User context model defines and stores the relevant user runtime context data which can be used for service selection.

Service repository stores different kinds of services' information organised by categories.

Runtime selection criteria are dynamically generated at service selection time and link to service NFPs and user context.

All these three components will be discussed in this chapter. User context modelling will be detailed in Section 3.1 which includes 4 context aspects. Service Data with detailed service repository and category will be discussed in Section 3.2. Context-aware service selection criteria generation process and implementation techniques will be introduced in Section 3.3. Finally, an example of context-aware criteria generation and chapter summary will be described in Section 3.4 and 3.5 respectively.

Parts of this chapter have been published in [YRM09a] and [YHHRM07].

3.1 User Context Modelling

To ensure the selection of the most suitable service for the user, correctly modelling user context is the first important step. Dey [DA99] identified problems arising from a context modelling point of view and [Vuk07] summarised them as 3 main requirements. Firstly, there is a need for a suitable context model, which describes the relationships between different types and facilitates inference and abstraction of context. Secondly, quality information is necessary to allow reasoning about the quality parameters of each context type and value, such as accuracy of location information. Finally, a suitable, easily developed infrastructure for context acquisition and management is required, which separates the acquisition from the utility of context.

	NFP model	Preferences model	Evaluate	Aggregation	Automation	Scalability	Context
Y. Liu	yes	average	no	low	low	average	no
H. Janicke	no	average	no	low	low	low	no
Y.Wang	no	low	no	average	low	high	no
S. Galizia	yes	low	no	low	low	high	no
Y.J. Seo	no	average	no	average	average	low	no
E. Al-Masri	unknown	average	no	low	average	high	no
X. Wang	yes	low	no	average	average	high	no
U.S. Manikrao	yes	low	yes	high	average	average	no
C. Schropfer	yes	high	no	high	low	low	no
E.M. Maximilien	yes	average	no	low	high	high	no
S. lamparter	yes	average	no	average	high	low	no
Cooltown-HP	no	low	yes	low	average	average	yes
S. Cuddy	yes	low	no	low	low	average	yes
A-R. El-Sayed	yes	low	no	low	average	average	yes
I. Sygkouna	no	high	unknown	unknown	low	unknown	yes

Table 2.1: Comparison results

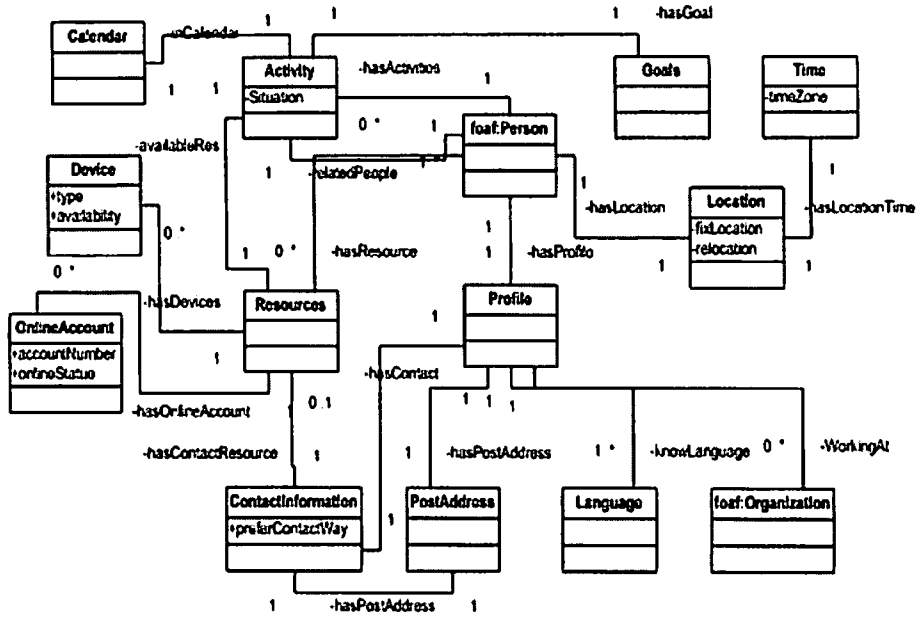

Figure 3.1: Top layer user context model

Supposing there are thousands of users and each one has various items of dynamic context information, then it will be very difficult and expensive for a centralized system to detect and update the context information at runtime. Consequently, the context information requires to be distributively stored and easily retrieved remotely. As we discussed in the related work section (Chapter 2), the OWL/RDF technologies can meet the context modelling requirements. Thus, we use OWL to model user context information and RDF to distributively store the instant user runtime context data.

By analysing the motivating selection scenarios and context information in general, our user context model has been divided into 4 packages of user profile, resources, activities and physical environment (location and time) which have shown in Figure 3.1. However, the 4 packages are not absolutely isolated from each others. Some of them may have overlapping subproperties or they connect to each other through their subproperties. In order to show the context structure more clearly, the OWL context model will be represented by using

OWL/RDF concept	UML concept
$< owl : Class >$	Class
$< owl : Property >$	Class Attributes
$< owl : Property >$	General association
$< rdf : domain >< rdf : range >$	Source and target of association
$< owl : subClassOf >$	Class inheritance
$< x\ rdf : ID =$ "X" $>$	Instance x of class X
Instance of $< rdf : Property >$	Instance of association

Table 3.1: Basic transformation relation from OWL/RDF to UML

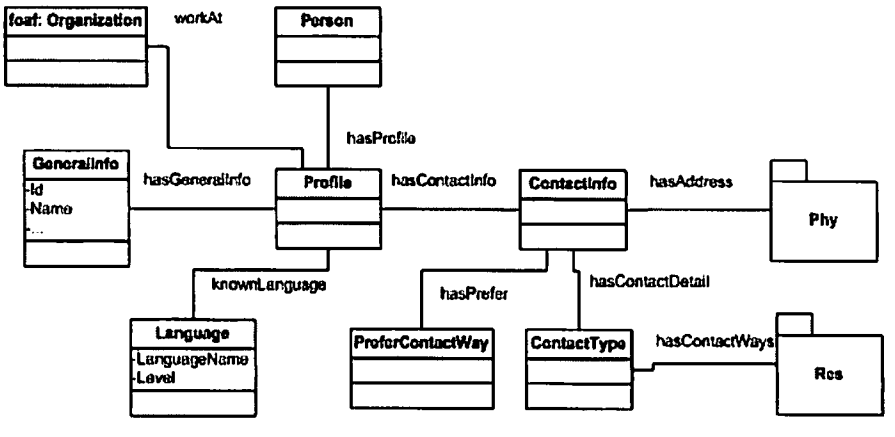

Figure 3.2: User profile package UML diagram

UML class diagrams in this chapter (The basic transformation relation from OWL/RDF to UML is listed in Table 3.1 and a more detailed explanation can be found in Appendix B).

3.1.1 User profile context

The user profile context stores user's personal data and contact information. The properties are represented in Figure 3.2. The profile property has the following subproperties.

ContactInfo class stores possible contact details of a particular person. *hasAddress* can store any kinds of address which defined in Physical environment context package. *hasContactDetail* contains other properties

including *hasDevice*, and *holdsAccount* which are defined within the Resource package. It is notable that *hasPrefer* property stores the best way to contact someone according to the person's preference setting. These context properties are related to the service non-functional properties of how to contact the person and where to contact the person in a dynamic environment.

GeneralInfo class saves the the user's identified id, name and other status. This context information is important to separate different users and specify relationships among them.

Organisation class holds the information of the users' working environment and roles that they may play in a task. It is understandable that different people who have different roles may require very different services for same task. The organisation class is inherited from the friend of a friend (foaf) ontology.

Language class shows how many languages this person knows and what level he/she can use. In service selection aspect, this property can restrict the values of language supporting NFP.

3.1.2 Resource context

The resource ontology (Figure 3.3) includes electronic documents which must be stored and read on a computing device, as well as physical resource such as communication devices and online accounts. The resource context connects to service NFPs, such as which devices are available currently and which communication type is the most fast way and so on.

ElectronicResources class is any electronic media content to be used in their electronic form. *hasDevice* holds the information of what electronic

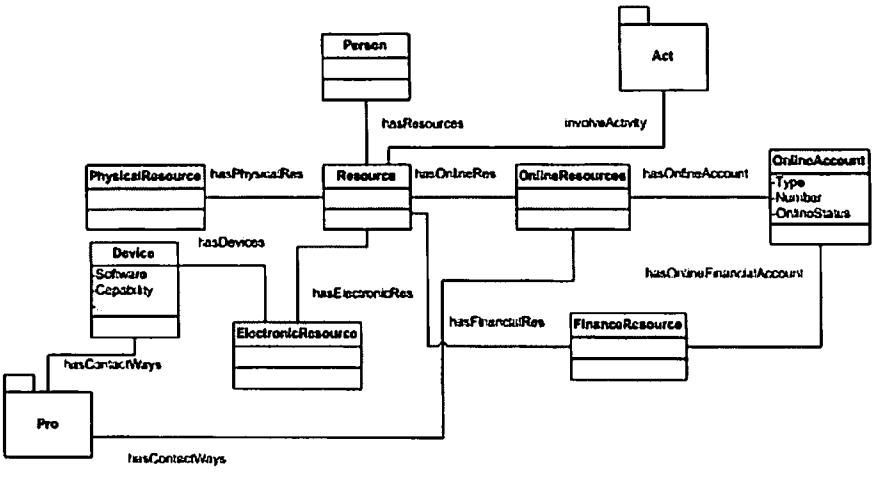

Figure 3.3: Resource package UML diagram

devices the user has currently. The device class also includes subclasses of installed softwares, device capabilities and so on, which are connected to user profile *hasContactway* properties. For service selection, this information is useful to support the matching between user's electronic devices and services' usability.

OnlineResources class saves the details information which relates to online resources that users have. The most important part is *hasOnlineAccount* used by any online service to identify oneself from another (e.g. IM, MSN, eBay, bank account etc.). hasOnlineRes property also connects to services'usability.

PhysicalResources class stores the information about hardware of the user has such as internet bandwidth, computer CPU and so on. This information is related to the restrictions of service's speed and hardware requirements for using the service.

FinancialResources includes the information of the financial restrictions on both cost and payment methods.

involveActivity property describes what activity is involved in using these resources.

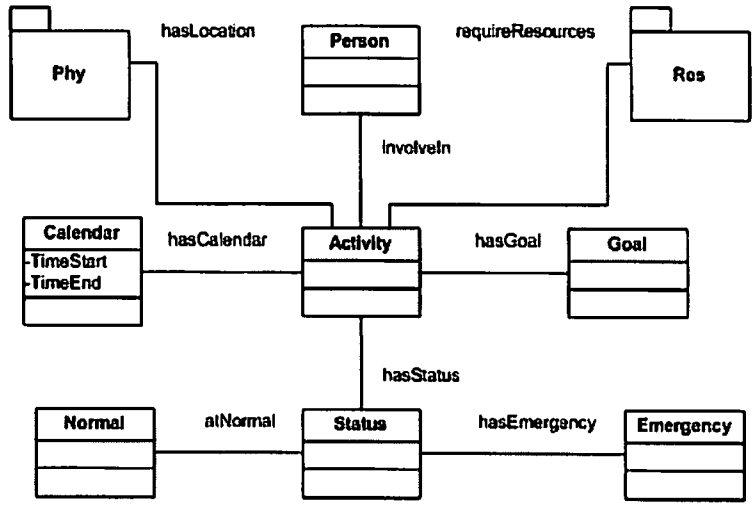

Figure 3.4: Activity package UML diagram

3.1.3 Activity context

Activity context (Figure 3.4) describes everything a person is doing (assigned tasks) in order to fulfil a goal. Each activity has a goal, a location and a person.

Goal class uses keywords (ontology based) to describe the performing task and its effects. This property implies the functional requirements of the service.

hasLocation states the location information to perform this activity which links to the location or distance constraints of selecting the service. This information stores in Physical environment context package.

Calendar class stores the schedule of the activity, which includes start time and end time.

Status class property can change the important level of different NFPs. For instance, emergency situation requires different set of important weights for the service non-functional properties from normal situation.

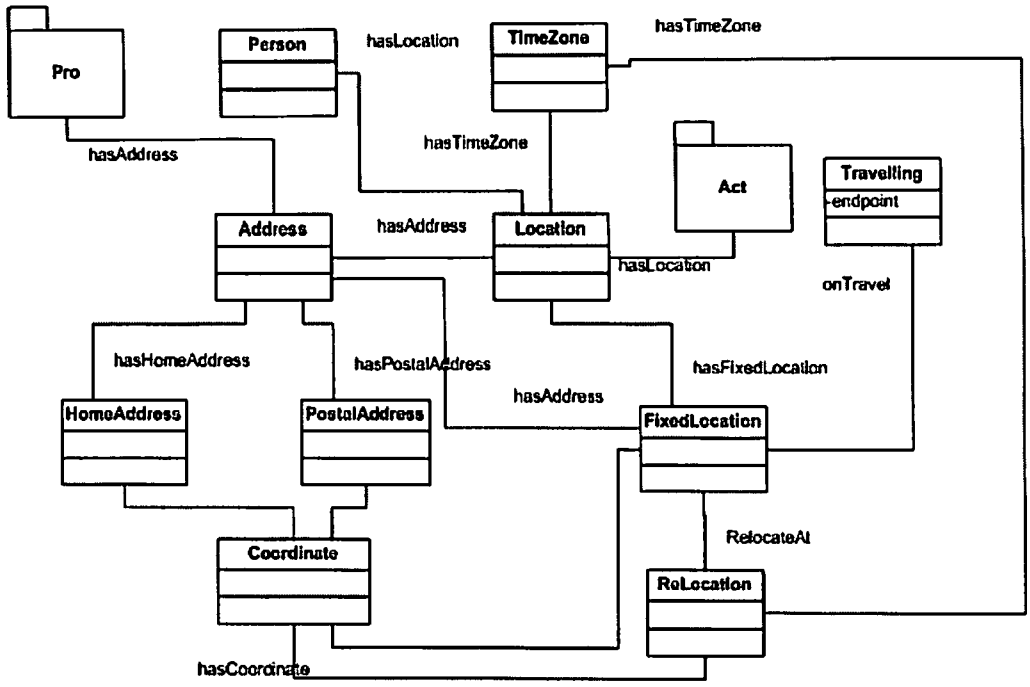

Figure 3.5: Physical environment package UML diagram

involveIn property links to people who are involved in this activity. Consequently, the context of involved people will be retrieved for the particular activity which requires service(s).

requireResources property stores the resources are needed for completing the activity. Resource information is stored in Resource package.

3.1.4 Physical environment context

Figure 3.5 shows the Physical Environment Package UML diagram. The physical environment context is the detail ontology of Location property. It utilises more precise indication of the location and time related constraints.

Address class is an abstract concept expressing fixed location. A fixed location may have different representations such as *GPS Coordinate* or

postal address. Class Coordinate describes real time position of a person/resource with a latitude longitude pair. *hasPostalAddress* covers country, city, street, building number and postcode, it is usually used for a fixed outdoor location of a working place. *hasHomeAddress* is similar to *hasPostalAddress* but for home address.

FixedLocation class is any location that does not move its position. It is usually defined by *PostalAddress*, a physical address or Coordinate, an exact spot (GPS) on the planet. *RelocateAt* property is used to specify movements of an entity. However, its current precise position can still be shown with *hasCoordinate* property. *onTravel* property specifies that the user is travelling at the moment. Therefore, there is no precise position information available but with the information of the destination of the travel.

Physical environment package also connects to other packages such as Activity and Profile Package.

3.2 Service Data

3.2.1 Service repository and category

The service repository contains different kinds of categories (see Figure 3.6) which classify services by their functional properties (e.g. notification service, flight booking service and medical support service). Service functional properties are described by the ontology-based key words and IOPE (Input, Output, Preconditions and Effects) specifications (this only related to service discovery step which is not the issue we are addressing in this dissertation). The formal definition of category is as follows:

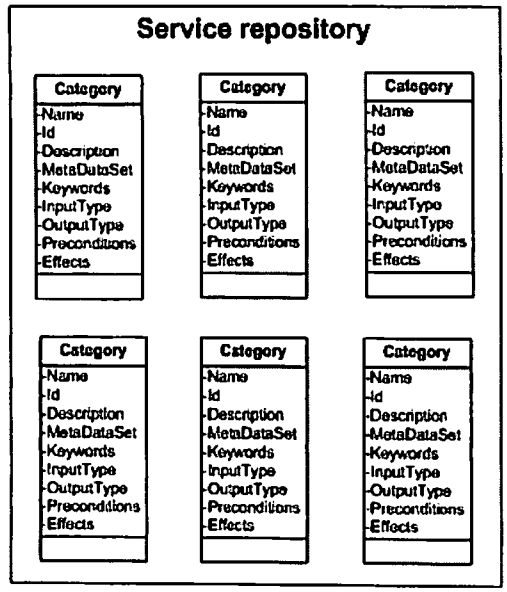

Figure 3.6: Service repository

Definition 3.1 *A category is a tuple < Name, Id, KeyWords, Description, MetaDataSet, InputType, OutputType, Preconditions, Effects>.*

Name is the syntactic name of the category.

Id is the unique identifier of the category.

Keywords are the ontology-based selection of words which describe the functional properties of the category.

Description is a name detailed explanation of the category in order to service providers to decide which category is appropriate for their services.

MetaDataSet is a set of meta data (non-functional properties) definitions. A MetaDataSet may include more than one definitions and they are different for different category domains.

InputType presents the input parameters requirement.

OutputType presents the output parameters requirement.

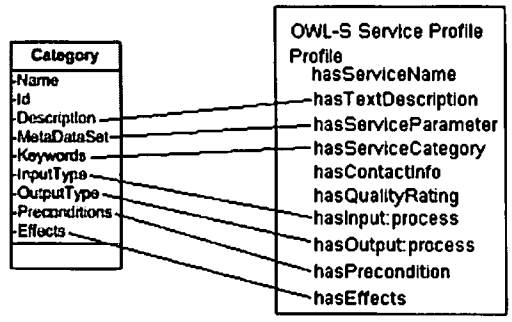

Figure 3.7: Relations between category concept and OWL-S profile ontology

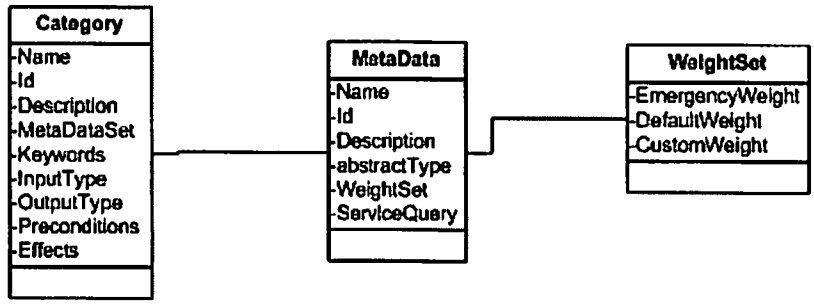

Figure 3.8: The conceptual model of MetaData

Preconditions present the required conditions for invoke the service.

Effects present the post-conditions after service was successfully invoked.

Figure 3.7 shows the mapping between *Category* concept and OWL-S service profile ontology.

3.2.2 Meta data of category

As we defined in Definition 3.1, a MetaDataSet in the category is a set of meta data which describes considered NFPs of the service. Each meta data is formally defined as Definition 3.2 (also see Figure 3.8)

Definition 3.2 *Meta data is a tuple of the form < Name, Id, Description, AbstractType, WeightSet, ServiceQuery >.*

Name is the syntactic name of the meta data item.

Id is the unique identifier of the meta data item.

Description is a detailed explanation of the meta data which introduces the way to specify this meta data inside a service OWL file (e.g., unit and data formation of a meta data).

AbstractType is the high level evaluation type to identify the adequate evaluation method (this is discussed in more details in Chapter 5). At this moment, it can be seen simply as a link syntax between meta data and its type of evaluation function.

WeightSet is the importance level of this meta data item in a particular category domain. Different situations imply different importance considerations of the meta data. So the weight are composed of EmergencyWeight, DefaultWeight and CustomWeight. In order to specify the situation that the minimum value is the best evaluation respect, we use negative values of (-1,0) to define the importance semantics of the meta. Otherwise we use (0,1) value. (The formal semantics of the weight values are defined in Chapter 5.)

ServiceQuery is a SPARQL (SPARQL Protocol and RDF Query Language) query statement [PS06] which can be used to locate the meta information from a published service description.

In order to clarify meta data should be specified, the NFPs are discussed more deeply here.

3.2.3 Service NFPs

The functional properties are normally described as IOPE specifications. The functional properties mainly specify the requirements of service invocation and promised deliveries from successful service execution. In contrast to functional

properties, NFPs refer to the attributes which describe the QoS (Quality of Service, e.g. security and speed) and other meta data about the service (e.g. provider, invoke policy and prices).

[Tom07] separated NFPs as annotation attributes and non-functional behavioural attributes. The annotations attributes are the properties which can apply to all element descriptions, e.g. services, goals, mediators, ontology. They simply provide a way to annotate and to provide meta data about any type of element description.

[LNZ04] distinguished NFPs by generic quality criteria and domain specific criteria. The generic criteria are applicable to all services, e.g. price and execution duration. The domain specific criteria refer to the different concerning aspects of the services from different service selection domains. For example, the printing service may concern NFPs of speed, colour options and quality. However, the weather service may consider accuracy and range. By summarizing previous NFPs definitions, we divide NFPs into following seven groups.

General properties are provided by service providers to describe the provider's information such as name of the provider, the location and price of the service and other provider related properties.

Trust properties are provided by the third parties who can monitor the services behaviours and give the feedback about the services, e.g. reputation, trust and satisfaction of the service.

Usability properties may have the elements of availability, policy, communication protocol and domain specific elements. Availability indicates whether the service is alive or not. Policy tells who, where the service can be used or not be used. For example, the service does not support invocation from mobile-application. Communication protocol explains

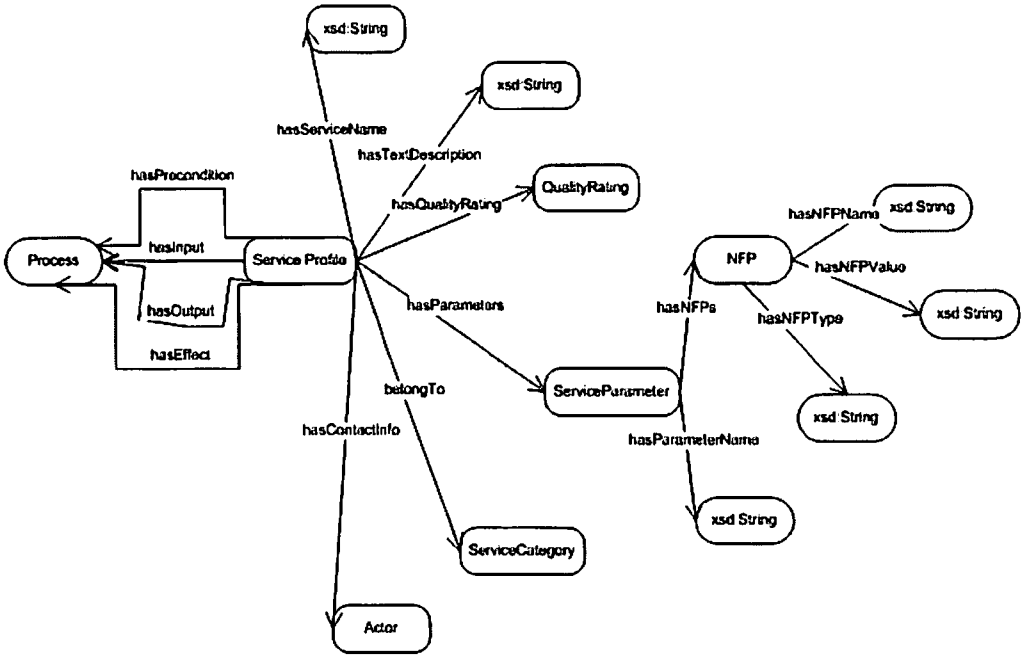

Figure 3.9: OWL-S: service profile ontology

how the services communicate with the surrounding environment, such as SOAP or REST (Representational state transfer).

Reliability properties may have the error rates and stability. Error rates shows chances of getting unsuccessful results from a service. The stability represents the long term stable performance of a service.

Efficiency properties may have the execution duration, accessibility and accuracy. The execution duration describes the speed of the service for completing a successful task. The accessibility shows the states of being accessible or not (e.g. the service is not accessible from wireless). The accuracy indicates the correctness and freshness level of the data or information produced by a service.

Security properties may include the privacy and security status. Privacy explains how a service protects its end-user's information. Security status

tell the level or standards of the service security.

Domain specific properties define the extra considerations for some specific types of services. For example, printer service may have colouring option as the extra NFPs, but weather service will not have.

In order to express NFPs syntax within OWL-S, we extended the OWL-S profile ontology by adding NFP class and hasNFPs property between serviceParameter and NFP classes. The extended service profile ontology is visualised as Figure 3.9. Code 3.1 is a fragment of NFPs specified in the OWL-S profile and a complete OWL-S profile example shows in Appendix A.

```
<profile:Profile>
...
    <profile:hasParameter rdf:resource="#parameter"/>
    <Parameter rdf:ID="parameter">
        <hasParameterName rdf:datatype="&xsd;string">NFPParameter</hasParameterName>
        <hasNFP rdf:resource="#typeOfMessage"/>
        <hasNFP rdf:resource="#price"/>
        ...
    </Parameter>
    <NFP rdf:ID="price">
        <hasNFPName rdf:datatype="&xsd;string">serviceFee</hasNFPName>
        <hasNFPType rdf:datatype="&xsd;string">numerical</hasNFPType>
        <hasNFPValue rdf:datatype="&xsd;string">0.5</hasNFPValue>
    </NFP>
    <NFP rdf:ID="typeOfMessage">
        <hasNFPName rdf:datatype="&xsd;string">messageType</hasNFPName>
        <hasNFPType rdf:datatype="&xsd;string">stringSet</hasNFPType>
        <hasNFPValue rdf:datatype="&xsd;string">IM</hasNFPValue>
    </NFP>
    ...
</profile:Profile>
```

Code 3.1

If we query the value of "typeOfMessage" NFP of the service, the SPARQL query should be stated as Code 3.2.

```
Namespace...

SELECT ?value

WHERE {:typeOfMessage :hasNFPValue ?value}
```

Code 3.2

3.2.4 Service register

According the specifications of category and meta data, the service provider can register a service into a category. A service is defined as Definition 3.3.

Definition 3.3 *A service is a tuple of < Name, Id, Provider, OWLURL, WSDLURL >.*

Name is the syntactic name of the service.

Id is the unique identifier of the service.

Provider is the service provider information.

OWLURL is a URL link of the location of the OWL-S description file. The OWL-S file should include the required NFPs meta information about the service.

WSDLURL is a URL link of the invoke endpoint of the service.

Registering a service requires 3 steps:

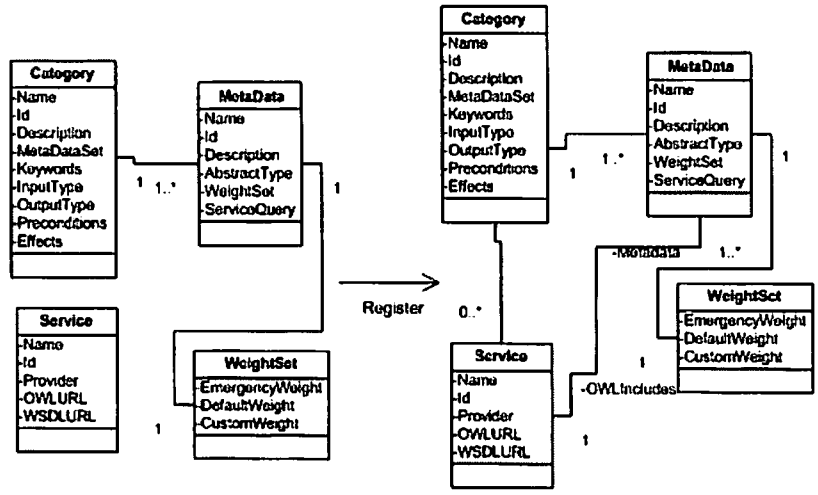

Figure 3.10: Service registration

1. Selecting category, a service provider should go through the service repository to search a service category that is the most suitable one for his service according to the category description and key words.

2. Describing the NFPs defined in the selected category inside the service OWL-S file.

3. The provider deploys the service information into the service repository with name of the selected category.

Figure 3.10 represents the service models before and after registration.

3.3 Context-aware Criteria Generation Process

At this stage, on the one hand, we defined user context information which related to the service selection. On the other hand, services's NFPs have been registered as meta information in the service category. To link the two sides for context-aware service selection goal, we use context-aware criteria concept which is the bridge between user context and service NFPs. Service

selection method ranks the competitive services according to these context-aware criteria. The criteria generation should have two main characteristics:

1. Context-awareness ensures correct criteria to be generated according to different user scenarios. For example, the location criterion should be generated differently and timely when user's location is changing. Another context-aware scenario is when a user asks a medical support service to help a minor injured person, the location criterion is more important than the support level criterion because all the service can deal with the minor injury situation. However, if it is a emergency case, the support service is more important than location since only high level support service can handle this emergency situation.suitable

2. Automation facilitates the service selection process to support automatically service evaluation. To achieve this aim, the evaluation criteria have to be generated dynamically without human interactions.

To meet these two requirements, a dynamic service selection criteria generation process is developed. The basic idea of criteria generation is that the service selection criteria are initialized from the category meta requirements at service selection time and the constrained values of each criterion are modified by the runtime user's context information (and composition context information for composition scenarios which will be discussed in Chapter 5). In order to illustrate the details of the generation process, the criterion and SPARQL matching table need to be discussed first.

Definition 3.4 *A criterion includes < Name, Id, AbstractType, Value, WeightSet ContextQuery, ServiceQuery >.*

Name is the understandable syntax of the criterion.

Id is the unique identification of the criterion.

AbstractType is the high level evaluation type mode to match the adequate evaluation methods to this criterion. This concept is as same as the one defined in Definition 3.2 (more discussion is in Chapter 5) and it is automatically generated from meta information in the category.

Value is the constraint value of the criterion.

For instance, "Leicester" is the location of the current user. Therefore, the desired service should be available and workable in this area. Actually, the value of this element is gained automatically from the user's context. This factor shows one aspect of context-awareness.

WeightSet is the importance level of the criterion and it is as same as WeightSet defined in Definition 3.2.

ServiceQuery is a SPARQL query statement which can be used to locate this meta information from published service description.

ContextQuery is similar to ServiceQuery, it stores the query statement to get the correct related context data as the criterion value.

We also need a SPARQL query matching table to indicate which ContextQuery statement is used for querying user runtime context information that is related to the criterion. Furthermore, SPARQL reasoning rules are also required to work together with ContextQuery in order to gain more correct and accurate user context value. For example, Code 3.3 shows the ContextQuery (SPARQL query statement) for obtaining user's current available devices.

```
Namespace...

SELECT ?devices
```

ServiceQuery	ContextQuery	rule
ServiceQuery1	ContextQuery3	rule13
ServiceQuery2	empty	empty
ServiceQuery3	ContextQuery9	rule3, rule5
...	...	...

Table 3.2: SPARQL query matching table

```
WHERE {:Resource :hasElectricRes ?er.

er:hasDevices ?devices}
```

Code 3.3

However, we are not going to details of how the context reasoning rules apply to the context query in this dissertation. Here, we have three query/rule pools for storing ServiceQuery, ContextQuery and Reasoning rules respectively. It allows an empty query and rule to be existed. Empty ContextQuery means that the runtime criterion constrained value is going to be empty for evaluation. Without value, the evaluation function will give the highest score 1 to the service that has the best value for the criterion, 0 score to the service that has the worse value and score $\frac{Currentservicevalue}{Bestservicevalue-Worstservicevalue}$ for other services in the between (more details about criteria evaluation and aggregation are introduced in Chapter 4). The SPARQL query matching table shows in Table 3.2.

The whole generation criterion process includes 5 steps which are represented in Figure 3.11.

1. Initialise all the criteria from matched service category MetaDataSet.
2. Search the SPARQL matching table to obtain the ContextQuery statements to each criterion.

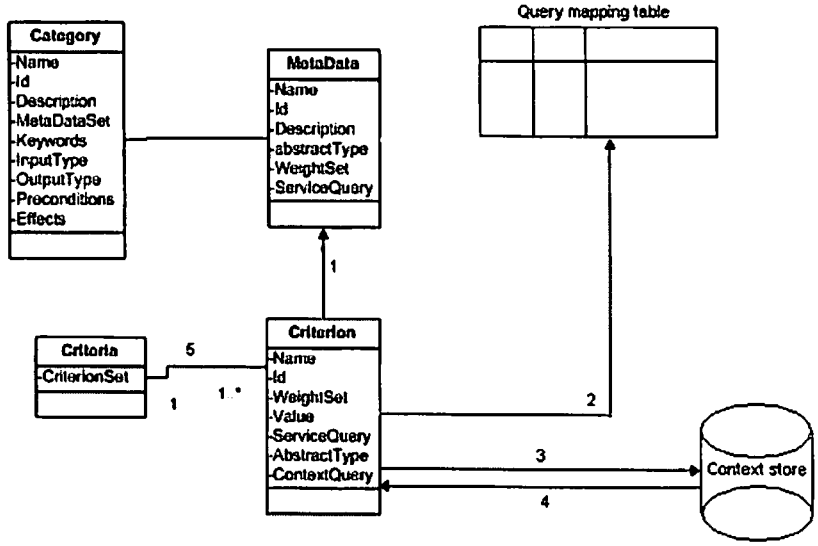

Figure 3.11: Criteria generation process

3. Using the matched contextQuery statement to retrieve the runtime user context information which related to the criterion. The value of the criterion is modified by using the *ContextQuery* string (it is a SPARQL query statement) to invoke the Jena context query engine for getting the correct and runtime constrain value from user context data.The weight set is selected by automatically searching what activity status are currently for requesting the service. If the statu is emergency, then the emergency weight set is used. Otherwise, the default weight set is selected. In single service selection scenarios, the custom weight set is selected when the weights are defined by users..

4. Assign the retrieved runtime context values and data to the values and WeightSet defined in each criterion.

5. Group all criteria into a runtime criteria set for service selection method to use.

Until now, the user context-aware issue is solved by applying this automatic context-aware criteria generation process.

3.4 Criteria Generation Example

To make the process of automatically generating the context-aware criteria more understandable, we take the previously discussed notification service selection scenario (see section 1.1.2) as an example in this section.

From service meta data aspect, the notification service category may consider the following NFPs.

- *Covered location*: the areas which can be covered by the notification service.
- *Type of message/devices*: the notification service may send SMS, IM, e-mail which may also be received by mobile phone, computer or PDA.
- *Price*: different notification service may have different costs.
- *Response time*: notification services require some time to respond for the service invocation. The response time may be affected by type of messages, network ability and other human or unhuman factors.
- *Prefer contact method*: user may consider what type of message she/he would like to receive.
- *Privacy*: the levels of the privacy are protected by the notification service.

The weights for different non-functional meta data are firstly defined by service domain experts as default, such as 1 for *Covered location* and *Type of message*, -0.3 for *Price*, 0.3 for *Prefer way*, -0.9 for *response time* and 0.5 *for Privacy*. Here, 1 means the non-function properties must be satisfied as hard requirements. The bigger absolute weight value means the criteria is more important. The weight of (-1,0) means the smaller meta data is desired. The

weight of (0,1) means the bigger meta data is desired. The detailed meaning of values from -1 to 1 will be explained in Chapter 4. However, the default weights setting can be redefined for emergency situation as emergencyWeight set. The emergencyWeight set should be designed by both domain experts and service client users learning from the emergency experiences. Moreover, the users can always set the customWeight for their special context requirements.

From user context side, the participate may has following context data recalling the scenarios in Chapter 1:

1. Bob is on holiday with only his mobile phone and PDA. The holiday location is in Spain. On holidays, Bob prefers to be contacted by PDA rather than mobile phone message.

2. Alice has switched off her mobile phone to conserve battery power, but she is online using IM (IM is also her preferred method of contact).

3. John is working in his UK office with access to a wide variety of communication devices (mobile phone, email, IM and PDA). However, he prefers to be contacted by email message.

From the user context, we can find the above three context constraints matched to three category meta considerations of *Covered location*, *Type of message/devices* and *Prefer way*. However, the non-mentioned three types of meta are hidden implied as high as possible for Privacy and as low as possible for price and response time. These hidden constraints do not specify the instant data values for generating the criteria.

Based on the category meta specification and user context information, the notification service selection criteria generated for Bob in normal situation is shown in Figure 3.12.

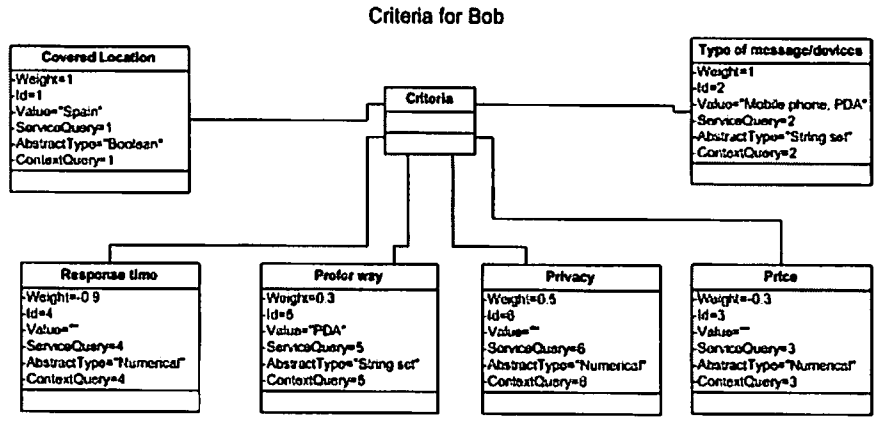

Figure 3.12: The context-aware selection criteria for Bob

3.5 Summary

In this chapter, we illustrated the first contribution of the thesis: using context information together with service non-functional meta data to automatically generate the service selection criteria. Because the generated criteria reflect the user context information, we call them *context-aware criteria*. The Context-aware criteria are going to affect the further service selection steps.

Refer to the proposed overall service selection process, we discussed three components as shown in Figure 3.13.

First. user context are modelled including four aspects: user profile, resources, activities and physical environment. We explained the relations between the context model and the service NFPs.

Secondly, we introduced a new service repository system to allow separating services into different kinds of categories which are not only based on the functional properties but also non-functional meta data considerations. Specially

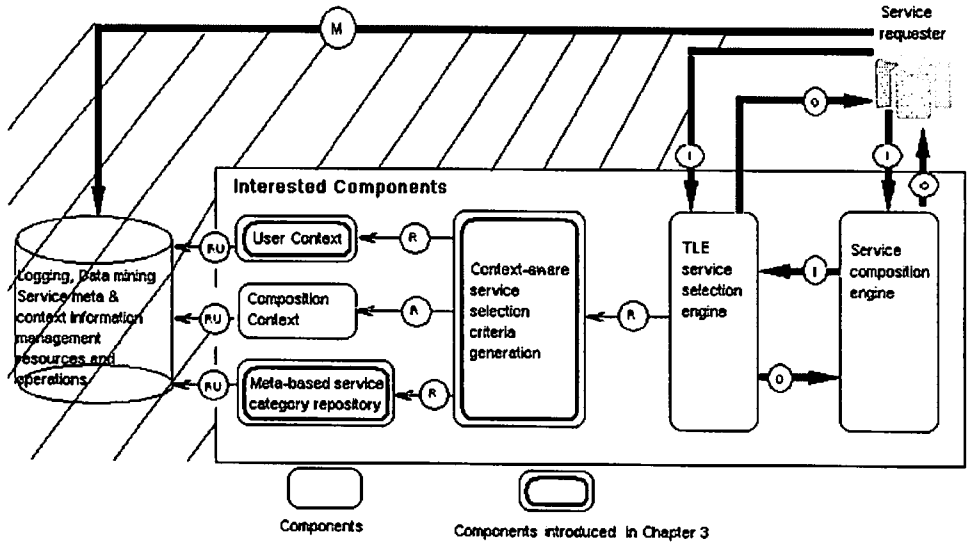

Figure 3.13: Components are discussed in Chapter 3

we explained the way to extend existing OWL-S profile ontology by adding NFP class associates to ServiceParameter class.

Finally, based on the context model and the new category system, the service selection criteria are automatically generated to reflect the user context by applying SPARQL query technology.

The context-aware service selection criteria are the first and key step to achieve runtime user context-aware automated service selection. The methodology applying these criteria for service selection will be discussed in the next chapter.

Chapter 4

Service Selection Method

We aimed to address the three challenges of context-awareness, automatic service selection and service composition and introduced the solution to the first in the previous chapter by dynamically generating the context-aware service selection criteria which will be used for service selection. As discussed in Chapter 2, there are 6 crucial requirements for developing a suitable and reliable service selection method, besides context-awareness. Our earlier survey concluded that the currently existing service selection methods are not capable to deal with at least 3 requirements: (1) combining criteria evaluation functions into the selection method, (2) dynamic selection of aggregation functions and (3) the automation of both the evaluating and aggregation processes. However, these 3 requirements are very important in order to achieve the context-aware service selection aim.

In this chapter, we will concentrate on addressing these 3 issues. The remainder of the chapter is organised as follows.

Section 4.1 illustrates the overview of the proposed service selection method. Section 4.2 explains the details of the proposal type-based criteria evaluation function. Section 4.3 discusses the LSP (Logic Scoring Preference) aggregation

function and our extension to it. A worked example in Section 4.4 makes the results more concrete and finally we draw a conclusion in Section 4.5.

Parts of this chapter have been published in [YRM08a] and [RMYT09].

4.1 Service Selection Method Overview

There are two foundations of our proposed TLE (Type-based Logic Scoring Preference Extension) service selection method. One is the type-based criteria evaluation function. The other is an extended LSP aggregation function. These two foundations will be fully explained in section 4.2 and 4.3. In this section, the overview of the selection method and the selection process will be presented. There are three functions involved in the selection method.

Type-based criteria evaluation function. This component is designated to dynamically evaluate the different criteria. The inputs of the evaluation function are the context-aware service selection criteria and a list of functionally competitive services with their meta URL. The outputs are the evaluation scores for different criteria and services. The function has two features. First, it automatically matches the evaluation methods to the criteria based on their abstract types. Second, it can query all services related meta data and evaluate them according to the context-aware criteria.

Orness degree calculation function. This is used to automatically decide which aggregation function should be applied to the selection function based on the criteria preferences and criteria evaluation results. The orness degree presents the degree position between simultaneity and replaceability.

LSP extended aggregation function. This is applied to aggregate different criteria evaluation scores of a service into a single overall score. As a result, all services have their own aggregated single evaluation score based on the same set of context-aware criteria. The aggregation uses the appropriate function as obtained by the orness degree calculation function.

4.2 Type-based Criteria Evaluation Function

Automatically evaluating different criteria is one of the challenges in a service selection method. Most of the traditional criteria evaluation functions strongly rely on human intervention. For example, criteria used for selection are tightly bound to a predefined evaluation function. In particular, the criteria have various types discussed when we introduced the abstract type in Chapter 3. The context-aware environment requires different evaluation functions to fit into their data types. Using static mapping techniques, when a new criterion is added into the evaluation environment, a new evaluation function or mapping needs to be manually predefined as well. Meanwhile, changing criteria requirements might also need manual modification of details of the mapped evaluation function. In a context-aware environment, however, context changes frequently and unpredictably. Therefore, automatic criteria evaluation becomes a crucial need in the service selection method.

Keeping different types and automatic function mapping in mind, we develop a type-based criteria evaluation function. This function can automatically match the evaluation methods to the criteria by detecting their abstract types rather than having to identify an evaluation function for each individual criterion.

By analysing the motivating scenarios in Chapter 1, we define four abstract types of criteria and each type has a related evaluation function.

1. *The numerical* type is used for criteria which take numerical input to the evaluation method such as cost, time and measurement values. The mapped evaluation method is given by equation 4.1, where w is the weight of the criterion. When the criterion is of numerical type, the weight can be in the range [-1, 0) or (0, 1]. [-1, 0) means that a smaller numerical value is desired (as e.g. for price properties). v is the value for the service under evaluation, v_{max} is the maximum value of all competitive services for the criterion. v_{min} is the minimum value of all competitive services (if no constraint is specified, it means the lowest value is preferable) or the constraint value of the requirement from the context information.

$$\varepsilon = \begin{cases} \frac{1-(v_{max}-v)}{v_{max}-v_{min}} & \text{iff } w \geq 0, \\ 1 & \text{iff } w \neq 1 \text{and } v_{max} = v_{min} \\ 0 & \text{iff } w = 1 \text{and } v_{max} = v_{min} \\ \frac{v_{max}-v}{v_{max}-v_{min}} & \text{otherwise.} \end{cases} \tag{4.1}$$

2. *The boolean* type is used for criteria which have a certain value evaluated as 1 or 0. The boolean type normally implies an exact match requirement, e.g. the yes/no criteria. It also can be single string and numerical match. The evaluation function is:

$$\varepsilon = \begin{cases} 1 & \text{iff criterion is met,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

3. *The string set matching* type is used to define the criteria which are measured by the size of the evaluation objects' satisfaction subset. For example, a person has Visa, Master and Solo cards (this is a value set) and a service supports payment by Visa card only, then the set match

level is $\frac{1}{3}$. If the set only contains one value, then the evaluation function becomes essentially identical to the boolean type.

$$\varepsilon = \frac{\varepsilon_1 + \varepsilon_2 + ... + \varepsilon_i + ... + \varepsilon_n}{n} \tag{4.3}$$

where ε_i is a score for each element of the set.

4. *The distance* type is defined to calculate the distance of two geographic coordinates:

$$\varepsilon = \begin{cases} R \times c & \text{iff } c \geq 1 \\ R \times 2 \times \arcsin(1) & \text{otherwise.} \end{cases} \tag{4.4}$$

$c = 2 \times \arcsin k$

$k = \sqrt{\sin 2(\mid L2 - L1 \mid /2) + \cos(L1) \times \cos(L2) \times sin2(\mid G2 - G1 \mid /2)}$

with L1 and L2 being latitude and longitude of the first point, L1 and L2 being latitude and longitude of the second point and R being the Earth mean radius of 6371 km.

We do expect there will be more types which need to be defined for different evaluation contexts and environment. However, these four types proved sufficient for our case studies discussed in Chapter 1. If further types are added, then this should not affect the feasibility achieved by our selection method, as each criterion has a specific type and has a related evaluation function.

4.3 LSP-based Aggregation Function

4.3.1 LSP function in literature

LSP (Logic Scoring Preference) aggregation function introduced in [Duj75] is one professional evaluation method initially designed for solving hardware selection problems [Duj]. LSP extends the traditional scoring techniques [III70] by adding continues logic to the aggregation function. The traditional scoring techniques captured by equations 4.5 through to 4.9 represent the totally different aggregation requirements, where W_i is the weight and E_i is individual criterion evaluation result.

$$E = \bigvee_{i=1}^{n} W_i E_i \tag{4.5}$$

Equation 4.5 represents the full disjunction aggregation function which means that the final aggregating score is the highest evaluation result of the criteria. For example, $E = 0.3 \cdot 0.4 \vee 0.2 \cdot 0.8 \vee 0.5 \cdot 0.3 = 0.16$. This function implies the highest score can replace the lowest score. Thus, full disjunction aggregation operates full replaceability.

$$E = \bigwedge_{i=1}^{n} W_i E_i \tag{4.6}$$

Equation 4.6 is the opposite function of full conjunction, which does not allow higher scores to affect the final aggregation result. For example, $E = 0.3 \cdot 0.4 \wedge 0.2 \cdot 0.8 \wedge 0.5 \cdot 0.3 = 0.12$. However, these two aggregation functions are two extremes useful for some evaluation tasks. For example, one person may consider price and performance as criteria to select the car. If the price is evaluated as full satisfaction, then the ranking result will be 1 without considering the other performance criterion by using equation 4.5. On the

other hand, if the price is not satisfaction at all, then the ranking result will be 0 without considering the performance criterion by using equation 4.6.

To fill the middle ground between the extremes of full disjunction and full conjunction, we can consider weaker disjunction, average aggregation and weaker conjunction function represented in equation 4.7, 4.8 and 4.9. In fact, these aggregation functions represent the fine-tuned logic relations between different evaluation criteria. Normally, these logic relations can be decided upon by human based analysis according to the evaluation requirements. For example, one person may consider both safety and performance criteria as important for buying a car. Therefore, the aggregation function should show a more conjunctive meaning rather than disjunctive meaning. In other words, the person does not like the high safety score to hide the low performance factor. In real world evaluation scenarios, there are more complex situations which require more than 3 aggregations between disjunction and conjunction.

$$E = \sum_{i=1}^{n} W_i E_i^2 \tag{4.7}$$

$$E = \sum_{i=1}^{n} W_i E_i \tag{4.8}$$

$$E = \prod_{i=1}^{n} W_i E_i \tag{4.9}$$

$$E = (\sum_{i=1}^{n} W_i E_i^r)^{\frac{1}{r}} \tag{4.10}$$

Equation 4.10 captures the initial idea of LSP method manually to amend the usages of the aggregation method by only modifying a single parameter r, where $\sum_{1=i}^{n} W_i = 1$ and $\sum_{1=i}^{n} E_i = 1$. The selection values of r are defined

Operation	Symbol	d	r2	r3	r4	r5
DISJUNCTION	D	1.0000	+infty	+infty	+infty	+infty
STRONG QD (+)	D++	0.9375	20.630	24.300	27.110	30.090
STRONG QD	D+	0.8750	9.521	11.095	12.270	13.235
STRONG QD (-)	D+-	0.8125	5.802	6.675	7.316	7.819
MEDIUM QD	DA	0.7500	3.929	4.460	4.825	5.111
WEAK QD (+)	D-+	0.6875	2.792	3.101	3.318	3.479
WEAK QD	D-	0.6250	2.018	2.187	2.302	2.384
SQUARE MEAN	SQU	0.6232	2.000			
WEAK QD (-)	D--	0.5625	1.449	1.519	1.565	1.596
ARITHMETIC MEAN	A	0.5000	1.000	1.000	1.000	1.000
WEAK QC (-)	C--	0.4375	0.619	0.573	0.546	0.526
WEAK QC	C-	0.3750	0.261	0.192	0.153	0.129
GEOMETRIC MEAN	GEO	0.3333	0.000			
WEAK QC (+)	C-+	0.3125	-0.148	-0.208	-0.235	-0.251
MEDIUM QC	CA	0.2500	-0.720	-0.732	-0.721	-0.707
HARMONIC MEAN	HAR	0.2274	-1.000			
STRONG QC (-)	C+-	0.1875	-1.655	-1.550	-1.465	-1.380
STRONG QC	C+	0.1250	-3.510	-3.114	-2.823	-2.606
STRONG QC (+)	C++	0.0625	-9.060	-7.639	-6.689	-6.013
CONJUNCTION	C	0.0000	-infty	-infty	-infty	-infty

Figure 4.1: GCD mean operators [Duj]

as GCD (Generalized Conjunction-Disjunction) function in Continuous Preference Logic [Duj07]. The currently defined 20 GCD operators with accepted symbols, disjunction degrees and parameter for r values (numbers of criteria range from 2 to 5) are presented in Figure 4.1 and more detail and large number of criteria mapping calculation from disjunction degree to r can be found in [Duj]. For example, the r value can be calculated by equation 4.11 for 2 criteria (or simply obtained from the table). In Figure 4.1 "QD" stands for quasi disjunction and "QC" stands for quasi conjunction. "d" shows the degree level between disjunction and conjunction. Each degree level has a set of matched r values according to different numbers of criteria. Disjunction degree level is also called *Orness* degree.

$$r = \frac{-0.742 + 3.363d - 4.729d^2 + 3.937d^3}{(1-d)d} \tag{4.11}$$

In short, LSP aims to evaluate quantitative features for the comparison of different entities. Recently, LSP has been used to deal with selecting Data Management systems, Hardware/Software systems and web sites evaluation

[Duj, SDB+87, OR02]. The research results show the selection problems are solved satisfactorily. However, there are three difficulties with adopting the initial LSP aggregation function into dynamic and automatic context-aware evaluation environment.

1. There is no automatic method defined to calculate the power r. Currently, the r value is defined based on manual Orness degree analysis by domain experts. This is clearly not suitable for dynamic context-aware evaluation requirements.

2. It is very difficult to give the semantics to the value of weights by having constraint of $\sum_{i=0}^{n} W_i = 1$ which limits the freedom to express the real important considerations for different criteria. For example, if there are two equal important criteria and one less desired criteria say 0.1, then we have to share the value of 1-0.1 to assign 0.45 to each of the important criteria. In this case, 0.45 means the most important. However, people may want to use 0.9 to express how the criteria are important for them rather than 0.45. Without weight semantics, it is impossible to argue the reason why give the criterion 0.6 weight but not 0.4 or 0.5. This problem is not only in the LSP function, but also in all of the existing aggregation functions.

3. The original LSP method does not clearly distinguish hard criteria(criteria that must be satisfied) and soft criteria (criteria that are desired to be satisfied). It only separates them by the value of weights. However, as we discussed on the above, without semantics for weights it is difficult to automatically recognise the hard criteria.

To overcome these limitations, we extend the LSP function in three aspects: (1)defining weight semantics, (2)separating hard criteria from soft criteria and (3)automatically computing the *Orness* degree.

Weight	Importance Level
$\mid W_i \mid = 1$	hard criterion
$0.9 \leq \mid W_i \mid < 1$	highest importance criterion
$0.8 \leq \mid W_i \mid < 0.9$	higher importance criterion
$0.7 \leq \mid W_i \mid < 0.8$	priori importance criterion
$0.6 \leq \mid W_i \mid < 0.7$	normal importance criterion
$0.5 \leq \mid W_i \mid < 0.6$	lower importance criterion
$0.4 \leq \mid W_i \mid < 0.5$	high desire criterion
$0.3 \leq \mid W_i \mid < 0.4$	higher desire criterion
$0.2 \leq \mid W_i \mid < 0.3$	low desire criterion
$0.1 \leq \mid W_i \mid < 0.2$	lower desire criterion

Table 4.1: Weight semantics

4.3.2 Defining weight semantics

Defining the semantics of the weights is crucial to allow users (e.g. category definers and service requesters) to understand (1) each value of the weight representing the defined importance level and (2) the semantics of human reasoning due to a shared understanding of the weight definition. In order to give the correct semantics, two extension having been applied.

1. The sum of weight value does not need to equal 1. As discussed $\sum_{i=0}^{n} W_i = 1$ limits the freedom of defining the semantics to the weights. Meanwhile, enabling to continue correctly using the LSP method, the $0 < \mid W_i \mid < 1$ must be satisfied. Hence the values have to be normalised, which can be achieved automatically by applying $W_i' = \frac{W_i}{\sum_{i=0}^{n} \mid W_i \mid}$, where W_i is the defined weight by users with semantics. W_i' is the weight which will be used in the LSP aggregation function, where $\sum_{i=0}^{n} W_i' = 1$.

2. A semantics for weights has been defined to match the weights to the different importance levels, see Table 4.1.

4.3.3 Separating hard criteria and soft criteria

We separate criteria by defining all the hard criteria's weights as 1 or -1 and soft criteria's weights value from (-1, 0) or (0, 1). This modification can be seen as a further extensions of the weight semantics. Based on the separation, a particular conjunctive partial absorption function has been defined in Figure 4.2, where C_i^h and C_j^s represent the hard criteria and the soft criteria respectively. The first aggregation function (denoted with DAC symbol in the Figure 4.2) uses the automatically calculated r value from the orness degree calculation algorithm (see Section 4.3.4) and the respective formula is shown in equation 4.12.

$$E = (\sum_{i=1}^{n} \mid W_i' \mid E_i^r)^{\frac{1}{r}} \tag{4.12}$$

The second aggregation function denoted (with CA symbol in the Figure) uses the CA GCD operator which has been introduced in the original conjunction partial absorption function in [Duj] to aggregate scores for hard criteria with the overall soft criteria score. HE_i is the "Hard Criteria Evaluation result" and SE is the "Soft Criteria Aggregation Evaluation result". Behaviors of the conjunctive partial absorption function are such that the global selection value of a service (denoted by GP) is an error when any of the hard criteria are not satisfied, in which case the service is discarded. On the other hand, a service that satisfies all hard criteria will be evaluated to a non-zero value, from which the degree of satisfaction of the soft criteria can raise or reduce the final global aggregation selection results. In another words, hard criteria can be seen as same as soft criteria when the service satisfies all the hard criteria. Therefore, in equation 4.13, the hard criteria and soft criteria are equally weighted (0.5) for the final aggregation step.

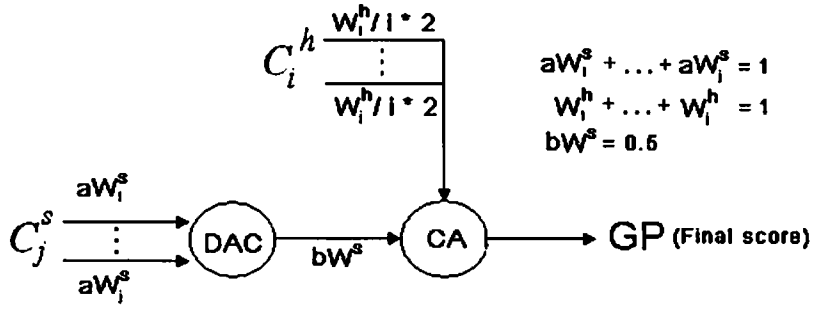

Figure 4.2: The conjunctive partial absorption function

$$E = ((\sum_{i=1}^{n}(\mid W_i^h \mid)/i2)HE_i^r)) + 0.5SE^r)^{\frac{1}{r}} \tag{4.13}$$

4.3.4 Automatic calculation of *Orness* degree

The last extension to the initial LSP function is the automatic Orness degree calculation function. The calculation function derives from equation 4.14, which was proposed by Fodor and Marichal independently to measure the orness of any mean operator $M(x)$ by studying the LSP degrees of conjunction and disjunction [FR94, Mar98], where $Max(x)$ is the pure conjunction (C) and $Min(x)$ is the pure disjunction (D). However, this definition bases on the definition of function $M(x)$.

$$orness_D(M(x)) = \frac{\int_{[0,1]^n} M(x)dx - \int_{[0,1]^n} Min(x)dx}{\int_{[0,1]^n} Max(x)dx - \int_{[0,1]^n} Min(x)dx} \tag{4.14}$$

Meanwhile, Yager [Yag88] introduced an aggregation technique based on the Ordered Weighted Averaging (OWA) operators as defined in Definition 4.1.

Definition 4.1 *OWA operator*

Let $W = (\omega_1, \omega_2, ...\omega_n)$ *with* $\sum_{i=1}^{n} \omega_i = 1$

Let $A = (a_1, a_2, ..., a_n)$ *and* $B = (b_1 + b_2 + ... + b_n)$ *be sets where* b_i *is the i-th largest element of* A.

An OWA operator of dimension n is a mapping

$F : R^n \rightarrow R$

such that

$F(a_1, a_2, ..., a_n) = \sum_{i=1}^{n} \omega_i b_i$

For example, $\omega = (0.4, 0.3, 0.2, 0.1)$, then $F(0.7, 1, 0.3, 0.6) = (0.4)(1)+(0.3)(0.7)+(0.2)(0.6) + (0.1)(0.3) = 0.76$.

A fundamental aspect of this operator is the re-ordering step. An aggregate a_i is not associated with a particular weight w_i but rather a weight is associated with a particular ordered position of the aggregate [CF97]. In [Yag88], the *Orness* measure of any kind of OWA operator is defined as follows:

$$orness(OWA) = \frac{1}{n-1} \sum_{j=1}^{n} ((n-j)w_j) \tag{4.15}$$

Theorem 4.1 has been proved in [SM03]. It gives a tool for linking the LSP method and OWA operators by bridging the gap between the *Orness* definition of equation 4.14 and the contributions of equation 4.15.

Theorem 4.1 *If the problem can be expressed as OWA problem, then:*

$Orness_D(M(x)) = Orness(OWA)$

On the one hand, we argued that the original LSP method does not provide an automatic mechanism to select a suitable r. On the other hand, [SM03] proves that if the selection problem can be transformed into OWA operators, then we

can use the OWA *Orness* definition to automatically calculate the LSP *Orness* degree. Therefore, we need to investigate whether the selection problem can be transformed into an OWA problem.

In general an OWA problem is characterised by being expressible by two sets: one with weights and the one with evaluation values. The latter comes in two forms: A and B, where A is ordered in the same order as the weights and B is in order of the value of the elements. Considering the services selection problem, we naturally have a set of weights, but we have multiple sets of values because we have one set of criteria evaluation values per service. Therefore, we can compute a set of values that contains the average score for each criteria across all services to be evaluated to look for an reasonable overall aggregator which should differentiate the services.

Assume that we are considering m services and n evaluation criteria. $W = \{w_1, w_2, ...w_n\}$ is a set of weights and $\{v_{11}, v_{21}, ...v_{n1}\}...\{v_{1m}, v_{2m}, ...v_{nm}\}$ are the evaluation results of each mapped criteria for the different services, with each set presenting results for the n criteria of one service.

Then:

$$V = \{\frac{\sum_{t=1}^{m} v_{1i}}{m}, \frac{\sum_{t=1}^{m} v_{2i}}{m}, ... \frac{\sum_{t=1}^{m} v_{ni}}{m}\} \tag{4.16}$$

is the set of average evaluation scores for our n numbers of criteria.

Using W and V (and its average version V') as the respective sets, we have recast the service selection problem into an OWA problem, and hence can use equation 4.15 to compute the *Orness* degree, which in turn provides the value r to be used in the global aggregation function 4.12 for the CAD operator. We use two examples seen in table 4.2 and 4.3 to show the computed *Orness* degree and provide insight about the operator expected for adequate aggregation.

Criteria	C1	C2	C3
Weight	0.7	0.2	0.1
Service1 evaluation scores	0.3	0.2	0.1
Service2 evaluation scores	0.1	0.7	0.9

Table 4.2: Simultaneity example

Criteria	C1	C2	C3
Weight	0.7	0.2	0.1
Service1 evaluation scores	0.9	0.2	0.1
Service2 evaluation scores	0.7	0.7	0.9

Table 4.3: Replaceability example

Recall that when calculating overall scores, we do not want the evaluation results of less import criteria to outweigh the important ones; we referred to this as simultaneity earlier on. In Example 1 we can see that both services have low evaluation values for the most important criterion C1, which means that to ensure simultaneity we require a conjunction LSP operator. By using formula 4.16 and 4.15, we calculate the *Orness* degree as 0.2 which maps to the strong quasi conjunction operator.

Example 2 shows a typical case of replaceability, where a good matching on an important criterion makes a service preferable. In the example, we can see that both services have higher evaluation results for the most import criterion. Recall that for replaceability we would expect a disjunction operator. Again, by applying our formula we can compute the *Orness* degree (this time as 0.75) and we find that it maps to the medium quasi disjunction operator.

4.4 Applying TLE method to the Notification Service Selection Scenarios

In this section, we apply the proposed TLE service selection method to the notification service selection scenarios discussed in Chapter 1.

Reminder that different notification services are required to be send the meeting notification messages to all participants after a meeting has been scheduled for an emergency situation. The notification service evaluation criteria are <Covered location, Message type, Price, Response Time, Privacy, Prefer contact method> with respected emergency weights set for the selection criteria <1, 1, -0.3, -0.9, 0.5, 0.3> and types of <string set, string set, numerical, numerical, numerical, string set>. Currently there are three different notification services available (see Table 4.4).

	Location w=1	Message Type w=1	Price w=-0.3	Response Time w=-0.9	Privacy w=0.5	Prefer Contacting w=0.3
notificationservice1	Europe	Mobile phone, Email	0.4 (UK 0.1)	0.4 s	0.6	*no* | *no* | *yes*
notificationservice2	Only UK	Email, Instant Messenger	0.1	0.8 s	0.9	*no* | *yes* | *yes*
notificationservice3	Europe	PDA, Instant Messenger	0.5(UK 0.1)	0.6 s	0.9	*yes* | *yes* | *no*

Table 4.4: Selection criteria and notification service NFPs meta data

	Location w=1	Message Type w=1	Price w=-0.3	Time w=-0.9	Privacy w=0.5	PCM w=0.3
S_1	1	0.5	0.25	1	0	0
S_2	0	0	1	0	1	0
S_3	1	0.5	0	0	1	1

Table 4.5: Notification service evaluation results calculated by Type-based evaluation functions for Bob

The participants may have the following context data as shown in Chapter 1:

1. Bob is on holiday with only his mobile phone and PDA. The holiday location is in Spain. On holidays, Bob prefers to be contacted by PDA rather than mobile phone message.

2. Alice has switched off her mobile phone to conserve battery power, but she is online using IM (IM is also her preferred method of contact).

3. John is working in his UK office with access to a wide variety of communication devices (mobile phone, email, IM and PDA). However, he prefers to be contacted by email message.

We take Bob as an example to illustrate the service ranking process.

For Bob, we invoke type-based evaluation functions to evaluate the different services with respect to the service selection criteria. For example, the "message type" criterion is string set type and we apply the string set matching evaluation function. For example for service 1:

$$E_{messagetype} = \frac{1(mobilephone)+0(PDA)}{2} = 0.5$$

The other individual criteria can be evaluated by the same process for different services using respective functions, and the final result is shown in Table 4.5.

Step 1 According to the values of weights, "covered location" and "message type" are hard criteria, the rest are soft criteria. Based on the OWA

orness degree calculation function, we obtain the soft criteria aggregation degree.

- the overall scores for "Price", "Time", "Privacy" and "Prefer Contact Method" on 3 services are ($0.25+1+0=1.25$, $1+0+0.75=1.75$, $0+1+\frac{1}{3}=1\frac{1}{3}$, $0+0+1=1$). Therefore, the criteria are reordered as new set of criteria C' "Time", "Privacy", "Price" and "Prefer Contact Method".
- we re-scale the weights to enable adding up to 1. The re-scaled weights for the C' are 0.45, 0.25, 0.15 and 0.15.
- we apply the OWA orness calculation function to obtain the orness degree $\frac{3*0.45+2*0.25+1*0.15}{3} \approx 0.667$.
- we map the degree value to the r value which is used as aggregation power as DAC operator and the calculated closest r value to the degree of 0.667 is 3.318 for 4 criteria.

Step 2 Application of the corresponding aggregation function to the soft criteria evaluation results:

$E_{S1}^{sagg} = (0.15*0.25^{3.318}+0.45*1^{3.318}+0.25*0^{3.318}+0.15*0^{3.318})^{1/3.318} \approx 0.787$

$E_{S2}^{sagg} = (0.15*1^{3.318}+0.45*0^{3.318}+0.25*1^{3.318}+0.15*0^{3.318})^{1/3.318} \approx 0.759$

$E_{S3}^{sagg} = (0.15*0^{3.318}+0.45*0.5^{3.318}+0.25*1^{3.318}+0.15*1^{3.318})^{1/3.318} \approx 0.784$

Step 3 Aggregation of soft criteria and hard criteria evaluation result by using the CA GCD operator. Because there are two hard criteria, the weights for the each hard criteria and aggregated soft criteria evaluation result are $0.5/2 = 0.25$ and 0.5.

$E_{S1}^{hagg} = (0.25*1^{-0.732}+0.25*0.5^{-0.732}+0.5*0.787^{-0.732})^{1/-0.732} \approx 0.728$

$E_{S2}^{hagg} = (0.25*0^{-0.732}+0.25*0^{-0.732}+0.5*0.759^{-0.732})^{1/-0.732} \approx dev/0\ error$

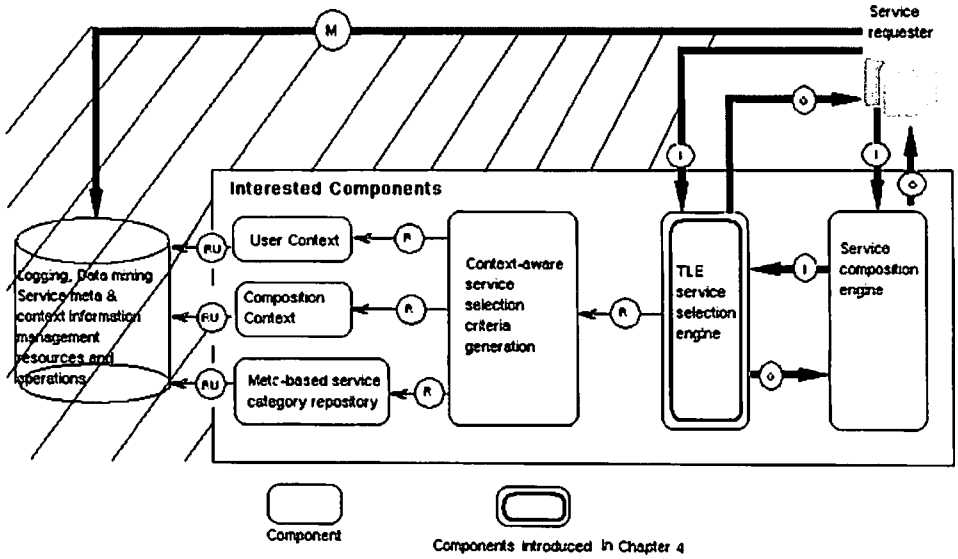

Figure 4.3: Components are discussed in Chapter 4

$$E_{S3}^{hagg} = (0.25 * 1^{-0.732} + 0.25 * 0.5^{-0.732} + 0.5 * 0.784^{-0.732})^{1/-0.732} \approx 0.727$$

Based on the final aggregation results, we find that S_1 is the most suitable service to be selected for sending the notification message for Bob, following by S_3 with S_2 not being suitable as it is not cover the area that Bob is in.

4.5 Summary

In this Chapter we have introduced the Type-based LSP extension (TLE) service selection method. Refer to the proposed overall service selection process, TLE is implemented as service selection engine (it is a Web service) in Figure 4.3.

It has three important parts: (1) Type-based criteria evaluation function;(2) Extended LSP criteria aggregation function and (3) automatic *Orness* degree calculation function.

The type-based criteria evaluation function solves the issue of identifying suitable evaluation functions at runtime. Currently, we defined 4 evaluation functions for numerical, boolean, string set and distance types. For different dynamic service selection frameworks, further types may be required, together with respective evaluation functions.

The extended LSP aggregation function enhances the LSP advantages to deal with dynamic changing aggregation requirements. The extension adds the semantics to the weight values in order to clearly express the criteria preferences and their relations.

Finally, we introduced the *Orness* degree calculation function which uses weight values. The *Orness* degree finally is applied to the extended LSP aggregation function for ranking the services.

In terms of context-aware automated service provision, the TLE service selection method has advantages over existing methods discussed in Chapter 2 with regarding to automatic evaluation and aggregation in dynamic environments. Further evaluation of the TLE method will be discussed in Chapter 6.

Chapter 5

Backwards Composition Context based Service Selection in Composition Scenarios

In Chapter 4, we introduced the automatic TLE service selection method for single service selection scenarios. However, for composition scenarios, additional information is required, we refer to this as composition context. *Composition context* generally refers to information that tells about services' collaboration histories, policies and financial terms. In this chapter, we introduce *composition context* and the composition mechanism to use it for service selection based on the proposed TLE service selection method. The assumptions in this chapter are that the composition activities are well defined as a workflow template and that the context information can be dynamically stored and retrieved by the selection method. We have briefly discussed two service composition scenarios (organising a meeting and planning a trip) in Chapter 1. In this chapter, we will discuss them in detail, and use them to explain the Backwards Composition Context based Service Selection (BCCbSS) approach.

In the composition workflow, each activity can be conducted by invoking a

type of service. Each type of service may have different services supported by different providers. Optimally composing different types of services to obtain the most suitable composite service is a crucial challenge. Meanwhile, the service composition process can be seen as performing multiple service selection steps following the workflow structure.

We firstly study the two typical scenarios in more detail to discuss what kind of information affects the service composition and should be considered as part of the composition context in Section 5.1. We then make precise what we mean by composition context in Section 5.2. The composition context based service composition approach is illustrated in Section 5.3. The research contributions will be discussed in Section 5.4. A comparison to related work on service composition will be discussed in Section 5.5. The chapter concludes with an example and summary in Section 5.6 and 5.7. Some parts of this chapter have been published in [YRMT08] and [YRM09b].

5.1 Composition Scenarios

Let us further consider the composition scenarios of organising a meeting and planning a trip which presents real world composition examples. The main purpose of studying these examples is to understand in more detail what affects service selection in composition and hence how to define and organise composition context.

5.1.1 Organising a meeting

Recall that a meeting is required to be held for discussing a plan to deal with an emergency. Organising a meeting needs a series of tasks that can be performed by services and integrated by a meeting organiser as a workflow template the

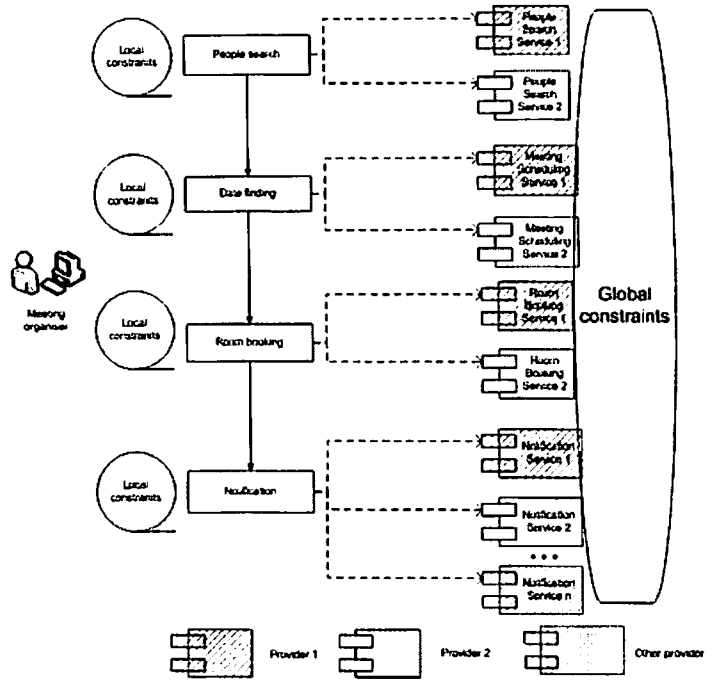

Figure 5.1: Workflow of organizing a meeting

same as the one shown in Figure 5.1.

Task 1: Searching for people who are required to attend the meeting; there are two relevant services offered by two different providers (P1 and P2).

Task 2: Finding a date that most of the invited people are available at according to their calendar context data. There are two relevant services offered by two different providers (P1 and P2) available.

Task 3: Booking a suitable room for the meeting. Again, there are two relevant services offered by two different providers (P1 and P2).

Task 4. Sending invitation messages including the meeting details. There are many relevant services offered by different providers which include P1 and P2.

In this scenario, it is further assumed that the target participants are in the organisation. By analysing this scenario, we conclude some information which needs to be considered for the service selection and composition.

Local constraints

The meeting organiser invokes the workflow template. For task 1, a people search service is required and two services are discovered. As discussed in Chapter 1, local constraints are a set of requirements for the service's NFPs. "Local" means the requirements are individual considerations for each type of the services. These requirements could either be hard criteria or soft criteria as we defined in Chapter 4. Each requirement has a weight for prioritizing. For instance, the organiser has preferences stating that accuracy of the search is more important than speed, only participants within the organisation are acceptable. If we only consider the local constraints, the selection problem will be as same as single service selection but with multiple steps for selecting 4 services independently.

Invocation error context

Supposing a service from provider 1 has been selected based on the local constraints (e.g. accuracy and speed). Invoking the selected service produces an invocation error, and hence, the other service from provider 2 has to be used instead. If this error can be saved as context information and retrieved for future service composition, it can reduce the composition time and increase the compositions reliability.

Coordination context

We assume the "people search service" from provider 2 has been selected for Task 1. For the current Task 2 of date finding, all target participants use online Google calendars. The only local preference is the availability rate (that is how many of the targeted people are available for the suggested date). There are two scheduling services which use people's calendar's URL address as input and return the most suitable date for all involved people as output. One scheduling service from provider 1 only has the ability to check Google and MSN online calendar systems and provides a 0.9 availability rate (that is 9 people out of

10 are available on the suggested date). The other service from provider 2 has the ability to check all kinds of currently existing online calendar systems and supports a 0.7 availability rate. When only considering the local constraints on their own, it is easy to see that the service from provider 1 is the better one because its availability rate is better. However, it is known that the selected people search service has more coordination failures with the scheduling service of provider 1 than that offered by provider 2, a fact is learned from historical composition records available in the context. Taking this into account, it is more difficult to decide which service is better. The difficulty is to balance local constraints and global constraints (in this case, coordination constraints). On the one side, the local constraints ensure that the service satisfies the user's preferences. On the other hand, the global constraints can reduce chances of the composition failure.

Provider distance

We suppose the date finding service from provider 2 has been selected, thus the previous two tasks are performed by the same provider. For Task 3, the aim is to book a room with some normal equipment. The booking service takes date and facility requirements as input and produces the place address and room information as output. There are two booking services available. One service supports to book the rooms cheaply but with some normal equipment. The other service supports to book all available business meeting rooms with a higher price but with some normal and special meeting equipment. Therefore, both services satisfy the task requirement. In this situation, a service can be selected at random as there is no other user's preference. However, the history of coordination activities show that services from the same provider have more efficient coordination rate, then service from provider 2 may be better because it has provided the previous two services.

5.1.2 Planning a trip

The previous case study shows services selection and composition inside an organisation. In a more open world, there may be additional economic and collaboration policy considerations due to business competition. Recall the scenario of planning a trip requires three tasks of booking transport, purchasing travel insurance and booking a hotel. Purchasing insurance and booking a hotel are two independent tasks but they both rely on the transport date and time.

Allowance policy

We assume that airline service A has been selected for the booking transport task based on the user local context constraints of covered locations, faster service response and cheaper service fee. For the hotel booking task, some business corporation policies are also applicable. It is useful to make selections depending on the previous selected services' corporate policy. The following two are examples.

- It is not allowed to continue invoking the service more than 10 times in one workflow.
- Provider A does not allow its services to be used by services owned by provider B.

Cost policy

In the commercial market, cost is an important factor to be considered from the global point of view. The aim of using the cost policy context is to find the cheapest composite service, which does not necessarily mean every single service is the cheapest one because the coordination among different providers has different prices. This is the main difference between local cost constraint

and global cost constraint. For example, one insurance service A takes £10 for travelling with airline service but the airline gives £8 discount for working with insurance service A. The other insurance service takes £5 for travelling with the same airline service A. Therefore, the first service is likely selected from global point of view.

Composition time

Time is also a crucial factor for the business. The time can be considered global and local. The local view focuses on the individual service response time. In contrast, the global view concentrates on the overall composite service response time. For example, one insurance service takes more time to complete with airline A. However, the other one will take less time to complete with the same airline. As results, it is more reasonable to select the second one, if we considering the composition time constraint.

5.2 Classifications of Composition Context

The composition context focuses on the context information which arises from execution services a part of a composition. Based on the case studies and scenario analysis, we defined eight composition context constraints in the three categories: execution context, coordination context and composition policy context (see Table 5.1). Again, we do not claim the elements defined here are completed, but they have shown sufficient for our scenarios. Furthermore, this chapter is about the concept of composition context and it's usages rather than about all possibilities of the concept. If further elements are added to the composition context, they should not affect the feasibility achieved by applying our selection mechanism. However, they should improve the service selection quality or accuracy.

Analysing these 3 groups of context, we find that composition context can also

Composition Context		Type	Description
Execution context	Execution error rate	Numerical	The workflow execution engine detected an exception, when the server is invoking.
	Connection error rate	Numerical	Two services worked fine independently, however an error appeared during their integration.
Coordination Context	Provider distance	Boolean	Do services belong to the same provider?
	Coordination time	Numerical	The coordination time for communications between services.
	Physical distance	Boolean	Are services deployed at the same physical location?
Composition policy context	Special Cost	Numerical	This captures special deals between services.
	Allowance	Boolean	This captures which services can or which cannot be used together, or "Is composition allowed?"
	Times limitation	Numerical	This specifies the allowed times of continuous invoking.

Table 5.1: Composition Context classifications

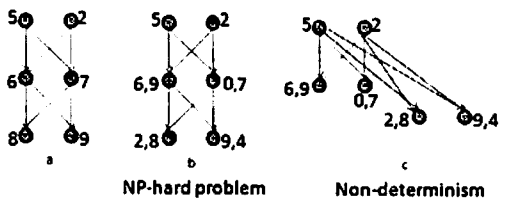

Figure 5.2: Composition complexity analysis

be separated as dynamic context and static context. The dynamic context (e.g. coordination time) means the context changes very frequently time to time. Thus, dynamic context needs to be detected, calculated and stored at runtime e.g. execution context of execution error rate and connecting error rate. Static context refers to the composition information which does not change frequently, e.g. provider distance and special cost.

5.3 BCCbSS: Backwards Composition Context based Service Selection

5.3.1 Challenges of dynamic service selection for composition

Our goal is to efficiently conduct the dynamic context-aware service selection for composition which rises some fundamental challenges over and above the dynamic service selection discussed in Chapter 4. These occur as we also need to consider the composition context. The challenges are:

The balance between globally optimal and locally optimal solutions is an important issue to achieve. Unlike general global optimization problems, the context-aware service selection requires to consider both the user's local context constraints as well as global composition context

constraints. Therefore, we need to select the service which satisfies both sides of the optimization, as otherwise, the selection result may not be adequate.

The balance between complexity/efficiency and adequacy. Composition context information is dynamic and has to satisfy multiple constraints, making it a difficult global optimization problem. Normally, global optimization problems have fixed values; for example, when computing cheapest path, each node has a fixed value in the graph, allowing for the use of greedy algorithms to find the cheapest solution (see Figure 5.2.a). However, in our case, the service composition context data is different to different services. For example, the composition price is £0.5 between Service1 and Service2, but is £0.0 (free) between Service1 and Service3 (see Figure 5.2.b). Additionally, considering the multiple constraint dimensions, the global optimization becomes an NP hard problem. Therefore, always finding the best possible solution is quite expensive and time consuming which devalues contributions to the dynamic context-aware service selection.

Control flow structure affects the global optimal strategy . The other unusual global optimal issue is that the composition specification is a workflow with control flow structures such as sequence, parallel (and) and split (or) operators. Especially, the split control flow has significant impact on the possible workflow paths, that is only when a service is selected and invoked, the workflow path can be determined based on the output data assessment as Figure 5.2.c shows. However, runtime output data cannot be predicated in advance, and only become known when the service is invoked. As a result, the global optimization result has potential to be completely wrong unless split we considered appropriately.

The investigations of current service composition approaches (see Section 5.5), showed that there is no suitable one for these needs. Therefore, we investigate a new composition approach to achieve our goal.

5.3.2 BCCbSS approach overview

We developed a backwards composition context based service selection algorithm (BCCbSS). Recall that we start with workflow templates and need to select actual service only, rather than creating the workflow. The approach starts with the selection of the first activity, with the steps being applied while more activities are encountered in the workflow. We also treat parallel control flow and split control flow differently in the algorithm. The parallel tasks are selected one by one from left to right, however, it will not be considered as sequence control flow in the algorithm. The split tasks will force to invoke the service before select service for next task. The detail description is follows.

Step 1 Searching and returning all candidate services from the registry for the *current task* in the composition workflow.

Step 2 Invoking the ranking function F (the function will be discussed in Section 5.3.3) to give an evaluation value to each candidate service by considering the user context information and composition context between available services for *current task* and services selected for the *previous task* and *next task*(Note the previous task or next task means the previous or next control flow task rather than directly previous or next task in the real time parallel task). If there is no *previous* selected service and no *next* service (this only happens to the start task), the ranking function bases the decision on the user context and *execution error rate* composition context. If the *next* workflow operator is "split",

then invoke the highest ranked service for *current task* and go back to Step 1. Otherwise directly go to Step 3.

Step 3 based on the highest ranked service for *current task*, re-rank the services for *previous task* if it exist and has not been invoked, then invoke the re-ranked highest service for *previous task*. If an error occurs when the service is invoked, record the error information in the composition context store. Repeat Step 3 with the next best service. Otherwise go to Step 4.

Step 4 If the current task is the last task in the composition workflow, then invoke the current selected service, log the execution details to the context store and finish. Otherwise, go to step 5 with next required task.

Step 5 Move to next activity and return to Step 1.

The algorithm implementation is represented in Algorithm 1 and 2.

5.3.3 BCCbSS optimization using TLE service selection method

Service selection based on both the user's context (local optimal criteria) and composition context is a vital step of BCCbSS. In Chapter 4, we introduced the Type-based LSP Extension service selection method to solve the problem of optimal service selection based on user context.

For adding the composition context into service selection, we define a global ranking function F as follows:

Let

$E_{1.1}$ = Soft local optimization criteria considering user requirements and derived from user context;

Algorithm 1 BackwardsServiceSelection(WF)

WF is the workflow template with a set of tasks $(T_1, T_2, ...T_i...T_n)$ and a set of control flow $(CF_1, CF_2, ...CF_i...CF_n)$.
let $s = null$, $s_p = null$ and $s_n = null$ be selected service for current task, selected service for previous task of T_{i-1} and selected service for next task of T_{i+1}.
let $S_p = empty$ be the set of candidate services for previous task of T_{i-1}.
for T_i=1 to T_i=n **do**
  let $S_c = empty$ be a set initial set of services.
  S_c = DiscoverServices(T_i)
  if $S_c! = empty$ **then**
    s=SelectingCompositionService(T_i, s_p, s_n, S_c)
    if CF_i=="split" **then**
      Invoke s
      while s is not successfully invoked **do**
        $S_c = S_c$ - s
        s=SelectingCompositionService(T_i, s_p, s_n, S_c)
        Invoke s
      end while
    else
      s_p=GetInvokedService(T_{i-1})
      if $s_p! = empty$ **then**
        s_n=s, s_p=GetInvokedService(T_{i-2})
        S'_c=DiscoverServices(T_{i-1})
        s'=SelectingCompositionService(T_{i-1},s_p, s_n, S'_c)
        Invoke s'
        while s' is not successfully invoked **do**
          $S_c = S_c$ - s'
          s'=SelectingCompositionService(T_i, s_p, s_n, S_c)
          Invoke s'
        end while
      end if
    end if
  else
    Failure to exist the algorithm
  end if
end for

Algorithm 2 SelectingCompositionService(T,s_p, s_n, S_c)

s_p and s_n are selected service for previous task of T and selected service for next task of T. S_c is a set of candidate service for T.
if S_c != empty **then**
  let R be a set of ranking scores for S_c and r_i is the individual score.
  for each service s_i in the S_c **do**
    Invoke function r_i=F(s_i, s_p, s_n)
    reorder S_c sorted descending by R values
  end for
  s=the first service in S_c
else
  s=empty
end if
return s

$E_{1.2}$ = Soft global optimization criteria amended and derived from composition context related to the previous selected service (if the previous selected service does not exist, then $E_{1.2} = 0$) for the task in the workflow;

$E_{1.3}$ = Soft global optimization criteria amended and derived from composition context related to the next selected service (if the next selected service does not exist, then $E_{1.3} = 0$)for the task in the workflow; and r=1, $\mid W_1 \mid = \mid W_2 \mid = \mid W_3 \mid = 1/3$ in

$$E_1 = (\mid W_1 \mid E_{1.1}^r + \mid W_2 \mid E_{1.2}^r + \mid W_3 \mid E_{1.3}^r)^{1/r}, \tag{5.1}$$

E_1 aggregates all the soft optimization criteria.

E_2 = Hard optimization criteria (including both global and local context) represent all mandatory requirements which must be satisfied. Any hard criterion evaluating to 0 will lead to an aggregation result of 0. The soft criteria handle all other preferences. Finally,

$$F = (\mid W_1 \mid E_1^r + \mid W_2 \mid E_2^r)^{1/r}, \tag{5.2}$$

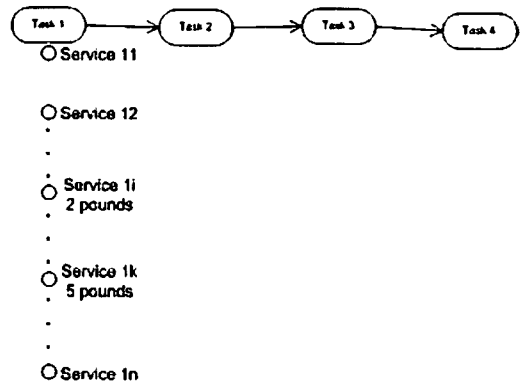

Figure 5.3: BCCbSS step 1

where r=-0.72, $\mid W_1 \mid = \mid W_2 \mid = 0.5$.

Here, r is assigned to the value of conjunctive partial absorption function which has been explained in Chapter 4. When we using the conjunctive partial absorption function, 0 score for hard criteria (E_2) delivers a overall 0 aggregation score for the service. If the service satisfies all the hard criteria (we have discussed in Chapter 4), then the hard criteria should be treated the same as the soft criteria. Therefore, we assign the equal weights (0.5) to soft and hard criteria.

5.3.4 An example and complexity analysis

To demonstrate the entire service composition process and complexity analysis, a simple composite price-based example is illustrated here. In this worked example, the user wants the cheapest composite service. The worst scenario is the tasks are sequentially ordered (all tasks require to be invoked finally). The workflow includes 4 tasks: T1, T2, T3 and T4. The service selection process is illustrated step by step below.

Step 1: n candidate services (S11, S12 ... S1n) are discovered for T1, containing S1i and S1k ($1 \leq i, k \leq n$) which have the cheapest cost of £2 and £5 (respectively see Figure 5.3). In this instance, there is no composition context

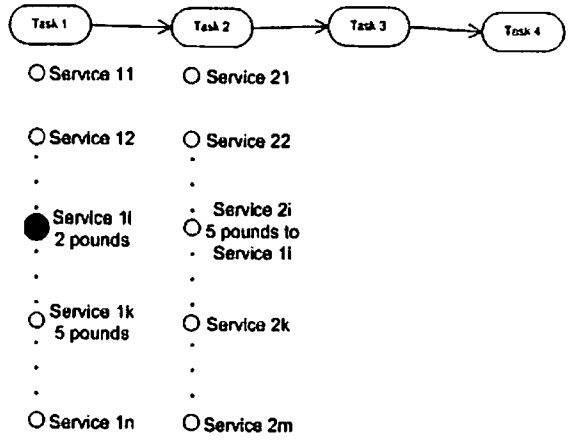

Figure 5.4: BCCbSS step 2

available, so S1i is selected as it is the cheapest. In this step, the complexity of the service selection process can be proved to be O(n) because it is a simple sort problem.

Step 2: m candidate services (S21, S22 ... S2m) are discovered for T2, containing S2i ($1 \leq i \leq m$). Since S1i has been selected for the first task in Step 1, we now only consider the composition context (special cost in this case), where S2i costs £5 if composed with S1i is the cheapest combination fee among the m candidate services. Therefore, the S2i is selected in this step (see Figure 5.4). The complexity of this step is still O(m) because it only considers the composition context with the fixed service S1i which is selected in previous step.

Step 3: Re-select the service for T1 in order to check if there is a better combination between the first two tasks with S2i selected. Based on the composition context with service S2i selected (there is no previous selected service for T1), S1k is re-selected for T1 and invoked in this step as S1k+S2i= £6 < S1i+S2i= £7 (see Figure 5.5). The complexity of the step becomes O(n) again as the next selected service is fixed.

Step 4: l candidate services (S31, S32, ... S3i, ... S3l) are discovered for

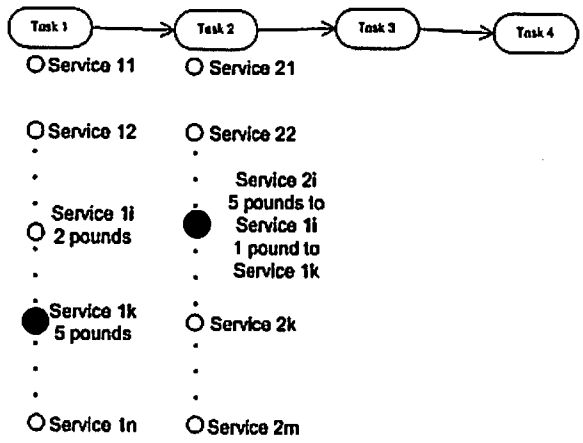

Figure 5.5: BCCbSS step 3

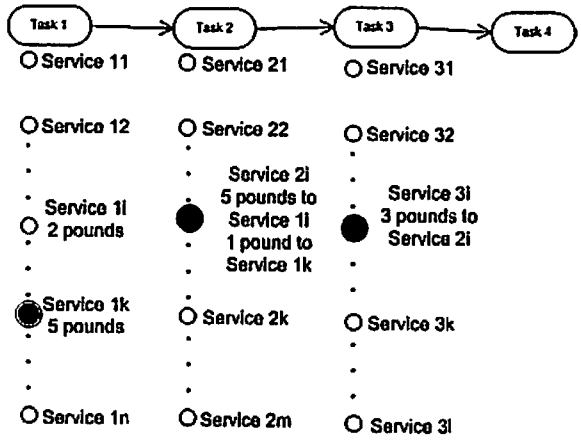

Figure 5.6: BCCbSS step 4

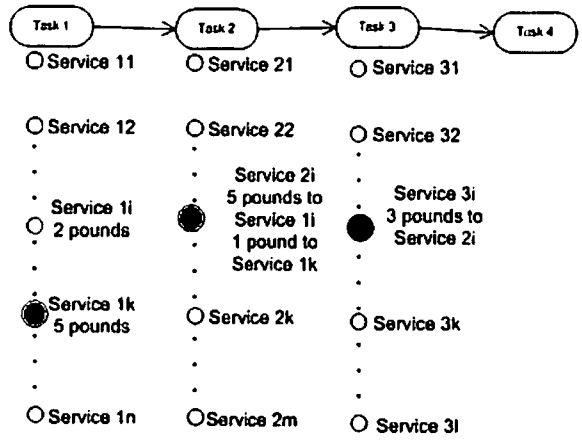

Figure 5.7: BCCbSS step 5

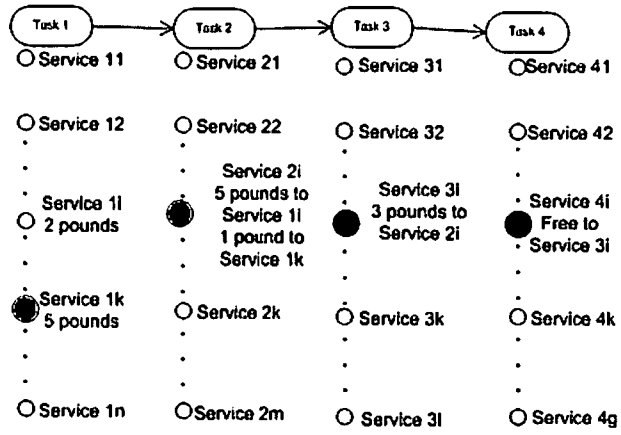

Figure 5.8: BCCbSS step 6

T3, where S3i costs £3 to be composed with S2i and presents the cheapest combination fee among the l candidate services. Therefore, S3i is selected as the cheapest composition service with S2i (see Figure 5.6). The complexity can again be shown to be O(l).

Step 5: the selection is restarted for choosing a service for T2 based on previous invoked service and next selected service. The S2i remains as the selected one and is invoked (see Figure 5.7). Although the previous and next selected services need to be considered, the complexity is O(m) as previous and next services are fixed.

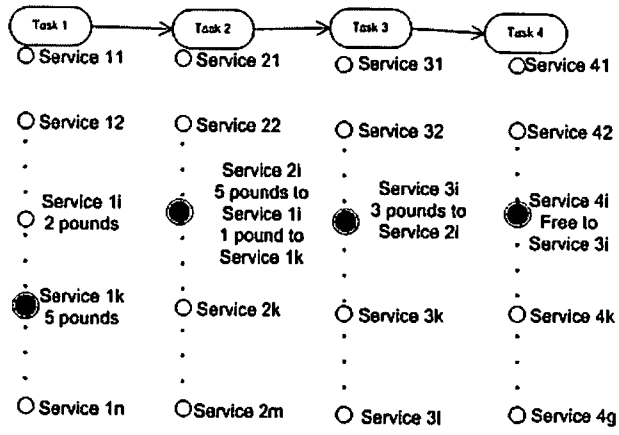

Figure 5.9: BCCbSS step 7

Step 6: g candidate services (S41, S42, ... S4i, ... S4g) are discovered for T4, where S4i is free if composed with S3i, which are selected for the previous task. Thus, S4i is selected (see Figure 5.8). The complexity is O(g).

Step 7: the selection is restarted for choosing a service for T3 based on the previously invoked services and next selected service. Again, S3i is the best one and is invoked. The complexity is O (l). Finally the last select service S4i is invoked (see Figure 5.9). As result, we can see that every step takes time linear in the amount of available services. If there are x tasks in the workflow, then the whole composition process has 2x-1 selection steps and the complexity has an upper bound of (2x-1)O(y), where y is the different numbers of candidate services for the task with the longest choice of services.

5.4 Contributions of BCCbSS

The overall services composition approach can be considered as a sequence of service selections and the length of that sequence depends on the number of tasks in the workflow template. The data of the composition context are gained through the addition of facts from observation and the history. The BCCbSS approach has following advanced characteristics:

1. The approach performs the selection and invocation step by step. Some research work [CPEV05, ZBN+05, YL05] suggests completing a service composition by selecting all the services for the whole workflow template. However, this is not an efficient way, if there are many tasks involved in the workflow and many candidate services are available for each of them. Taking the organising meeting scenario as an example, if there are 4 services for each of the 4 tasks, then the selection method has to compare the entirely 256 different composition solutions for identifying the adequate service composition choice. With more service available, the state explosion problem will affect the approach efficiency and scalability. The step by step strategy can essentially avoid such a problem because each step has only to consider a small number of the services, which is the number of the available services for the task, previous selected or invoked service and next selected service (only in re-selection process).

2. The approach can guide the selection method to make a choice based on existing knowledge of the composition context. In the organising meeting scenario, when the people search service from Provider 1 has been selected for task 1, the rest of the selection tasks should only consider the composition context related to the selected service because other people search services' composition context is no longer useful.

3. The approach considers not only local constraints (user context) but also the composition context as inputs for the selection method. The local constraints specify the user's preferences for the individual service, e.g. quality, execution duration and prices. Since it is difficult for user to judge the global view of the composition service, the composition context should be automatically applied. The composition context or global constraints consider the interaction and composition properties among the selected services and available services for the current task.

4. The approach is a run-time approach. The user's preferences may fre-

quently change according to his/her current status, as will the service's NFPs and composition context. These dynamic features require a run-time composition approach rather than a design-time composition approach. The run-time composition approach can make sure that the composite solution is the most suitable one for the current user's status and composition context.

5. The approach is fault tolerant. When a selected service can not perform adequately in a certain step, the approach allows for the next best service to substitute the current selected one in order to complete the composition task. Fault tolerance is very important for a run-time approach and real service composition scenarios. It increases the likelihood of successful completion of the workflow. In the example of planning a trip, when the air ticket has been booked, the composition requires the insurance purchase to be successful, because the previous step is costly and irreversible.

5.5 Related Work

Two kinds of service selection approaches have been developed in order to address the Web service composition problem, which are found on local optimal selection and global optimal selection.

Local optimization based service selection refers to selection methods which only take certain selection constrains related to the current activity in the workflow into account without specifying and considering the constraints implied by the workflow context and the consequences that the choice will have on later activities. For example, a policy based BPEL workflow Web service selection method is presented in [KHC+05]. It extends BPEL for run-time adaptation of service by adding the policy reference to each node. The policy

documents provide the local optimization rules which are independent from each other. The service selection process is applied at each node separately. A similar approach was also presented in the earlier e-Flow project [CIJ+00]. The biggest advantage of the local optimization methods is efficiency in selection time - the worst case can be solved in polynomial time. However, they not necessarily select the optimal or even close to optimal service in the global composition context.

Global optimization based service selection, on the other hand, considers the global selection constraints to select a group of services rather than one service for a node in the composition workflow. The key assumption of this strategy is that all suitable services for each node have already been discovered and are inside the global optimization search space. [CPEV05, ZBN+05] are two example approaches. By studying these approaches, we find they surely narrow the disadvantages pointed out for local optimization. However, they introduce their own problems.

- Low scalability: In general, multi-QoS constrained service selection with optimization is an NP-complete problem [YL05], which reduces scalability of the methods.
- Lack of fault tolerance: Global optimization methods return a set of combined services as the final solution package. However, if one service is not available or throws an exception at run-time, then the whole solution package fails.
- Low flexibility: Global optimization methods need to know all constraints at design time. However, some selection constraints are only known when certain data is produced at run-time. For example, considering a conditional choice in a composition, the complete global constraints are available only after the condition is evaluated.

- Lack of reflection of local constraints that are important to reflect the user's context.

In contrast, the BCCbSS approach does not need to predict all the global constraints in advance. It makes the selection decisions on an activity by activity bases on the currently existing local and global composition context. The composition context is growing as we proceed through the activities. Based on these context constraints, we select the best service for the next activity according to run-time knowledge. As we continue to select services, the composition context grows allowing for more fine-grained selection.

5.6 A Worked Example

Let us reconsider the planning a trip scenario. Assume that a person is located in the UK and plans a trip to China. The user context information is

- Activity: travelling from UK to China;
- Payment method: Visa or MasterCard.

For the first task of booking transport, only local constraints derived from user context are considered. The service is selected by using the context-aware TLE service selection method as illustrated in the single service selection scenarios. We assume flight booking service B is selected.

Once the first task is completed by invoking service B, the composition context related to service B is considered for selecting the the service. We understand that the purchasing insurance task and reserving hotel task are parallel activities, which means the composition context between those two types of services does not need to be considered. We can use the purchasing insurance services

Context	PIS1	PIS2	PIS3	SB
Execution error W=-0.2	4	3	0	0
connection error W=-0.3	0 with	1 with	2 with	0
Provider W=0.2	0 with	0 with	1 with	1
Coordination time (seconds) W=-0.1	2.2 with	0.5 with	1.2 with	0
Physical location W=0.1	0 with	0 with	1 with	1
cost W=-0.1	£4 with	£10 with	£2 with £5 with others	0

Table 5.2: Composition context of the example conjunction with service B (SB)

PIS = Purchasing Insurance Service

selection process to show how context composition context is used for service composition.

Supposing we find 3 candidate services for purchasing insurance, Table 5.2 lists the context information for the 3 available insurance purchasing services. As shown, the payment method is a hard criterion while the others are soft criteria.

Ranking results for the services are gained by applying the type-based criteria evaluation functions. The results for the given context constraints are presented in Table 5.3.

Then, we aggregate all context constraints to obtain the final score for each service. To achieve this goal, we use equation 5.1 and 5.2 to aggregate the soft and hard criteria constraints. In this example, there is no soft local optimization criterion, so we ignore $E_{1.1}$ in equation 5.1. r = 2.384 is selected for the formula 5.1 by analysis and automatic calculation (see Chapter 4). Soft composition criteria aggregation results are: PIS1 = 0.64, PIS2 = 0.47 and PIS3 = 0.97. As result, equation 5.2 produces the final selection results by

Context	TBS1	TBS2	TBS3
Execution error W=-0.2	0	0.25	1
connection error W=-0.3	1	0.5	1
Provider W=0.2	0	0	1
Coordination time (seconds) W=-0.1	0	1	0.59
Physical location W=0.1	0	0	1
cost W=-0.1	0.75	0	1
Payment method W=1	0	1	0.5

Table 5.3: The composition service selection results
TBS=Type-based evaluation scores for service

considering the hard financial support criterion: PIS1= 0, PIS2 = 0.65, PIS3 = 0.67. PIS3 is the best suitable service after calculation.

We use the same process to select a hotel booking service at the same time.

Finally, we backward refine the first service selection because it has not been invoked. If the flight booking service B is still a best one for considering the composition context with PIS3 and the other selected hotel booking service, then service B is invoked. Since there is no other task is required, then the rest of selected services are invoked. If any of the selected services has failed to complete the task, the error will be logged into the composition context store and the second highest scoring services will be tried instead.

5.7 Summary

Selecting the most suitable services to complete a complex composite service is an important research topic. In terms of context-awareness, we believe that

the global composition context should be considered together with the local non-functional properties as a crucial factor for the service selection in the composition process.

In this chapter, we introduced the concept of composition context which is divided into 3 classes and 8 specific elements identified by analysing the workflow service selection scenarios of organising a meeting and planning a trip. The 8 elements include runtime dynamic information, history mining information and static description information.

Based on the composition context, we presented a novel Backward Composition Context based Service Selection (BCCbSS) approach to compose different type of services. The BCCbSS composition process fully considers both user's local context and composition context factors by adopting the TLE service selection method with a backward step refining mechanism.

Above research work is developed as composition context model and service composition engine shows in Figure 5.10. At the end of this chapter, we have discussed all the relevant components in our proposed process architecture.

Comparing the approach to the context-aware service composition requirements and other composition approaches, our approach has several advantages:

The approach is a fault tolerant step by step process. The method scales well for large workflow as well as large numbers of services, as the ranking considers only services for the current task and has access for the wider workflow condition through the composition context. The selected services are dynamic bound to and invoked at run-time rather than statically bound at design time.

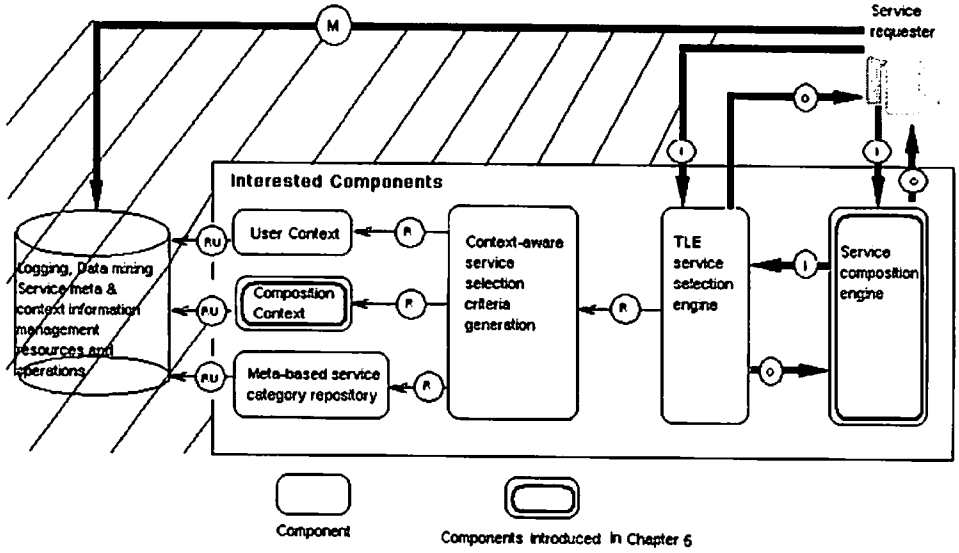

Figure 5.10: Components are discussed in Chapter 5

Chapter 6

Implementation and Evaluation

We introduced the context-aware TLE service selection method to tackle both single service and composition service scenarios in the previous two chapters by considering user context and composition context respectively. In this chapter, we discuss the method evaluation through a web-based simulation system and focus on two major aspects: *adequacy* and *scalability*.

Adequacy means that the highest scoring service should be the most suitable one. For single service selection, the suitability is computed according to NFPs preferences and the user run-time context. For service composition, the suitability additionally takes the composition context into account.

Scalability refers to the evaluation questions of how the method copes with an increasing number of services, criteria (context-aware), workflow steps and their combination.

Adequacy and scalability will be more clearly defined in Section 6.2 and 6.3. In addition to the case studies discussed in Chapter 1, we use more complex sim-

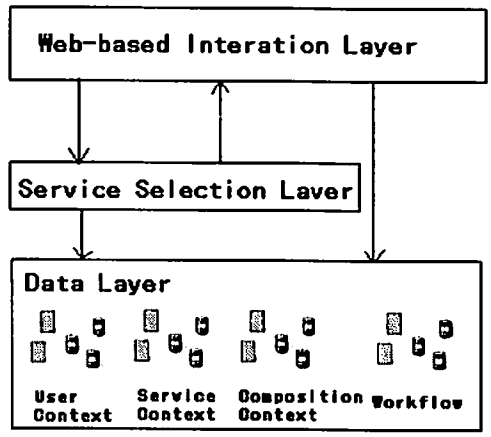

Figure 6.1: Implementation layers

ulation test cases in order to obtain stronger evaluation results. The reminder of the chapter is structured as follows:

Section 1 introduces the implementation of the TLE simulating system. Section 2 concentrates on the evaluation of adequacy. Section 3 evaluates the scalability of the selection method for both single service selection and composition scenarios. Section 4 draws the conclusion of the chapter.

6.1 Implementation

To implement and evaluate the proposed context-aware TLE service selection method, we designed a three-layer simulation system (see Figure 6.1).

Interaction layer provides a web-based user interface to allow service providers to register their services, workflow specifications and to allow service users to register their context to the system and send service selection requests.

Service selection layer essentially consists of a Web service called relevance engine that implements the TLE service selection method.

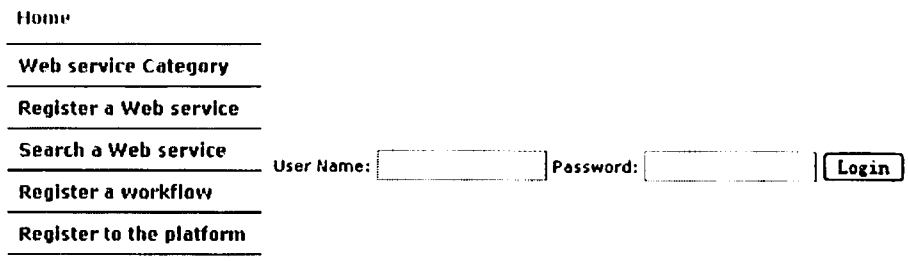

Figure 6.2: Work bench index page

Data layer includes the registered services information (NFPs' meta data) and user's context information as discussed in Chapter 3. Additionally, it contains the composition context and workflow specifications.

In order to simplify the system implementation and focus on the research issues of interest. The implementation only covers the important parts in terms of service selection and composition.

Figure 6.2 shows the snapshot of the web-based UI. The user needs to register their context to the system first in order to search a service (or a composite service based on the selected workflow). The page also allows to browse for service category information, registered services, workflows and registered users.

After a user successfully logs into the system, he/she can go to the "search a service" page (see Figure 6.3.a) which enables the user to enter the service keywords describing functionality or select an existing workflow specification file. After submission, the service selection or composition results will be displayed as shown in Figure 6.3.b. The services will be sorted by the ranking scores and the most suitable one will be at the top.

The service providers can register their services through the web-based user interface by specifying the service name, provider name, URL of meta information and service category with detailed descriptions. If there is no existing suitable category for the service, the provider can also create a new category

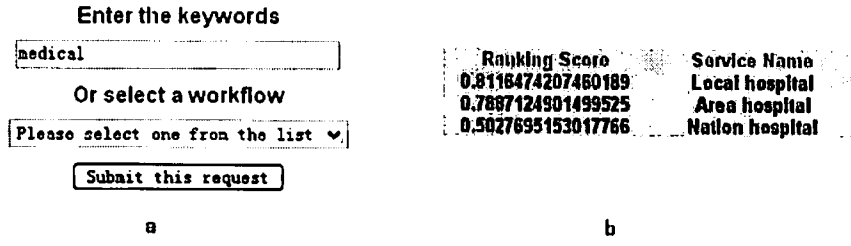

Figure 6.3: Service ranking results pages

Function	Input	Output	Description
FindRelevance()	keywords : String	Service[ ] : Service	The operation takes user's functional requirements keywords as input and matches the correct service for categories returning all functional suitable services as output.
ServiceRanking()	Service[ ]: Service, userId : String	Service[ ] : Service (sorted by ranking scores)	The operation takes *FindRelevance()* operation's output as one input parameter and user's id as another parameter to return a ranked list of services.

Table 6.1: Relevance engine

as shown in Figure 6.4. For that, the service provider needs to specify the category name, description and importantly define which meta data is related to this category by ticking the appropriate check boxes.

The relevance engine is implemented in Java using Axis2 technologies on the Apache Tomcat server platform. The service interface is defined in Table 6.1.

The data layer is simply implemented as a storage of OWL files which contain context information of users and NFPs' data of services (as we discussed in Chapter 3).

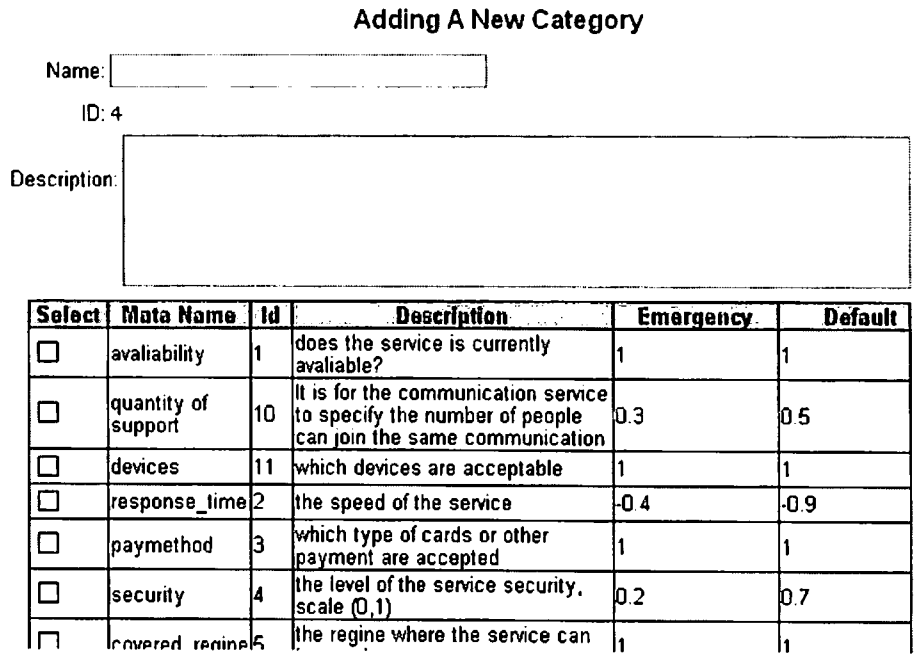

Figure 6.4: Create a new service category

6.2 Adequacy Evaluation

6.2.1 Adequacy

It is desirable to show the adequacy of the proposed service selection method. However, we need to explain what is meant by adequacy in this context. As discussed informally in the beginning of the chapter, adequacy means that the most suitable service with respect to the user. However, *most suitable* is difficult to define in general because it depends on different views and concerns. In terms of context-aware service selection, the suitability is composed of three important factors: (1) suitability with respect to user context, (2) suitability with respect to NFP preferences and (3) suitability with respect to the aggregation of results.

1. Suitability with respect to user context is the basic level of adequacy, because the users should have the ability and convenience to use the

selected service. Therefore, the selected service should definitely match the users context information.

2. Suitability with respect to NFP preferences is the satisfaction level adequacy. When services reach the basic level, the service is a better choice if it also satisfies the service selection preferences, such as security, cost, time and cooperation properties. Normally, different NFPs have different important concerns in different situations. Therefore, the most suitable service requires to reflect the NFP preferences.

3. Suitability with respect to the aggregation of results is the aggregation adequacy by considering selection requirements between replaceability and simultaneity as discussed before. This cannot be easily detected by users.

The adequate selection result satisfies the maximum suitability described above. An inadequate service does not hold the context-aware suitability.

Context information, NFP preferences and aggregation functions are different for different selection cases, therefore, these three levels of suitability need to be evaluated through case by case analysis in order to find out which selection choice is better in the particular scenario, and this decision is not an automatic or completely objective one.

6.2.2 Adequacy measurement

In order to evaluate the adequacy of our proposed service selection method, we apply our TLE service selection method together with three selection methods mentioned in Chapter 4 to the scenarios of selection notification service and medical support service. Recall that these three methods are the most used methods by other service selection approaches:

(1) Weighted sums:

$$E = \sum_{i=1}^{n} W_i E_i \tag{6.1}$$

(2) Weighted Conjunctions:

$$E = \bigwedge_{i=1}^{n} W_i E_i \tag{6.2}$$

(3) Weighed Disjunctions:

$$E = \bigvee_{i=1}^{n} W_i E_i \tag{6.3}$$

There are other selection mechanisms (e.g. random selection method) that could be applied to the service selection issue. However, these methods are rarely used.

Notification service selection adequacy evaluation

The test cases are designed based on the notification service selection scenarios introduced in Chapter 1. A meeting organizer wants to send a meeting notification to the invited people (assuming the organisation is in the UK) using the most suitable notification services based on their different context and service selection criteria. The notification services' NFPs meta data is represented in Table 6.2. Recall that there are three different typical scenarios.

	Location Weight=1	Devices Weight=1	Price Weight=-0.3	Respond Time Weight=-0.9	Privacy Weight=0.5	Prefer Contacting Weight=0.3
notificationservice1	Europe	Mobile phone, Email	0.4 (UK 0.1)	0.4 s	0.6	*no* | *no* | *yes*
notificationservice2	UK	Email, IM	0.1	0.8 s	0.9	*no* | *yes* | *yes*
notificationservice3	Europe	PDA, IM	0.5(UK 0.1)	0.6 s	0.9	*yes* | *yes* | *no*

Table 6.2: Selection criteria and service NFPs meta data
UK 0.1 means that in UK the service price is 0.1.

	Location	Devices	Price	Time	Privacy	PCW
S_1	CbC	CbC	0.25 (1)	1	0	0 | 0 | 1
S_2	CbC	CbC	1 (1)	0	1	0 | 1 | 1
S_3	CbC	CbC	0 (1)	0.5	1	1 | 1 | 0

Table 6.3: Evaluated scores for each service's NFPs by Type-based evaluation functions

1. Bob is on holiday with only his mobile phone and PDA. The holiday location is in Spain. On holidays, Bob prefers to be contacted by PDA rather than mobile phone message.

2. Alice has switched off her mobile phone to conserve battery power, but she is online using IM (IM is also her preferred method of contact).

3. John is working in his UK office with access to a wide variety of communication devices (mobile phone, email, IM and PDA). However, he prefers to be contacted by email message.

We firstly obtained the possible evaluated individual criteria evaluation results for all services in different scenarios as shown in Table 6.3. In the table, CbC means that the evaluation scores may different for different scenarios. Finally, the selection results for all four service selection methods are shown in Table 6.4 (our method's selection results are gained by using the simulation system and the other methods' selection results are gained by manual calculation). Now we will exam the adequacy of our method and compare it to other three methods.

Scenario 1. Four types of context information need to be considered in this case. Firstly, Bob is out of UK, which requires a service that can serve in user current location. Secondly, Bob has mobile phone and PDA for getting the messages, hence a service which does not support sending mobile phone and PDA messages is not suitable. Thirdly, Bob prefers to be contacted by PDA message on holiday, which means a service

	Sum	Conjunction	Disjunction	TLE
Scenario 1				
S_1	2.475	0	**1**	**0.728**
S_2	0.8	0	0.5	discard
S_3	**2.75**	0	**1**	0.727
Scenario 2				
S_1	2.2	0	**1**	discard
S_2	3.1	0	**1**	0.911
S_3	**3.55**	**0.167**	**1**	**0.923**
Scenario 3				
S_1	**3.0**	0	**1**	**0.758**
S_2	2.6	0	**1**	0.749
S_3	2.75	0	**1**	0.727

Table 6.4: Test results for emergency situation

which can send PDA messages is more desirable from Bob's point of view. According to the context information, S_2 should be penalized because it only works in the UK and can not send any mobile phone and PDA messages. S_1 and S_3 are suitable services based on context. From the NFP preferences side, price and participant prefer method are less important than time and privacy properties (see Table 6.2). It is difficult to judge which service is better at the moment because two services both have one higher score on less important properties and one higher score on important properties. Therefore, we need to consider aggregation level suitability to distinguish which service is better. From Table 6.3, we can see that S_1 gets 0 (Bob prefer PDA message, but S_1 only can send Mobile phone message) for prefer method and S_3 gets 0 for price. Since these two preferences are less important ones, the aggregation should make these 0 scores affect the final selection decision less and make the higher scores for the more important preferences substitute the lower scores. Through the analysis, this scenario is a typically replaceability aggregation case. The most important preference is time in this case and S_1 has the higher score. Thus, S_1 should be the most suitable service. Looking at the selection results table, the conjunction method penalized

all services because no service satisfies all the NFP preferences, while the disjunction method can not justify which service is better between S_1 and S_3. In contrast, summary and TLE methods both selected the suitable services, but only TLE selected the most suitable services and discarded the unsuitable service.

Scenario 2. Participant Alice has a very different context. She is in the UK and can only be contacted by IM message. For context-aware suitability, S_1 should be filtered out because it does not support IM message. From the preference perspective, both S_2 and S_3 have the same price in UK (so they both score 1 for the price) and satisfy the prefer mathod preference. In Bob's case, we already discussed that the replaceability should be applied, which allows higher scores for important preferences replace the low scores for less important preferences due to the 0 score values. For Alice, we only need to compare the two important factors of time and privacy which should not be overshadowed by higher scores for less important price and prefer method preferences. Because S_3 got 0.5 and 1 for the time property and privacy property and S_2 got 0 and 1 for the time and privacy properties, the best suitable service should be S_3 because it has highest score for the other important criterion of privacy. The selection results show that exception the disjunction method, all methods produced the adequate selection. However, only the conjunction and TLE method disqualified the unsuitable service.

Scenario 3. Participant John is in the UK and all four types of devices are available. Consequently, all three services are suitable from a context point of view. Concentrating on time and privacy, it is difficult to justify which service is better than others because they all have some good or bad side and we can say all the services are suitable. However, time is more important than privacy between them, then S_1 is the most suitable one for this case because only S_1 got 1 for time criterion. This time, only

Services	Availability	Response Time	Location	Support Level
Normal weights	1	0.6	0.9	0.3
Emergency weights	1	0.5	0.2	0.9
S_1	yes	2 minutes	27.01, 56.37	minor
S_2	yes	3 minutes	28.35, 57.29	minor, severe

Table 6.5: Two available medical support services

TLE and summary methods choose the adequate service.

Medical support service selection adequacy evaluation

Recall that there are three typical scenarios for medical service selection.

1. The injury has been reported close to service 1 and it is a minor injury.
2. The injury has been reported close to service 2 and it is a minor injury.
3. The injury has been reported close to service 1 but it is a severe injury (severe injuries raise an emergency status).

Two medical support services are available with 4 selection criteria as shown in Table 6.5. We gained the selection results for the 4 service selection methods and they are shown in Table 6.6.

Scenario 1. The person only gets a minor injury and is close to S_1. Therefore, from the context suitability aspect, S_1 is the most suitable one. Moreover, the preferences show that location and response time are more important than service level in the normal situation, which means that S_1 is much better than S_2. For aggregation level, S_1 satisfies all criteria and S_2 only matches the unimportant service level criterion. As a result, the high score for an unimportant criterion will not strongly affect the

	Sum	Conjunction	Disjunction	TLE
Scenario 1				
S_1	**2.8**	**0.3**	**1**	1
S_2	1.3	0	**1**	0.1
Scenario 2				
S_1	**1.9**	0	**1**	0.4807
S_2	2.2	0	**1**	**0.6714**
Scenario 3				
S_1	**2.5**	0	**1**	0.87
S_2	1.3	0	**1**	**0.9086**

Table 6.6: Medical support service selection results

final selection scores. Thus, S_1 satisfies all three levels of suitability and it is the most suitable one. In this scenario, all service selection methods select the right service but the disjunction method cannot separate the two services.

Scenario 2. This case is very similar to the pervious case only that the injury is close to S_2 rather than S_1. Therefore, S_2 should be the most suitable based on the same analysis as before. From the selection results, we found that only TLE produces the adequate answer.

Scenario 3. This case has a very important difference from the previous two cases as the person has a severe injury (emergency situation). The emergency situation breaks down normal service selection preferences to consider the support level as the most important criterion. Therefore, it is easy to understand that S_2 should be the most suitable medical service and only the TLE method found the adequate service.

By analyzing the different service selection cases, our method always provided the expected selection results, and hence is adequate with respect to the adequacy definition. The weighted Sum method gave just about 50% adequate selections but without discarding unsuitable services. Conjunction method got two suitable selection result (around 33.3%) and the disjunction method only

manage to produce 1 adequate answer overall (around 16.7%). Therefore, we can conclude that our service selection method is adequate and performs better than other commonly used selection functions.

In the composition scenario, we take the composition context criteria into account as an extra factor and the selection scenarios become more complex. However, the fundamental selection calculation does not change and the complexity does not affect the adequacy of the method especially as the otherwise is a repeated individual service selection. Thus, we will not discuss the adequacy evaluation in more detail.

6.3 Scalability Evaluation

Scalability is an important issue for service selection methods. In order to accurately discuss the scalability, we constructed two and three test cases for single selection scenarios and composition scenarios.

6.3.1 Single selection mode scalability

There are two scenarios for scalability evaluation: increasing the number of services with fixed number of criteria and increasing the number of criteria with fixed number of services

Test case 1: Increasing number of services

The first test case focus on measuring the selection time with regard to an increasing number of services. We selected 10 different increasing number of services as $2, 2^2, 2^3, ..., 2^{10}$ with 6 criteria (6 is the maximum number of criteria in our example scenarios and hence can be seen as realistic).

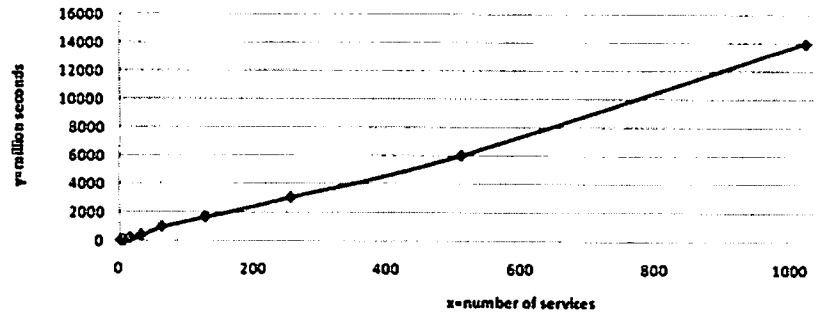

Figure 6.5: Evaluation results for single service selection test case 1

Figure 6.5 shows a linear increase in runtime, hinting at good scalability of our method with regard to the number of services with a fixed number of criteria.

Test cast 2: Increasing number of criteria

In order to evaluate the second scalability aspect of increasing the number of criteria, we tested 6, 12, 24, 48, 96 and 192 criteria with 4 competitive services for each case (4 services are common number of services for a service selection task in our example scenarios). The test results are presented in Figure 6.6. By analyzing the results, we see that again we have a linear increasing in runtime.

We can see that the proposed method has a good scalability against large number of criteria. In real world scenarios, the number of criteria is rarely more than a couple of hundreds, which is the reason why we tested up to 192 criteria.

From these 2 tests, the most important conclusion is our method scalability can be shown as a linear function (O(n) complexity) when increasing number

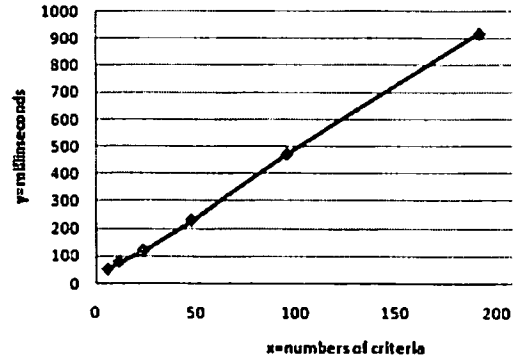

Figure 6.6: Evaluation results for single service selection test case 2

of criteria and services respectively.

6.3.2 Composition mode scalability

In this subsection, we evaluate the selection scalability for composition scenarios. As a starting point, we need to look at which types of test cases are required. Based on the evaluation results of single selection mode, we understand that the scalability of increasing number of services, criteria and their combinations. Without considering the composition context involvement, we can get Lemma 1, where n is the number of workflow activities, T_{nw} is the cost time of a completing services selection for a workflow with n composition steps and T^i_{nsc} is the time for suggesting a best service in a single selection scenarios on i^{th} activity and the worst case that all activities are required to be executed (without "split" control flow).

Lemma 1 $T_{nw} = \sum_{i=1}^{n} T^i_{nsc}$

Prove: *If a workflow which has n activities, then each step can be seen as a service selection issue individually without considering the composition context. Therefore, the overall time can be measured as*

$T_{nw} = T^1_{nsc} + T^2_{nsc} + ... + T^n_{nsc} = \sum_{i=1}^{n} T^i_{nsc}$.

In fact, we also did the test by using our simulation system. The test results show the same indication from the macroscopic view, although there are some small inaccuracies because of precision of measurements.

Now, the big evaluation question is the scalability with composition context. In order to give the answer, we designed 3 test scenarios to analysis.

1. Increasing the composition context with 4 services involved in each step of 3 composition steps.
2. Increasing the service numbers for each steps of 3 by considering all composition composition context.
3. Increasing the composition steps with 4 services and all composition context involved in each composition step.

Test scenario 1

We defined $6 + i$ selection criteria for this scenario where 6 are the single selection scenarios' criteria and i is the increasing number of the composition context criteria raising from 1 to 8 (we defined 8 composition context criteria in Chapter 5). From the test results data (see Figure 6.7), we can see that the time line is very close to a linear function.

Test scenario 2

In this scenario. we combine the maximum of 8 composition context criteria and increasing numbers of services from 2^1 to 2^8 for each of the 3 steps. Figure 6.8 represents that the method is close to a linear function too.

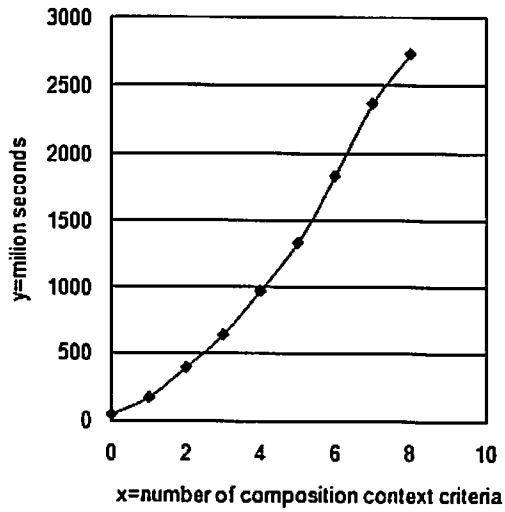

Figure 6.7: Evaluation results for composition selection test case 1

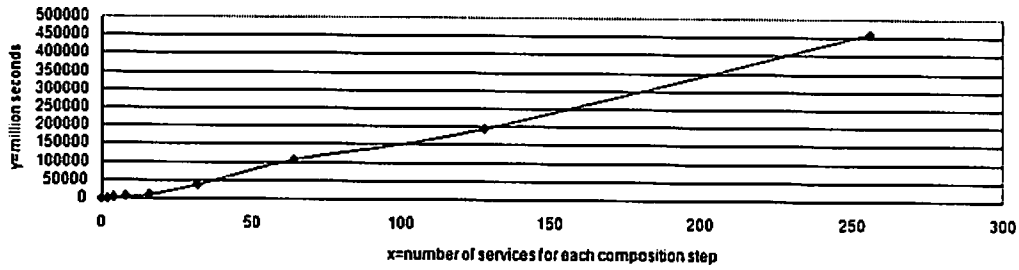

Figure 6.8: Evaluation results for composition selection test case 2

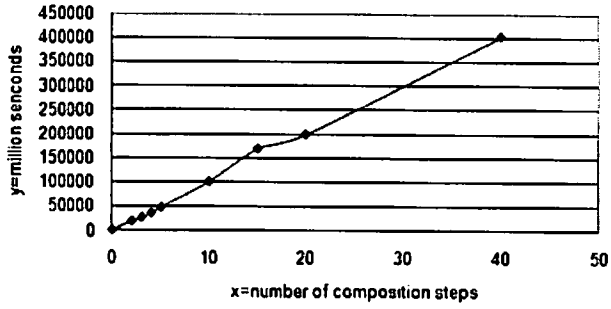

Figure 6.9: Evaluation results for composition selection test case 3

Test scenario 3

The last scenario is designed to test the scalability of number of composition steps. There are maximum 8 number of composition context criteria and 4 services for each step. The test results in Figure 6.9 shows that the method is also a linear function to the increasing numbers of composition activities.

From above three case analysis and the test figure, the proposed method always shows a linear function line which implies the method has a good scalability.

6.3.3 Discussion

From the scalability evaluation, we learn that the method shows a linear function for single service selection scenarios when increasing number of services and criteria. We expect that complexity is a quadratic when increasing both of them at the same time. However, the number of criteria is no more than a hundred in the realistic situation (normally it is less than 10 in our studied cases). Therefore, the evaluation results show that the method is very efficient to tackle single service selection.

For service composition scenario, the method efficiently works with under 10 depth of steps and works acceptably under 20 depth of steps. It also shows that more than 5 minutes cost in more than 40 steps situation. However, we

need to discuss two things here: (1) from business process point view, it is not a good workflow design for composing around 40 services to enable achieving a business goal. The basic reason is that one more transaction added into workflow, more than one risks will be added as well, such as failure error, over cost, transaction control, security issues to fail a successful composition. (2) The BCCbSS approach is a step by step mechanism means that the composition is divided into individual selection and invocation process. When a service is selected and invoked, the output comes out step by step for service user to use rather than waiting for the overall completion of the workflow.

6.4 Summary

In this chapter, we evaluated adequacy and scalability of the context-aware TLE service selection method by using the simulation system. We implemented the method inside a 3 layered architecture which includes *Interaction layer*, *Service selection layer* and *Data layer*.

In order to discuss the adequacy, we defined what it means in this context, adequate selection means that the highest ranked service is the most suitable service which satisfying suitability with respect to context, NFP preference and aggregation.

Our adequacy evaluation was based on 3 different notification service selection scenarios and 3 different medical support service selection scenarios. In order to compare our method to other service selection methods, we also evaluated the weighted sum, pure conjunction and pure disjunction methods. The test results show that all selection decision made by the context-aware automatic TLE service selection method are adequate and it outperforms other selection methods.

On the scalability side, the evaluation was divided into single selection and

composition by testing different numbers of services, criteria, composition steps and their combinations. The test results show that the context-aware TLE service selection method has a O(n) scalability, which is our desired research object.

Chapter 7

Conclusion

The overall aims of our research are (1) to generate context-aware service selection criteria based on user's context information; (2) to develop an efficient and automated service selection method and (3) to investigate a service composition algorithm which is composition context sensitive and able to use the proposed selection method.

7.1 Research Contributions

We divided our research aims into three aspects and each aspect comes with several subgoals and objectives. We will now discuss our research contributions against each of them.

7.1.1 Context-aware criteria generation

Reflecting a user's run-time context situation. Applying the user's current context information to the service ranking method is an important part of

our research and objectives. The research challenge is to ensure the context information gathered dynamically and automatically to affect service selection. As introduced in Chapter 3, our research solution is to add the user's context factors into the definition of the context-aware service selection criterion. Firstly, we modelled the user context information using 4 OWL/RDF models. Then, each criterion has an attribute of value which obtains the constraint data from the user's context through an OWL/RDF context query language. The attribute is finally used to evaluate the services' NFPs. Moreover, the user's emergency status also affects which weight set should be used for the criterion evaluation. Therefore, the ranking criteria are reflecting the user's current context and affect the service selection and composition results.

Criteria are meaningful and easily mapped to service NFPs. In Chapter 3, we detailed the service category structure which incudes service NFPs specification for different types of services. The selection criteria are initialized by the NFPs defined in the category and can be modified later by user context information. As a result, all dynamically generated criteria are derived from the service NFPs.

Overall, the TLE service selection method uses a user context information as ranking criteria and the criteria are context-aware and NFPs related. In this sense, our dynamic criteria generation process achieved the context-awareness aim.

7.1.2 Efficient and automatic service selection

Automatically evaluating services NFP against selection criteria. One of the major purpose of investigating the service selection method is to assist people in getting the best service when choosing services. People may not be enable to make the correct decision because the selection constraints are complex. Therefore, avoiding people's interaction and automatically ranking

services is very useful. Moreover, if this is part of a large system, users should not be concerned with what is eventually an implementation issue. However, it is difficult to evaluate different types of criteria because each criterion might be expressed in a different way and requires a specific evaluation function. To solve the difficulty, the type-based evaluation functions have been developed in our research. The functions are applied to the correct observed criteria by mapping the type of the functions to the type of the criteria. The solution has been successfully used in the ranking method with 4 different types of evaluation functions.

Different criteria evaluation results are adequately and automatically aggregated. This is the second difficulty when selecting the most suitable service. We applied the Logic Scoring Preference (LSP) and Ordered Weighted (OWA) methods to select the most suitable service. The LSP supports a set of selectable aggregation functions and OWA supports the calculation method to automatically select the adequate aggregation function. As results, our proposed method is automatic by using context-aware service selection criteria.

The TLE service selection method has a good scalability. We have tested the scalability of the proposed TLE method. The test results show that the context-aware automated service selection method provides good scalability in terms of increasing numbers of services, criteria and composition context.

7.1.3 Composition context sensitive service composition

Both user context and composition context are considered. In the service composition problem, the service selection for each required service in the composition flow needs to consider both local and global context constraints. The local context means the user's preferences/context and the global context means the composition aspects of the optimization context and service collaboration records. To enable the combination of the local context with global

context, we defined the concept of composition context and populated it with eight criteria by studying our service composition scenarios. The combination reuses the proposed service selection method by applying the LSP aggregation functions to calculate and combine local and global evaluation results. The global context knowledge grows during the composition process by applying the Backward Composition Context based Service Selection (BCCbSS) algorithm.

The composition mechanism has a low complexity. In Chapter 5, we compared our composition mechanism with other existing service composition strategies. We concluded that our composition mechanism together with the service selection method is of low complexity, allowing runtime decisions and additionally providing good fault tolerance.

7.2 Future Research Directions

The context-aware TLE service selection method and BCCbSS composition mechanism support a platform for experimenting with context-aware SOA applications. Although we have successfully achieved our research aims, there are some remaining and arising research issues and directions in context-aware SOA research.

Context reasoning for setting individual criteria weight. We presented a technique to allow user's context information to change the weight settings. We divided the weights into three groups of default weights, emergency weights and user's custom weights. We reasoned based on the user's context information to understand which group of the weights should be applied. However, the weights are pre-defined by service category creator, service provider or users during service selecting time. We can see that the weights are statically defined as a group and used according to user's context. In future work, it is very useful

to develop a more advanced reasoning process to enable dynamically catching individual criteria weights at runtime from the user's context information. In other words, the criteria weights are not pre-defined, but defined according to user's run-time context.

Extending the type-based evaluation function to enable using ontology information. So far, we defined 4 types of evaluation functions: numerical type, boolean type, string set type and distance type. These evaluation functions are enough to solve our specified service selection scenarios. However, it will be very useful and interesting to add more types into the evaluation systems to adapt to other domains of service selection. In particular, to add an ontology based type for evaluating more complex service NFPs.

Enabling the BCCbSS mechanism to be applied to current workflow execution engine. We developed the BCCbSS mechanism to consider composition context information for service runtime composition. At the current stage, we only managed to simulate the mechanism without really executing the workflow specification. There are two major future steps for completing the mechanism. One is to dynamically transform our abstract workflow templates to the workflow specifications, such as BPEL. The main difference between the abstract workflow template and executable workflow is that the former only specifies the workflow structure and required services without binding to actually services and their invocation endpoints; the latter does both the service specification and service binding. The abstract workflow templates are not executable before runtime binding is completed. It is required to transform to the executable workflow specifications at runtime during the service selection and binding process. The other approach is to embed the service selection and binding activities into the workflow execution engine. The engine modification should allow the engine to have breaking and continuing points when selecting and binding to the most suitable services.

7.3 Concluding Remarks

Since SOA was firstly introduced nearly a decade ago, runtime service selection and service composition became a crucial research area for developing more flexible, more open and business process oriented system. Thus, a wide range of service selection methods and composition frameworks have been proposed to deal with variety of these problems. In runtime service selection domain, however, there is no real investigation on where the selection requirements come from, what information affects service selection and how to using the information for service selection. This is the initial goal of our research to introduce context information for runtime service selection.

However, our proposed solution is constructed based on some important assumptions of advanced technologies, such as the general service concept. There are different kinds of SOA implementations, and not every service is described and published using OWL standards as we required. Similarly, composite services are not always modelled using an abstract business process specification language.

Currently, work fulfilling these assumptions is in research or promotion stages and has not really been fully migrated into industry practice. However, they are not far a way from realisng this. First of all, SaaS (Software as a Service), HaaS (Hardware as a Service) and IaaS (Infrastructure as a Service) have already been proposed and prototyped by Google, IBM and other leading IT organisations in the form of Cloud Computing, which makes the service concept more general than just Web service. Secondly, Researchers have developed tools (e.g. Protege) for easily editing and accessing OWL descriptions, which means service provider can produce the OWL efficiently. The only issue is when industry will accept this technology. Finally, abstract workflow in the form the specification language has already been introduced. Abstract BPEL specification and other research communities have proposed further suggestions

in this area. Therefore, we believe these technologies will be very realistic in near future.

We hope that the context-aware automatic service selection approach presented in this thesis will contribute to the development and usage of SOA in dynamic collaborative systems in the future.

Appendix A

An Example of OWL-S Profile for A Notification Service

```
<?xml version="1.0" encoding="ISO-8859-1" ?> <!DOCTYPE uridef[
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns"> <!ENTITY
rdfs "http://www.w3.org/2000/01/rdf-schema"> <!ENTITY owl
"http://www.w3.org/2002/07/owl"> <!ENTITY xsd
"http://www.w3.org/2001/XMLSchema"> <!ENTITY service
"http://www.daml.org/services/owl-s/1.1/Service.owl"> <!ENTITY
process "http://www.daml.org/services/owl-s/1.1/Process.owl">
<!ENTITY profile
"http://www.daml.org/services/owl-s/1.1/Profile.owl"> <!ENTITY actor
"http://www.daml.org/services/owl-s/1.1/ActorDefault.owl"> <!ENTITY
concept "http://localhost:8080/TestServiceConcept.owl"> ]> <rdf:RDF
    xmlns:rdf= "&rdf;#"
    xmlns:rdfs= "&rdfs;#"
    xmlns:owl= "&owl;#"
    xmlns:xsd= "&xsd;"
    xmlns:service= "&service;#"
    xmlns:process= "&process;#"
    xmlns:profile= "&profile;#"
    xmlns:actor= "&actor;#"
    xml:base= "http://www.cs.cmu.edu/naveen/profile.owl"
```

```
>
<owl:Ontology about="">
    <owl:versionInfo>
        $Id: OWLSServiceProfileEmitter.java,v 1.1 extension with NFPs
    </owl:versionInfo>
    <rdfs:comment>
        Add Comment
    </rdfs:comment>
    <owl:imports rdf:resource="&service;" />
    <owl:imports rdf:resource="&process;" />
    <owl:imports rdf:resource="&profile;" />
    <owl:imports rdf:resource="&actor;" />
    <owl:imports rdf:resource="&concept;" />
  <owl:ObjectProperty rdf:ID="hasNFP">
    <rdfs:domain rdf:resource="#Parameter"/>
    <rdfs:range rdf:resource="#NFP"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="hasNFPName">
    <rdfs:domain rdf:resource="#NFP"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNFPType">
    <rdfs:domain rdf:resource="#NFP"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNFPValue">
    <rdfs:domain rdf:resource="#NFP"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasParameter">
    <rdfs:domain rdf:resource="#Profile"/>
    <rdfs:range rdf:resource="#Parameter"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="hasParameterName">
    <rdfs:domain rdf:resource="#Parameter"/>
    <rdfs:range rdf:resource="&xsd;string"/>
```

```
</owl:DatatypeProperty>
<owl:Class rdf:ID="NFP"/>
</owl:Ontology>


<profile:Profile  rdf:ID="NotificationSerivceA">
    <profile:serviceName>
        Notification Service
    </profile:serviceName>
    <profile:textDescription>
        A service can send notification
    </profile:textDescription>
    <profile:contactInformation>
        <profile:Actor rdf:ID="CompanyA">
        </profile:Actor>
    </profile:contactInformation>
    <profile:hasInput>
        <process:Input rdf:ID="parameters_IN">
            <process:parameterType rdf:datatype="&xsd;#anyURI">
                &concept;#NotiRequest
            </process:parameterType>
        </process:Input>
    </profile:hasInput>
    <profile:hasOutput>
        <process:Output rdf:ID="parameters_OUT">
            <process:parameterType rdf:datatype="&xsd;#anyURI">
                &concept;#NotiResponse
            </process:parameterType>
        </process:Output>
    </profile:hasOutput>
  <profile:hasParameter rdf:resource="#parameter"/>
  <Parameter rdf:ID="parameter">
        <hasParameterName rdf:datatype="&xsd;string">NFPParameter</hasParameterName>
        <hasNFP rdf:resource="#typeOfMessage"/>
    </Parameter>
  <NFP rdf:ID="price">
```

```
            <hasNFPName rdf:datatype="&xsd;string">serviceFee</hasNFPName>
            <hasNFPType rdf:datatype="&xsd;string">numerical</hasNFPType>
            <hasNFPValue rdf:datatype="&xsd;string">0.5</hasNFPValue>
          </NFP>
        <NFP rdf:ID="typeOfMessage">
            <hasNFPName rdf:datatype="&xsd;string">messageType</hasNFPName>
            <hasNFPType rdf:datatype="&xsd;string">stringSet</hasNFPType>
            <hasNFPValue rdf:datatype="&xsd;string">IM</hasNFPValue>
          </NFP>
        <NFP rdf:ID="responseTime">
            <hasNFPName rdf:datatype="&xsd;string">responseTime</hasNFPName>
            <hasNFPType rdf:datatype="&xsd;string">numerical</hasNFPType>
            <hasNFPValue rdf:datatype="&xsd;string">200</hasNFPValue>
          </NFP>
          <NFP rdf:ID="coveredLocation">
            <hasNFPName rdf:datatype="&xsd;string">coveredLocation</hasNFPName>
            <hasNFPType rdf:datatype="&xsd;string">boolean</hasNFPType>
            <hasNFPValue rdf:datatype="&xsd;string">UK</hasNFPValue>
          </NFP>
          <NFP rdf:ID="privacy">
            <hasNFPName rdf:datatype="&xsd;string">privacy</hasNFPName>
            <hasNFPType rdf:datatype="&xsd;string">numerical</hasNFPType>
            <hasNFPValue rdf:datatype="&xsd;string">0.8</hasNFPValue>
          </NFP>
      </profile:Profile>
</rdf:RDF>
```

Appendix B

The Detailed Mapping Between OWL Diagram to UML Diagram

The OWL (Web Ontology Language) is built on top of RDF but adds more complex properties, characteristics and restrictions. Some features such as Object property, Transitive property, Symmetric Property, InverseFunctional Property , and advanced cardinality restriction are only support in OWL, which is an extension of RDF. A mapping between class diagrams and OWL can be defined as shown in Figure B.1 [YHHRM07].

UML concept	RDF/OWL concept [1]
class diagram	RDF/OWL Schema
object diagram	RDF/OWL document (instant of schema)
Basic data structure	built-in XML Schema datatypes
Class	<rdfs:Class> <owl:Class>
property (attribute)	<rdf:Property>
general association	<rdf:Property>
source and target of the association	<rdfs:domain>< rdfs:range>
specified association (transitive association, symmetric association, etc) between classes	<owl:ObjectProperty> <owl:TransitiveProperty> <owl:SymmetricProperty> <owl: InverseFunctionalProperty >
class Inheritance (generalization)	<rdfs:subClassOf>,
N/A [2]	< rdfs:subPropertyOf>
cardinality, OCL restriction (Size-related)	<owl:cardinality> <owl:maxCardinality> <owl:minCardinality >
instance x of class X	<x rdf:ID='X'>
links (instance of association) between objects	instance of <rdf:Property> etc.

Figure B.1: The detailed mapping between OWL diagram to UML diagram

Appendix C

An Example of User's OWL Context Information

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
    <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
    <!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >
]>


<rdf:RDF xmlns="http://www.cs.le.ac.uk/user.owl#"
     xml:base="http://www.cs.le.ac.uk/user.owl"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
     xmlns:swrl="http://www.w3.org/2003/11/swrl#"
```

```
 xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
 xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:owl="http://www.w3.org/2002/07/owl#">
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="Activity"/>
<Activity rdf:ID="Activity_Harry1">
    <hasStatu rdf:resource="#Status_Harry"/>
    <requireResources rdf:resource="#Resource_Harry"/>
    <hasCalendar rdf:resource="#Calendar_Harry1"/>
    <hasActivityLocation rdf:resource="#Location_Harry"/>
    <hasGoal rdf:resource="#Goal_Harry"/>
</Activity>
<owl:Class rdf:ID="Address"/>
<owl:Class rdf:ID="Calendar"/>
<Calendar rdf:ID="Calendar_Harry1">
    <start rdf:datatype="&xsd;string"></start>
    <end rdf:datatype="&xsd;string"
        >10:30-07-07-2009</end>
</Calendar>
<owl:Class rdf:ID="ContactInfo"/>
<ContactInfo rdf:ID="ContactInfo_Harry">
    <rdfs:comment rdf:datatype="&xsd;string"></rdfs:comment>
    <hasContactDetail rdf:resource="#ContactType_Harry"/>
    <hasPrefer rdf:resource="#preferContact_Harry"/>
</ContactInfo>
<owl:Class rdf:ID="ContactType"/>
<ContactType rdf:ID="ContactType_Harry">
    <hasContactWays rdf:resource="#MobilePhone"/>
    <hasContactWays rdf:resource="#OnlineRes_Harry"/>
</ContactType>
<owl:DatatypeProperty rdf:ID="Coordinate1">
    <rdfs:domain rdf:resource="#Location"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID="Coordinate2">
    <rdfs:domain rdf:resource="#Location"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="Device"/>
<owl:Class rdf:ID="ElectronicResource"/>
<ElectronicResource rdf:ID="ElectronicResource_Harry">
    <hasDevices rdf:resource="#MobilePhone"/>
</ElectronicResource>
<owl:DatatypeProperty rdf:ID="end">
    <rdfs:domain rdf:resource="#Calendar"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="FinanceResource"/>
<owl:Class rdf:ID="GeneralInfo"/>
<GeneralInfo rdf:ID="GeneralInfo_Harry">
    <hasUserID rdf:datatype="&xsd;string">1</hasUserID>
    <hasUserName rdf:datatype="&xsd;string">Harry Yu</hasUserName>
</GeneralInfo>
<owl:Class rdf:ID="Goal"/>
<Goal rdf:ID="Goal_Harry">
    <name rdf:datatype="&xsd;string">comunicated</name>
</Goal>
<Person rdf:ID="Harry">
    <hasResource rdf:resource="#Resource_Harry"/>
    <hasLocation rdf:resource="#Location_Harry"/>
    <hasProfile rdf:resource="#Profile_Harry"/>
    <involveIn rdf:resource="#Activity_Harry1"/>
</Person>
<owl:ObjectProperty rdf:ID="hasActivityLocation">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Activity"/>
    <rdfs:range rdf:resource="#Location"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasAddress">
    <rdfs:domain rdf:resource="#Profile"/>
```

```
    <rdfs:range rdf:resource="#Address"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasCalendar">
    <rdfs:domain rdf:resource="#Activity"/>
    <rdfs:range rdf:resource="#Calendar"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasContactDetail">
    <rdfs:domain rdf:resource="#ContactInfo"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasContactInfo">
    <rdfs:domain rdf:resource="#Profile"/>
    <rdfs:range rdf:resource="#ContactInfo"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasContactWays">
    <rdfs:domain rdf:resource="#ContactType"/>
    <rdfs:range>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Device"/>
                <owl:Class rdf:about="#OnlineResource"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasDevices">
    <rdfs:domain rdf:resource="#ElectronicResource"/>
    <rdfs:range rdf:resource="#Device"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasElectronicRes">
    <rdfs:domain rdf:resource="#Resource"/>
    <rdfs:range rdf:resource="#ElectronicResource"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasFinanceRes">
    <rdfs:domain rdf:resource="#Resource"/>
    <rdfs:range rdf:resource="#FinanceResource"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="hasGeneralInfo">
    <rdfs:domain rdf:resource="#Profile"/>
    <rdfs:range rdf:resource="#GeneralInfo"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasGoal">
    <rdfs:domain rdf:resource="#Activity"/>
    <rdfs:range rdf:resource="#Goal"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasLanguage">
    <rdfs:domain rdf:resource="#Profile"/>
    <rdfs:range rdf:resource="#Language"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="hasLanguageLevel">
    <rdfs:domain rdf:resource="#Language"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasLanguageName">
    <rdfs:domain rdf:resource="#Language"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasLevel">
    <rdfs:domain rdf:resource="#Status"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasLocation">
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range rdf:resource="#Location"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasLocationAddress">
    <rdfs:domain rdf:resource="#Location"/>
    <rdfs:range rdf:resource="#Address"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasOnlineAccount">
    <rdfs:domain rdf:resource="#OnlineResource"/>
    <rdfs:range rdf:resource="#OnlineAccount"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasOnlineFinanceAccount">
    <rdfs:domain rdf:resource="#FinanceResource"/>
```

```
    <rdfs:range rdf:resource="#OnlineAccount"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasOnlineRes">
    <rdfs:domain rdf:resource="#Resource"/>
    <rdfs:range rdf:resource="#OnlineResource"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasOrganization">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Profile"/>
    <rdfs:range rdf:resource="#Organization"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPhysicalRes">
    <rdfs:domain rdf:resource="#Resource"/>
    <rdfs:range rdf:resource="#PhysicalResource"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPrefer">
    <rdfs:domain rdf:resource="#ContactInfo"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPreferWays">
    <rdfs:domain rdf:resource="#PreferContactWay"/>
    <rdfs:range>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Device"/>
                <owl:Class rdf:about="#OnlineResource"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasProfile">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range rdf:resource="#Profile"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasResource">
    <rdfs:domain rdf:resource="#Person"/>
```

```
    <rdfs:range rdf:resource="#Resource"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasStatu">
    <rdfs:domain rdf:resource="#Activity"/>
    <rdfs:range rdf:resource="#Status"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="hasUserID">
    <rdfs:domain rdf:resource="#GeneralInfo"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasUserName">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#GeneralInfo"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="involveIn">
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range rdf:resource="#Activity"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Language"/>
<Language rdf:ID="Language_Harry1">
    <hasLanguageName rdf:datatype="&xsd;string">English</hasLanguageName>
    <hasLanguageLevel rdf:datatype="&xsd;string">3</hasLanguageLevel>
</Language>
<Language rdf:ID="Langue_Harry2">
    <hasLanguageName rdf:datatype="&xsd;string"></hasLanguageName>
    <hasLanguageLevel rdf:datatype="&xsd;string"></hasLanguageLevel>
</Language>
<owl:Class rdf:ID="Location"/>
<Location rdf:ID="Location_Harry">
    <Coordinate2 rdf:datatype="&xsd;string">800</Coordinate2>
    <Coordinate1 rdf:datatype="&xsd;string">500</Coordinate1>
    <hasLocationAddress rdf:resource="#UK"/>
</Location>
<Device rdf:ID="MobilePhone"/>
<owl:DatatypeProperty rdf:ID="name">
```

```
    <rdfs:domain rdf:resource="#Goal"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Number">
    <rdfs:domain rdf:resource="#OnlineAccount"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="OnlineAccount"/>
<OnlineAccount rdf:ID="OnlineCommunication_Harry">
    <OnlineStatu rdf:datatype="&xsd;string"></OnlineStatu>
    <Type rdf:datatype="&xsd;string">Email</Type>
    <Number rdf:datatype="&xsd;string"></Number>
</OnlineAccount>
<OnlineResource rdf:ID="OnlineRes_Harry"/>
<owl:Class rdf:ID="OnlineResource"/>
<owl:DatatypeProperty rdf:ID="OnlineStatu">
    <rdfs:domain rdf:resource="#OnlineAccount"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="Organization"/>
<owl:Class rdf:ID="Person"/>
<owl:Class rdf:ID="PhysicalResource"/>
<PhysicalResource rdf:ID="PhysicalResource_harry"/>
<PreferContactWay rdf:ID="preferContact_Harry">
    <hasPreferWays rdf:resource="#MobilePhone"/>
</PreferContactWay>
<owl:Class rdf:ID="PreferContactWay"/>
<owl:Class rdf:ID="Profile"/>
<Profile rdf:ID="Profile_Harry">
    <hasContactInfo rdf:resource="#ContactInfo_Harry"/>
    <hasGeneralInfo rdf:resource="#GeneralInfo_Harry"/>
    <hasLanguage rdf:resource="#Language_Harry1"/>
    <hasLanguage rdf:resource="#Langue_Harry2"/>
    <hasAddress rdf:resource="#UK"/>
    <hasOrganization rdf:resource="#UniversityOfLeicester"/>
</Profile>
```

```
    <owl:ObjectProperty rdf:ID="requireResources">
        <rdfs:domain rdf:resource="#Activity"/>
        <rdfs:range rdf:resource="#Resource"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="Resource"/>
    <Resource rdf:ID="Resource_Harry">
        <hasPhysicalRes rdf:resource="#PhysicalResource_harry"/>
        <hasElectronicRes rdf:resource="#ElectronicResource_Harry"/>
        <hasOnlineRes rdf:resource="#OnlineRes_Harry"/>
        <hasFinanceRes rdf:resource="#Visa"/>
    </Resource>
    <owl:DatatypeProperty rdf:ID="start">
        <rdfs:domain rdf:resource="#Calendar"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:ID="Status"/>
    <Status rdf:ID="Status_Harry">
        <hasLevel rdf:datatype="&xsd;string">Normal</hasLevel>
    </Status>
    <owl:DatatypeProperty rdf:ID="Type">
        <rdfs:domain rdf:resource="#OnlineAccount"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <Address rdf:ID="UK"/>
    <Organization rdf:ID="UniversityOfLeicester"/>
    <FinanceResource rdf:ID="Visa"/>
</rdf:RDF>
```

Bibliography

[ACKM04] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services Concepts, Architectures and Applications.* Springer Book, 2004.

[AMM07] E. Al-Masri and Q. H. Mahmoud. Discovering the best web service. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 1257–1258. ACM, 2007.

[BBB01] C. Boutilier, F. Bacchus, and R.I. Brafman. Ucp-networks: A directed graphical representation of conditional utilities. In Jack S. Breese and Daphne Koller, editors, *UAI*, pages 56–64. Morgan Kaufmann, 2001.

[BCM+08] F. Baader, D. Calavanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider. *The Description Logic Handbook, 2nd Edition.* Cambridge University Press, 2008.

[BDS06] R.I. Brafman, C. Domshlak, and S.E. Shimony. On graphical modeling of preference and importance. In *J. Artif. Intell. Res. (JAIR)*, volume 25, pages 389–424, 2006.

[CF97] C. Carlsson and R. Fullér. *OWA operators for decision support.* in Proceedings of EUFIT'97 Conference, Vol. II, pages 1539–1544, 1997.

[CIJ+00] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M. C. Shan. *Adptive and Dynamic Service Compostion in eFlow.* Proceedings of the 12th International Conference on Advanced Information Systems Engineering, pages 13-31, LNCS volume 1789, 2000.

[CKL05] S. Cuddy, M. Katchabaw, and H. Lutfiyya. Context-aware service selection based on dynamic and static service attributes. In *Proceedings of IEEE International Conference on Wireless And Mobile Computing, Networking And Communications*, volume 4, pages 13–20. IEEE, 2005.

[Cor07] Microsoft Corporation. *Windows Workflow Foundation.* http://netfx3.com/content/WFHome.aspx, 2007.

[CPEV05] G. Canfora, M. D. PentaRaffaele, R. Esposito, and M. L Villani. *An approach for QoS-aware service composition based on genetic algorithms.* Proceedings of the 2005 conference on Genetic and evolutionary computation, pages 1069-1075, 2005.

[DA99] A.K. Dey and G.D. Abowd. *Towards a Better Understanding of Context and Context-Awareness.* Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pages 304-307. LNCS Volum 1707, 1999.

[DCG+06] J. Domingue, L. Cabral, S. Galizia, V. Tanasescu, A. Gugliotta, B. Norton, and C. Pedrinaci. *IRS-III: A broker-based approach to semantic Web services.* Book of The Semantic Web - ISWC'06, pages 201-214. LNCS Volume 4273/2006, 2006.

[DFvH03] J. Davies, D. Fensel, and F. van Harmelen. *Towards The Semantic Web Ontology-driven Knowledge Management.* John Wiley and Sons, LTD, 2003.

[DLP] A. Dan, H. Ludwig, and G. Pacifici. *Web Service Differentiation with Service Level Agreements.* White paper, IBM Corporation.

[Duj] J.J. Dujmovic. A method for evaluation and selection of complex hardware and software systems. In *Procedings of 22nd International Conference for the Resource Management and Perfromance Evaluation of Enterprise Computer Systems.*

[Duj73] J.J. Dujmovic. *Mixed Averaging by Levels (MAL)- A System and Computer Evaluation Method.* Proceedings of the Informatica Conference, paper d28, Bled, Yugoslavia, 1973.

[Duj75] J.J. Dujmovic. *Extended Continuous Logic and the Theory of Complex Criteria.* Journal of the University of Begrade, EE Dept., Series Mathematics and Physics, No. 537, pages 97–216, 1975.

[Duj07] J.J. Dujmovic. *Continuous Preference Logic for System Evaluation.* IEEE Transactions on Fuzzy System, vol. 15, No. 6, 2007.

[Erl04] T. Erl. *Service-oriented architecture : a field guide to integrating XML and Web services.* Prentice Hall PTR, 2004.

[ESB07] A-R. El-Sayed and J.P. Black. *Semantic-Based Context-Aware Service Discovery in Pervasive-Computing Environments.* in Proc. of IEEE Workshop on Service Integration in Pervasive Environments (SIPE), In conjunction with IEEE International Conference on Pervasive Services (ICPS), pages 556-557, IEEE, 2007.

[FR94] J.C. Fodor and M. Roubens. *Modelling and Multicriteria Decision Support.* Kluwer, Dordrecht, 1994.

[GGD07] S. Galizia, A. Gugliotta, and J. Domingue. A trust based methodology for web service selection. In *Proceedings of International Conference on Semantic Computing*, pages 193–200, 2007.

[Gro04] Object Management Group. *Common Object Request Broker Architecture: Core Specification.* Object Management Group, http://www.omg.org/docs/formal/04-03-12.pdf, 2004.

[Gro05] (WSMO) Working Group. Web service modeling ontology (wsmo) - an ontology for semantic web services, [online], http://www.w3.org/2005/04/fsws/submissions/1/wsmo-position-paper.html, 2005.

[HPSMW08] I. Horrocks, P.F. Patel-Schneider, D.L. McGuinness, and C.A. Welty. *The Description Logic Handbook: Chapter 14, OWL: A Description-Logic-Based Ontology Language for the Semantic Web, 2nd Edition*, pages 458–486. Cambridge University Press, 2008.

[HTW04] A. Helsinger, M. Thome, and T. Wright. Cougaar: A scalable, distribute multi-agent architecture. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pages 1910–1917. IEEE, 2004.

[III70] J.R. Miller III. *Professional Decision-Making.* Praeger Publishers, 1970.

[JS07] H. Janicke and M. Solanki. Policy-driven service discovery. In *Proceedings of the 2nd European Young Researchers Workshop on Service Oriented Computing*, pages 56–62, 2007.

[KBS04] D. Krafzig, K. Banke, and D. Slama. *Enterprise SOA: Service-Oriented Architecture Best Practices.* Prentice Hall PTR, 2004.

[KEKW04] N. Klimin, W. Enkelmann, H. Karl, and A. Wolisz. *A Hybrid Approach for Location-based Service Discovery in Vehicular Ad Hoc Networks.* citeseer.ist.psu.edu/klimin04hybrid.html, 2004.

[KHC+05] D. Karastoyanova, A. Houspanossian, M. Cilia, F. Leymann, and A. Buchmann. *Extending BPEL for Run Time Adaptability.* Proceedings of the 9th IEEE International EDOC Enterprise Computing Conference, pages 15-26, 2005.

[LASG07] S. Lamparter, A. Ankolekar, R. Studer, and S. Grimm. Preference-based selection of highly configurable web services. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 1013–1022. ACM, 2007.

[LH03] C. Lee and S. Helal. Context attributes: An approach to enable context-awareness for service discovery. In *Proceedings of 2003 Symposium on Applications and the Internet*, pages 22–30. IEEE Computer Society, 2003.

[LNZ04] Y. Liu, A.H. Ngu, and L.Z. Zeng. Qos computation and policing in dynamic web service selection. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 66–73. ACM, 2004.

[Mar98] J.L. Marichal. *Aggregation operations for multicriteria decision aid.* PhD. Thesis, Institute of Mathematics, University of Liège, Belgium, 1998.

[MBH+04] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. *OWL-S: Semantic Markup for Web Services.* W3C Member Submission, http://www.w3.org/Submission/OWL-S, 2004.

[MD01] T. P. Moran and P. Dourish. Introduction to this special issue on context-aware computing. *Special Issue of Human-Computer Interaction*, 16:1-8, 2001.

[MS04] E.M. Maximilien and M.P. Singh. A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, 8(5):84-93, 2004.

[MT05] U.S. Manikrao and T.V.Prabhakar. Dynamic selection of web services with recommendation system. In *(NWESP'05) Proceedings of the International Conference on Next Generation Web Services Practices*, page 117. IEEE Computer Society, 2005.

[NP05] R. Neto and M. Pimentel. *Toward a domain-independent semantic model for context-aware computing.* In Proceedings of the Third Latin American Web Congress, IEEE Computer Society pages 10-18, 2005.

[OR02] L.A. Olsina and G. Rossi. *Measuring Web application quality with WebQEM.* IEEE Multimedia, 9(4), pages20-29, 2002.

[Org04a] OASIS Orgnization. *UDDI Version 3 Specification.* OASIS Standard, 2004.

[Org04b] W3C Orgnization. *OWL Web Ontology Language Overview.* W3C Standard, 2004.

[Org07a] Oasis Organization. *Web Services Business Process Execution Language Version 2.0 - Primer.* http://docs.oasis-open.org/wsbpel/2.0/Primer/wsbpel-v2.0-Primer.pdf, 2007.

[Org07b] W3C Orgnization. *SOAP Version 1.2.* W3C Standard, 2007.

[Org07c] W3C Orgnization. *Web Services Description Language (WSDL) Version 2.0.* W3C Standard, 2007.

[oTOG06] SOA Working Group of The Open Group. *Service-Oriented Architecture (SOA):Definition of SOA.* Open Group SOA Definition, 2006.

[Pac04] Hewlett Packard. *Cooltown Project.* http://www.cooltown.com/cooltown/, 2004.

[PBMW98] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. http://citeseer.ist.psu.edu/page98pagerank.html.

[PS06] E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf, [online], http://www.w3.org/tr/rdf-sparql-query/, 2006.

[RDN05] M.A. Razzaque, Simon Dobson, and Paddy Nixon. Categorisation and modelling of quality in context information. In *Proceedings of the IJCAI 2005 Workshop on AI and Autonomic Communications*, pages 1–10. Springer, 2005.

[Red97] F. E. Redmond. *DCOM: Microsoft Distributed Component Object Model.* IDG Books Worlwide, 1997.

[RMYT09] S. Reiff-Marganiec, H. Q. Yu, and M. Tilly. Service selection based on non-functional properties. In *Proceedings of Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop NFPSLA-ICSOC'07*, pages 128–138. Springer Lecture Notes in Computer Science, 2009.

[RPC+04] E. Rukzio, G. N. Prezerakos, G. Cortese, E. Koutsoloukas, and S. Kapellaki. *Context for Simplicity: A Basis for Context-aware Systems Based on the 3GPP Generic User Profile.* In Proceedings of International Conference on Computational Intelligence, pages 17-19, 2004.

[RT06] O. Riva and S. Toivonen. A hybrid model of context-aware service provisioning implemented on smart phones. *ACS/IEEE International Conference on Pervasive Services*, (4):47 - 56, 2006.

[SAW94] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 85–90, Santa Cruz,CA, USA, 1994. IEEE Computer Society.

[SBS+07] C. Schropfer, M. Binshtok, S.E. Shimony, A. Dayan, R. Brafman, P. Offermann, and O. Holschke. Introducing preferences over nfps into service selection in soa. In *Proceedings of Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop 2007*, pages 68–79. Springer, 2007.

[SDB+87] Y.W. Su, J.J. Dujmovic, D.S. Batory, S.B. Navathe, and R. Elnicki. *A cost-Benefit Decision Model: Analysis, Comparison, and Selection of Data Management Systems.* ACM Transactions on Database Systems, Vol. 12, No. 3, pages 472–520, 1987.

[SJS05] Y.J. Seo, H.Y. Jeong, and Y.J. Song. A study on web services selection method based on the negotiation through quality broker: A maut-based approach. In *Book Embedded Software and Systems*, pages 65–73, 2005.

[SM03] J.M. Fenández Salido and S. Murakami. *Extending Yager's orness concept for the OWA aggregators to other mean operators.* Fuzzy Sets and Systems, Elsevier B.V. 139, pages 515–542, 2003.

[SVC+03] I. Sygkouna, S. Vrontis, M. Chantzara, M. Anagnostou, and E. Sykas. *Context-Aware Services Provisioning on Top of Active Technologies.* Book Series Lecture Notes in Computer Science Publisher Springer Berlin / Heidelberg ISSN 0302-9743, Volume

Volume 2881/2003 Book Mobile Agents for Telecommunication Applications, Category Service Management - Service Provisioning pages 67-76 Subject Collection Computer Science, 2003.

[Tom07] I. R. Toma. On describing and ranking services based on non-functional properties. In *Proceedings of the Third International Conference on Next Generation Web Services Practices*, pages 61–66. IEEE Computer Society, 2007.

[TRMY07] M. Tilly, S. Reiff-Marganiec, and H.Q. Yu. *Design and Implemetationn of monitoring and aggregation mechanisms for context-based services - Version 1.* inContext project deviverables, D3.2 V1, 2007, http://www.in-context.eu/page.asp?PageRef=10, 2007.

[TYRM+08] M. Tilly, H.Q. Yu, S. Reiff-Marganiec, D. Schall, and S. Peray. *Design and Implemetationn of monitoring and aggregation mechanisms for context-based services - Version 2.* inContext project deviverables, D3.2 V2, 2008, http://www.in-context.eu/page.asp?PageRef=10, 2008.

[UF06] European Union and FP6 Framework. *inContext (Interaction and Context Based Technologies for Collaborative Teams) project.* IST IST-2006-034718, 2006.

[Vuk07] M. Vukovic. *Context aware service composition.* Technical Report, Computer Laboratory, University of Cambridge,UCAM-CL-TR-700 ISSN 1476-2986, 2007.

[WV07] Y. Wang and J. Vassileva. Vassileva: Toward trust and reputation based web service selection: A survey, [online], http://bistrica.usask.ca/madmuc/papers/yao-julita-ws-mas-survey.pdf, 2007.

[WVKT] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma. A qos-aware selection model for semantic web services. In *Proceedings of Service-Oriented Computing ICSOC 2006*, pages 390–401. Springer LNCS.

[Yag88] R.R. Yager. *On ordered weighted averaging aggregation operators in multi-criteria decision making.* IEEE Transactions on Systems, Man and Cybernetics 18, pages 183–190, 1988.

[YHHRM07] H.Q. Yu, Y. Hong, R. Heckel, and S. Reiff-Marganiec. *Context-sensitive Team Formation: Towards Model- Based Context Reasoning and Update.* Sixth International and Interdisciplinary Conference on Modeling and Using Context, Doctorial Consortium, pages 115-129, 2007.

[YL05] T. Yu and K. Lin. *Service Selection Algorithms for Composing Complex Services with Multiple QoS Constrains.* ICSOC2005, LNCS, vol: 3826, pages 130-143, 2005.

[YRM08a] H. Q. Yu and S. Reiff-Marganiec. A method for automated web service selection. In *2008 IEEE International Conference on Services Computing*, volume 0, pages 513–520, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[YRM08b] H.Q. Yu and S. Reiff-Marganiec. Non-functional property based service selection: A survey and classification of approaches. In *Proceedings of Non-Functional Properties and Service Level Agreements in Service Oriented Computing Workshop co-located with The 6th IEEE European Conference on Web Services*, Dublin, Ireland, 2008. Sun SITE Central Europe.

[YRM09a] H.Q. Yu and S. Reiff-Marganiec. Automated context-aware service selection for collaborative systems. In *Proceedings of The*

21st International Conference on Advanced Information Systems, pages 193–200. Springer Lecture Notes in Computer Science, 2009.

[YRM09b] H.Q. Yu and S. Reiff-Marganiec. A backwards composition context based service selection approach for service composition. In *IEEE International Conference on Services Computing (SCC'09.* IEEE, 2009.

[YRMT08] H.Q. Yu, S. Reiff-Marganiec, and M. Tilly. Composition context for web services selection. In *IEEE International Conference on Web Service, Work In Progress Track*, pages 304–307, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[ZBN+05] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. *QoS-aware middleware for web services composition.* IEEE Transactions on Software Engineering, pages 311-327, 2005.

[ZMN05] F. Zhu, M. W. Mutka, and L. M. Ni. Service discovery in pervasive computing environments. *IEEE Pervasive Computing*, 4(4):81–90, 2005.