SWWS – Semantic Web Enabled Web Services

Title:
B2B Demonstrator Description

Version: **1.0**
Date: **01/09/2004**
Pages: **24**

Responsible Authors:
Javier.Esplugas-Cuadrado@hp.com

Hewlett Packard Labs, Bristol

Co-Author(s):

Status:

[ ] Draft
[ ] To be reviewed
[ ] Proposal
[ü] Final / Released to CEC

Confidentiality:

[ü] Public - for public use
[ ] INT - for SWWS consortium (and Project Officer if requested)
[ ] Restricted - for SWWS consortium and Project Officer only

Project ID: IST-2002-37134

Deliverable ID: D8.3

Workpackage No: 8

Title: WP8 B2B Demonstrator Description

Summary / Contents:

This document presents the description of the demonstrator as a concrete technical architecture of the case study. It follows D8.2 Ontologies and Services which included the analysis of the case study as a set of web services. Please refer to deliverable D8.1 to find the Case Study requirements and the conceptual architecture of this domain.

SWWS Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2002-37134. The partners in this project are: Leopold-Franzens Universität Innsbruck (IFI, Austria)); National University of Ireland, Galway (NUI, Galway, Ireland); Forschungszentrum Informatik (FZI, Germany); Intelligent Software Components S.A. (iSOCO, Spain); OntoText Lab. - Sirma AI Ltd. (SAI, Bulgaria); Hewlett Packard (HP, UK), British Telecom (BT, UK)

Leopold-Franzens Universität Innsbruck (IFI)

Institut für Informatik
Technikerstrasse 13
A-6020 Innsbruck Austria

Tel: +43 512 507 6489
Fax: +43 512 507 9872

Contact person: Juan Miguel Gomez
E-mail: juan.miguel@uibk.ac.at

National University of Ireland, Galway (NUI)

National University of Ireland,
University Road
Galway, Ireland

Tel: +353 91 750414
Fax: +353 91 562894

Contact person: Liam Caffrey
E-mail: Liam.Caffrey@nuigalway.ie

FZI – Forschungszentrum Informatik

Haid-und-Neu-Str. 10-14
76131 Karlsruhe, Germany

Tel: +49 721 9654816
Fax: +49 721 9654817
Contact person: Adreas Abecker
E-mail: abecker@fzi.de

Intelligent Software Components S.A. (iSOCO)

Francisco Delgado 11, 2nd Flor
28108 Alcobendas, Madrid, Spain

Tel: +34 913 349797
Fax: +34 913 349799

Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

OntoText Lab.- Sirma AI Ltd. (SAI)

OntoText Lab.
38A Chr. Botev Blvd.
Sofia 1000, Bulgaria

Tel: +35 92 9810018,
Fax: +35 92 9819058

Contact person: Atanas Kiryakov
E-mail: Atanas.Kiryakov@sirma.bg

Hewlett Packard (HP)

HP European Laboratories
Filton Road, Stoke Gifford
BS34 8QZ Bristol, UK

Tel: +44 117 3128631
Fax: +44 117 3129285

Contact person: Janet Bruten
E-mail: janet.bruten@hp.com

Associated Partner:
British Telecommunications plc. (BT)
Orion 5/12, Adastral Park
Ipswich ip5 3RE, UK
Tel: +44 1473 609583
Fax: +44 1473 609832
Contact person: John Davies
E-mail: john.nj.davies@bt.com

Change Log

Vers.	Date	Author	Description
1.0	06/07/04	J. Esplugas	First version

List of Acronyms and Abbreviations

Acronym/ abbreviation	Resolution
SWWS	Semantic Web enabled Web Services
SWWS-AA	SWWS project Abstract Architecture
WS	Web Services
B2Bi	Business-to-Business Integration
WSMF	Web Service Modeling Framework
CRM	Customer Relationship Management
SCM	Supply Chain Management
X.12	ASC Standard for EDI
PIP	Partner Interface Process
ASN	Advanced Shipment Notice
SSM	Shipment Status Message
POD	Proof of Delivery
FI	Freight Invoice
EDI	Electronic Data Interchange

Table of Contents

Table of Figures

1 Introduction

This document follows the document D8.2, which presented the case study as the set of web services and ontologies. As in D8.1 and D8.2, the main players of this scenario are a multi-leg logistic coordinator and a freight forwarder. The multi-leg logistic coordinator will use the service of a marketplace to find the concrete FF.

The purpose of the multi-leg logistic coordinator (aka logistic provider) is to find a suitable freight forwarder to complete a supply-chain that has been broken. The freight forwarder offers his service publishing its capabilities using a semantic web enabled service.

In alignment with SWWS Abstract Architecture, the case study scenario holds the following conceptualization:

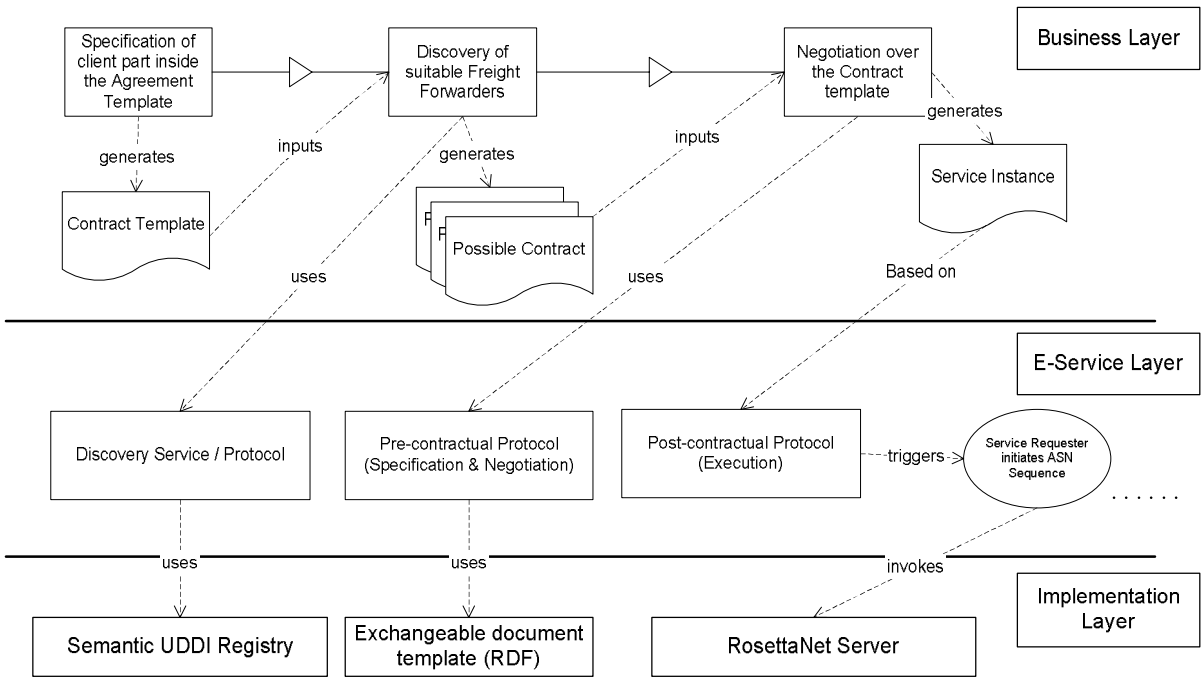

Figure 1 : Scenario view as a SWWS- AA

To have an in-depth analysis of the different stages related to the contract formation, the service execution and the ontologies please refer to the D8.2. This document will present basically the physic “entities” and the technologies they use in their relationships.

2 Architecture Design

This section will cover the specification of the services that will be deployed to support this case study. We will present the case study storyboard, for each of the steps; we will specify the impact that these stages have in the services.

2.1 Existing scenario storyboard (extracted from D8.2):

The reader could find a more detailed explanation about the scenario in the requirements document. Briefly, the situation is the following:

For an existing reason (optimization process, contract violation, etc) a logistic provider is about to be substituted. The multi-leg logistic provider stands as a communication broker between all partners. There is always 2 partner communications.

The Freight forwarder 2 is the component that is going to be substituted, and the following sections will present different aspects of the service substitution.

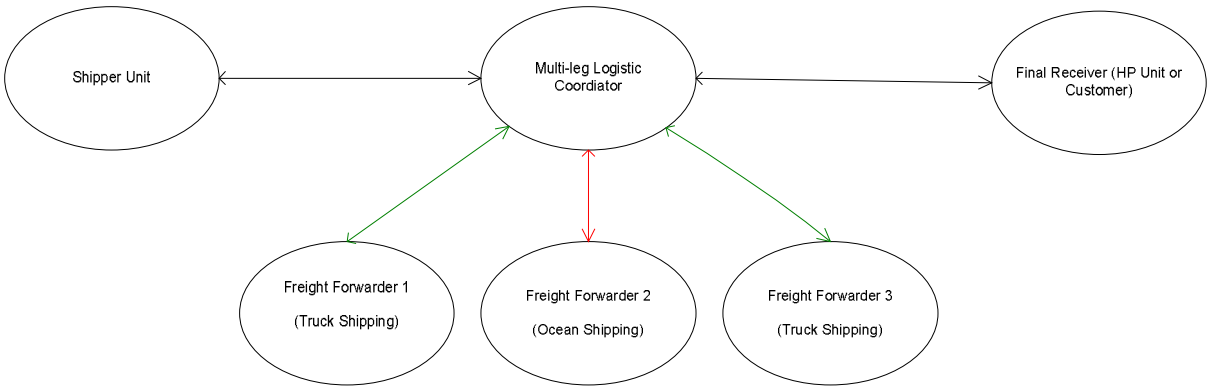

Figure 2: Actors involved in multi-leg logistic communications

1. Specification of the requirements of the solution.
2. HP will use a discovery service to locate the possible candidates to that could enter into an agreement. (Discovery Phase)
3. This agreement or contract will be specified by HP and the service must acknowledge the exploration of a possible contract. (Contract Formation)
4. HP will examine the process model of the new logistic provider and will check if it could be integrated or not (Matchmaking). This process will be the following:
    a. Compare the process model with the existing partners in the communication chain
    b. Specify the Mediation guidelines
    c. Check the consistency of the ontology representation lifting and lowering messages.

2.2 General architecture of the WP8 SWWS Demonstrator

The demonstrator, and by extension, the technical architecture will be centred in the process of discovery of a new provider and the process of integration of this service provider. Therefore will not cover at any point, messages with Shipper Units, Customers or other freight forwarders except the ones that exist in the interface of communication with them (e.g. ASN's are forwarded from one Logistic provider to another)

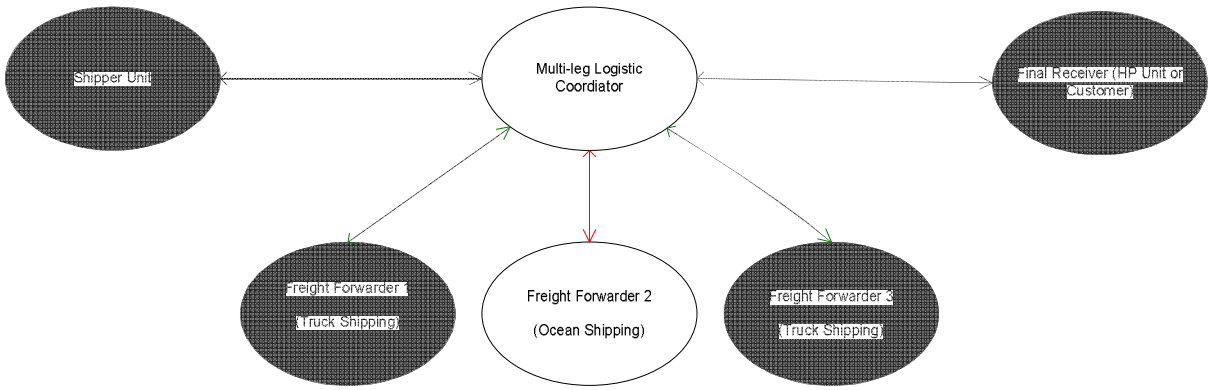

Figure 3 : Focus of the technical architecture document over the storyboard

The whole architecture will be represented using a Service Oriented Architecture, in this architecture there are three main components, the Service Consumer (in our case, the Multi-leg Logistic coordinator), the Service Provider (in our case the Freight Forwarder) and the Directory Service or Registry (in our case is the Marketplace).

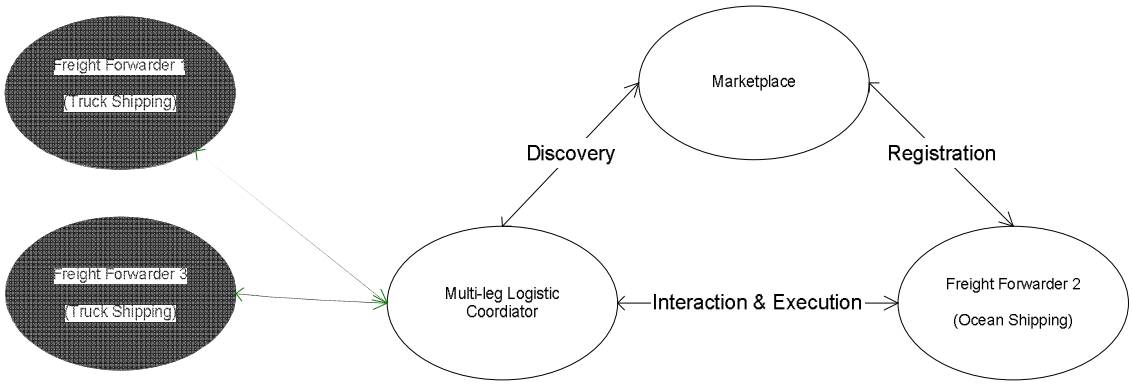

Figure 4 : WP8 Technical architecture in Service Oriented Architecture view.

2.3 The existing actors in the architecture

There are three main actors in this architecture. As presented in the introduction of this chapter, the main participants are the marketplace, the multi-leg logistic coordinator and the freight forwarder B, also a mediator and two dummy freight forwarders (to ensure the interfaces) will appear in this architecture.

2.3.1 The Marketplace

The marketplace will be implemented as a stand-alone component. This agent will have several capabilities:

- Ø S/Mime: The protocol used in communication will need at least S/Mime handling, any variation or new protocol specification is also possible (SOAP, BPEL4WS, …)
- Ø UDDI Capabilities: As a Discovery service, this marketplace needs to offer the existing web service discovery capabilities.
- Ø OWL & OWL-S Capabilities: The Marketplace will have to be able to serve OWL ontologies as well as store OWL-S descriptions of the services registered there.
- Ø Knowledge Base: As a registry all this information has to be persistent.

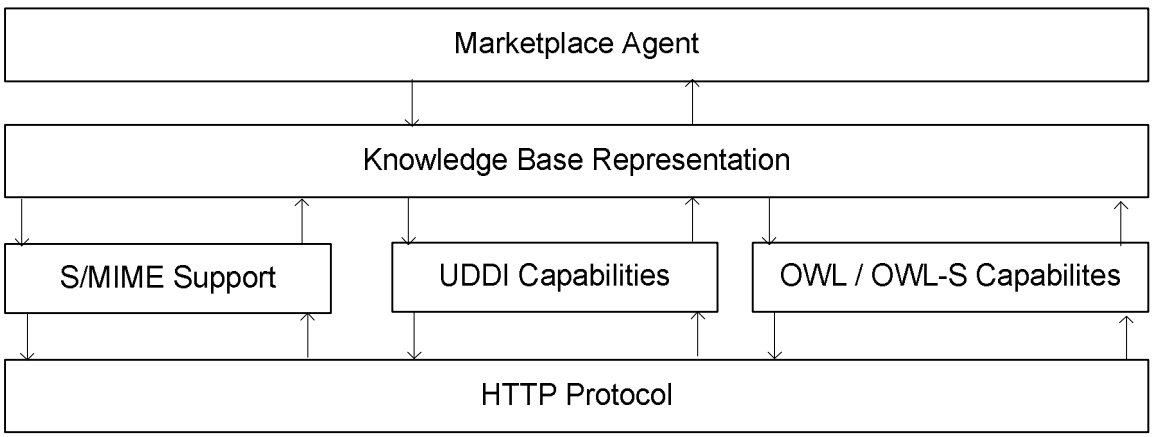

Figure 5 : Marketplace technical implementation

2.3.2 The Multi-leg logistic coordinator

The agent that implements the Multi-leg logistic coordinator will have a centre-piece which is the union of the Knowledge base representation and a DL reasoning engine. This unit will be feed by the capabilities of reading and interpreting OWL & OWL-S, as well as WSDL. This knowledge unit will create the information necessary for consumption of the data and protocol mediation creators. Incoming messages will trigger different actions in the Agent as well.

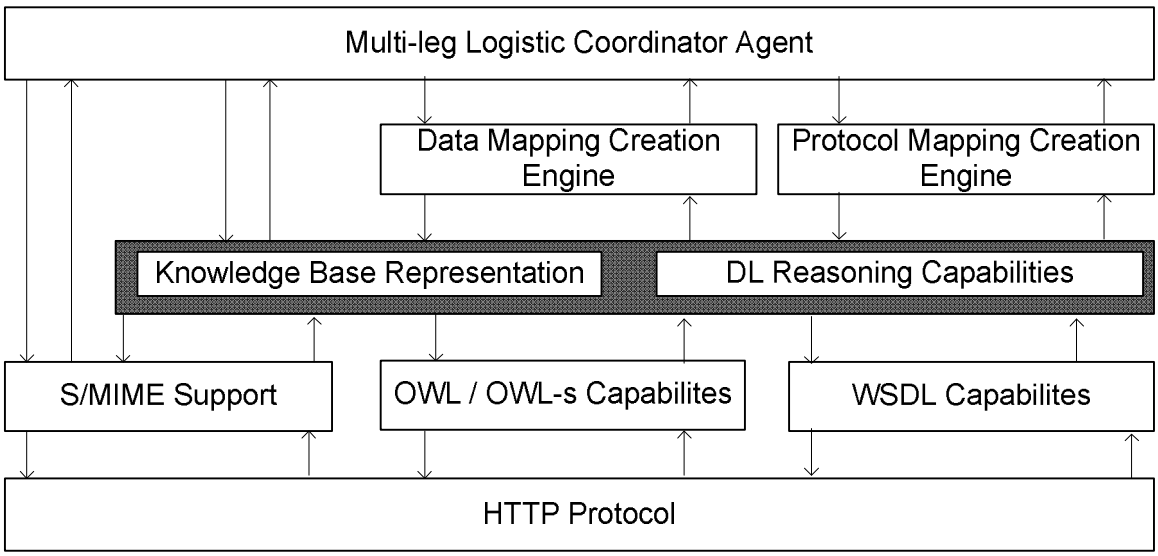

Figure 6 : The multi-leg logistic Coordinator technical implementation

2.3.3 The Dummy Freight Forwarder A

This is just a RosettaNet ASN sender and a waiting loop until an ack is received. It is necessary to ensure the sequence of actions correspond to the real execution environment.

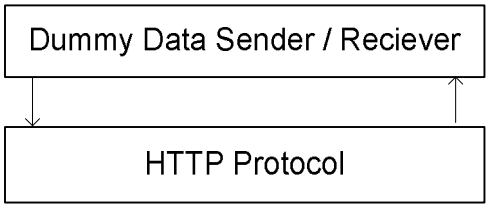

Figure 7 : Freight Forwarder technical implementation

2.3.4 The Mediator

The mediator once is created is just a simple lower and lifting machine, internally the mediation agent will have conscience of which protocols is he between and will react to incoming messages accordingly. The translator modules will have “data mediation” scripts that will upload the content of those messages into the knowledge base.

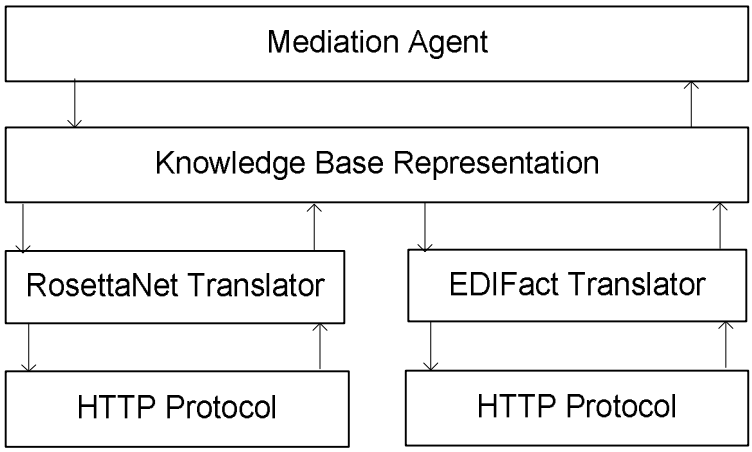

Figure 8 : Mediator technical implementation

2.3.5 The Freight Forwarder B

The Freight forwarder agent will have three main components. A “black box” EDIFact engine where the only thing we can ensure from the Freight forwarder is that is going to be an EDIFact compliant partner (follows an internal State machine) and sends XML-fied messages. It has also an OWL-S description that will be published in the marketplace, and it has a registration module (UDDI + transport capabilities) that interacts with the marketplace for its registration as a semantic web service.

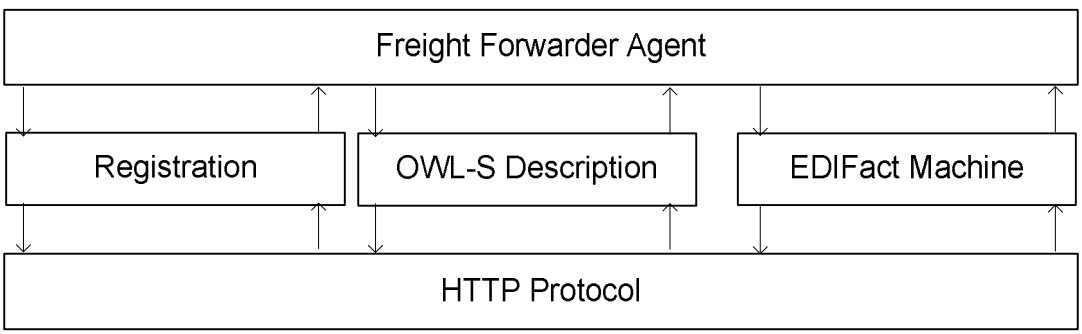

Figure 9 : The Freight Forwarder technical implementation

2.3.6 The Dummy Freight Forwarder C

As expressed in the dummy freight forwarder A (this component maybe implemented jointly with it), this component will be on a blocking loop waiting for an ASN to arrive, when that happens an ack is answered.

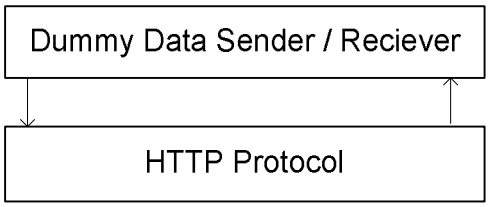

Figure 10 : Freight Forwarder technical implementation

3 Use Cases

We will present which are the different use cases or activities presented by the main components in this following UML use case diagram.

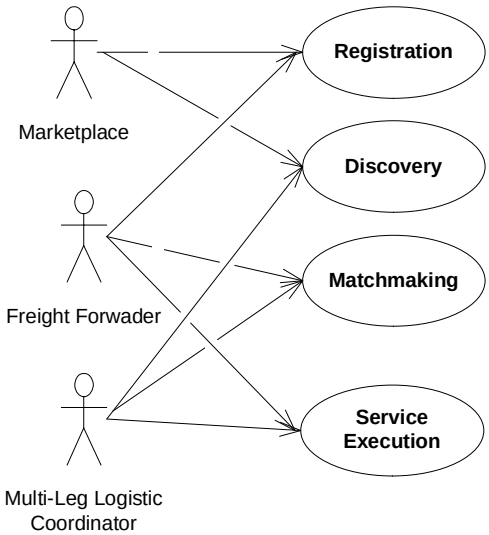

Figure 11 : Use Cases of WP8 Demonstrator

All those use cases will be executed sequentially, first there will be a registration, after that, there will be a discovery, this discovery will produce one matchmaking and finally there will be a simulation of a service execution.

3.1 Registration

This use case involves as participants the Freight Forwarder and the marketplace, this use case emerge when the freight forwarder wants to make him visible to the semantic web.

Precondition: The service that we want to register is the same type as the services accepted by the marketplace.

Post-Condition: The service is registered in the Marketplace.

In the logistic world there are different types of service, therefore we can imagine a first classification of services offered by the marketplace in terms of “type of service”, in our case we are talking about “freight forwarding”, other kind of service could be “stock maintenance”, etc.

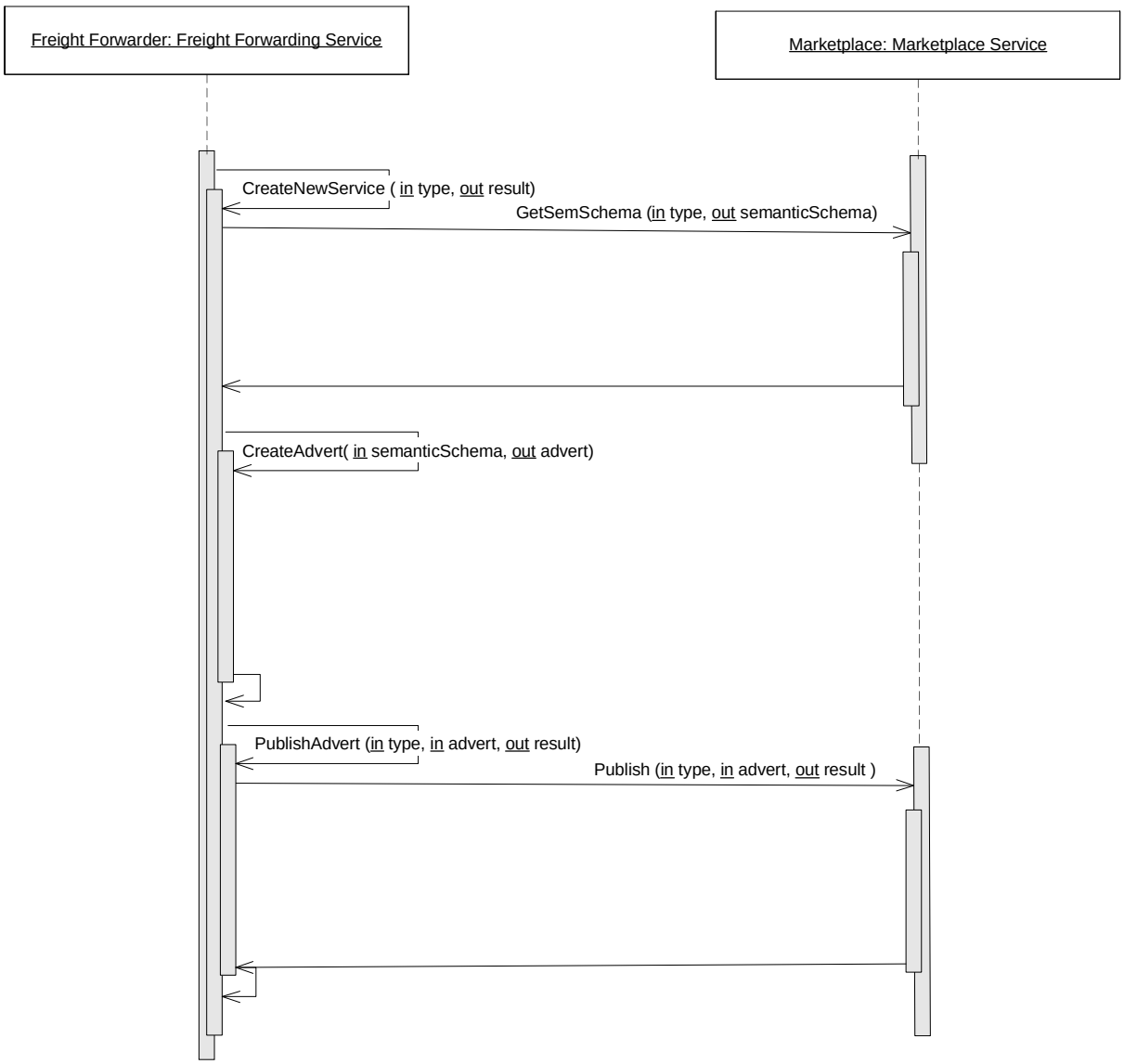

Figure 12 : UML Sequence diagram of operations of the Registration

The main operation of the use case is *CreateNewService (in Type, out Result)*. This operation will create a new entry in the "discovery" service in our case this discovery service is implemented by the marketplace. To create this entry, first we will retrieve the semantic structure of how the information should be published (This is the semantic Schema, a combination of an RDF document plus a DL ontology). This schema will be the provider side of the future contract agreement template.

	Consumer	**Provider**	Supporting Ontologies	Dependencies	Consumer fixed?	Provider fixed?
Terms						
Location	Concrete Pair	**Restrictions on possible locations**	Location Ontology	Ø Range Ø Pricing	YES	NO
Package Size	List of Triple of Measures	**Maximum measures**	Domain Ontology	Ø Pricing	YES	NO
Package Weight	List of values	**Maximum weight**	Domain Ontology	Ø Pricing	YES	NO
Payment						
Price	None or Request For Quotation.	**Quotation**	Payment Ontology		NO	YES
Type of Service	List of values	**List of values**		Ø Pricing	NO	NO
Means of Payment	List of accepted means	**List of accepted means**	Payment Ontology	Ø Preferences	NO	NO
Terms of Payment	List of accepted terms	**List of accepted terms**	Payment Ontology	Ø Preferences	NO	NO

Figure 13 : Information elements to be described in the Contract

We need a structure to store {The restrictions on possible locations, the maximum measures to be handled, the maximum weight the provider accepts, the form of a request of quotation, the list of Incoterms related type of services, the list of accepted means and the list of accepted terms of payment}.

Once the provider has this structure, the invocation to *CreateAdvert (in SemanticSchema, out Advert)* will fill this structure with the concrete values given by the business of the provider itself (e.g. If it is a local truck company that operates in UK, this information will be presented in the **restrictions of possible locations**). Once this information is filled, the system will call the procedure *PublishAdvert (in Advert, out Result)* with hopefully satisfactory results.

Note: Apart from the information described here, we will need to present also the interaction patterns of the semantic web services, this area is still a verge area in the project and no commitments are made from WP8.

3.2 Discovery

This use case involves the three parties in the architecture, after Multi-leg logistic coordinator deals with the marketplace to find a suitable provider, the marketplace asks for approval to pass more information to this Multi-leg logistic coordinator to start with the integration and negotiation process.

Precondition: The type service that we want to discover is offered by the marketplace.

Post-Condition: We will negotiate and try to integrate If a service that matches our requirements also agrees to trade.

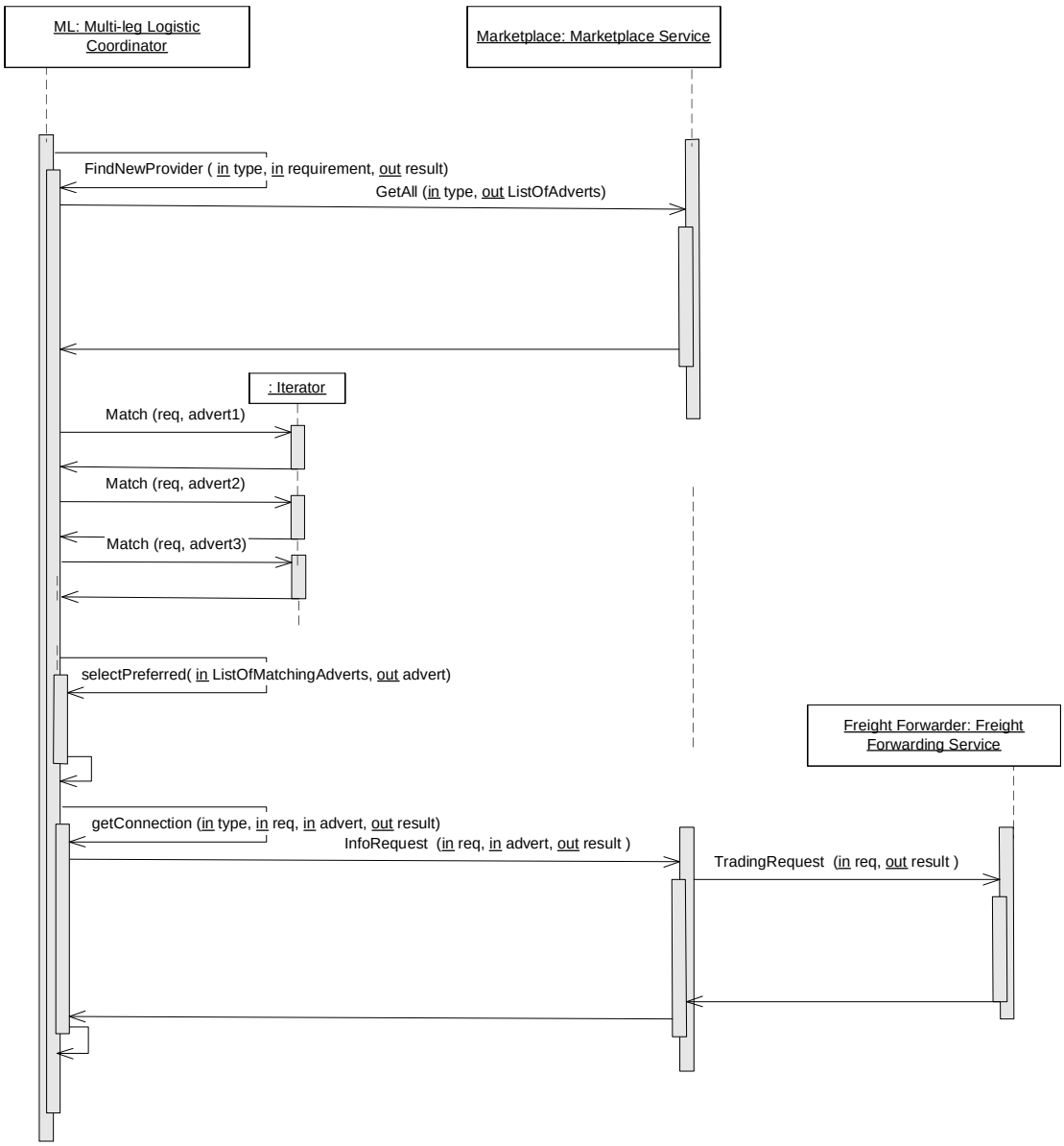

Figure 14 : UML Sequence diagram of operations of the Discovery

The main operation of the use case is *FindNewProvider (in Type, in requirements, out Result)*. This operation will examine all the entries in the "discovery" service of the type provide. First of all, we will retrieve all the instances of adverts, after having all the adverts, we will examine one by one if it matches with our requirements (our requirements in bold, the match will be made between elements in Italic)

	Consumer	Provider	Supporting Ontologies	Dependencies	Consumer fixed?	Provider fixed?
Terms						
Location	***Concrete Pair***	*Restrictions on possible locations*	Location Ontology	Ø Range Ø Pricing	YES	NO
Package Size	***List of Triple of Measures***	*Maximum measures*	Domain Ontology	Ø Pricing	YES	NO
Package Weight	***List of values***	*Maximum weight*	Domain Ontology	Ø Pricing	YES	NO
Payment						
Price	***None or Request For Quotation.***	*Quotation*	Payment Ontology		NO	YES
Type of Service	***List of values***	*List of values*		Ø Pricing	NO	NO
Means of Payment	***List of accepted means***	*List of accepted means*	Payment Ontology	Ø Preferences	NO	NO
Terms of Payment	***List of accepted terms***	*List of accepted terms*	Payment Ontology	Ø Preferences	NO	NO

Figure 15: Information elements to be described in the Contract (2)

We can imagine more than one advert will match our requirements, therefore there will be an operation to select the most appropriate implementing our desires. This operation is *selectPreferred (in ListofMatchingAdverts, out Advert).* Once a single advert is selected, the Multi-leg logistic coordinator will ask for details of connection *getConnection(in type, in advert, out result),* This will invoke a request to the marketplace *InfoRequest(in request, in advert, out result).* This request for trading (and negotiation) will be forwarded from the marketplace to the freight forwarder to ask for an explicit approval with hopefully satisfactory results.

Note: Apart from the information described here, we will need to present also the interaction patterns of the semantic web services, this area is still a verge area in the project and no commitments are made from WP8.

3.3 Matchmaking

This use case appears after the discovery. The service consumer will examine the interfaces offered by the service provider and being those compatible, will accept to sign a contract of service

Precondition: The contract reflects our requirements.

Post-Condition: If the interaction is possible, the contract is signed.

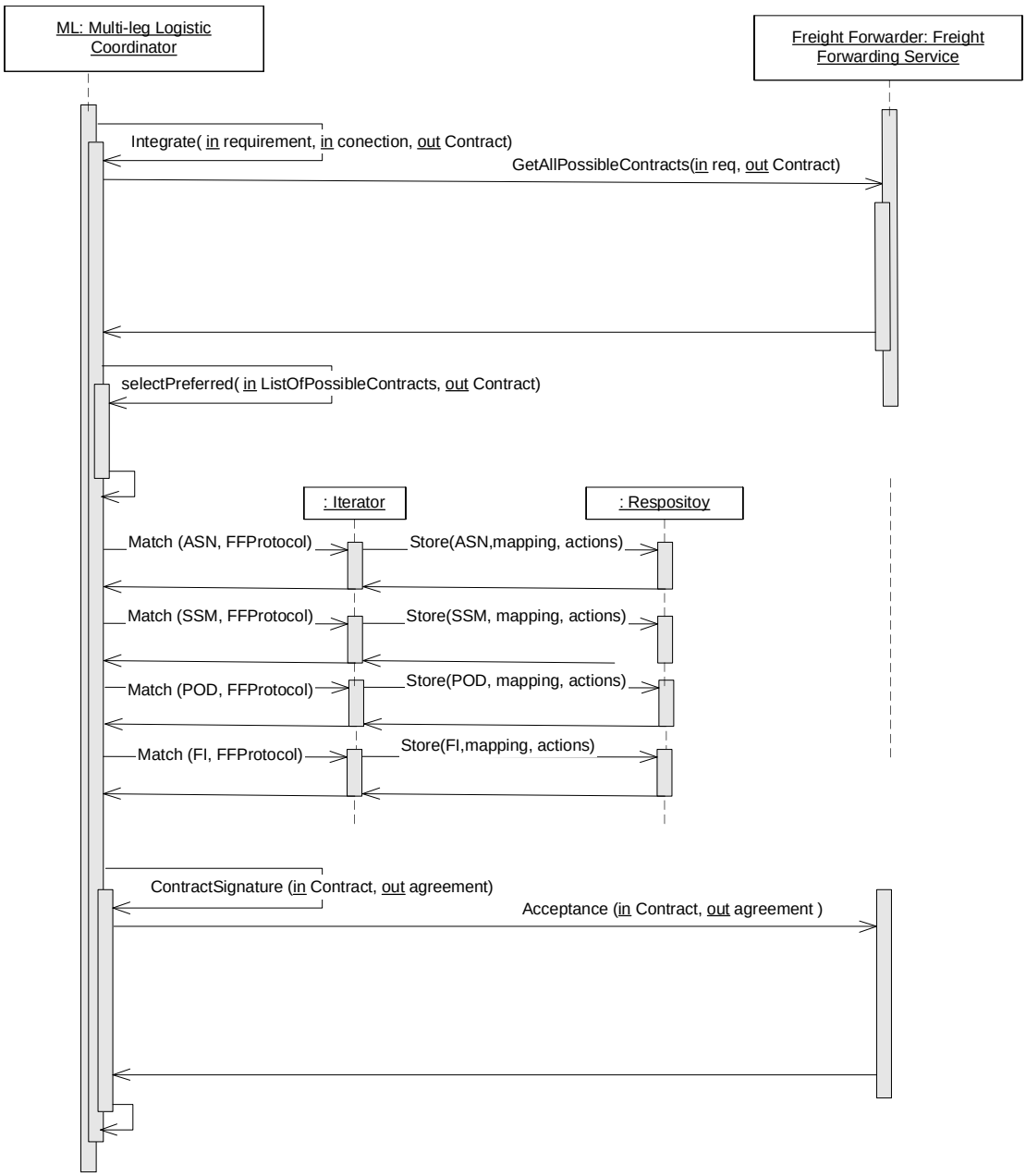

Figure 16 : UML Sequence diagram of operations of the Matchmaking

The main operation of the use case is *Integrate (in requirement, in connection, out Contractt)*. This operation will take the information about the connection provided by the marketplace and will start the interaction with the Freight forwarder. To begin with, the Multi-leg coordinator will retrieve all possible contracts (or ways to deliver a service). From all those, the agent will have an specific candidate, for this candidate, the service consumer will examine how it is implemented and will "derive" which are actions needed for a mediation (protocol and data). All this information will be stored in our repository for mediation. In the case of data, most of the time will be inspect the scripts that will allow the lift (and lower) of all the information into the knowledge base. In the case of the protocol mediation, the mediator has to be a transparent component in the communication (none of the interfaces of the edges could change) but the sequence may not match completely on the message exchange between partners. Therefore, somehow this information has to be stored. As the case study has appointed before, this is the case of the sending / receiving an ASN between a RosettaNet party and a EDIFact party.

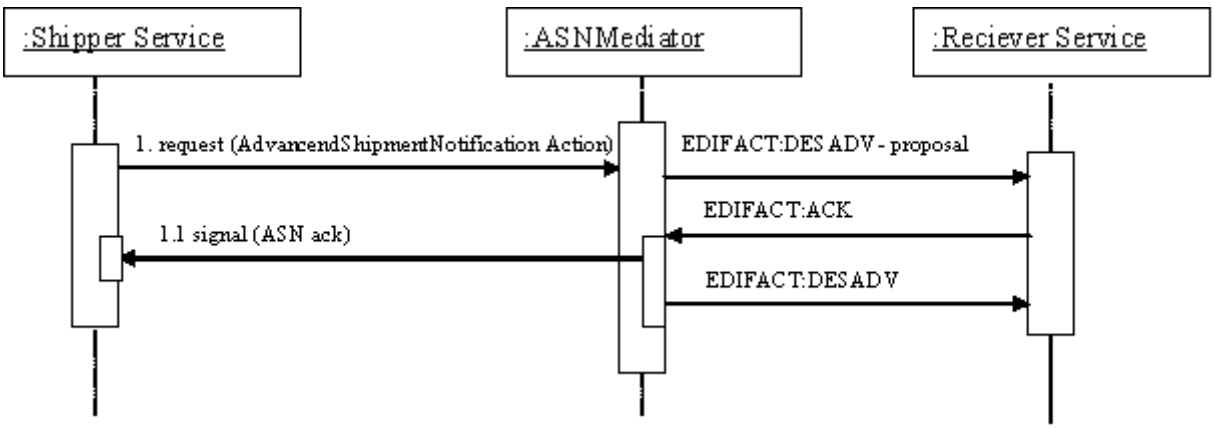

Figure 17 : Use of a mediator in the message exchange definition

Once all the interactions have been examined, if all of them could be performed either by direct invocation or by the use of mediators, the Multi-leg logistic coordinator will propose the signature of a contract reaching both parties an agreement of collaboration.

3.4 Execution

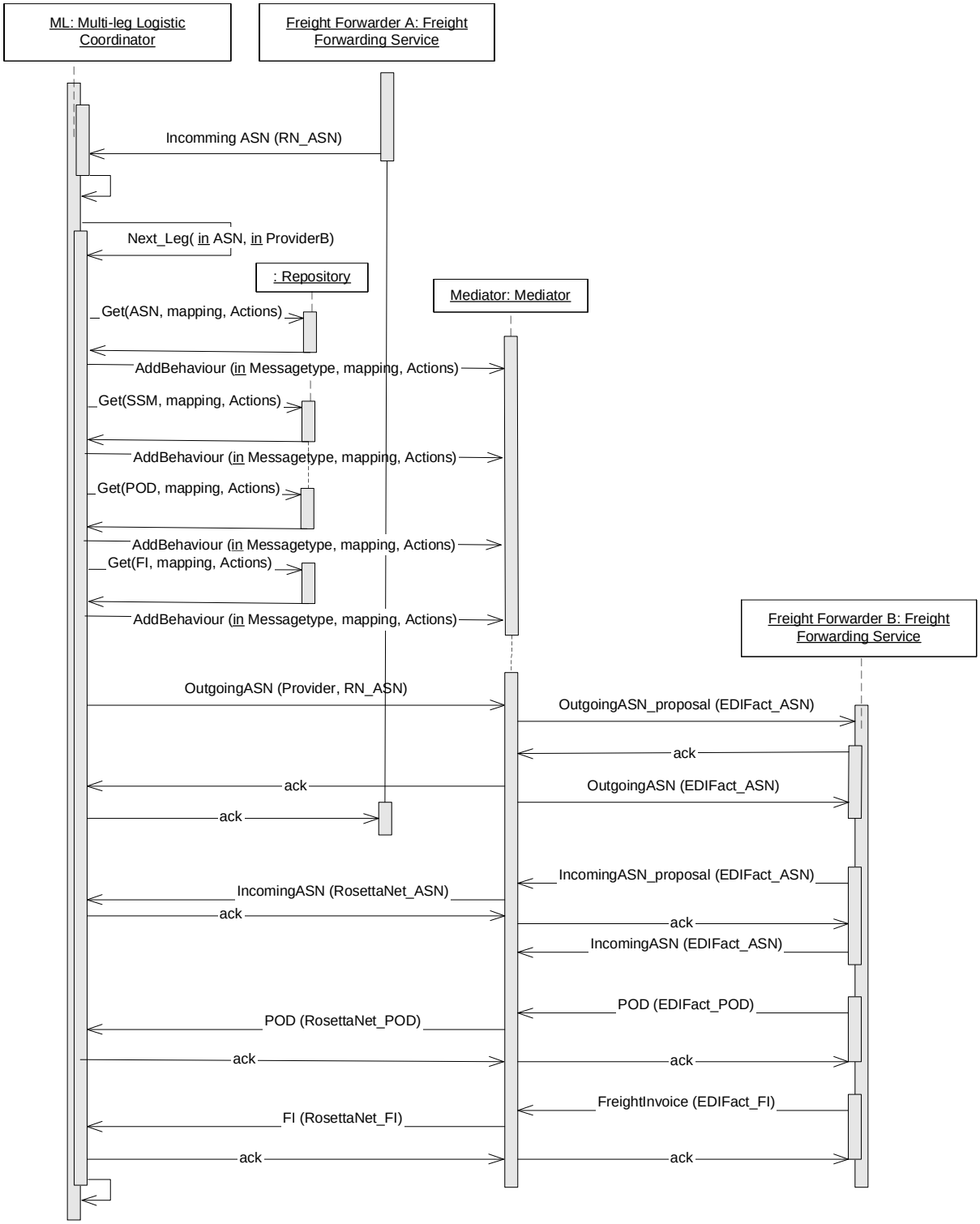

Figure 18 : UML Sequence diagram of operations of the Service Execution

This is the main use case of the case study, we will look at the service when the transition of items is due between the first freight forwarder and the freight forwarder we have integrated. An incoming ASN will trigger the main operation of the use case *Next_Leg (in ASN, in ProviderB, out Result)*.

The first thing to be done is to generate in runtime the mediator component that will allow any further communication with this new provider. We will look-up in our internal repository to upload its behaviour. Once this mediator is set-up, we will forward the ASN request that we had received to the Freight Forwarder B, once we receive the confirmation to go ahead we will give this confirmation to the previous Freight Forwarder.

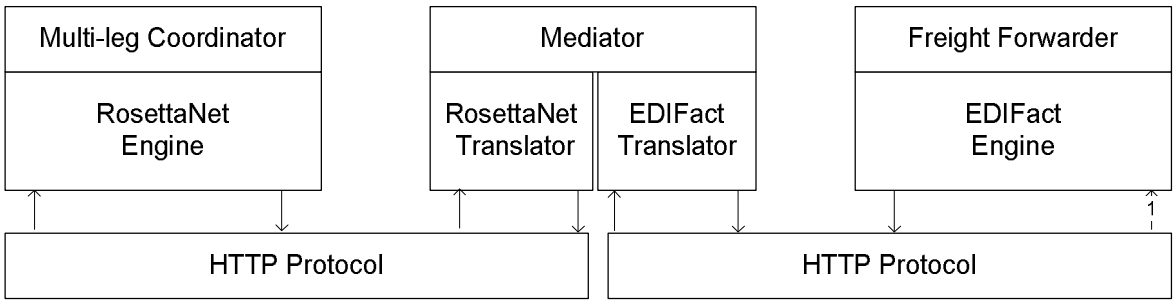

Figure 19 : Communication of a RosettaNet Partner to an EDIFact one through a mediator

After the mediation is setup and the logistic provider holds the item, we execute the standard logistic message exchange sequences.

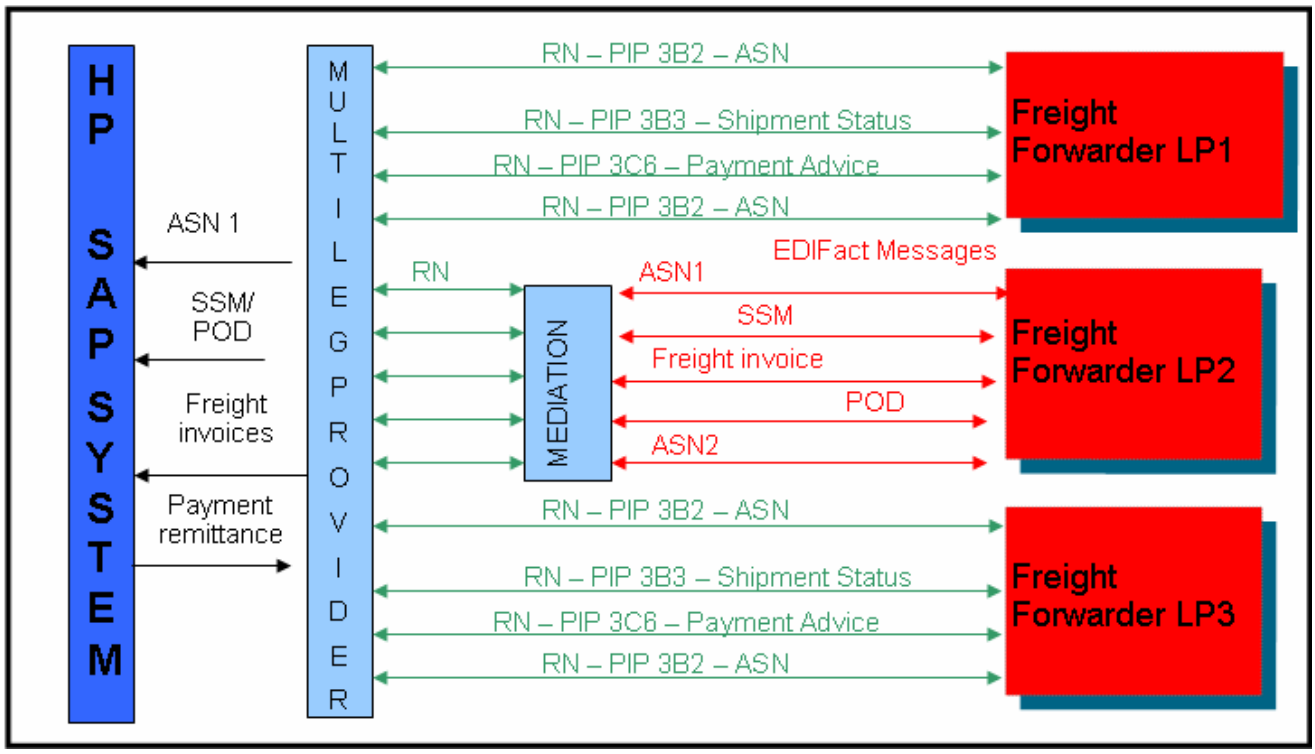

Figure 20 : Message exchange during logistic transactions

4 UML Class diagrams of the components

This section covers the data model and what operations are available for each of the components (already introduced in the use case sequence diagrams). After the general diagram, each of the classes is presented individually.

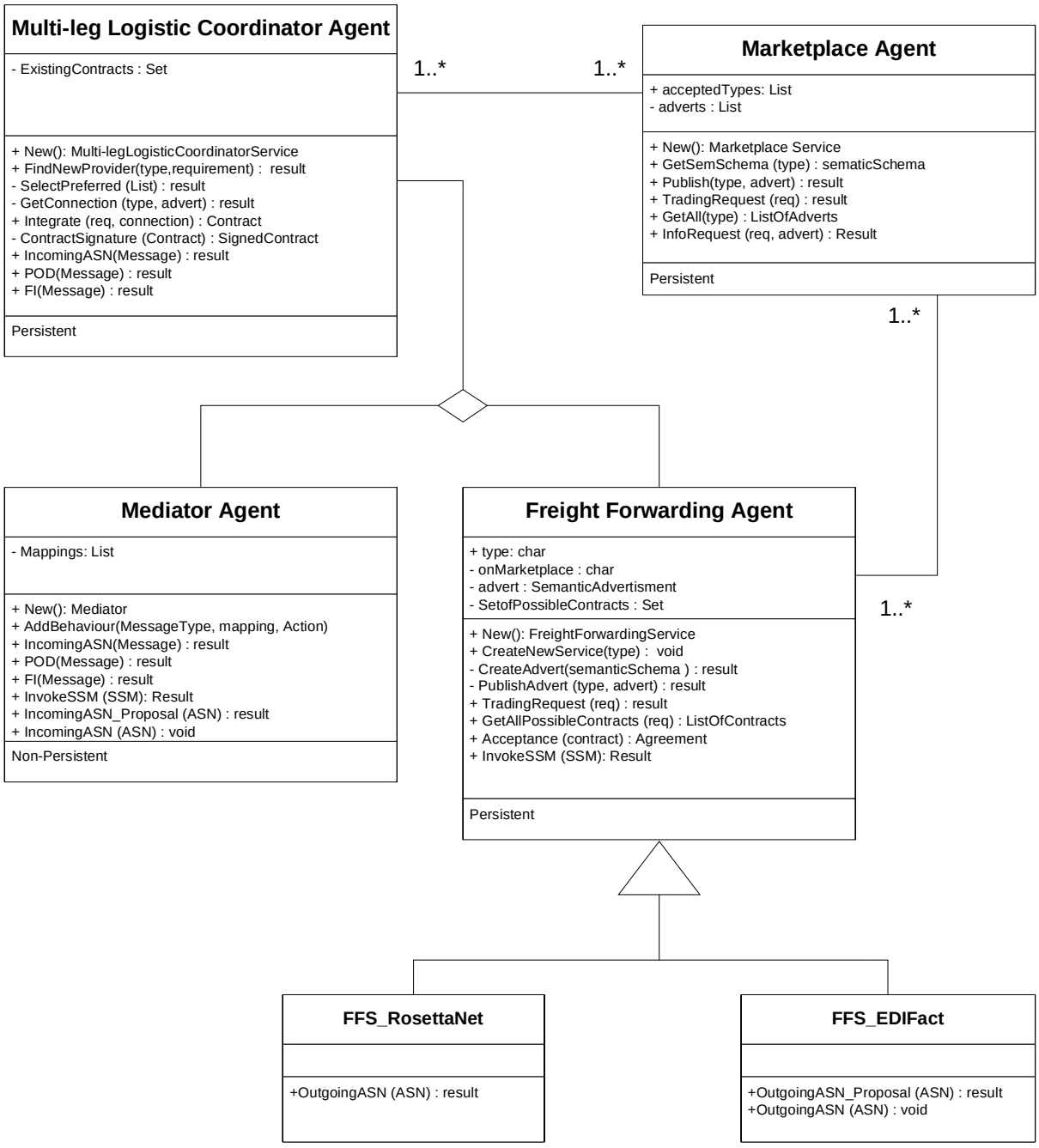

Figure 21 : WP8 Technical Architecture UML Class Diagram

4.1 The Marketplace

We have seen during the previous sections how the marketplace fulfils the role of the main look-up service; it holds a private list of adverts only accessible by invocation of the operation *GetAll.* It publishes publicly the accepted types of service. There are no private operations.

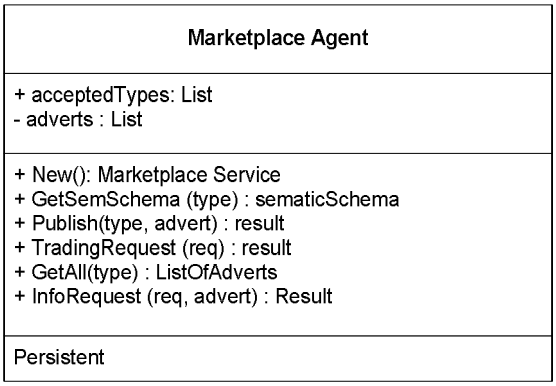

Figure 22 : UML Class diagram of the Marketplace

4.2 The Multi-leg logistic provider

The multi-leg coordinator is another persistent entity in the system, it has 6 public operations: The creation [*New()*], the discovery invocation [*FindNewProvider(…)*], the matchmaking [*Integrater(…)*] and the ones related to the execution [ASN, POD, FI].

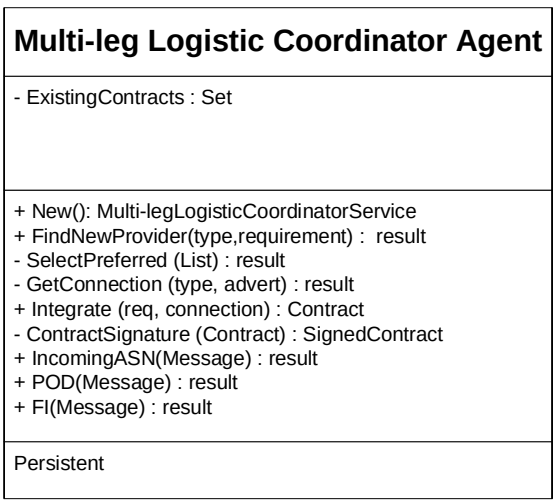

Figure 23 : UML Class diagram of the Multi-leg Logistic Coordinator

4.3 The Freight Forwarder

The freight forwarder component will be a parent class that will hold all the operations that had appear in the sequence diagrams, this class will have two different specializations because the signature of the operation “incoming ASN” is different in the case of RosettaNet and EDIFact, therefore we can not override a single operation. All the classes here has to be persistent as the calls will be concurrent (check in the execution sequence that the Freight forwarding A do not receive the ack until it has been granted from the Freight forwarder B)

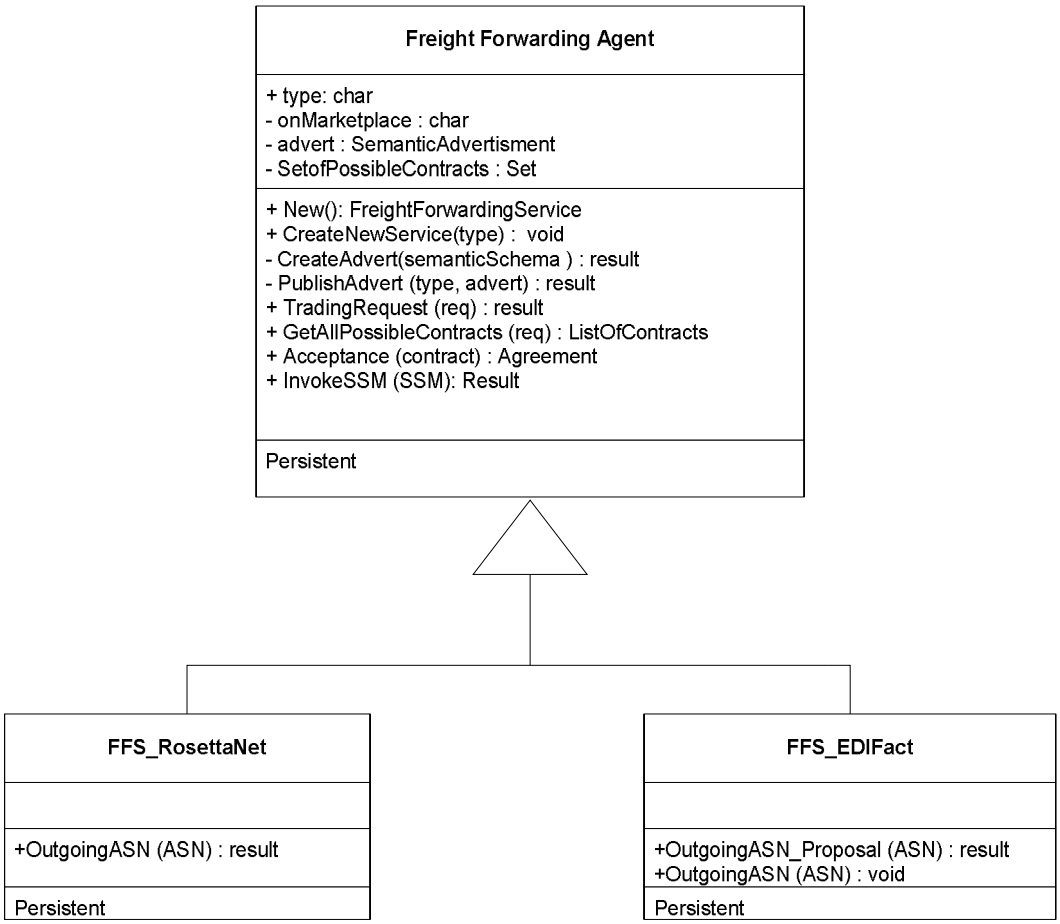

Figure 24 : UML Class diagram of Freight Forwarding Service

4.4 Mediator component

The single non-persistent component in our architecture is the mediator. The mediator component is created in run-time, its behaviour will be inducted by the public call *AddBehavior()*. This will load its knowledge base with the mappings from one protocol to the other and the actions to be taken when different operations are invoked.

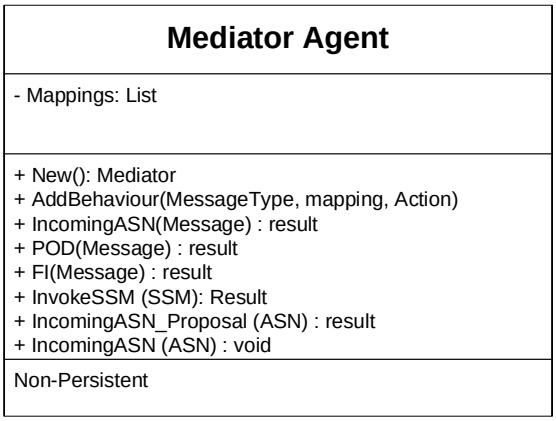

Figure 25 : UML Class diagram of the Mediation