Modelling and Brokering Emergency Requests-For-Assistance

Markos F. Fragkakis

Master of Science
School of Informatics
University of Edinburgh
2006

Abstract

The scale of the December 2004 Indian Ocean Tsunami required a huge humanitarian relief effort, the participants in which were numerous and varied in nature and resources. Given the diversity and the numbers of participants in relief efforts such as this, there is need for central coordination, which is provided by organisations like MPAT. One of the roles of MPAT is to act as a broker for Requests-for-Assistance (RFAs), attempting to match requests for services or expertise with the corresponding service-providers and experts.

Currently, RFAs are brokered manually by the MPAT personnel, which has several drawbacks. The purpose of this project is to consider ways in which this brokering task may be formalised and (perhaps partially) automated. This aims to improve the brokering process, both by achieving more accurate brokering results and by reducing the amount of time needed for the task to be completed.

Acknowledgements

I would like express my gratitude to my supervisor, Stephen Potter, who helped me realize that Research is as much about answering questions as about questioning answers.

I would also want to thank my parents, Fotios and Ioulia Fragkakis, who keep enduring me and the financial implications I entail.

Finally, special thanks go to my good friend Petros Pistofidis, who has been supporting me all those years.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Markos F. Fragkakis*)

Table of Contents

List of Figures

Chapter 1

Introduction

The scale of the December 2004 Indian Ocean Tsunami required a huge humanitarian relief effort. The participants in that effort were numerous and varied in nature and resources. Nations, armies, organisations and other groups of people offered their capabilities, expertise, man power and money for the common cause.

Given the diversity and the numbers of participants in relief efforts such as this, there is need for central coordination. This task is undertaken by organisations like the Multinational Planning Augmentation Team (MPAT)[1], which helps establish task command and planning headquarters. One of the roles of MPAT during the response is to act as an information exchange[2] , and as part of this role it acts as a broker for Requests-for-Assistance (RFAs), attempting to match requests for services or expertise with the corresponding service-providers and experts. As an example, the following was one of the RFAs that were successfully brokered during the tsunami relief effort:

> *Transport 400 MTs (441 tons) of High Energy Biscuits (HEBs) from Bangladesh (Zia Int'l Airport) to Banda Aceh and Medan, IN.*

The brokering of this RFA was performed manually. In other words, a person or a group of people decided which of the relief effort participants could best meet the particular RFA. The purpose of this project is to consider ways in which this brokering task may be formalised and (perhaps partially) automated. This aims to improve the brokering process, both by achieving more accurate brokering results and by reducing the amount of time needed for the task to be completed.

Formalizing and automating this brokering process is a very important task: by adequately modelling the RFAs and developing a brokering methodology that is accurate

[1]`http://www2.apan-info.net/mpat/`

[2]`https://team.apan-info.net/tsunami`

and sound, we may ensure that needs are covered by more adequate or appropriate resources. Furthermore, during crisis periods, time is a crucial factor. Automating the RFA brokering may result in accelerating the whole process, which in turn may help to save lives.

Besides the practical benefits involved, this project has some interesting scientific aspects as well. As will be seen later in this thesis, one possible way to model RFAs and capabilities is by using the OWL-S [14] ontology, developed primarily for describing Web Services. As OWL-S has been mainly used for Web Services, it would be interesting to test its limitations when describing different types of "real" services and resources, like capabilities and RFAs.

This thesis will have the following structure:

In Chapter 2, we will attempt to discover the nature of requests that might occur in an crisis by examining actual RFAs. Furthermore, the brokering procedure that is currently used by the MPAT will be reviewed. We will conclude by setting some desiderata for the modelling and brokering methodologies to be proposed.

In Chapter 3, some approaches for the modelling of RFAs will be considered. In particular, we will present the advantages and disadvantages of databases, RDF and OWL, and attempt to discover the degree to which they seem to meet the previously set desiderata. The chapter will continue by considering the use of the OWL-S ontology and will end by reviewing two service brokering approaches.

In Chapter 4, we will analyse how the OWL-S ontology can be adopted for RFAs. In particular, we will consider ideas from the two brokering approaches and propose solutions to problems that might occur. The chapter will end by proposing an approach to model RFAs and services in OWL-S.

In Chapter 5, we will propose an algorithm to broker RFAs and services. Each of the three parts of the algorithm will be analysed in detail, and some hypothetical scenarios will be considered.

In Chapter 6, we will present the RFA Broker, an implementation of the brokering algorithm. In particular, we will review its architecture, and analyse the role and function each of its components.

In Chapter 7, we will attempt to evaluate this project. The evaluation will begin with some test case scenarios that were processed by the RFA Broker. We will continue by assessing the degree to which the desiderata of Chapter 2 are met, and conclude with an analytical evaluation of certain design choices.

In Chapter 8, we will draw some conclusions and propose future work for this project.

Chapter 2

Requests-For-Assistance

Requests-For-Assistance, or RFAs, are descriptions of local needs that are identified by or communicated to the MPAT personnel. As mentioned in the previous chapter, one of the goals of this thesis is to formally model RFAs and enable their automatic brokering. However, before we discuss their formal modelling, in this chapter there will first be a review of the current modelling and brokering procedures used by MPAT. The purpose of this review is to identify factors that may directly or indirectly affect the brokering process, and also to identity any mistakes or possible areas of confusion in the current approach, which can then be avoided in the formal approach.

The resources that were available for this “exploration” of the nature of RFAs were found in the Asia-Pacific Area Network (APAN)[1] web portals for particular crises. On two of these portals, namely the 2004 Tsunami portal[2] and the 2006 Mudslide in Leyte portal[3], were published the list of submitted RFAs, as well as some additional material for describing RFAs and tracking their status.

Unfortunately, crises portals other than the two mentioned have restricted access to sections that may contain important material for this thesis. For instance, the RFA repositories in portals of ongoing crises have restricted access to certified users. As a result, it is not known whether the unavailable RFAs concerned different needs, if the description form has changed, or if the MPAT has altered their brokering process.

On the other hand, it has to be noted that the material that was put together proved to be sufficient to reveal several shortcomings of the then current RFA procedures. It also made possible the proposal of improvements and the development of ideas that will be presented later in this chapter. To conclude, despite the fact that resources

[1]`http://www2.apan-info.net/apan/`
[2]`https://team.apan-info.net/tsunami`
[3]`https://www2.apan-info.net/apan/leyte/`

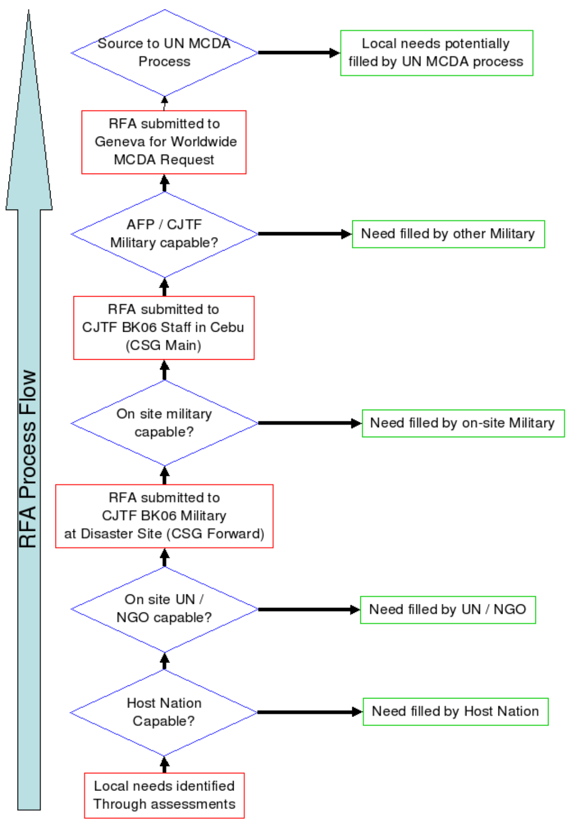

RFA Process Flow
Source to UN MCDA Process
Local needs potentially filled by UN MCDA process
RFA submitted to Geneva for Worldwide MCDA Request
AFP / CJTF Military capable?
Need filled by other Military
RFA submitted to CJTF BK06 Staff in Cebu (CSG Main)
On site military capable?
Need filled by on-site Military
RFA submitted to CJTF BK06 Military at Disaster Site (CSG Forward)
On site UN / NGO capable?
Need filled by UN / NGO
Host Nation Capable?
Need filled by Host Nation
Local needs identified Through assessments

Figure 2.1: The RFA process flow chart

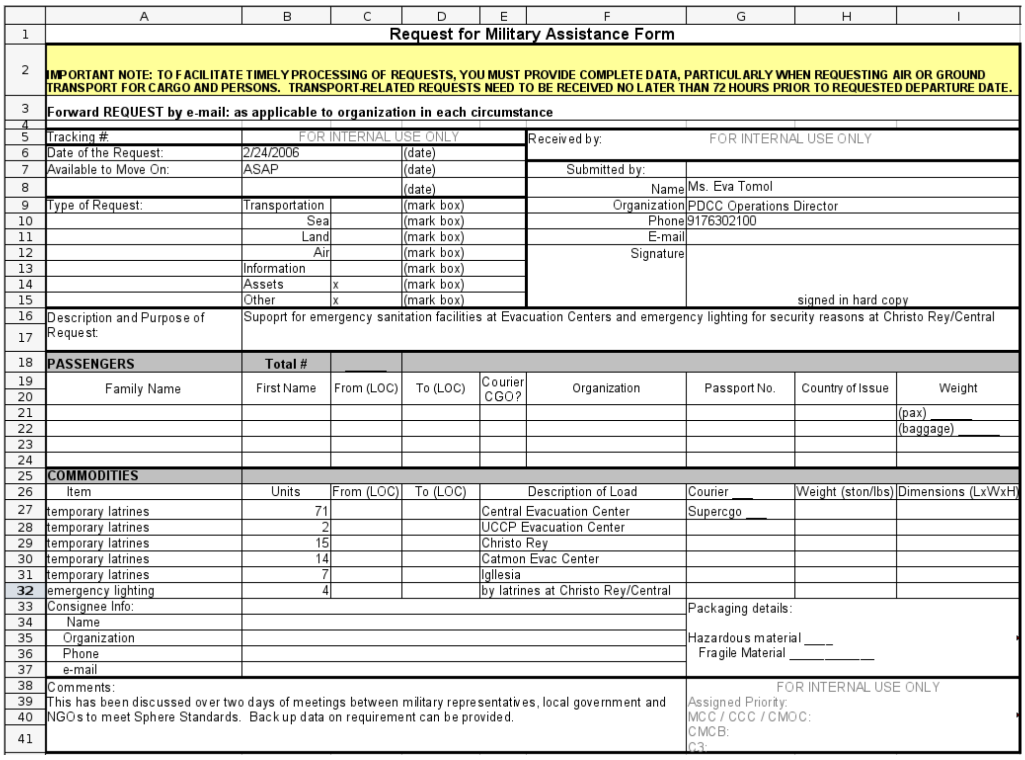
which would have clarified certain points were unavailable, the ones that actually were considered provided valuable insights to guide the modelling and brokering process attempted during this project.

2.1 Current Brokering Process

Currently, there is no automated method for the brokering of RFAs and available resources. The procedure is implemented by people, and generally comprises the following stages:

1. A person or group of people (who may not be MPAT personnel) identifies a "local need" and records it on a form that is available on the MPAT website. The form is then forwarded to MPAT and the information it contains is stored in an RFA Form (an Excel spreadsheet).

2. An attempt is then made to broker the current RFAs in one of two ways:



	A	B	C	D	E	F	G	H	I
1	**Request for Military Assistance Form**								
2	**IMPORTANT NOTE: TO FACILITATE TIMELY PROCESSING OF REQUESTS, YOU MUST PROVIDE COMPLETE DATA, PARTICULARLY WHEN REQUESTING AIR OR GROUND TRANSPORT FOR CARGO AND PERSONS. TRANSPORT-RELATED REQUESTS NEED TO BE RECEIVED NO LATER THAN 72 HOURS PRIOR TO REQUESTED DEPARTURE DATE.**								
3	**Forward REQUEST by e-mail: as applicable to organization in each circumstance**								
4									
5	Tracking #:	FOR INTERNAL USE ONLY				Received by:	FOR INTERNAL USE ONLY		
6	Date of the Request:	2/24/2006		(date)					
7	Available to Move On:	ASAP		(date)		Submitted by:			
8				(date)		Name	Ms. Eva Tomol		
9	Type of Request:	Transportation		(mark box)		Organization	PDCC Operations Director		
10		Sea		(mark box)		Phone	9176302100		
11		Land		(mark box)		E-mail			
12		Air		(mark box)		Signature			
13		Information		(mark box)					
14		Assets	x	(mark box)					
15		Other	x	(mark box)				signed in hard copy	
16	Description and Purpose of Request:	Supoprt for emergency sanitation facilities at Evacuation Centers and emergency lighting for security reasons at Christo Rey/Central							
17									
18	**PASSENGERS**	**Total #**	______						
19	Family Name	First Name	From (LOC)	To (LOC)	Courier CGO?	Organization	Passport No.	Country of Issue	Weight
20									
21									(pax) ______
22									(baggage) ______
23									
24									
25	**COMMODITIES**								
26	Item	Units	From (LOC)	To (LOC)	Description of Load		Courier ___	Weight (ston/lbs)	Dimensions (LxWxH)
27	temporary latrines	71			Central Evacuation Center		Supercgo ___		
28	temporary latrines	2			UCCP Evacuation Center				
29	temporary latrines	15			Christo Rey				
30	temporary latrines	14			Catmon Evac Center				
31	temporary latrines	7			Igllesia				
32	emergency lighting	4			by latrines at Christo Rey/Central				
33	Consignee Info:						Packaging details:		
34	Name								
35	Organization						Hazardous material ____		
36	Phone						Fragile Material __________		
37	e-mail								
38	Comments:						FOR INTERNAL USE ONLY		
39	This has been discussed over two days of meetings between military representatives, local government and						Assigned Priority:		
40	NGOs to meet Sphere Standards. Back up data on requirement can be provided.						MCC / CCC / CMOC:		
41							CMCB: C3:		

Figure 2.2: The MPAT RFA Form

- Representatives of organisations who can provide their resources/services (hereafter simply called services) as part of the relief effort can browse through the online RFA directory and decide whether they can cover some of the needs.
- Additionally, MPAT staff may contact organisations they think might be able to help address certain RFAs. The request is passed to appropriate organisations which can then say if they can meet it.

The process of assigning available services to a submitted RFA can be seen in Figure 2.1[4]. It can be seen that the search for a prospective response to an RFA is escalated through local and locally sited civilian organizations, through military organizations, all the way up to a request for a response at the global level. Irrespective of who performs the brokering, for the purposes of this project the important issue here is that it is always performed by people. This may hold certain disadvantages:

- The MPAT form, shown in Figure 2.2, provides some structure and uniformity to information on the RFAs. However, most of this information is recorded in

[4]The figure was taken from the Combined Support Groups form for RFAs (see 2.2.1).

natural language, containing acronyms, military jargon and typos. This makes it prone to misinterpretation, especially when the people involved are not native English speakers, which is often the case.

- The number of RFAs and available services may be too large for a person or group of people to process in a reasonable amount of time. This may cause successful matches between RFAs and services to be overlooked.
- There are several cases where two or more available services may be combined to cover an RFA. This composition may be too complicated for people to perform, especially when the number of participants in the relief effort is large.
- The MPAT does not have an explicit and formal description of the capabilities of the relief effort participants. This results in brokering based on the knowledge and previous experience of the human broker, rather than brokering based on the actual available services.

The disadvantages listed above can be very limiting and may gravely affect the results of the brokering. Therefore, they have to be taken into account during the design of an automated brokering process.

2.2 Issues

In addition to addressing the limitations mentioned above, there are some other aspects that have to be taken into account. This section examines some of those and attempts to draw some guidelines that will be followed during the design process.

2.2.1 Types of RFAs

The Combined Support Groups (CSGs) of Indonesia, Sri Lanka and Thailand have introduced a form for describing RFAs. All RFAs that were submitted to both the Tsunami portal and the Leyte mudslide portal were recorded using this form. Unfortunately, the 2006 Java earthquake (the most recent MPAT relief effort at the time of writing) website RFA section was restricted and, consequently, it is not clear whether the RFA form is still being used. However, the fact that it was used in two distinct incidents, separated in time by over a year, indicates that it may be an established template, and this is how it will be considered here.

The form is intended to hold information in a way that humans can broker, whereas this project aims to allow brokering to be performed or assisted by a machine. Therefore, the representation contained in this form might not be the most appropriate for the purposes of this project. However, it gives clear indications and directions of what the MPAT people believe is enough information to accurately describe a request. And this will be a common factor, whatever modelling and brokering method is adopted.

The form is a spreadsheet with predefined fields describing who submitted the form and the type of need to be satisfied. The field named "Type of request" has four predefined choices: "Transportation", "Information", "Assets" and "Other". This, as well as the fact that the two largest areas of the form are used to describe the passengers and the commodities that need to be used, shows that the form was created with focus on particular types of RFAs. In turn, this indicates that these are the types of RFAs that most commonly occur.

In order to draw a clearer picture of how common transport and provision RFAs are, actual data had to be used. There are about 80 RFAs in the APAN portals, others described in RFA forms, others in RFA Status forms[5]. A review of these showed that:

- 90% are RFAs concerning or involving transport of equipment or personnel (and which sometimes also imply provision of the equipment or personnel). Typical example:

  Transport 400 MTs (441 tons) of High Energy Biscuits (HEBs) from Bangladesh (Zia Int'l Airport) to Banda Aceh and Medan, IN.

- 4% are RFAs requesting for a certain good to be provided or donated. The difference to the above category is subtle Typical example:

  UNJLC requests donation of mobi-mating. CSF resolution of whether or not the mating can be donated remains sticking point.

- 2% are RFAs requesting for certain task be carried out. Typical example:

  Logistic assessment of NW coast landing sites, Sabang Island and K-Raja port infrastructure.

- The remaining 4% are RFAs that are not categorized, like more complicated RFAs. Example:

[5]The RFA Status forms describe the current status of pending RFAs. They, too, are stored as an Excel spreadsheet.

Require expert in vector control and malaria is needed to provide guidance to Indonesian vector-control teams. Need person w/experience in leading teams in the field, familiarity w/Indonesia, ability to speak Bahasa Indonesian.

It was noted before that the complete set of RFAs across all emergencies was not available due to restricted access. However, this small survey indicates that the overwhelmingly dominant type of RFA concerns or involves transportation and sometimes the provision of goods. Given this, it would be sensible to have as main priority the correct brokering of transport and provision RFAs, possibly at the expense of adopting a more generic approach. Of course, success in handling the diversity of RFAs is also desirable.

The survey also makes apparent that RFAs can vary from expressions with two or three parameters to more complicated ones, like the last example quoted above. Therefore, it is desirable for a modelling approach to be able to express both simple and complicated expressions. Similarly, a brokering method should be robust enough to handle complicated RFAs without demanding excessive computational resources.

2.2.2 Combining Resources

An interesting attribute of the form that should be pointed out is the fact that more than one service may be requested to meet the need. This can be done either explicitly, by filling in more independent requests. However, what the person who fills out the form perceives as a single RFA may actually require several resources or services to be invoked. As an example, several of the RFAs were requesting for particular good to be provided, like in this case (4 Feb 2005, Indian Ocean Tsunami relief effort):

Provide water purification unit to Royal Thai Navy.

Although the action for this request is not clarified in the RFA Status form, it is reasonable to assume that the purification unit manufacturer / provider may be located in place different from the destination stipulated by the RFA and furthermore, that this provider does not necessarily have the capability to ship its products to the destination. This would create the need for an additional, cargo transporting, service to be invoked once the purification unit had been secured. Even more, there may not be a sole transport service from the location of the provider to the destination where the commodity is needed, thus requiring the coordination of several transport services.

It seems logical to assume that the case where an RFA may need the collaboration of multiple service providers is not occasional, but in fact, very common. It was mentioned earlier that managing large numbers of RFAs and available services may be too large for a person to handle. As the number of service combinations may approach exponential to the original number of services, the difficulty of the task can significantly increase, and plausible solutions may be overlooked.

In conclusion, it would be desirable for an automated brokering process to be able to propose "orchestrations" of services to satisfy RFAs.

2.2.3 Missing Information

A final remark concerns the fact that several of the fields in the RFA forms of the APAN portals were found to be left blank. More specifically, while it appeared that all the RFA forms that were examined contained the minimum information to clearly describe the request, it is hard to know if this information is enough for all possible matches to be discovered. The problem presented here is twofold: firstly, it is not obvious to the person filling out the form exactly which fields will be needed to successfully broker the RFA. Secondly, even if one knows which fields are actually needed, this information may not be available to them. To judge by the stages in the brokering process depicted in Figure 2.1, it seems that no reconsideration of an RFA Form takes place once it has been submitted.

For example, an RFA submitted on the 24 Feb 2006 for the Leyte mudslide incident, requests the delivery of temporary latrines for the central evacuation center (Figure 2.2). One can notice that only the fields of the form describing the asset and the place of the request are filled in. This information is sufficient to describe the ongoing need, and possibly find a role-filling service that can satisfy it. However, additional role-filling services or groups of services could be discovered if more fields were available.

In order to cope with this problem, an automated brokering approach could gradually assign available services to the RFA, enriching it at the same time with any additional information these assignments may entail. The brokering would carry on until the constantly rephrased RFA is satisfied by the assignment of a group of services. In the case presented above, if there is no latrine provider in Leyte, the brokering would not fail, but rather try assigning a latrine provider located in a remote place. This assignment would enrich the RFA with information like the dimensions and the weight of the latrines, and would enable searching for a service capable of transporting the latrines to Leyte. Hence, the brokering process now involves elaborating, enriching or refining the RFA itself.

2.3 Modelling and Brokering Desiderata

The modelling and brokering of the RFAs are closely interrelated tasks. This is because the way brokering will be carried out will depend heavily on the model that is used. Similarly, the RFA model should be designed in such a way that will be possible for it to being brokered. In an effort to clarify some goals for modelling and brokering, separate desiderata were set for them.

Based on the issues discussed above, the following properties would be desirable for an RFA **modelling** approach:

- it provides a formal description of RFAs that is not prone to misinterpretation.
- it can precisely model at least transportation and provision RFAs and services, which are the most common types.
- it is flexible enough to describe many diverse types of RFAs ans services, even new ones that may emerge.
- it is able to describe RFAs and services with varying detail.

As far as an RFA **brokering** approach is concerned, the following properties would be desirable:

- it can successfully broker at least transportation and provision RFAs, which are the most common types.
- it can propose complicated solutions to an RFA, in the form of orchestrated services.
- it can cope with large numbers of RFAs and services.
- it can enrich RFAs with additional information, as the brokering process progresses.

In a later chapter we shall return to these desiderata to assess the extent to which the approach to brokering adopted here satisfies them and, hence, use them as some measure of the adequacy of the approach.

Chapter 3

Background

This chapter contains a review of some commonly used approaches to the task of brokering. These may differ not only in the methodologies they use, but also in the type of objects they try to model and broker. This chapter will list the advantages and disadvantages of each approach, focused particularly on the purpose of this thesis. In order to achieve this, there will be an attempt to describe the extent to which an RFA could be modelled and brokered in each of these approaches.

3.1 Non-Semantic approaches

The simplest and most straightforward way to model RFAs and available services would be using by using a database. The set of services can be stored in a database table, with each one occupying a table row and each parameter being stored in a cell. Another table with similar fields may be used for storing pending RFAs. An RFA can then be brokered by formulating an appropriate query to the database, asking for services with matching values in corresponding fields.

Databases have some features that would be useful for the purpose of this thesis. The fact that records can be added, removed and updated, gives direct and low-level control over the data. They offer some flexibility, in the aspect that they enable the user to pass custom queries to the database and access the available data in whatever way is convenient. Finally, the fact that database technology has been evolving for many decades has helped develop increased speed, scalability and reliability, all of which are very important for brokering RFAs and services.

However, using databases has some drawbacks as well. RFAs have been shown to vary greatly, which means that it may become apparent that a new design is needed to uniformly describe the new set of RFAs. This is difficult when using databases, as their structure is relatively rigid throughout their use. Although new fields and tables can be easily added and populated at any time, the transfer of existing data to a differently designed database may be needed. This migration can be troublesome and, in the worst case, may result in information loss.

The most important drawback of using databases is the fact they typically use string comparison to match data. This can be a problem when comparing an RFA description to an available service description, as different words may have been used for the same information. For instance, an RFA may be requesting for a *vehicle* to transport injured people, whereas an appropriate service may be providing an *ambulance*. If this brokering was performed by a person, a match would be identified. However, a database will only compare the words “vehicle” and “ambulance”, which are different strings. The same type of error can also be caused by typos or acronyms, which are common in RFAs.

The underlying reason for this kind of behaviour is that databases make syntactic, and not semantic comparison. In other words, they only compare strings, and not their meaning. There is no direct way of taking into account that an *ambulance* is a type of *vehicle*. Workarounds, like creating tables to contain such hierarchical structures would be very time consuming and inefficient, as there would always be information that will be left out.

Given the nature of RFA descriptions, using databases or other approaches that rely on string comparison is inadequate, as it fails meet the most important of the modelling desiderata: it is error-prone. Furthermore, as far as brokering is concerned, there is no evident way of creating orchestrated solutions, at least when using databases. For these reasons, the following methods that are examined are based on semantic descriptions.

3.2 RDF / RDFS

The Resource Description Framework (RDF) [12] is a W3C[1] specification for knowledge description. The structural elements of RDF are resources, properties and statements. The idea behind RDF is to model objects as resources, which are described and interrelated through their properties. The resources and their properties are stated in the form of triples, called statements. Each statement has a subject, a predicate and an object, where the subject is a resource, the predicate is a relation and the object is

[1]http://www.w3.org/

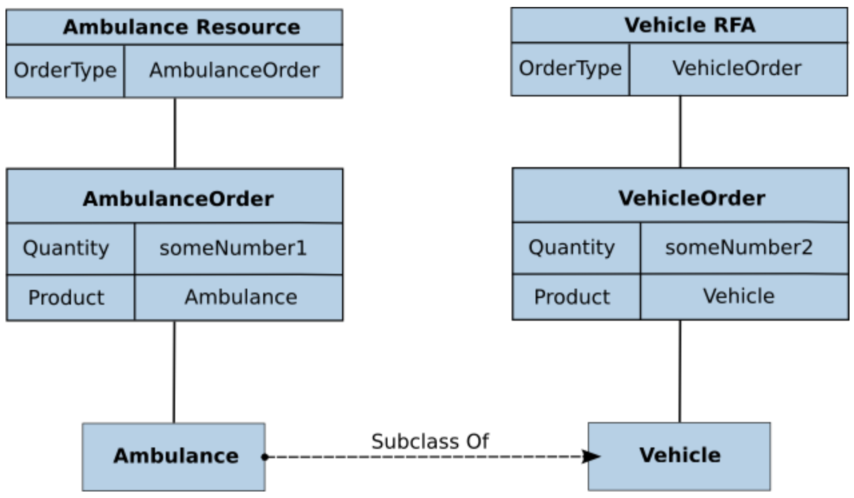


Figure 3.1: Resource and RFA described in RDF

another resource or a literal.

Resources and properties are uniquely identified by Universal Resource Identifiers (URIs)[2]. Although URIs usually begin with the http:// prefix, they don't have to point to retrievable documents, but rather they act as a unique identifier. RDF can be represented in XML [4], from which it inherits several advantages, like extensibility and the use of namespaces. If the assignment of URIs to objects is seen as a vocabulary, the use of XML namespaces, along with extensibility, allows the importing and use of remote vocabularies. In this way the same vocabulary can be used by several parties to express knowledge of the same domain.

Despite the fact that RDF can be used to detect equality of resources through the underlying URIs, it cannot express general semantic information about them. This is the reason it is used in conjunction with RDF Schema (RDFS) [5], an extensible knowledge representation language. RDFS can define URIs to represent classes and properties of classes. It also enables the refinement of properties' definitions by restricting their domain and range. Finally, it allows the definition of hierarchies of both classes and properties through subclass and subproperty relationships. By using RDFS definitions in RDF descriptions, one can describe a richer domain of individuals as members of particular conceptual classes, and can perform simple reasoning based on their hierarchical and other relationships.

RDF and RDFS can been used for semantic matchmaking of services, which means assigning a service advertisements to service requests, based on semantic criteria. In

[2]URLs are just a type of URIs.

contrast to service matchmaking, service brokering may result to the assignment of more that one collaborating services to a request, which is called service orchestration.

In [25], the authors propose a way to describe e-commerce services in RDF, as well as an algorithm that uses these descriptions to conduct semantic matchmaking. The service advertisements and requests have the same form, which is essentially an RDF graph. The brokering algorithm can traverse the graphs looking for matching nodes, where a match would mean equality or subsumption of the two nodes or their children.

Despite RFAs not requesting e-commerce services, but rather services in the physical world, the description and the algorithm presented are generic enough to be adopted. The first step would be to create class hierarchies for locations, goods and types of resources and RFAs. If the previously used example is modelled in the way proposed, the available service and the RFA would both have the form of RDF graphs, as shown in Figure 3.1. Here, *vehicle* and *ambulance* are both defined as classes, with the ambulance class being a subclass of vehicle. There will be no direct match, as the class URIs are different. However, if the class ambulance is defined as a subclass of the class vehicle, the subsumption can be detected, which is considered to be an indirect match.

On the other hand, RDFS has some limitations itself, which unfortunately impringe on the way RFAs can be described using RDFS:

- The range of a property has to be common for all the classes of a property. As a result, if VehicleResources are defined to provide Vehicles, some of them cannot only provide Ambulances.

- Two or more classes cannot be defined to be disjoint. As a result, Vehicles cannot be defined to be disjoint from, say, Locations.

- A class cannot be defined as the union, intersection or complement of other classes. As a result, Vehicles cannot be defined as the union of Cars, Motorcycles, etc.

- Cardinality constraints cannot be imposed on properties. As a result, an transporting service cannot be defined to be able to carry at most 3 types of cargo (if there is such reason).

- Properties cannot be defined to be transitive. As a result, if aircraft A has greater radius than aircraft B, and aircraft B has greater radius than aircraft C, it cannot be inferred that A has greater radius than C.

Some of these RDFS limitations may seem too trivial to be limiting to RFA descriptions. However, the relations between classes can be more complicated than direct subclass relations. If there is no way to express them, this will eventually cause matches to be overlooked. Although RDFS does use semantic information to do the brokering, its expressiveness is fairly limited. As a result, it fails to meet the modelling desiderata.

3.3 OWL

The expressive limitations of RDF led W3C to the development of the Web Ontology Language (OWL) [21]. There are three 'species' of OWL, each with increasing expressiveness and computational complexity. This means that there is a trade-off between expressiveness and efficiency. As a result, the selection between the three varieties of OWL depends on the need for expressiveness and underlying logic, but also on the computational resources that are available. The three species of OWL are:

OWL Full : This is the complete OWL, which contains all the primitives of the language, and also allows the use of RDF and RDFS primitives. OWL Full is the only variety of OWL where a class can be regarded both as the collection of instances and as an instance itself. Furthermore, limitations can be applied on the number of subclasses a class can have, and consequently, on the number of classes in an ontology. The extended expressiveness of OWL Full comes at the cost of undecidability. As stated in [17], it is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

OWL DL : OWL DL is the maximum subset of OWL that maintains two important characteristics: computational completeness (all conclusions can be computed to be true or false) and decidability (all computations will finish in finite time). All the expressions of OWL are available in OWL DL, except the ones that could jeopardize completeness and decidability (i.e. owl:inverseFunctionalProperty). Its name comes from its correspondence to Description Logics [1], a decidable subset of First Order Logic [24].

OWL Lite : OWL Lite is the lightweight subset of OWL DL. It allows the expression of taxonomies and simple relations among classes and instances. However, cardinalities other that 0 and 1, disjointness and enumeration of classes are not allowed. In effect, OWL Lite is the simplest subset of OWL from a computational point of view.

About the compatibility between the three (sub-)languages, OWL DL allows all

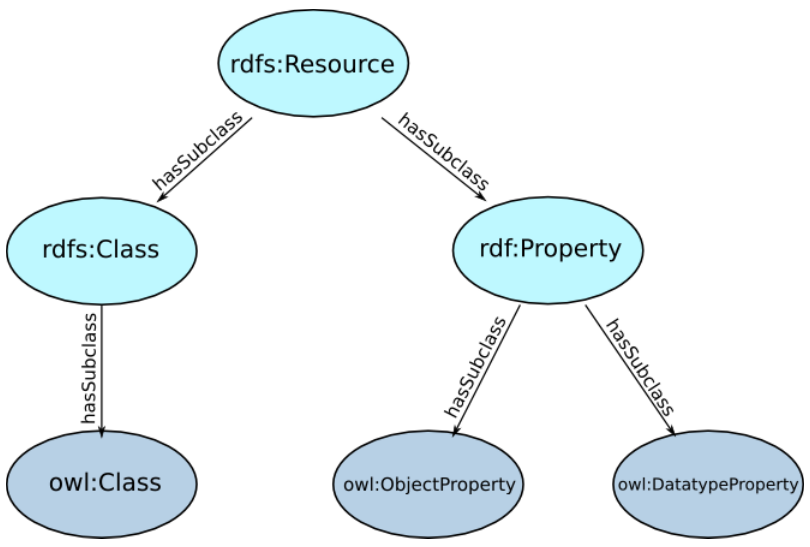


Figure 3.2: OWL and RDF primitives

valid expressions of OWL Lite, as well as some additional ones. Similarly, OWL Full allows all the valid expressions of OWL Lite and OWL DL, as well as some others. This results in the following conclusions on compatibility:

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

OWL features greater expressiveness than RDF. Some important capabilities not present in RDFS are:

- OWL can define classes and individuals, like RDF(S) can, through its constructs owl:Class and owl:Individual respectively. However, in contrast to RDFS, each property can be either an object property (owl:ObjectProperty), or a datatype property (owl:DatatypeProperty). The object properties can have a class type as range, whereas the datatype properties can only have XML Schema datatypes. The relation of OWL classes and properties to their RDF counterparts can be seen in Figure 3.2.
- OWL can define the complement (owl:complementOf), the union (owl:unionOf) and the intersection (owl:intersectionOf) of classes. Furthermore, two classes

can be defined to be disjoint (owl:disjointWith) or equivalent (owl: equivalentWith).

- Properties can be defined to be inverse (owl:inverseOf) or equivalent (owl: equivalentProperty). Furthermore, properties can have more than one class for domain or range, like the union or intersection of classes.

- OWL can define that a class A fulfills certain restrictions on its properties. This can be done indirectly, by creating a superclass of A (which may be unnamed), and whose instances fulfill the desirable restrictions. Because of the subsumption, A inherits the restrictions. Such restrictions are defined using the expression owl:Restriction. Here is an example of a service that delivering goods in Indonesia:

```
<owl:Class rdf:about="resourceDeliverToIndonesia">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="deliveryLocation">
      </owl:allValuesFrom rdf:resource="LocationInIndonesia">
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

  The above resource description places the restriction on members of the class resourceDeliverToIndonesia such that all values of the property deliveryLocation must be drawn from the set LocationInIndonesia (i.e., in simple English, this says that a necessary condition of membership of this class is that the individual delivers something to Indonesia). A similar restriction to owl:allValuesFrom is owl:someValuesFrom, which is used when at least one value needs to be an instance of a particular class. In addition to restrictions on the range of a property, cardinality restrictions may be defined, which concern the number of values of a particular property that an individual has.

- A property can be defined to be transitive, symmetric, functional or inverse functional (only in OWL Full).

- In contrast to RDF, OWL does not make the Unique Name Assumption. This means that different concepts can be given the name, and that differently named concepts may be the same. Naturally, this increases the complexity of the language.

OWL seems to be a much better language for modeling RFAs than RDF(S). It is capable of richer expression than RDF(S), when describing RFAs. For example, a brokering algorithm must be able to distinguish the RFA descriptions from the available services descriptions. As argued in the next chapter, this may be done by defining two corresponding classes. Obviously, a description can either be an RFA description or an available service description (or neither one). In order to make sure that it is not legal for a profile to be of both types, the two classes must be declared to be disjoint.

Another desirable property for the modelling approach was to be flexible enough to model types of RFAs that will emerge in the future, along with the existing ones. Defining new RFA types will probably involve restricting the range of existing properties. This can be done in OWL, but not in RDF(S) (the combination of RDF and RDF-S).

Another advantage of OWL is that it guarantees decidability and completeness (when OWL Lite or DL is used). This is important, as it is desirable for a brokering approach to be able to cope with large numbers of RFAs and services. If an undecidable ontology language is used, there can be no computational guarantees for large, or even small numbers of RFAs. From the above, OWL seems to meet all the modelling desiderata, and create the preconditions for the brokering desiderata to be met.

3.4 OWL-S

OWL-S [14] is an ontology for the description of web services. It was originally created as DAML-S in 2001 by DARPA[3], and the current version at the time of writing is 1.2. As its name implies, OWL-S is written in OWL. It aims to enable the discovery, invocation, composition and monitoring of services by human and computer agents. OWL-S has three main sub-ontologies: the *service profile*, the *process model* and the *grounding*. Figure 3.3 shows the top level of the ontology. Although OWL-S is primarily intended for describing web services (that is, services that are invokable through the web), many of the notions of a service that it contains are sufficiently general that they

[3]`http://www.darpa.mil/`

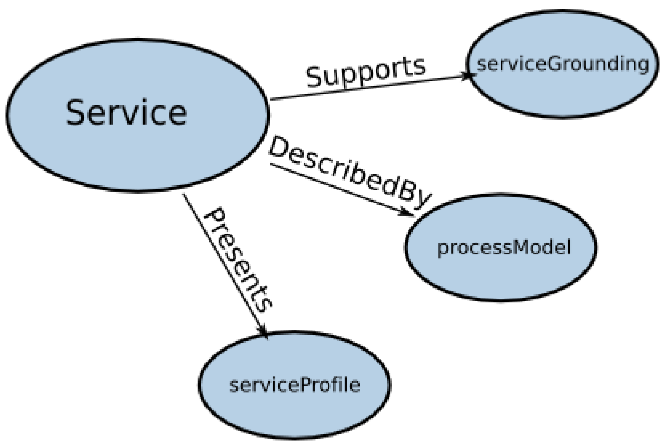


Figure 3.3: The OWL-S ontology

might also be applied to ‘real world’ services. The rest of this chapter will attempt to decide whether OWL-S can be used to describe RFAs in such a way that brokering is possible.

Service Profile : The service profile gives a general description of the service, so that a agent can decide whether the service is suitable for their needs. Typical information that are given by the service profile are the inputs the service needs and the outputs its execution will provide. Furthermore, a classification of the service may be available (using external classification systems), or information about the quality of service.

Process Model : The process model contains information about how the service works. It too describes the inputs and the outputs, but focuses on the stages of execution of the service. The process model is able to describe the execution flow of both atomic (standalone) services and composite ones, and so is intended to be used to interact with the service appropriately during its execution. This information can also be used by interested parties to conduct a more fine-grained decision about whether the service meets their needs, or perhaps to automatically invent a way to combine more than one services into a composite service. The process model also contains information that can be used to monitor the service.

Grounding : The service grounding contains information about how the service can be accessed via the web. Essentially, it binds the abstract definitions given in the process model to concrete implementations. In other words, it attaches software engineering information to the semantic definitions of the model. Information that can typically be found in the grounding may be the address and port where the service can be accessed, the (computational) types the inputs must have, as well as descriptions of any messaging protocols that must be employed.

There have been several attempts to automatically broker services modelled in OWL-S. Although OWL-S has a specific structure, it will be shown that services can be modelled in more that one ways. The following sections will consider different approaches to both modelling and brokering of services using the OWL-S ontology.

Some points that may need to be clarified before these approaches are presented:

- Generally, both service advertisements and requests use the OWL-S profile. Despite this similarity in modelling, they are semantically different. A advertisement profile describes the properties of an existing service, whereas a request profile in effect describes a hypothetical service that *would* meet the requester's needs. However, it is convenient to describe them in a uniform way (the OWL-S profile ontology).

- The fact that the following proposals use the DAML-S ontology and not the OWL-S is not fundamentally important. DAML-S is an older version of the OWL-S, and little has changed as far as the essential organisation and structure of the ontology are concerned; with the appropriate changes to syntax, terminology, etc, the described proposals should work equally with OWL-S.

- The authors conduct matchmaking of Web Services, whereas this project aims at brokering RFAs. However, their work is relevant, as the algorithms they propose use the OWL-S ontology, and at the brokering stage there is little in the service descriptions that relates exclusively to Web Services.

3.4.1 First approach

In [19], the authors use the DAML-S ontology to conduct semantic matchmaking of Web Services. In particular, they make use of the service profile to describe advertisements of and requests for Web Services. They also propose an algorithm which compares the service profiles to determine whether there is a semantic match between an advertisement and a request.

The algorithm proposed by the authors decides whether there is a match between a service advertisement and a service request by attempting to match the inputs and the outputs of the two profiles. More specifically, in order to have a perfect match between an advertisement and a request, for every output the request specifies, there has to be a matching output in the advertisement. Furthermore, for every input the advertisement specifies in order to function, there has to be a matching input the request declares to be available.

From the above description, it can be seen that the comparison of service profiles is reduced to comparing the values of those profiles' properties. The authors have adopted a flexible method of matching property values. They identify a perfect match when the values are identical, but they also identify 'weaker' types of matches, depending on the way the two values are subsumed. In more detail and in order of decreasing 'goodness' of match, these are the types of match that are identified, when comparing outputs (or inputs):

- **Exact :** If the two outputs are equal (or if the two inputs are equal).
- **Plugged In :** If the output of the advertisement subsumes the output of the request (or if the input of the request subsumes the input of the advertisement).
- **Subsumed :** If the output of the request subsumes the output of the advertisement (or if the input of the advertisement subsumes the input of the request).

For every request that is brokered, points are assigned to each advertisement, depending on the number and type of matches between its inputs and outputs and the ones of the request. As a result, not only perfect matches will be returned, but also advertisements that are "sufficiently similar", as described by the authors.

The fact that the algorithm is flexible and identifies different levels of matching is important, and could be added to the desiderata listed in a previous chapter.

However, there are some points that need to be addressed, if this algorithm is to adopted for RFAs. Conducting matchmaking based solely on the comparison of inputs and outputs may be deceiving. This is because there may exist services with the same inputs and outputs, which are completely different. For instance, a service that rents cars and a request for a service that sells cars both may specify the model name as input and return a receipt number as output. The algorithm proposed by the authors would faultily identify a match in this case.

3.4.2 Second approach

Horrocks and Li [13] propose a software framework for semantic matchmaking of services. As in the previous approach, the ontology used is DAML-S and the services are

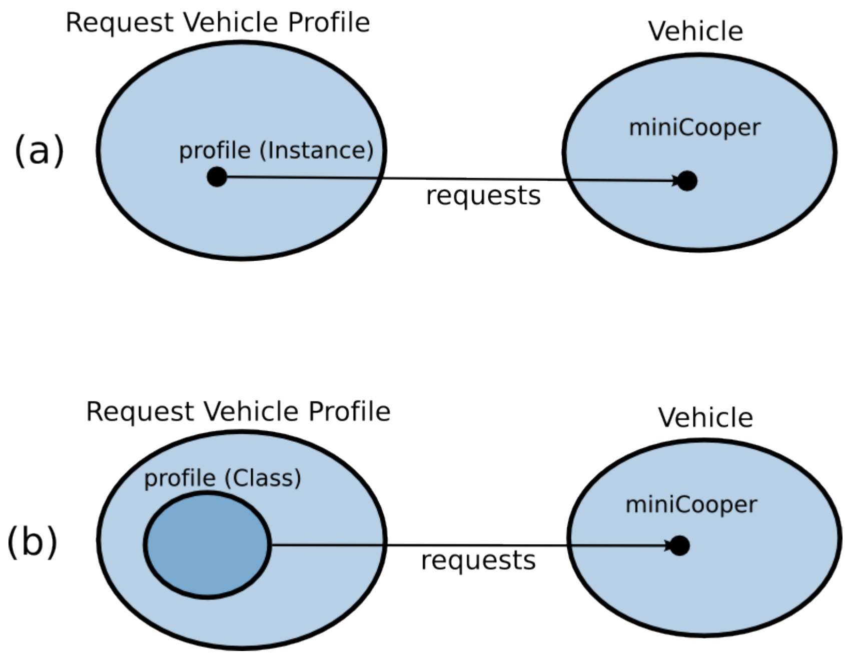


Figure 3.4: Service profiles for Matchmaking

described using the DAML-S profile. However, here the matchmaking is not merely based on inputs and outputs, but on all the properties described in the service profile.

Instead of sequentially comparing all the request properties against the advertisement properties, the comparison takes place at a higher level. For example, a request for a service that provides a Mini Cooper would normally be represented as an instance of the profile class, where the car model input would have Mini Cooper model as range (Figure 3.4.a). Here, the advertisement and request profiles are not represented as instances of the DAML-S profile class, but rather as subclasses of it (Figure 3.4.b). In comparison to the DAML-S profile class, they refine themselves by fixating the values of their properties. Here, the same request is represented as the class of all service profiles, where the car model input has Mini Cooper as range.

Describing services as classes is somewhat contrary to intuition, as both advertisements and requests refer to specific services, and not categories of services. The reason for this rather unorthodox approach is to enable TBox[4] reasoning, which the authors consider to be more effective than ABox[5] reasoning. More specifically, matchmaking can be conducted by detecting equality or subsumption on the entire services themselves, which is much simpler than the previous approach.

[4]TBox is the component of an ontology that contains assertions about concepts and roles.
[5]ABox is the component of an ontology that contains assertions about individuals.

Again, there are types of match other than the exact match. If A is the advertisement profile class and R is the request profile class, the types of (overall, in this case) match that are identified (again, in decreasing order of 'goodness' of match) are:

- **Exact :** If advertisement A and request R are equivalent classes.
- **Plugged In :** If request R is a subclass of advertisement A.
- **Subsumed :** If advertisement A is a subclass of request R.
- **Intersection :** If the intersection of advertisement A and request R is satisfiable.

In this case there is an additional type of match, the intersection match. This match will be detected if an advertisement and a request concepts overlap. It has to be noted that Plugged In and Subsumed matches are expected to be rare, because in order for them to occur, *all the pairs* of corresponding properties have to have the same hierarchical relation (either all advertisement property values to be subsumed by the corresponding request property values for a Subsumed match, or all request property values to be subsumed by the corresponding advertisement property values for a Plugged In match).

The approach followed here does take the service type into account. As all profile properties are used for the comparison, so is the *serviceCategory* property. As a result, two profiles must have the same category value in order to produce an exact match. However, although this will prevent the faulty identification of exact, plugged in and subsumed matches between different services with identical inputs and outputs, it will not prevent the identification of intersection matches. Of course, the Intersection match is the weakest of the types of match, but is still considered to be a match.

The fact that both functional and non-functional properties are used for the comparison entails another implication. Functional properties of a service profile are the ones describing the information needed to actually function the service (like inputs, outputs, preconditions, effects, service categorisation, etc). On the other hand, non-functional are the properties that concern additional information (like the provider of the service, the service name, text descriptions etc). The presence of non-functional properties can reduce an otherwise exact match to a weaker one. For instance, a request for a service is unlikely to specify a desired provider. When this request is compared to an almost identical advertisement, which additionally specifies the actor of the service, there will be a match, but not an exact one.

In conclusion, modeling services as classes has the advantages of making matchmaking much simpler and taking service categorisation into account. It is possible to separate non-functional from functional properties by expanding the DAML-S ontology with additional classes and roles. However, it would be more adequate conceptually to model service descriptions as individuals. For this reason, the approach will not be used.

Chapter 4

Modelling RFAs / Services

The OWL-S ontology was created primarily for Web Services. Although it appears generic enough to be used for describing and brokering RFAs and relief services, their diversity and nature required some adjustments. This chapter describes a series of design issues that arose when modelling RFAs and services as OWL-S profiles, as well as the decisions that were taken to resolve them.

4.1 OWL-S Profile: Instances Vs. Classes for RFAs / Services

The previous chapter concluded with an examination of two different approaches for using OWL-S to model services, which can be considered to apply to RFAs and services as well. One option is to model RFAs/services as instances of the profile class, as proposed in the first approach (section 3.4.1). The second option is to model RFAs/services as subclasses of the profile class, as proposed in the second approach (section 3.4.2).

One of the drawbacks of the second approach was that non-functional information (service name, service provider, test description etc.) would interfere when subsumption was detected between profiles. However, the brokering algorithm can be modified to overcome this problem. Specifically, a solution would be to detect subsumption between selected properties of the profiles, rather than the profiles themselves. However, brokering based on the subsumption of profiles, which seems to have been the reason profiles were modelled as classes in the first place, will no longer be the case.

In addition to its brokering problems, modelling RFAs/services as classes does not seem conceptually correct. Semantically, a service advertisement refers to a particular service, with specific parameters and properties. One could argue that a profile could be regarded as the class of all individual invocations, with particular inputs, outputs etc. However, such an interpretation would collide with the OWL-S definition of the profile.

Modelling RFAs/services as instances of the profile class seems correct from a semantic point of view. However, this comes at the cost of inventing and using a fairly complicated brokering algorithm to navigate through the profiles and compare corresponding properties. On the other hand, greater control would be available over the way the comparison is made. Specifically, certain types of parameters could be compared differently than others, in a way defined by the user. This could enable the detection of matches that would be impossible when relying on mere subsumption.

Based on the above discussion, it was decided to model RFAs and services as instances.

4.2 Inputs-Outputs Vs. Preconditions-Effects

An important issue concerned the question of which properties of the OWL-S Profile should be used for matchmaking. The first approach that was considered used only the inputs and outputs, whereas the second approach used only the functional (inputs, outputs, service catogory etc.) properties and excluded the non-functional ones (service name, service provider, test description etc.). A factor that may have to be taken into account is that RFAs and available services are primarily services in the physical world, whereas OWL-S was developed primarily to describe Web Services.

An available service will most naturally need some information in order to function correctly. For instance, a service transporting air cargo will need to know the departure airport and the destination airport, as well as the date the transportation needs to take place. This kind of information can be provided to the service in the form of inputs. In turn, the service may provide the invoker with a receipt number, or a unique transaction identifier. This information can be returned in the form of service outputs. Contrast this with, say, a Web Service that computes the square of a number. In that case, the number would be naturally modelled as a input, and the result of squaring it as the output, since this is what the service *actually* does when invoked. In the ‘real’ physical world of RFAs, however, does not really make sense to think of, say, the intended destination as the output of the service, but rather as the physical outcome of the service.

There is more functional information to an available service than inputs and outputs. That information concerns the conditions that need to be met for the service to be invoked, and the impact the execution of the service will have on the world. In OWL-S, this kind of information can be expressed through the *preconditions* and *effects* of the profile class. Preconditions and effects are represented as logical formulas, and can be written in a number of languages, either XML-based (SWRL [11], RDF), or not (KIF [7], PDDL [16]). In the case of the transporting service, a precondition could state that the location of the cargo before the execution of the service is the same as the place of the departure input. Similarly, an effect could state that, should the service be executed, the location of the cargo will then be the same as the intended destination input.

It should be clear that the set of inputs, outputs, preconditions and effects (together called IOPEs in OWL-S) is very important to determine the functionality of a service. The inputs and outputs represent the change of data the execution of the service will cause, whereas the preconditions and effects represent the change in the state of the world. Matchmaking services using both inputs/outputs and preconditions/effects seems to be promising, as services are compared in a more complete way.

Despite the fact that the use of preconditions and effects contributes to a precise description of a service, they would further complicate the brokering process. OWL-S is written in OWL, and as a result reasoning about it requires reasoning about Description Logics, a task which is known to be decidable. On the other hand, preconditions and effects may be written in one of several languages, each corresponding to a different logic which has its own computational properties (not necessarily including decidability) and reasoners. When implementing a service broker, properties would be matched using the DL reasoner, whereas preconditions and effects would need to be extracted and, then separately processed using a different type of reasoner. Combining the two parts of the algorithm to an overall matchmaking algorithm could be overly complicated.

The conclusion is that although preconditions and effects contribute to the completeness of a service description, they are hard to use when matchmaking and brokering of RFAs and corresponding services are concerned. For this reason, it was decided to only use inputs, outputs and the rest of the profile properties.

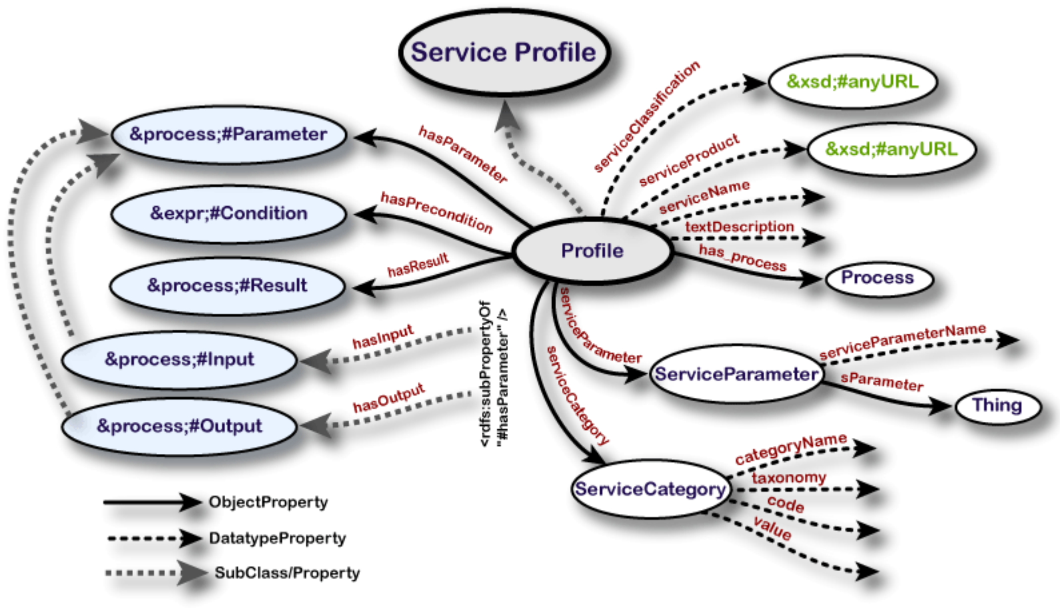


Figure 4.1: Selected classes of the OWL-S Profile

4.3 Categorization of Services

When the first approach was being considered in Section 3.4.1, it was pointed out that services of different types may have identical inputs and outputs and yet may not be a match. Therefore, matchmaking and brokering cannot rely solely on the comparison of inputs and outputs.

One way of overcoming this problem is through the use of preconditions and effects to constrain further the relationship between inputs and outputs; but their use leads to the problems outlined earlier in this chapter. Another approach is to pre-define service categories, where members of a category all conform to some preconceived notion of the behaviour of services of that type. In order to determine the category of a service in OWL-S profile, one can use the property *serviceCategory* (Figure 4.1, from [14]). The serviceCategory receives as range an instance of the ServiceCategory class, which describes the category of the service based on some specific taxonomy. Specifically, an instance of the ServiceCategory class represents a specific service type (i.e. on-line flower seller) in a specific taxonomy.

The taxonomies most frequently used are NAICS[1] [6] and UNSPSC[2] [22]. These taxonomies are not related to OWL-S, and are not written in OWL or any other for-

[1]http://www.census.gov/epcd/www/naics.html
[2]http://www.unspsc.org/

mal ontology language. This means that any semantic subsumption between types of services, like *transporting service* and *personnel transporting service* will not be reflected on the ServiceCategory instances. In fact, the two instances will only differ in the *code* and *categoryName* properties, where two unequal string identifiers will be used. Although it can be determined whether services are of the same type, this can only happen if both profiles are categorized using the same taxonomy. In conclusion, the use of the serviceCategory property does not seem to be intended for functional description, but rather as complementary information, which is outside the scope of OWL-S.

The problem of classifying services in a way convenient for brokering is solved in this case by defining a taxonomy of service profiles in OWL. In this way, in order to ensure that only services of the same type are matched, a brokering algorithm can only search for available services of the same (or subsuming) profile type as the RFA. For instance, the OWL-S profile could have the subclass *transportationProfile* for profiles of transportation RFAs and services. When such an RFA would be brokered, the candidate service profiles would also have to be members of the transportationProfile class.

Another categorization is needed to distinguish between RFA profiles and available services profiles. Like before, there were defined two more subclasses of the OWL-S profile class: the rfaProfile class and the resProfile class, corresponding to RFA profiles and resource (available services) profiles respectively. Furthermore, the two classes were defined to be disjoint, so that a profile can be either an RFA profile or service profile, but not both. (The fact that disjointness cannot be expressed in RDFS provides some justification for the use of OWL.)

Some additional classes were defined to combine the two profile hierarchies. For instance, the rfaTransportProfile is an RFA profile requesting a transporting service. In analogy, a resProvisionProfile is an available service providing a certain good. It should be noted that the definition of combinational classes was not mandatory, as OWL supports multiple class membership for individuals. In other words, a profile individual can be declared to be a member of both the rfaProfile class and the transportProfile class.

4.4 Inputs

Both matchmaking approaches that were considered in 3.4 seem to be using any class as Input or Output, without restriction. This would be very convenient, as the classes of two instances could be directly compared to detect any subsumption. However, the hasInput property of the OWL-S profile is defined to range over instances of the class Input, or XML Schema datatypes. Consequently, for every non-datatype input, the range has to be member of a class, that is subclass of the Input class. A simplistic approach would be to define Location, Weight, Good, or any other class to be subclasses of the Input class. However, this would not be correct for several reasons:

- Intuitively, locations, weight measurements and goods are completely different concepts to inputs. Expressing that a service has an input of type Location means that the *actual location* is used for input, which does not make sense.
- It will not be possible to define outputs of the same type as inputs, because the Output class is disjoint to the Input class.

The problem was resolved by defining specialised subclasses of the Input class, like LocationInput, WeightInput and GoodsInput, to act as mediators between the profiles and the actual concepts. In addition, corresponding subclasses of the Output class were created to be used as outputs (LocationOutput, WeightOutput, GoodsOutput). Only the above three sub-types of Input were defined, because they are the most commonly used by actual RFAs/services. However, this is by no means restricting, as any service profile can define and use its own types of Input and Output.

There will now be an examination of the type-specific issues that emerged when defining the three types of inputs.

4.4.1 Location Inputs

In accordance to the above discussion, the LocationInput class has the property hasLocation to associate the input to a location. However, there is more information about location inputs than just their location. This becomes apparent in the case of services that receive more than one location input. When matchmaking such services based on their Inputs, false matches may be proposed. For example, a location input referring to the destination point of an RFA may be mistakenly matched to the departure point of an available service. Although the algorithm would consider this correct (inputs of the same type are matched), the two location inputs refer to different parameters.

The above problem indicates that the two types of location inputs (departure and destination) must somehow be distinguished. For guidance at this point, we may turn to the "Bravo Air Service"[3] profile was advised, The Bravo Air Service profile is one of the example profiles published along with every version of OWL-S, and, in this case, one that should be especially relevant since it too describes a 'real world' physical airline transport service (albeit an imaginary one). Consequently, the way departure and destination location inputs are distinguished in this profile might be that thought the most appropriate by the authors of OWL-S. Interestingly enough, they too have defined a specific type of Location Input. However, they do not distinguish between the two location inputs of the profile. Instead, they seem to rely on the names of the location inputs (airport_in and airport_out) to determine which one of the two inputs refers to the destination and which to the departure point.

This solution is not applicable when matchmaking, as it would require the brokering algorithm to rely on the names of inputs to determine their identity, and moreover, that the incoming queries use the same labels to distinguish their inputs. Besides the fact that naming conventions may be too restricting, the brokering algorithm would essentially be using string comparison to identify matches: this would be syntactic, and not semantic matchmaking, undermining the use of OWL-S in the first place. Relying on syntactic matches among the names or part of the names for certain inputs should be avoided.

Another solution that was considered would be the further subclassing of the LocationInput class to DepartureLocationInput and DestinationLocationInput. However, this was not considered to be a good solution either, as unspecified location inputs (which could arise, for example, for the specification of a service branch by location) could be matched to any type of location input, possibly producing false matches.

In order to solve the problem, the hasDirection property was added to the LocationInput class. This property has as range the Direction class, which has two instances: destinationPoint and departurePoint. By setting both the hasLocation and the hasDirection properties of a LocationInput, there can be no ambiguity between destination and departure points.

[3] `http://www.ai.sri.com/daml/services/owl-s/1.2/examples.html`

4.4.2 Goods Inputs

Provision services need to be invoked with the type of good that need to be provided, and the quantity of this good. As provision RFA profiles may contain requests for multiple goods, it was considered a good strategy to have one type of input to determine both the type of good and the quantity. This input is the GoodsInput, which has two properties: the requestGood to determine the good and requestQuantity to determine the quantity.

One important issue was the semantics of the requestQuantity property. Specifically, RFAs usually request an exact or minimum number of items. Although descriptions of provision services were not available to us, it seems reasonable to assume that they would specify the maximum number for items they can provide. In some cases, they could specify both a maximum and a minimum number of items (provision of small orders may be too costly). Obviously, the requestQuantity property should be somehow refined to be able to handle all types of orders.

Like in the LocationInput case, creating subproperties of requestQuantity was not considered to be a good strategy. Instead, a more generic solution was needed, to apply for other types of inputs where exact, maximum and minimum values may needed to be used (like the weight inputs). The problem was solved by creating the class BoundInput (subclass of Input) to describe inputs that specify a numeric value. In turn, three subclasses of the BoundInput class were defined: ExactBoundInput, MaximumBoundInput and MinimumBoundInput, for inputs specifying exact, maximum and minimum values, respectively.

An RFA requesting at least 3 water pumps will have a goods input that is a member of both GoodsInput and MinimumBoundInput. Similarly, a provision service that can deliver at most 5 water pumps will have a goods input that is a member of both GoodsInput and MaximumBoundInput. For convenience, additional classes were created to combine the GoodsInput class and the BoundInput hierarchy, like MaxGoodsInput, or MinGoodsInput.

The ExactBoundInput class was defined as the intersection of MaximumBoundInput and MinimumBoundInput. This way, an RFA requesting exactly n items will be matched by both a service providing at most n items and a service providing at least n items.

4.4.3 Weight Inputs

The participants in a relief effort usually originate from several countries. Part of the confusion this may cause concerns the different measurement systems used. For instance, an input that is commonly used for transporting services is the weight input. In order to avoid ambiguity, both weight measure and weight unit should be specified. This is the reason the WeightInput has two properties: the hasWeight and the hasWeightMeasureUnit. The first one takes as a value a non-negative integer, and the second takes an instance of the WeightMeasureUnit class.

Like in the goods inputs, there are cases where maximum, minimum or exact weights must be specified. For instance, a transporting service may have the capacity to carry cargo weighing at most 10 tons. On the other hand, RFAs usually request an exact number of specific items which, in turn, has an exact weight. In order to solve the problem, the generic classes MaximumBoundInput, MinimumBoundInput and ExactBoundInput were used one more. For convenience, some additional classes were created to combine the WeightInput class and the BoundInput hierachy, like the MinWeightInput and the ExactWeightInput.

4.5 Using XML Literals for Resources

In general, services do not provide a specific good (a specific item), but rather unspecified goods of a certain type. In accordance, an RFA will most probably request a type of good and not a particular item. Specific goods ontologically correspond to instances of the class Good. However, in the case of an unspecified good of a certain type, the important information is the type of the good, which ontologically corresponds to a class. The same applies with locations, as a service may be able to transport to either a particular location (instance of the Location class) or any location of a wider area (subclass of the Location class). Consequently, there are cases where an instance of the Input class needs to be associated with a class.

However, OWL ABox statements only relate individuals to individuals. In practice, this means that there cannot be an RFA where the requestGood property of a GoodsInput can point to the class of vehicles. This issue applies for other types of inputs as well, like location inputs. For example, a transporting service may be able to deliver to any place of a country, which conceptually corresponds to the class of all locations in the particular country.

In order to solve the problem, there is need for a uniform way to refer to either instances or classes, which in addition could be used as a range to a property. The solution adopted for this problem uses the fact that both instances and classes are uniquely identified by URIs. So, if the properties in question are defined to have range of type URI (which is an XML Schema datatype), the problem can be solved.

There are some shortcomings to this approach. An input is allowed to have any valid URI as range, and not only URIs that belong to particular classes or instances. In other words, valid OWL-S profiles may be defined, which are semantically incorrect. However, the above method was finally adopted, mostly due to time restrictions which did not allow a more elegant solution to be found.

4.6 The RFA domain ontology

Most of the above issues can be resolved by extending the original OWL-S ontology with additional classes and properties. In addition, service profiles need to refer to concepts outside the scope of OWL-S. For those definitions, an RFA domain ontology is needed.

The purpose of the RFA domain ontology is not to define every possible class or property that might be needed to describe an RFA. Such effort would be futile, given the diversity of the RFAs/services. The RFA domain ontology must only define the class hierarchies that are needed for the examples created for this project, as well as the extensions to OWL-S, which are generally used by all RFA and service profiles. However, the extent of the RFA domain ontology is by no means limited. As an OWL ontology, it will be able to be imported, extended and used by any OWL document, allowing new types of RFAs/services to be modelled.

4.7 An RFA profile

Following all the above design decisions, we propose RFAs and services to be modelled as OWL-S profiles, with their parameters being modelled as inputs, outputs and other properties. Figure 4.2 shows an RFA modelled in the proposed method. The rfaProfile is an instance of the class rfaProfile, which is a subclass of the OWL-S Profile. The rfaProfile has several properties, both functional (hasInput) and non-functional (hasName, textDescription). As far as the functional properties are concerned, two of the inputs are instances of the class LocationInput, which is a subclass

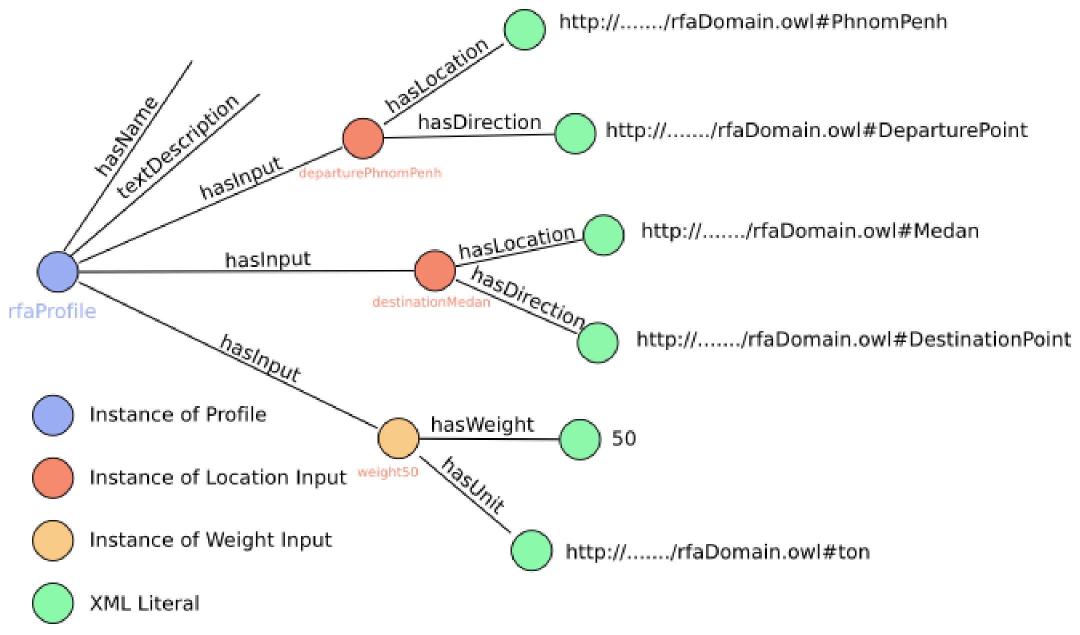

hasName
textDescription
hasInput
hasLocation
http://......./rfaDomain.owl#PhnomPenh
hasDirection
http://......./rfaDomain.owl#DeparturePoint
departurePhnomPenh
rfaProfile
hasInput
hasLocation
http://......./rfaDomain.owl#Medan
destinationMedan
hasDirection
http://......./rfaDomain.owl#DestinationPoint
hasInput
hasWeight
50
weight50
hasUnit
http://......./rfaDomain.owl#ton
Instance of Profile
Instance of Location Input
Instance of Weight Input
XML Literal

Figure 4.2: An RFA profile

of the OWL-S Input, and another input is instance of the class MaxWeightInput, which is also a subclass of the OWL-S Input. As described above, the first of the two location inputs is about the departure point and the other is about the destination point. Likewise, the weight input describes both the weight value and the weight unit. Finally, as explained above, XML literals are used to contain the URIs of resources.

Chapter 5

Brokering RFAs

The purpose of this chapter is to propose an algorithm that is capable of brokering RFAs described in OWL-S in the manner outlined in the previous chapter. The brokering algorithm is able to identify available services that can independently satisfy an RFA. Besides this type of “direct brokering”, the algorithm is capable of proposing orchestrations of services, for satisfying transportation and provision RFAs. In this case, sequences of services, called *chains*, are proposed to satisfy cooperatively an RFA.

5.1 Direct Brokering

Whenever an RFA profile is submitted, the brokering algorithm will look for its type (the class of which the profile instance is a direct member). In the case where this is neither a transportation nor a provision RFA, there will only be direct brokering attempted, that is, looking only for services that independently meet the requirements of the RFA.

The first step is to filter out the available services of different type. This is done by only comparing the RFA profile to ones whose type is a superclass of the type of the RFA. Generally, in order to identify a match between an RFA profile and an available service profile of the same type, there has to be a correspondence between their functional properties. The only functional properties used by the brokering algorithm are inputs and outputs. In order to identify a match, the following two conditions must be satisfied:

1. For every input the available service needs, there has to be a matching input in the RFA.

2. For every output the RFA requests, there has to be a matching output in the available service.

The brokering algorithm slightly differentiates itself for inputs and outputs. For the sake of simplicity, only the case of inputs will be thoroughly analysed here, but the case of outputs is analogous. The inputs of OWL-S profiles are instances of the class Input. The first condition to identify an "input match" between an input of the RFA profile and an input of an available service profile, is for the two inputs to be of the same type (e.g. location inputs). The second condition concerns the properties of the two inputs. In order to identify an input match, there has to be a "property match", for all the properties of the RFA inputs.

The property values of the RFA input must be either equal to or subsumed by the values of the available service input. Intuitively, this means that the RFA is requesting something equal, or 'contained within' what the available service can provide. However, the property values of the profile inputs may be either instances or XML literals, where subsumption is not defined. Depending on whether the inputs are instances or different types of XML literals, there are the following cases where two properties match:

1. **Property values are instances :** A property match is identified only if the two instances are equal.

2. **Property values are URIs :** The URIs should either represent an instance or a class. A property match will be identified in the following cases:

   - If both the RFA URI and the available service URI identify instances, which are equal.
   - If the RFA URI identifies an instance *i*, the available service URI identifies a class *C*, and *i* is a member of *C*.
   - If the RFA URI belongs to a class C_1, the available service URI belongs to a class C_2 and C_1 is a subclass of C_2.

3. **Property values are integers :** The criteria for a match depend on the kind of input (whether it declares minimum, maximum or exact value). Specifically:

   - In the case where the RFA input is of class *MaximumBoundInput*, then there will be a property match if the value of the corresponding available

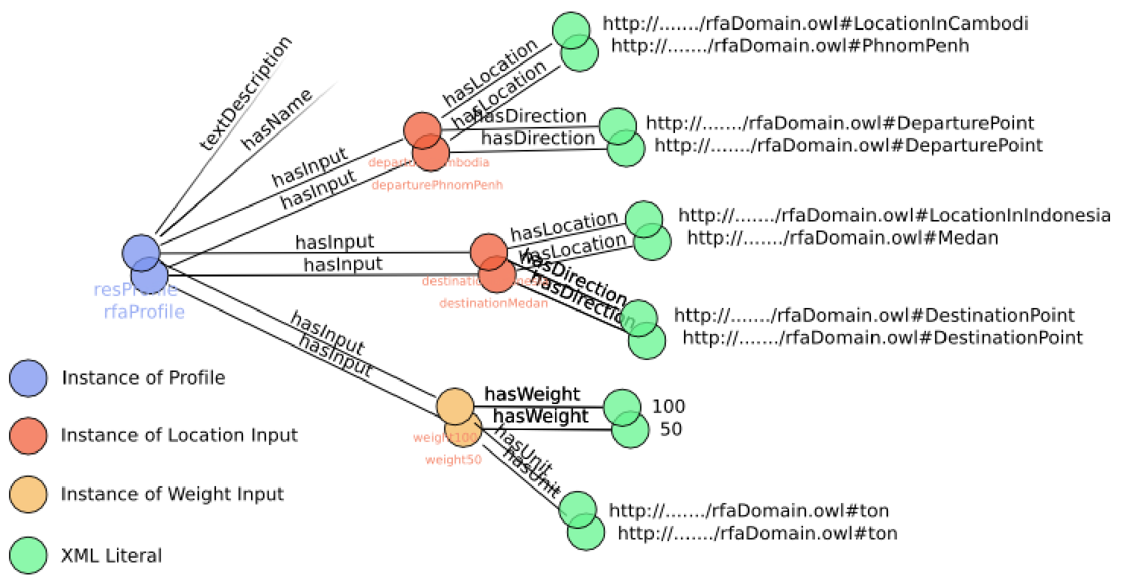

textDescription
hasName
hasInput
hasInput
hasInput
hasInput
hasInput
hasInput
resProfile
rfaProfile
hasLocation
hasLocation
hasDirection
hasDirection
departurePhnomPenh
destinationMedan
weight50
http://......./rfaDomain.owl#LocationInCambodi
http://......./rfaDomain.owl#PhnomPenh
http://......./rfaDomain.owl#DeparturePoint
http://......./rfaDomain.owl#DeparturePoint
http://......./rfaDomain.owl#LocationInIndonesia
http://......./rfaDomain.owl#Medan
http://......./rfaDomain.owl#DestinationPoint
http://......./rfaDomain.owl#DestinationPoint
hasWeight
hasWeight
100
50
hasUnit
hasUnit
http://......./rfaDomain.owl#ton
http://......./rfaDomain.owl#ton
Instance of Profile
Instance of Location Input
Instance of Weight Input
XML Literal

Figure 5.1: Direct Matchmaking of an RFA showing the matches among the individual terms. The 'lower' of the two profiles is the RFA, and the 'upper' is the matching available service.

service is the same or greater, and the class of that service input is MaximumBoundInput.

- In the case where the RFA input is of class *MinimumBoundInput*, then there will be a property match if the value of the corresponding available service input is the same or less, and the class of that service input is MinimumBoundInput.
- In the case where the RFA input is an *ExactBoundInput*, then there will be a property match in the following cases:
  - If the value of the corresponding available service input is the same and the class of that service input is ExactBoundInput.
  - If the value of the corresponding available service input is the same or greater and the class of that service input is MaximumBoundInput.
  - If the value of the corresponding available service input is the same or less and the class of that service input is MinimumBoundInput.

As far as the outputs are concerned, the brokering algorithm is slightly different. In order to identify a property match of two outputs, the property of the available service value must subsume the RFA value, and not the other way around.

An example of an RFA profile and a matching available service profile can be seen in Figure 5.1. The two profiles match because for every input of the RFA there is a matching input in the available service. The departurePhnomPenh input of the RFA profile matches the departureCambodia input of the available service firstly because they are of the same type (location inputs). Secondly, their URI values in the hasDirection property are equal (they both identify the departurePoint instance of the Direction class). Finally, the RFA term indicated by the URI value in the hasLocation property subsumes the term indicated by the URI value for the same property in the available service: in this case, the RFA domain ontology defines phnomPenh as an instance of the class LocationsInCambodia. Similarly there is a match for the other location input (locationInMedan).

Regarding the weight inputs, the weight50 input of the RFA matches the weight100 input of the available service because firstly, they are both of the same type (maximum weight inputs). Secondly, the terms indicated by their URI values in the hasUnit property are equal (they both identify the ton instance of the WeightUnit class). Finally, the RFA integer value in the hasWeight property is smaller than the available service integer value in the same property (MaxWeightInputs are defined as MaximumBoundInputs).

5.2 Brokering Transportation RFAs

Transportation RFAs are one of the two types of RFAs that can be satisfied both directly and indirectly. In the case of transportation RFAs, an indirect match would be a "chain" of transportation services that, when invoked sequentially, can satisfy the RFA cooperatively. These types of ordered chains will be called transportation chains. The brokering algorithm attempts to search for these chains irrespective of whether any direct matches have been found or not.

In order to propose orchestrations of transportation services, the brokering algorithm is based on the structure of this particular type of RFAs and services. Specifically, it follows the convention that a transportation RFA has at least three inputs: a departure location input, a destination location input and a weight location input, describing the departure point, destination point and weight of cargo, respectively. Based on these three inputs, any matching chain of transportation services must have the following characteristics:

1. The destination input of the RFA must be matched by the destination input of the last transportation service in the chain.

2. The departure input of the RFA must be matched by the departure point of the first service in the chain.

3. For any two adjacent transportation services, the destination point of the first one must be matched by the departure point of the second one (i.e., the first one must arrive at the location from which the second one leaves).

4. Every one of the transportation services of the chain must have a weight input that matches that specified in the RFA.

The brokering algorithm uses recursion to detect all the matching transportation services and chains. In each run, the algorithm is given a departure location, a destination location and a weight, and returns the list of all services or chains of services that are capable of the transportation. There are two steps: In the first step, the algorithm will detect any directly matching services, and add them to its list of solutions. The second step is more complicated. Essentially, the algorithm will look for all the services that can deliver the cargo to the requested location, and then, for each one of them, it will find all the ways to transport the cargo to a place where it can be picked up. In practice, the algorithm will select a service that can make deliveries of the specified weight to the specified destination point, and then call itself to locate any services or chains of services that can transport the cargo to the departure point of the selected service. There are checks to ensure that a service can only occur once in each proposed chain. The solutions of the second step of the algorithm will eventually be added to the list of solutions, and returned.

Intuitively, when the algorithm first runs, it ‘begins’ from the RFA destination point and progresses its way to the RFA departure point. It can be seen from the description above that the search for chains is exhaustive. The algorithm will end after all available transportation services have been tested for a direct or orchestrated match. After the execution of the algorithm, every possible chain of transportation services, as well as every direct match to the RFA will be returned.

An example of a RFA profile and a matching transportation chain can be seen in Figure 5.2. For the sake of simplicity, the properties of the profile Inputs are not included. The RFA requests the transportation of cargo weighing exactly 50 tons from London to the Polonia International Airport (located in Medan, Indonesia). Having

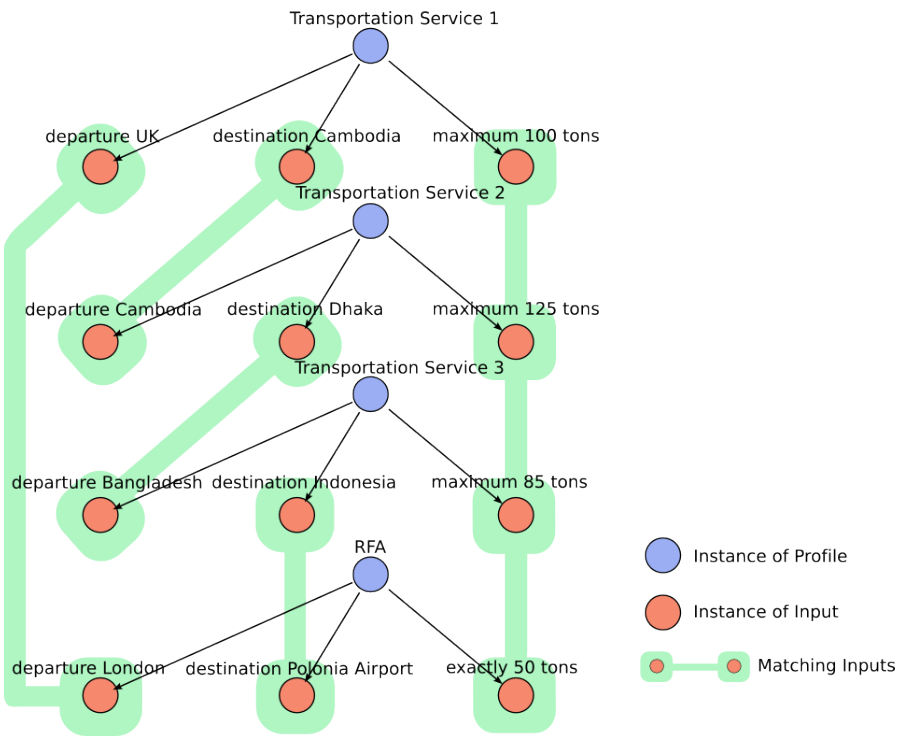


Figure 5.2: Orchestrating Transportation Services

failed to find a service that directly matches this request, the algorithm will begins to search for service chains by first finding a services reaching the Polonia airport. Such a service is the "Transportation Service 3", which can make deliveries to Indonesia and, hence, to the Polonia airport. If no such service was found, the algorithm would terminate - the RFA can not be satisfied using the current set of available services. Furthermore, the service is capable of transporting cargo of maximum 85 tons and, thus, is capable of transporting the requested 50 tons. If the service also had a departure point that 'included' London, that would have been a direct match. However, the departure point can be any place in Bangladesh.

The algorithm will continue by next finding services that reach any place in Bangladesh. Such a service is the "Transportation Service 2", which can make deliveries to Dhaka (located in Bangladesh), and this service also respects the weight restrictions, carrying weights of maximum 125 tons. Again, the service does not have a departure point that 'includes' London. The algorithm will continue by finding a services that makes deliveries to Cambodia. Such a service is "Transportation Service 1", that reaches any location in Cambodia. The service respects the weight restrictions, carrying weights of maximum 100 tons. Finally, the service can have any departure point in the United

Kingdom, including London. This concludes the discovery of a chain of transportation services - these three services, when used sequentially, appear to meet the original RFA.

5.3 Brokering Provision RFAs

Provision RFAs are the second type of RFAs that can be matched both directly and indirectly. An indirect match would be a chain of services, where the first service is capable of providing the requested good and the remaining ones can transport it to the destination location of the RFA through intermediate locations, where necessary. Essentially, these provision chains consist of a provision service, followed by a transportation chain.

Again, in order to propose orchestrated solutions, the structure of provision RFAs was taken into account. Typically, a provision RFA has two inputs: a goods input and a location input, describing the good type and quantity, and the destination location at which it is required, respectively.

A problem that emerged concerned the fact that in order to determine whether a transportation service is capable of a particular transportation or not, the weight of the cargo needs to be known. However, as stated in a previous chapter, this information may not be available to the person submitting the RFA. This is also reflected in the way provision RFAs were modelled: they do not have a weight parameter that could be used to determine which transportation services can be used.

In order to solve this problem, the algorithm will search the domain ontology for the weight of the good in request, and will compute the total weight of the cargo by multiplying the weight of a single item by the quantity that is requested. Once this is done, a virtual weight input can be composed to be used as a matching target to determine which transportation services can be used. This, in a way, “enriches” the RFA with additional information, as was proposed in an earlier chapter. Consequently, the use of different provision services (whose products have different weights) may result in the selection of different transporters and, ultimately, different chains.

Based on these inputs, a matching chain of services should have the following characteristics:

1. The destination input of the RFA must be matched by the destination input of the last transportation service of the chain.

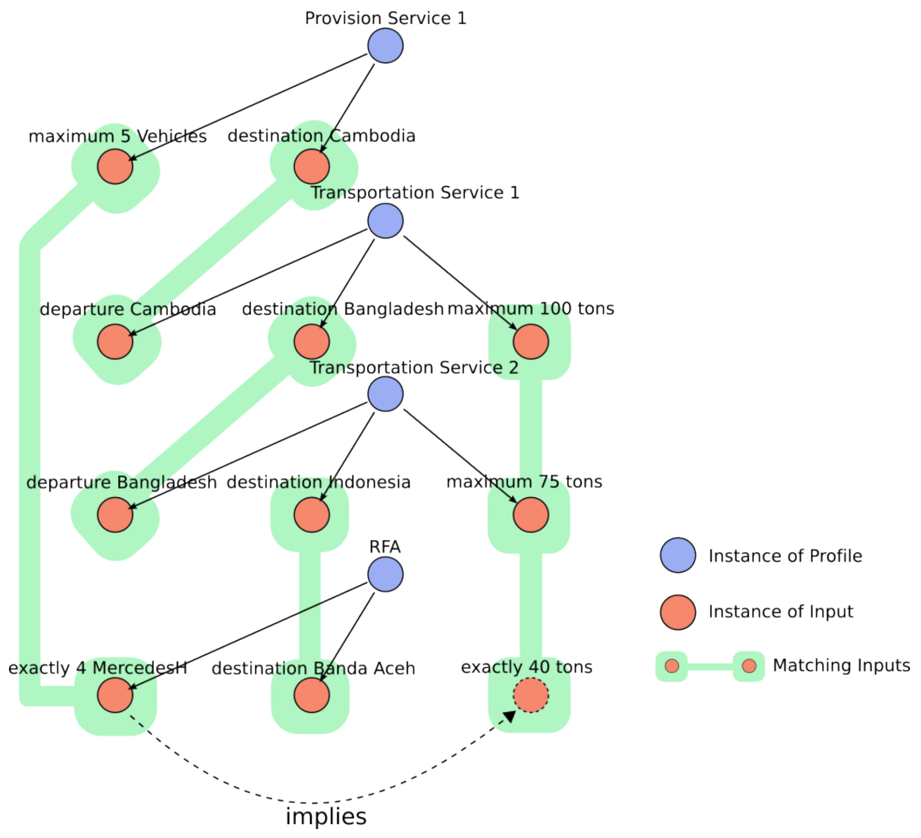


Figure 5.3: Orchestrating Provision and Transportation Services

2. The goods input of the RFA must be matched by the goods input of the first service of the chain.

3. For any two adjacent transportation services, the destination point of the first one must be matched by the departure point of the second one.

4. Once the virtual weight input is composed, every one of the transportation services of the chain must have a matching weight input to the virtual weight input of the RFA.

In the case of provision RFAs, the brokering algorithm begins by finding all the available services that can provide the good requested by the RFA. For every one of those, it checks for a direct match (in the way described earlier), or for a transportation chain to the RFA destination. A virtual weight input will be composed as described above and used for the detection of possible transportation chains. The provision service is then appended to the beginning of every transportation chain detected, with the resulting sequence of services then returned as a matching provision chain.

An example of a RFA profile and a matching transportation chain can be seen in Figure 5.3. The RFA requests for exactly 4 Mercedes H trucks (this is a hypothetical model) to be delivered to Banda Aceh (located in Indonesia). The algorithm will begin by finding a service that can provide the requested quantity of goods. Such a service is “Provision Service 1”, which can provide and deliver a maximum of 5 vehicles (including trucks), to any place in Cambodia. As Banda Aceh is not in Cambodia, the algorithm will next try to find a transportation chain from Cambodia to Banda Aceh. The virtual weight input that must be matched will be composed as described above. The Mercedes H model of the example weighs 10 tons. Consequently, the total weight of exactly 4 Mercedes H trucks is exactly 40 tons. Note that the virtual weight input does not belong to the RFA profile, but can be considered to be implied by this profile, and so is used as if it were part of the profile.

The algorithm will continue by finding a service reaching Banda Aceh. Such a service is “Transportation Service 2”, which can make deliveries to Indonesia and, thus, to Banda Aceh. The service is capable of transporting cargo of maximum 75 tons and, thus, the 40 tons of Mercedes trucks. If the service had a departure point that was in Cambodia, that would conclude the provision chain. However, the departure point can only be in Bangladesh. The algorithm will carry on by finding a service that reaches any place in Bangladesh. Such a service is “Transportation Service 1”, which also respects the weight restrictions, carrying weights of maximum 100 tons. Furthermore, the departure point of the service can be any place in Cambodia, which matches the delivery point of the provision service. This concludes the discovery of a chain of provision and transportation services.

Chapter 6

Implementation: The RFA Broker

In an attempt to find the extent to which brokering RFAs is feasible, the brokering algorithm proposed in the previous chapter was implemented in an application named *RFA Broker*. The RFA Broker can download service and RFA profiles from user-specified URLs, and then broker them using the algorithm proposed in a previous chapter. After the end of brokering, it proposes all services or combinations of services that can satisfy the brokered RFA.

The system was implemented using the Java programming language. The most important reason for this choice is that Java features a multitude of APIs and frameworks for managing and processing OWL ontologies. Furthermore, an object-oriented language facilitates the partitioning of the system into separate interacting modules. In turn, this contributes to the development of a structured system, easier debugging and code re-use. By using Java the same code can run on several platforms, without significant adjustments and additional compilations. Finally, a number of robust Integrated Development Environments (IDEs) exist, that enable faster development. In particular, the RFA Broker was implemented in Netbeans[1].

One aim of implementation was the separation of functionality from the graphical interface. This enables the use of the brokering back-end with future interfaces that may be developed. The architecture of the overall system can be seen in Figure 6.1. The brokering system comprises two parts: the RFA Broker and a graphical interface, through which the user controls the brokering process. The RFA Broker uses the Jena framework to manage the ontologies containing the RFA and service profiles.

[1]`http://www.netbeans.org/`

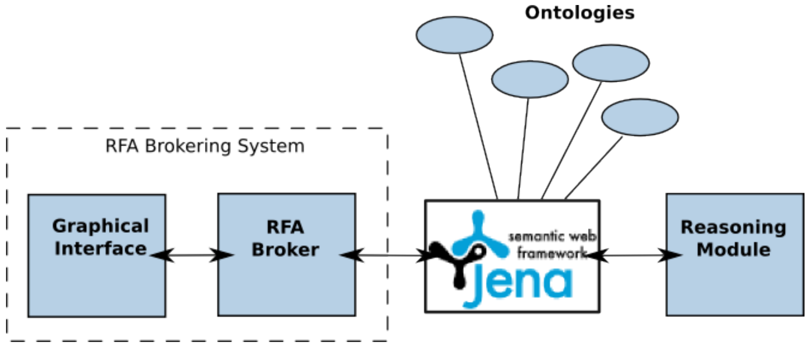


Figure 6.1: The RFA Broker architecture

6.1 Jena Framework

Jena[2] [15] is a framework for the deployment of applications for the Semantic Web [3]. Jena is an open-source project and is being developed by Hewlett-Packard. It is written in Java and, as of the time of writing, its current version (and the one used for the implementation) is 2.4. Jena features several APIs, each enabling different functionalities.

RDF API: Jena features an RDF API to create and manage RDF documents. The RDF documents are internally represented as graphs, called *models*. Models are created through a "model maker" called model maker. RDF resources and properties are represented in the model as nodes and edges, respectively. Using the RDF API methods, the user can create new models or modify existing ones. There is also support for operations such as the union, intersection and difference of models. Finally, Jena features an implementation of the SPARQL [23] querying language, which can be used to query models.

Ontology API: The Jena ontology API is built on the RDF API and supports ontology languages such as RDFS, OWL and DAML+OIL [10]. A special type of model, the *ontology model*, is used to handle documents of these languages. The ontology model is also represented as a graph, whose nodes are ontology resources (generally instances or classes) and edges are ontology properties (datatype properties or object properties).

Inference support: Jena also features a reasoning module, which can be used through the ontology API. The Jena internal reasoner supports RDFS and OWL Lite reasoning. In the case where reasoning support is required for OWL-DL, any external

[2]http://jena.sourceforge.net/index.html

Description Logics reasoner implementing the DIG [2] interface can be used (i.e. Pellet [20], Racer [8] or FaCT [9]). Irrespective of which reasoner is used, different levels of inferencing can be set. Using higher levels of inference means that more complicated expressions are can be processed, at the cost of computational load. Finally, in addition to the internal reasoner, Jena features a general purpose rules engine.

In practice, inferencing is enabled by assigning a reasoner (either an internal or an external one) to an ontology model upon its creation. Every time inference is needed for a query to be answered, the reasoner is automatically invoked. This is completely transparent to the programmer, not requiring him to alter the code, nor to know the specific syntax each reasoner adopts.

Data persistency: Models and ontology models can be stored in a database, so that they do not have to be downloaded and parsed every time a Jena application starts. In order to do this, models are converted into triples of the form subject - predicate - object (every RDF statement can written in a triple form), where each of the three parts is the URI of the corresponding resource or property. Database systems that can be used include MySQL[3], Oracle[4], PostgreSQL[5] and Microsoft SQL Server[6]. Similarly to the case of inference support, whether a model will be persistent or not is determined upon its creation. Consequently, any changes to the model will automatically be stored in the database.

6.2 Ontologies

In order to test the capabilities of the RFA Broker, several ontologies were created. The cornerstone of these is the RFA Domain ontology, which contains both the required extensions to OWL-S, and a number of hierarchies describing locations, types of goods, directions and other things. The classes of the RFA Domain ontology can be seen in Figure 6.2. In addition, several RFA and available services profiles were created. The RFA Domain ontology, as well as the OWL documents containing the RFA and service profiles were uploaded to a publicly accessible domain[7].

All the ontologies were created using the Protégé[8] [18] ontology editor. Protégé

[3]http://www.mysql.com/
[4]http://www.oracle.com/technology/products/database/oracle10g/index.html
[5]http://www.postgresql.org/
[6]http://www.microsoft.com/sql/default.mspx
[7]http://homepages.inf.ed.ac.uk/s0570652/ontologies/
[8]http://protege.stanford.edu/index.html

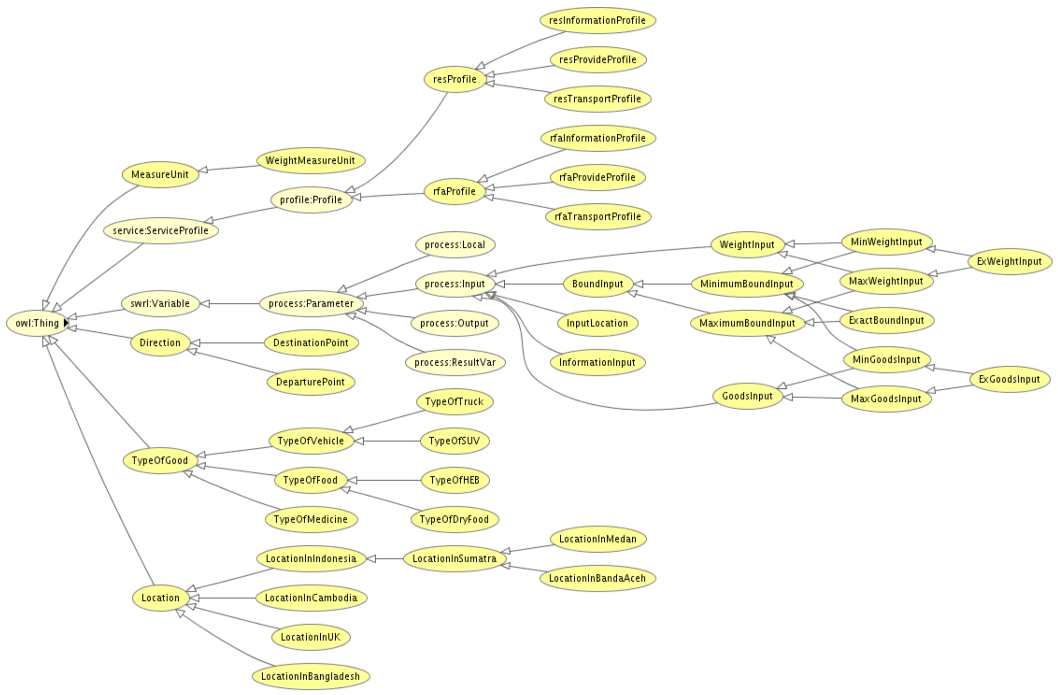


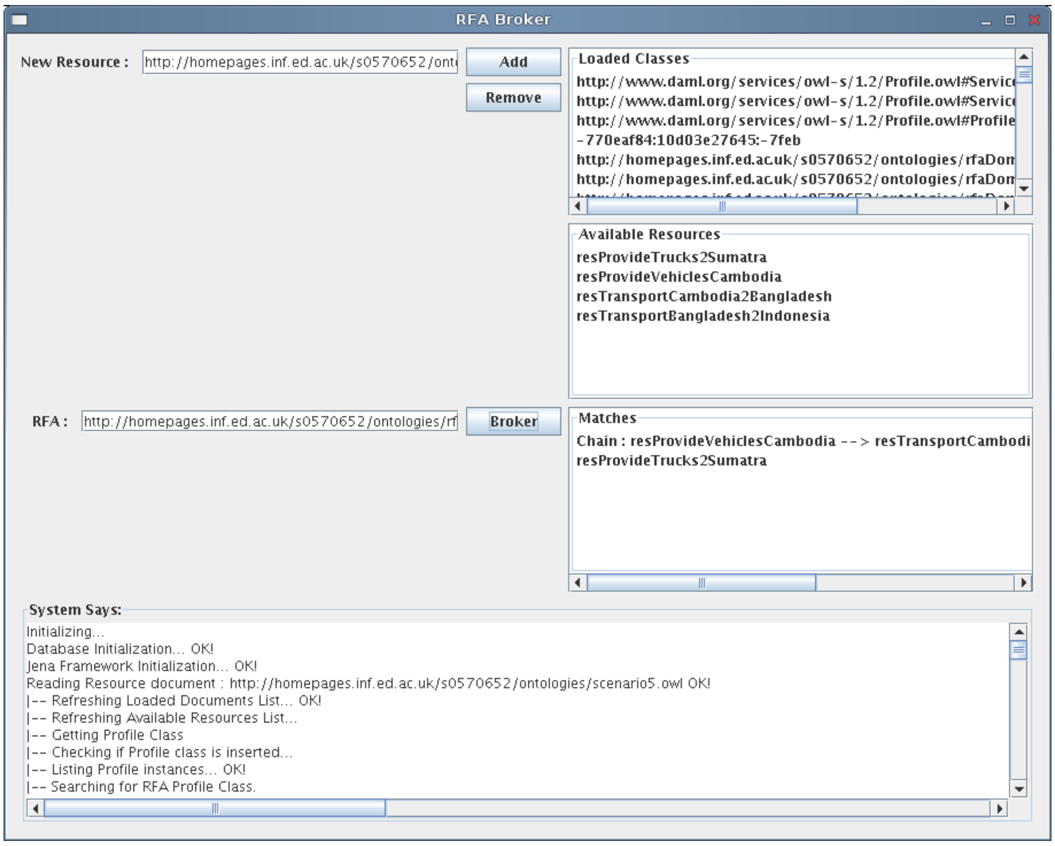
Figure 6.2: The RFA Domain ontology

was selected because it features several plugins and extensions for both OWL-S and OWL that facilitated the creation of the databases. It was also used to check the consistency of the ontologies by connecting to RACER through the DIG interface.

6.3 Graphical Interface

The graphical interface of the RFA Broker (Figure 6.3) was created using the standard Java Swing library. The user can start by loading available service profiles. This is done by typing the URL of the OWL documents where the profiles are described. The two sections on the upper right of the window show the classes and the available service profiles that were found in the loaded documents. Each time a new OWL document is loaded, the two lists are updated.

When the user has finished loading available service profiles, an RFA profile can be brokered. This is done by typing the URL of the OWL document describing the RFA profile in the “RFA” text field. Clicking on the Broker button then starts the brokering process (see below). If there are any services or chains of services found to match the RFA, they will then appear in the “Matches” section of the graphical interface. While the graphical interface is being used, messages will appear in the bottom text area of



Loaded Classes

```
http://www.daml.org/services/owl-s/1.2/Profile.owl#Servic
http://www.daml.org/services/owl-s/1.2/Profile.owl#Servic
http://www.daml.org/services/owl-s/1.2/Profile.owl#Profile
-770eaf84:10d03e27645:-7feb
http://homepages.inf.ed.ac.uk/s0570652/ontologies/rfaDon
http://homepages.inf.ed.ac.uk/s0570652/ontologies/rfaDon
```

Available Resources

```
resProvideTrucks2Sumatra
resProvideVehiclesCambodia
resTransportCambodia2Bangladesh
resTransportBangladesh2Indonesia
```

Matches

```
Chain : resProvideVehiclesCambodia --> resTransportCambodi
resProvideTrucks2Sumatra
```

System Says:

```
Initializing...
Database Initialization... OK!
Jena Framework Initialization... OK!
Reading Resource document : http://homepages.inf.ed.ac.uk/s0570652/ontologies/scenario5.owl OK!
|-- Refreshing Loaded Documents List... OK!
|-- Refreshing Available Resources List...
|-- Getting Profile Class
|-- Checking if Profile class is inserted...
|-- Listing Profile instances... OK!
|-- Searching for RFA Profile Class.
```

Figure 6.3: The RFA Broker application

the window. These provide feedback about the progress of the brokering algorithm and the status of the system.

6.4 The RFA Broker

The RFA Broker is the back-end of the system, where all functionality is implemented. When the system starts, the RFA Broker initializes the Jena components that will be used. Specifically, a model maker supporting OWL-DL expressions and reasoning is created. The model maker is then used to create a central ontology model. The role of the central model is to act as a container for all the sub-models of the imported documents. In other words, it can be considered to be a knowledge base where the profiles of both RFAs and available services can be stored. The sub-models are also created by the same model maker.

The RFA Broker also initializes the connection to a MySQL database. Afterwards, the connection was passed to the model maker, so that the models created would be persistent. Unfortunately, the transformation of models to triples proved to add significant computational load. For this reason, data persistency was finally disabled[9] and the models were stored in run-time memory.

Each time the user types the URL of an available service description, the RFA Broker uses Jena to download the corresponding OWL document, parse it and store it as a submodel of the central model. The same happens when an RFA URL is typed, with the difference that the brokering algorithm is initiated after the creation of the sub-model. In the case where the RFA sub-model contains two or more RFA profiles, only the first one to be found is brokered. In its current version, the RFA Broker can only broker one RFA profile during its execution. In other words, the application needs to be restarted to broker new profiles.

The RFA Broker uses the brokering algorithm that was proposed in a previous chapter. For instance, when brokering begins, it checks the type of RFA (the class of which its profile is member) and only compares it to available service profiles of the same type. In general, the RFA Broker makes use of Jena classes and methods to navigate through the central model and make the checks that are required to determine which part of the algorithm should follow next.

Finally, the RFA Broker is responsible for updating the graphical interface. Each time a new document is loaded, it refreshes the GUI sections that list available services and classes. Furthermore, after the end of the brokering algorithm, it presents the matches in the "Matches" section of the interface. Equally importantly, especially when matches are not found, the RFA Broker prints feedback in the corresponding section of the interface, which provides the user with some indication of the progress of the algorithm..

[9]Data persistency can be enabled by just uncommenting the corresponding part and re-compiling.

Chapter 7

Evaluation

A series of test cases were created to display the capabilities of the proposed modelling methodology and the brokering algorithm. To a certain extent, the test cases model real-life scenarios, in the respect that actual RFAs from the 2004 Tsunami relief effort were modelled. However, as explained in Chapter 2, neither the complete results of the brokering done by MPAT nor the full set of services available to MPAT at that time were available. For this reason, fictional services had to be created to be matched against the RFAs. This action restricts the conclusions that can be drawn about these tests and their results in several ways.

Firstly, there is no way of knowing whether the modelling used for RFAs is capable of describing available (real) services as well. Although it seems reasonable to assume that services and RFAs have the same form, there may be non-apparent attributes of a service that are not present in RFAs of the same type, which could be difficult to express as parameters of the OWL-S profile. This would render the modelling approach inadequate for available services.

Furthermore, one way to evaluate the brokering algorithm would be to compare the services proposed by the RFA Broker to the services that were actually used to satisfy RFAs. However, because the manual brokering results and the set of services that were available to MPAT are not known, this is not possible. Consequently, if the RFA Broker proposes different services from the ones actually used, this might be because such services were not available to MPAT. In any case, there is no way to infer whether the RFA Broker performs better or worse brokering in comparison to the MPAT manual process.

Despite the afore-mentioned difficulties, these test-cases should provide some insights into the brokering problem and the solution proposed here, leading to a more general consideration of the strengths and weaknesses of this approach.

7.1 Test Cases

The following scenarios were modelled and tested:

7.1.1 Scenario 1

This scenario concerns the direct brokering of a transportation RFA. The RFA, submitted on the 8th of January 2005, was: *"Transport 10x WFP trucks (40 tons) from Phnom Penh, Cambodia to Dhaka, Bangladesh"*. As described in Section 5.2, transportation profiles have at least three inputs, describing the departure point, destination point and weight of cargo. The additional information provided by the RFA about the type of cargo (WFP trucks) and the number of items (10) was not modelled in the profile. Consequently, the RFA that was brokered was equivalent to: *"Transport 40 tons from Phnom Penh, Cambodia to Dhaka, Bangladesh"*.

The scenario involves only one available transportation service. When the brokering took place, the following profiles were present (here, and throughout this section, to make the text more readable the RFAs and services are presented in the equivalent "English" form of the actual OWL-S description used):

- **Service:** *Can transport cargo of maximum 100 tons from any location in Bangladesh to any location in Indonesia.*
- **RFA:** *Transport 40 tons from Phnom Penh, Cambodia to Dhaka, Bangladesh.*

The RFA Broker identified a match between the RFA and the available service.

7.1.2 Scenario 2

This scenario concerns the direct brokering of a provision RFA. The RFA, submitted on the 13th of January 2005, was: *"Request for exactly 4 Mercedes Model 1 trucks in Banda Aceh (Indonesia), to assist with dual off-load ops for HDR supplies."*. Again, as described in Section 5.2, provision profiles have at least two inputs, describing the destination point and the type of good that needs to be provided. The additional information provided by the RFA about the task for which the particular good is needed was not modelled in the profile. Consequently, the RFA that was brokered essentially was: *"Request for exactly 4 Mercedes Model 1 trucks to Banda Aceh (Indonesia).".*

This time there are two provision services, only one of which is capable of satisfying the RFA. The following profiles were present:

- **Service 1:** *Can provide maximum 20 SUVs to any location in Indonesia.*
- **Service 2:** *Can provide maximum 5 vehicles to any location in Indonesia.*
- **RFA:** *Request for exactly 4 Mercedes Model 1 trucks to Banda Aceh (Indonesia).*

The RFA Broker identified only the second available service as a match.

7.1.3 Scenario 3

There were cases where RFAs requested information about particular places, especially airports. The RFA of the scenario was, submitted on the 26th of January 2005, was: *"Request any available information on the following airfields on Sumatra. Bland Pidie, Tapaktuan (Teuku Cut Ali), Singkil, Sinabang (Lasikin) - Polonia.".* The RFA was split into 5 separate RFA profiles, each requesting information about a particular airfield. As the 5 RFAs would be brokered in the exact same way, here we consider only the last one: *"Request for information about the Polonia International Airport (Medan, Indonesia)".*

There are two available services, one of which is capable of satisfying the RFA. Note that although the services "provide" information, this is not considered to be a case of brokering provision services. All profiles are defined to be members of informationProfile class, which is disjoint from the provisionProfile class. The central model contains the following profiles:

- **Service 1:** *Can provide information about Medan.*
- **Service 2:** *Can provide information about vehicles.*
- **RFA:** *Request for information about the Polonia International Airport (Medan, Indonesia).*

The RFA Broker identified only first service as a match.

7.1.4 Scenario 4

This scenario concerns the orchestrated brokering of a transportation RFA. In fact, this is the same scenario pictured in Figure 5.2, which is a modification of Scenario 1, presented above. There are four transportation services available, only one of which is capable of directly satisfying the RFA. In addition, the remaining three services can be combined to collaboratively satisfy the RFA. The central model contains the following profiles:

- **Service 1:** *Can transport cargo of maximum 100 tons from any location in the United Kingdom to any destination in Cambodia.*
- **Service 2:** *Can transport cargo of maximum 125 tons from any location in Cambodia to Dhaka (Bangladesh).*
- **Service 3:** *Can transport cargo of maximum 85 tons from any location in Bangladesh to any destination in Indonesia.*
- **Service 4:** *Can transport cargo of maximum 75 tons from any location in the United Kingdom to any destination in Medan (Indonesia).*
- **RFA:** *Request for transportation of exactly 50 tons from London to Polonia (Medan, Indonesia).*

The RFA Broker identified two matches:

1. Direct match: Service 4.
2. Orchestrated Match (Chain): Service 1 $\longrightarrow$ Service 2 $\longrightarrow$ Service 3.

7.1.5 Scenario 5

This scenario concerns the orchestrated brokering of a provision RFA. In fact, this is the same scenario pictured in Figure 5.3, which is a modification of Scenario 2, presented above. There are four services available, only one of which is capable of directly satisfying the RFA. In addition, the remaining three services can be combined to collaboratively satisfy the RFA. The central model contains the following profiles:

- **Service 1:** *Can provide maximum 5 vehicles to any location in Cambodia.*

- **Service 2:** *Can transport cargo of maximum 100 tons from any location in Cambodia to any location in Bangladesh.*
- **Service 3:** *Can transport cargo of maximum 75 tons from any location in Bangladesh to any location in Indonesia.*
- **Service 4:** *Can provide maximum 5 trucks at any location in Sumatra (Indonesia).*
- **RFA:** *Request for exactly 4 trucks to Banda Aceh (Sumatra, Indonesia).*

The RFA Broker identified two matches:

1. Direct match: Service 4.
2. Orchestrated Match (Chain): Service 1 $\longrightarrow$ Service 2 $\longrightarrow$ Service 3.

7.2 Complex RFAs

In some of the scenarios, information in the actual RFA was not modelled in the RFA profile. This should not necessarily be mistaken for a modelling or brokering shortcoming. For example, the transportation RFA of Scenario 1 was specifying the type of cargo (WFP trucks) and the quantity of items (10). This information was not included in the RFA profile mainly because it had no apparent use in brokering. Furthermore, it could unnecessarily complicate the brokering process. A transportation service would more likely need to know the weight and dimensions of the cargo, and not the type of vehicles that are being transferred. However, if for some reason this information needed to be present in the RFA profile, it *could* be modelled, say, by having an extra input specifying the type of cargo and its quantity.

There are other cases where information in transportation RFAs / services could not be modelled, at least not in the approach proposed in this thesis. For example, the RFA *"Use currently scheduled airlift from Bangkok to move 20 Thai volunteers from Don Muang Intl (wing 6) to Phucket"* requests an already scheduled transportation to be used. In order to broker this RFA, the transportation schedule should also be formalised and described in an ontology, and made available to the system. The current modelling considers brokers RFAs independently of any wider relief context and, as a result, it cannot associate an RFA with a scheduled flight.

There are cases where modelled information in an RFA profile can be hard to broker. An RFA, submitted on the 25th of January 2005, requests the transportation of 20 Thai volunteers. Despite the fact that information such as the nationality of the transportees would normally not be used in brokering, there are cases where a human broker might take it into account. For instance, a human broker would avoid assigning, say, a Red Cross transportation service to people from a nation that openly opposes Red Cross. The same would be the case if the nation that provided a service had political or cultural differences with the nation of the transportees (for example, any country that was known to oppose US foreign policy might not be prepared to accept US Army transportation services).

Such information can, in theory, be modelled in an RFA / service profile, although the potential extent of such information is so wide-ranging as to make it practically impossible to cover all possible socio-cultural factors. For example, a transportation RFA could model the nationality of the transportees as an extra input. However, brokering using such information becomes much more complicated. The way the human broker acted was rather based on general knowledge and intuition, and not on any formalised general brokering process. The brokering algorithm proposed on this thesis would fail to broker this kind of information.

7.3 Meeting the Desiderata

In Chapter 2 some desirable properties were defined for modelling and brokering RFAs. This section will examine the extent to which these properties have been met with the approach adopted in this project.

7.3.1 Preventing misinterpretation

As discussed in Chapter 2, one of the desirable characteristics for an RFA / service modelling approach was to prevent possible misinterpretation that could be caused by typos, acronyms or other reasons. Using an ontology language such as OWL to model services, seems to have solved this issue, as RFAs / services are defined in terms of predefined concepts, instead of strings. This means that the RFAs / services, as well as the concepts that are used to describe them, are expressed in a universal, unambiguous way.

This, in turn, facilitates correct brokering. For example, consider an RFA requesting and a service providing the transportation of a certain good from Cambodia to Indonesia. Obviously, since they are constructed using the same structure and the same ontologies, the two profiles are matching. Even if the corresponding location inputs may be given different labels (strings), the fact that they are both related to the same URI unambiguously determines their location. The same would be the case if the names of the two inputs were written in different languages, or if they were meant to be identical, but a typo occurred.

Consequently, it can be concluded that modelling RFAs / services in OWL prevents misinterpretation, always assuming that there is agreement on and consistency in the use of OWL-S and the underlying domain ontologies.

7.3.2 Modelling and Brokering Transportation and Provision RFAs

The most common types of RFAs are the ones requesting the transportation of some type of cargo from one location to another. One of the desirable characteristics for a modelling and brokering approach was to be able to handle this type of RFAs and services. Scenarios 1 and 4 describe how two transportation RFAs are modelled and brokered. All the information in the RFA textual description (departure point, destination point and cargo weight) was modelled as inputs of the OWL-S profile, indicating, by this measure at least, that modelling was successful.

The case was the same for the other common type of RFA, requesting the provision of a certain good to a location. Scenarios 2 and 5 describe the modelling and brokering of two provision RFAs. Again, all the information contained in the textual description of the RFA (quantity and type of good, delivery location) was modelled as inputs of the OWL-S profile.

The brokering algorithm could only be tested using fictional services to match against RFAs. As explained before, this probably entails different results from the action taken in reality. However, the fact that the matches produced by the RFA Broker were all reasonable and that all unreasonable solutions were ruled out is encouraging. Furthermore, whatever the form of actual transportation services may be, they would most probably have at least the three inputs used in the fictional transportation services (departure input, destination input, weight input). The fact that brokering of transportation services is based on those inputs is a strong indication that real services will probably be brokered correctly as well.

7.3.3 Modelling various types of RFAs / Services

The RFAs that involve neither transportation nor provision may have varying natures and parameters. For this reason, one of the desirable properties for a modelling approach was to be able to describe diverse types of RFAs and services. The modelling approach proposed in this thesis achieves this goal to a certain extent, by defining subclasses of the OWL-S profile to describe different types of RFAs. Then, the attributes of an RFA can usually be expressed in terms of inputs and outputs. In this case, RFAs are modelled and brokered successfully. An example of such an RFA that was successfully modelled and brokered is the one in Scenario 3.

However, there are certain types of RFAs that could not be modelled, at least not directly. Such a type is the one requesting for a particular task to be carried out. For instance, an RFA submitted the 10th of January 2005 requests: *"Conduct sea-based UN-WHO health assessment of western Sumatra via helicopter off Abraham Lincoln"*. There is no evident way to directly model this task as inputs of the RFA profile. However, a task may be decomposable to simpler tasks that can be handled easily. Indeed, the particular RFA could be split to a request for medical personnel that could carry out the health assessment, as well as two requests for their transportation from Abraham Lincoln aircraft carrier to western Sumatra, and back. However, the decomposition of certain tasks to simpler ones may require domain-specific expertise, which may not be available to the MPAT personnel.

Finally, there is another subset of RFAs that were too complicated to be modelled. For example, this RFA was submitted on the 28th of January 2005: *"Expert in vector control and malaria is needed to provide guidance to Indonesian vector-control teams. Need person with experience in leading teams in the field, familiarity with Indonesia, ability to speak Bahasa Indonesian"*. The modelling and brokering approach proposed here could only handle a generic version of the RFA, requesting for an expert in a particular medical field.

In general, most of the RFAs that were available in the relief efforts web sites could be modelled using the proposed approach. Unfortunately, certain RFAs had to be decomposed to simpler ones in order to be modelled, and others could not be modelled at all. However, these RFAs represented only a small percentage of the RFAs that were examined.

7.3.4 Modelling RFAs / Services with varying detail

The last of the modelling desiderata was for the modelling approach to allow descriptions of RFAs / services with varying detail. The modelling approach proposed here achieves that by allowing any number of inputs and outputs to be specified for a profile. In other words, some profiles may be more detailed and contain more information than others. However, in order to prevent the existence of too generic profiles that have very few or no inputs at all, cardinalities can be defined for the inputs of new types of services. In this way, certain inputs can be defined to be mandatory, whereas others can be optional.

7.3.5 Coping with large numbers of services

The test cases mentioned earlier on this chapter were run on a laptop with a 1.6 GHz Intel Centrino processor and 1 GB of RAM. Generally, all the test cases were simple, with a maximum of four available services present. However, the time needed for the RFA Broker to broker them was considerably high, varying from 6.2 seconds (Scenario 3) to 15.7 seconds (Scenario 5). Despite the lack of a test case involving search over numerous available services, the low performance in the existing test cases shows that the RFA Broker is unlikely to be able to handle large numbers of services.

There are several factors that result in high computational load. The main one is the complexity of the brokering algorithm, which increases significantly in the case of indirect brokering. There are other factors that contribute to the performance problem. Firstly, the brokering algorithm makes heavy use of the Jena reasoning module to detect implicit relations between classes and instances. Furthermore, the conversion of OWL documents to their corresponding models has proved to be a resource consuming process. Finally, the fact that the RFA Broker and Jena are both written in Java, which is interpreted and not compiled, has a direct impact on performance.

In a real life scenario, the number of available services and RFAs to be brokered may rise into the hundreds. As a consequence, the RFA Broker could not be practically deployed in its present form. There are ways to improve the efficiency of the algorithm, but this will be discussed in the next chapter.

7.3.6 Enriching RFAs

One of the problems identified in the current brokering process concerned the fact that several required pieces of information were missing from the RFAs. In order to solve this, it would be desirable for brokering approach to gradually enrich RFAs with information, as brokering progresses.

The brokering algorithm that was proposed is achieves to enrich provision and transportation RFAs. Normally, only direct matches to a provision RFA would be detected. However, when a provision RFA is brokered indirectly, the total weight of the cargo is automatically computed and used to locate transportation services that are capable of carrying it to the location where it is requested. In other words, the provision RFA is enriched with additional information that was originally unavailable, allowing the detection of matches that may not be obvious.

7.4 Analytical Evaluation

In this section two important design decisions will be evaluated: the use of OWL and OWL-S.

7.4.1 The selection of OWL

In Chapter 3 OWL was selected for the description of RFAs and available services. This selection was justified in a number of occasions. First of all, OWL provided the means to avoid misinterpretation. As mentioned in the discussion on desiderata, through the use of URIs it provided the means to identify resources in an unambiguous and unique way.

Furthermore, OWL enabled expressions that were not possible in other ontology languages, such as RDF(S). For instance, some classes were defined as the union or intersection of other classes (ExactBoundInput $\equiv$ MaximumBoundInput $\sqcap$ Minimum-BoundInput). Others were defined to be disjoint (resProfile $\sqcap$ rfaProfile $\equiv \bot$). Finally, cardinalities can be used to refine the definitions of specific types of profile. For example, transportation profiles can be defined to have exactly two location inputs, one for the destination and the other for the departure point.

In conclusion, the use of OWL seems to have enabled modelling capabilities that would otherwise be impossible. Thus, its selection over RDF(S) can be considered a good choice.

7.4.2 Evaluation of OWL-S for RFAs

In order to answer the question of whether the use of OWL-S was a correct choice or not, one must first consider the extent to which it was able to model RFAs (and, by extension, services). Given that transportation and provision RFAs are very similar to the ones modelled in the test cases, it would be safe to assume that most of them can be modelled in OWL-S, in the method proposed in an earlier chapter. In addition, Scenario 3 shows that RFAs of different type and nature can also be modelled, as long as their attributes can be 'mapped' to inputs and outputs. However, only direct brokering can be used for these. Generally, OWL-S has proven to be capable of describing the majority of RFAs, and in a way that can be brokered.

On the other hand, there were RFAs that could not be modelled in OWL-S, at least not in the proposed methodology. One of the factors for this, is that OWL-S was designed primarily to describe Web Services, which can have specific attributes and properties. RFAs, on the other hand, may be requests of virtually any form. Despite the fact that OWL-S is sufficiently general to model services of other types, there were cases of RFAs where such modelling was not possible.

Another option was to create and use a custom OWL ontology for RFAs, instead of OWL-S. However, it would not be possible to guarantee that such an ontology would be able to model more types of RFAs / services than OWL-S can. Furthermore, it would be very difficult to create an ontology as well structured and generic as OWL-S, which has been developed and constantly improved for years. Finally, the design and creation of such an ontology would require amounts of time that were not available for this project.

In conclusion, the use of OWL-S has achieved its goal by modelling the majority of RFAs. The present approach can be further improved in a number of ways, which will be discussed in the next chapter.

Chapter 8

Conclusions and Future Work

Having presented and analysed a modelling and brokering approach for RFAs, in this chapter we summarize this work and discuss the conclusions we are able to draw. Afterwards, based on the evaluation detailed in the previous chapter, we will propose ideas for future work on the project.

8.1 Conclusions

In this section we will re-examine the stages of this project discussing any final conclusions about what has been achieved.

The description and brokering of RFAs is a real problem faced by emergency relief organisations such as MPAT, and one for which AI techniques may provide assistance. This idea formed the basis of the work described in this thesis. We started by examining several RFAs that were submitted during the 2004 Tsunami relief effort and the 2006 mudslides in the Philippines, in order to discover what forms an RFA might take. This examination revealed that most RFAs requested or involved transportation or provision of goods, whereas the remaining ones could have significantly varying nature. Based on this examination and on other identified issues, we concluded on a number of characteristics that would be desirable for any RFA modelling and brokering approach.

In Chapter 3 several approaches were considered that could be used to model and broker RFAs and services. We first ruled out non-semantic approaches, as the syntactic comparison they are using would cause successful matches to be overlooked. The next approach to be considered was RDF in conjunction with RDF-Schema (together referred to as RDF(S)). We argued that, although RDF(S) is able to unambiguously specify concepts, its expressive power is insufficient for RFAs/services. The last approach that was considered was OWL. After an analysis of its expressive capabilities, we argued that a number of expressions that were unavailable in RDF(S) were allowed in OWL. Therefore, we concluded that OWL was the most appropriate of the considered approaches. The Chapter ended with a presentation of the OWL-S ontology and two modelling and brokering approaches for services that use it.

In Chapter 4, a modelling methodology for RFAs using the OWL-S ontology was proposed. This chapter included an analysis of several design issues that emerged. For some of them we considered adapting solutions proposed by the reviewed approaches, and argued on possible ways these would affect RFA modelling. One of the most important issues concerned the use or not of preconditions and effects. It was argued that modelling would be more complete if preconditions and effects were used. We concluded, however, that this would significantly complicate the brokering algorithm. The chapter went on to propose a specific structure for the common types of transportation and provision RFAs and services, by means of the particular OWL-S inputs they will be using. The chapter ended by giving an example of an RFA modelled as an OWL-S Profile in the proposed method.

In the next two chapters we proposed a brokering algorithm, able both to find 'direct' matches between RFAs and services and to compose 'chains' of services to meet an RFA, and then presented an implementation of it, called RFA Broker. Following this, in Chapter 7 we evaluated the design decisions that were taken and the modelling and brokering methodologies that were proposed. The evaluation began by running a set of tests, derived from actual RFAs, on the RFA Broker to show that it worked as expected. Afterwards, we attempted to decide on the degree to which the initial desiderata have been met. Specifically, we concluded that by using OWL in the way suggested here, we avoid possible misinterpretations and that the most common types of RFAs and services can be modelled, with few exceptions in the devised formalism, but that a small percentage proved to be too complicated for the modelling approach to describe. In general, however, we believe that most RFAs in this group are within the capabilities of the modelling/brokering approach. The project also succeeded in allowing RFAs to be modelled with different levels of detail. We also displayed that it is possible to gradually enrich RFAs with information, as their brokering progresses. This project succeeded in meeting most of the desiderata, at least to a certain extent.

The evaluation chapter finishes by concluding that the selection of OWL over RDF(S) was a correct choice for the description of RFAs. To support this claim, we listed a number of occasions where OWL expressions with no equivalent in RDF(S) were used. The chapter ends with a similar consideration for the use of OWL-S. We argued that, despite the fact that some RFAs could not be modelled, OWL-S was capable of modelling the majority of RFAs - at least the ones that were available.

There are several reasons for which the RFA Broker cannot be used in its present form. First of all, only a fragment of the data needed for thorough and complete testing was available. For this reason, we cannot be sure that the modelling and brokering methods we propose are adequate and efficient as indications show. Furthermore, the current implementation of the RFA Broker is not computationally efficient and scalable enough to be used in practice.

If the modelling and brokering approaches proposed in this thesis are adopted by MPAT, the current brokering process will need to to be accordingly adapted. Human brokers would still be needed, but their duty will now will be to select one of the results proposed by the RFA Broker, or override them for a self-proposed solution. As we saw in the previous chapter, some RFAs might need to be decomposed to simpler ones, which the RFA Broker can handle. This process will need to be carried out in an additional stage, before brokering begins.

If the above problems are solved (see next section) and adaptations are made, we believe that the RFA Broker can be a useful asset for MPAT. Firstly, possible solutions will not be overlooked, which may be often the case when brokering is performed by humans. Besides the services that directly match an RFA, the RFA Broker can propose orchestrated solutions that may be too complicated for a human to identify. Furthermore, can perform brokering much faster that a human can, saving time and, hopefully, lives.

8.2 Future work

In this section we consider ways in which the modelling and brokering methodologies could be improved. The following proposals were not implemented due to the limited time that was available for this thesis.

8.2.1 Additional Testing

The work that was carried out for this thesis could not be evaluated appropriately. The fact that the set of services available to MPAT was not available limited the results drawn from the evaluation in a number of ways, described in the previous chapter.

Consequently, it would be beneficial for this work if it was tested in a more complete way. This would include testing the RFA Broker with the actual set of services available to the RFA, and comparing the results proposed by the RFA Broker to the action proposed.

It would also be interesting to test how the RFA Broker performs against real MPAT personnel in a set of scenarios, by comparing the results proposed by the two parties. Besides the evaluation of the current implementation, the MPAT personnel could contribute to its improvement in a number of ways. Firstly, in case the two propositions are different, they could help clarify the reasons this might have happened. This could illuminate any factors that were not modelled properly, or were not taken into account by the Broker. Furthermore, since they are meant to be the primary users of the RFA Broker, they could give valuable feedback on possible ways it could be improved or request additional functionality. For instance, they might find it more convenient if the results are presented in a particular order, or request changes on the interface. Finally, it would be important to find out whether they can use the RFA Domain ontology to model RFAs and services, extend it if needed, or identify mis-modelled concepts.

8.2.2 The dimension of time

An important parameter that was not taken into account is time, which is particularly important in the case of transportation RFAs and services. Currently, the transportation services are matched to RFAs in terms of their departure and destination locations, and the weight they can carry. In reality, however, an RFA might be requesting for the transportation to be carried out at or before a particular time. Furthermore, a transportation service is likely to be able to undertake the transportation in specific time slots, when no other transportation is scheduled. Therefore, it would be more realistic to broker RFAs in terms of time as well.

There are many ways this could be elaborated, as diverse factors may be taken into account. For instance, the number of transportations that a typical service can carry out simultaneously depends on the numbers of its aircrafts and pilots that are currently available. Furthermore, the duration of transportations could be computed based on the departure and destination locations, the weight of the cargo, as well as the aircraft to be used. Finally, the brokering algorithm could be modified to only propose chains whose services have compatible schedules.

8.2.3 Cost estimation

Another parameter that could be taken into account is the cost of invocation of services. In its current form, the algorithm proposes all the services or combinations of services that are in position to satisfy the RFA. However, all the possible solutions are not equivalent, at least not from a cost point of view. For example, the transportation of a person may be carried out by a helicopter, a military C-130 aircraft or a chain of 5 air transportation services. All three solutions would satisfy the RFA equally well, according to the RFA Broker. However, a human broker would choose the first one, because of its significantly smaller cost.

The above adaptation would help reduce the cost of satisfying RFAs, or equivalently, in the satisfaction of more RFAs at a certain cost. Furthermore, the above addition would help in a better distribution of the financial contribution among the participants in the relief effort.

An additional, practical benefit of talking factors such as cost into account might be to provide some metrics for ranking the solutions returned by the broker (listing them in order of ascending cost), or even for making the (currently exhaustive) search for solutions more efficient: services that are more costly than the amount that a requester is willing or able to pay can be ignored, and chains whose cost exceeds this amount can be discarded.

8.2.4 Degrees of match

In its current form, the brokering algorithm only proposes services that 'subsume' an RFA. In other words, only services that can completely satisfy the RFA are identified as a match. In practice, however, it may be unrealistic to expect that such services will always be available. In this case, proposing services that partly satisfy an RFA is preferable than proposing no services at all.

The problem could be solved in a way similar to the one adopted in the first brokering approach considered in Chapter 3. The 'goodness' of a match would be expressed in terms of points, which would be assigned for every matching parameter between the RFA and service profile. Only exact matches would reach the highest possible point values, whereas weaker types of match would be available as well.

8.2.5 Physical Dimensions

Another refinement to the brokering algorithm concerns transportation services in particular. As mentioned before, the brokering of transportation services is based on departure and destination locations, and the weight of the cargo. However, even if these parameters are met, the physical dimensions of the cargo may prevent it from being transported by certain services. Therefore, a useful addition to the algorithm would be to take into account the physical dimensions of the cargo.

A simplistic approach would be to follow the same method that was used for weight restrictions. However, the problem would be complicated if several goods were to be transported, as the dimensions of the cargo would also depend on the shape of the items and the shape of the space in which they are to be stowed. Even if a complete solution is too complicated to be found, considering the dimensions of a single item would still be able to rule out some inappropriate services.

Bibliography

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[2] S. Bechhofer. The DIG description logic interface: DIG/1.1. *Proceedings of the 2003 Description Logic Workshop (DL 2003)*, 2003.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic Web. *Scientific American*, 284(5):28–37, 2001.

[4] T. Bray, J. Paoli, C.M. Sperberg-McQueen, et al. Extensible Markup Language (XML) 1.0. *W3C Recommendation*, 6, 2000.

[5] D. Brickley and RV Guha. Resource Description Framework (RDF) Schema Specification 1.0. *W3C Candidate Recommendation*, 27, 2000.

[6] U.S.C. Bureau. North American Industry Classification System (NAICS). *US Census Bureau*, 2001.

[7] M.R. Genesereth et al. Knowledge Interchange Format. *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference (KR91). Morgan Kaufmann Publishers, San Francisco, California*, 1991.

[8] V. Haarslev and R. Moller. RACER system description. *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, 2001.

[9] I. Horrocks. The FaCT system. *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux*, 98:307–312, 1998.

[10] I. Horrocks et al. DAML+OIL: a Description Logic for the Semantic Web. *IEEE Data Engineering Bulletin*, 25(1):4–9, 2002.

[11] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. *W3C Member Submission*, 21, 2004.

[12] O. Lassila, R.R. Swick, et al. Resource Description Framework (RDF) Model and Syntax Specification. *W3C Recommendation*, 22:2004–03, 1999.

[13] L. Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. *International Journal of Electronic Commerce*, 8(4):39–60, 2004.

[14] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, et al. OWL-S: Semantic Markup for Web Services. *W3C Member Submission*, 22, 2004.

[15] B. McBride. Jena: A Semantic Web Toolkit. *IEEE Internet Computing*, 6(6):55–59, 2002.

[16] D. McDermott et al. PDDL-the planning domain definition language. *The AIPS-98 Planning Competition Comitee*, 1998.

[17] D.L. McGuinness, F. van Harmelen, et al. OWL Web Ontology Language Overview. *W3C Recommendation*, 10:2004–03, 2004.

[18] NF Noy, M. Sintek, S. Decker, M. Crubezy, RW Fergerson, and MA Musen. Creating Semantic Web contents with Protege-2000. *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]*, 16(2):60–71, 2001.

[19] M. Paolucci, T. Kawamura, T.R. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. *Proceedings of the 1st International Semantic Web Conference, Eds. I. Horrocks and J. Hendler, LCNS*, 2342.

[20] B. Parsia and E. Sirin. Pellet: An OWL DL Reasoner. *Proceedings of the International Workshop on Description Logics*, 2004.

[21] P.F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. *W3C Recommendation*, 10, 2004.

[22] U.S. Products. Services Classification (UNSPSC). *URL: http://eccma.org/unspsc*.

[23] E. Prudhommeaux and A. Seaborne. SPARQL Query Language for RDF. *World Wide Web Consortium*, 2004.

[24] R.M. Smullyan. *First-Order Logic.* Dover Publications, 1995.

[25] David Trastour, Claudio Bartolini, and Javier Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, 2001.