Functional Specification of 'Delivery'

Put author(s) here

(This document was generated by Ampersand v2.2.0.337, build time: 30-Nov-11 8:00.16)

Wed Nov 30 08:27:19 W. Europe Standard Time 2011

Contents

Chapter 1

Introduction

This document defines the functionality of an information system called 'Delivery'. It defines business services in a system where people and applications work together in order to fullfill their commitments. A number of these rules have been used as functional requirement to assemble this functional specification[1]. Those rules are listed in chapter 2, ordered by theme.

The diagnosis in chapter 3 is meant to help the authors identify shortcomings in their Ampersand script.

The conceptual analysis in chapter 4 is meant for requirements engineers and architects to validate and formalize the requirements from chapter 2. It is also meant for testers to come up with correct test cases. The formalization in this chapter makes consistency of the functional specification provable. It also yields an unambiguous interpretation of all requirements.

Chapters that follow have the builders of 'Delivery' as their intended audience. The data analysis in chapter 7 describes the data sets upon which 'Delivery' is built. Each subsequent chapter defines one business service. This allows builders focus on a single service at a time. Together, these services fulfill all commitments from chapter 2. By disclosing all functionality exclusively through these services, 'Delivery' ensures compliance to all rules from chapter 2.

[1] To use agreements as functional requirements characterizes the Ampersand approach, which has been used to produce this document.

Chapter 2

Shared Language

This chapter defines the natural language, in which functional requirements of 'Delivery' can be discussed and expressed. The purpose of this chapter is to create shared understanding among stakeholders. The language of 'Delivery' consists of concepts and basic sentences. All functional requirements are expressed in these terms. When stakeholders can agree upon this language, at least within the scope of 'Delivery', they share precisely enough language to have meaningful discussions about functional requirements. All definitions have been numbered for the sake of traceability.

2.1 Deliveries

Requirement 1:

Requirement 2:

Requirement 3:

Requirement 4:

Requirement 5:

Requirement 6:

Requirement 7:

Requirement 8:

Requirement 9:

Requirement 10:

Requirement 11:

We want orders to be delivered correctly, with only items that are mentioned on the order.

Requirement 12 (correct delivery): Each item in a delivery is mentioned on the order.

We want orders to be delivered completely, with no items missing.

Requirement 13 (complete delivery): Every item on an order must also be in the corresponding delivery.

Accepting an order is always done by the provider to whom the order was addressed. To prevent an order to be accepted or rejected by anyone else, we need this requirement.

Requirement 14 (proper address): A provider can only accept or reject orders that are addressed to that provider.

In this context, providers only deliver when there is an order. So, if a delivery is made by a provider, we assume the existence of an order that is accepted by that provider.

Requirement 15 (order based delivery): For every delivery a provider has made, there exists an accepted order.

To prevent arbitrary payments, we enforce that every invoice is paid by the client to whom it was sent.

Requirement 16 (correct payments): Payments are made only for invoices sent.

To make sure that deliveries are billed to the right customer, there must be a delivery for each invoice sent.

Requirement 17 (correct invoices): Invoices are sent to a customer only if there is a delivery made to that customer.

2.2 Sessions

Requirement 18:

Requirement 19:

2.3 Loose ends...

This paragraph shows remaining fact types and concepts that have not been described in previous paragraphs.

Requirement 20 (login):

Orders should be deliverable and payable. For such purposes, it is necessary to know the client (customer) that created the order

Requirement 21 (create orders): Each order is created by precisely one client.

Not every order received by a provider leads to a delivery. The provider may decide to accept or to reject an order. Eventually, all orders must be acknowledged, either positively (accept) or negatively (reject).

Orders are issued by clients for the purpose of making a sales transaction. At some point in time, a provider must accept or reject the order. In this simplistic model, every order will be accepted. A more realistic model should at least ensure that orders will not be lost.

Requirement 22 (accept orders): Orders addressed to a provider must be accepted or rejected by that provider.

Ultimately, each order accepted must be shipped by the provider who has accepted that order. The provider will be signalled of orders waiting to be shipped.

Requirement 23 (ship orders): Each order that has been accepted by a provider must (eventually) be shipped by that provider.

A client who receives an invoice must eventually pay.

Requirement 24 (pay invoices): All invoices sent to a customer must be paid by that customer.

The ordered goods must be delivered at some point in time to the client. This is done in one delivery.

Requirement 25 (receive goods): Every delivery must be acknowledged by the client who placed the corresponding order.

In order to induce payment, a provider sends an invoice for deliveries made.

Requirement 26 (send invoices): After a delivery has been made to a customer, an invoice must be sent.

Chapter 3

Diagnosis

This chapter provides an analysis of the Ampersand script of 'Delivery'. This analysis is intended for the authors of this script. It can be used to complete the script or to improve possible flaws.

Delivery assigns rules to roles. The following table shows the rules that are being maintained by a given role.

rule	Client	Provider
login	⊙	⊙
create orders	⊙	
accept orders		⊙
ship orders		⊙
pay invoices	⊙	
receive goods	⊙	
send invoices		×

Concepts Order, Client, Product, Delivery, Provider, Invoice, and Session remain without a purpose.

The purpose of relations *sProvider*, *sClient*, *from*, *addressedTo*, *item*, *rejected*, *accepted*, *of*, *delivery*, *from*, *sentTo*, *paid*, *item*, *provided*, and *deliveredTo* is not documented.

Relation *from* is not being used in any rule.

Rule *login* on line 142 of file F:\RJ$\Prive\CC model repository\Adlfiles\Delivery.adl is not documented.

All rules in process Delivery are linked to roles.

All role-rule assigments involve rules that are defined in process 'Delivery'.

The following table represents the population of various relations.

Concept	Population
Order	3
Client	3
Product	3
Delivery	3
Provider	2
Invoice	3
Session	3

Relation	Population
from : *Order* × *Client*	3
item : *Order* × *Product*	3
item : *Delivery* × *Product*	3
of : *Delivery* × *Order*	3
provided : *Provider* × *Delivery*	3
accepted : *Provider* × *Order*	3
rejected : *Provider* × *Order*	3
addressedTo : *Order* × *Provider*	3
deliveredTo : *Delivery* × *Client*	3
sentTo : *Invoice* × *Client*	3
delivery : *Invoice* × *Delivery*	3
from : *Invoice* × *Provider*	3
paid : *Client* × *Invoice*	3
sClient : *Session* × *Client*	3

The population in this script does not specify any work in progress.

The population in this script violates no rule.

Chapter 4

Conceptual Analysis

This chapter provides an analysis of the principles described in chapter 2. Each section in that chapter is analysed in terms of relations and each principle is then translated in a rule.

4.1 Deliveries

Figure 4.1 shows a conceptual diagram of this theme.

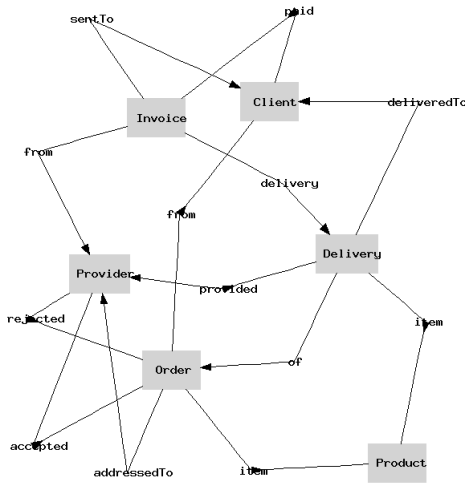

Figure 4.1: Concept diagram of Deliveries

correct delivery We want orders to be delivered correctly, with only items that are mentioned on the order.

To arrive at the formalization in equation 4.4, the following three relations are introduced.

$$\begin{aligned} item &: Order \times Product && (4.1)\\ of &: Delivery \to Order && (4.2)\\ item &: Delivery \times Product && (4.3) \end{aligned}$$

This means:

$$item_{[Delivery \times Product]} \vdash of; item \tag{4.4}$$

This corresponds to requirement 2.1 on page 4.

complete delivery We want orders to be delivered completely, with no items missing.

We use definitions 4.1 (*item*), 4.2 (*of*), and 4.3 (*item*). This means:

$$of; item \vdash item_{[Delivery \times Product]} \tag{4.5}$$

proper address Accepting an order is always done by the provider to whom the order was addressed. To prevent an order to be accepted or rejected by anyone else, we need this requirement.

To arrive at the formalization in equation 4.9, the following three relations are introduced.

$$\begin{aligned} addressedTo &: Order \to Provider && (4.6)\\ rejected &: Provider \times Order && (4.7)\\ accepted &: Provider \times Order && (4.8) \end{aligned}$$

This means:

$$accepted \cup rejected \vdash addressedTo^{\smile} \tag{4.9}$$

This corresponds to requirement 2.1 on page 4.

order based delivery In this context, providers only deliver when there is an order. So, if a delivery is made by a provider, we assume the existence of an order that is accepted by that provider.

In order to formalize this, a relation provided is introduced (4.10):

$$provided : Provider \times Delivery \tag{4.10}$$

We also use definitions 4.8 (*accepted*) and 4.2 (*of*) to formalize requirement 2.1 (page 4): This means:

$$provided \vdash accepted; of^{\smile} \tag{4.11}$$

correct payments To prevent arbitrary payments, we enforce that every invoice is paid by the client to whom it was sent.

To arrive at the formalization in equation 4.14, the following two relations are introduced.

$$sentTo \quad : \quad Invoice \rightarrow Client \tag{4.12}$$
$$paid \quad : \quad Client \times Invoice \tag{4.13}$$

This means:

$$paid \vdash sentTo^{\smile} \tag{4.14}$$

This corresponds to requirement 2.1 on page 4.

correct invoices To make sure that deliveries are billed to the right customer, there must be a delivery for each invoice sent.

To arrive at the formalization in equation 4.17, the following two relations are introduced.

$$from \quad : \quad Order \times Client \tag{4.15}$$
$$delivery \quad : \quad Invoice \rightarrow Delivery \tag{4.16}$$

We also use definitions 4.12 (*sentTo*) and 4.2 (*of*). This means:

$$sentTo \vdash delivery; of; from \tag{4.17}$$

This corresponds to requirement 2.1 on page 4.

4.2 Sessions

Figure 4.2 shows a conceptual diagram of this theme.

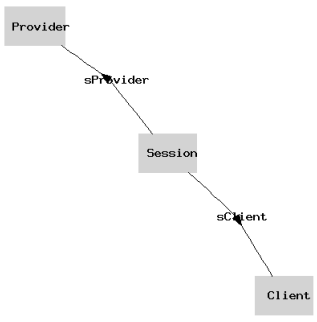

Figure 4.2: Concept diagram of Sessions

Chapter 5

Process Analysis

Delivery assigns rules to roles. The following table shows the rules that are being maintained by a given role.

Role	Rule
Client	login create orders pay invoices receive goods
Provider	login accept orders ship orders send invoices

Delivery assigns roles to relations. The following table shows the relations, the content of which can be altered by anyone who fulfills a given role.

Role	Relation
Client	*sClient* *from* *item* *paid* *deliveredTo*
Provider	*sProvider* *accepted* *rejected* *item* *item*
	$of : Delivery \times Order$ $provided : Provider \times Delivery$ $addressedTo : Order \times Provider$ $sentTo : Invoice \times Client$ $delivery : Invoice \times Delivery$ $from : Invoice \times Provider$

5.1 Delivery

Figure 5.1 shows the process model.

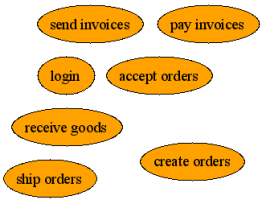

Figure 5.1: Process model of Delivery

The conceptual diagram of figure 5.2 provides an overview of the language in which this process is expressed.

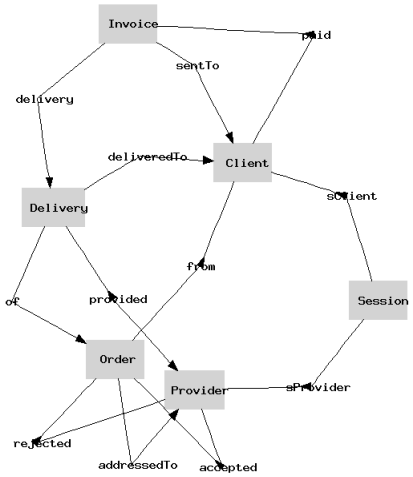

Figure 5.2: Basic sentences of Delivery

login We use definitions **??** (*sProvider*) and **??** (*sClient*). Activities that are defined by this rule are finished when:

$$I_{[Session]} \vdash (sProvider; V \cap \overline{sClient}; V) \cup (sClient; V \cup \overline{sProvider}; V) \quad (5.1)$$

These activities are signalled by:

$$I_{[Session]} \vdash (sProvider; V \cap \overline{sClient}; V) \cup (sClient; V \cup \overline{sProvider}; V) \quad (5.2)$$

create orders Orders should be deliverable and payable. For such purposes, it is necessary to know the client (customer) that created the order

We use definition 4.15 (from). Activities that are defined by this rule are finished when:

$$I_{[Order]} \vdash from; from^{\smile} \tag{5.3}$$

These activities are signalled by:

$$I_{[Order]} \vdash from; from^{\smile} \tag{5.4}$$

accept orders Not every order received by a provider leads to a delivery. The provider may decide to accept or to reject an order. Eventually, all orders must be acknowledged, either positively (accept) or negatively (reject).

Orders are issued by clients for the purpose of making a sales transaction. At some point in time, a provider must accept or reject the order. In this simplistic model, every order will be accepted. A more realistic model should at least ensure that orders will not be lost.

We use definitions 4.8 (*accepted*), 4.7 (*rejected*), and 4.6 (*addressedTo*). Activities that are defined by this rule are finished when:

$$addressedTo^{\smile} \vdash accepted \cup rejected \tag{5.5}$$

These activities are signalled by:

$$addressedTo^{\smile} \vdash accepted \cup rejected \tag{5.6}$$

ship orders Ultimately, each order accepted must be shipped by the provider who has accepted that order. The provider will be signalled of orders waiting to be shipped.

We use definitions 4.2 (*of*), 4.10 (*provided*), and 4.8 (*accepted*). Activities that are defined by this rule are finished when:

$$accepted \vdash provided; of \tag{5.7}$$

These activities are signalled by:

$$accepted \vdash provided; of \tag{5.8}$$

pay invoices A client who receives an invoice must eventually pay.

We use definitions 4.12 (*sentTo*) and 4.13 (*paid*). Activities that are defined by this rule are finished when:

$$sentTo^{\smile} \vdash paid \tag{5.9}$$

These activities are signalled by:

$$sentTo^{\smile} \vdash paid \tag{5.10}$$

receive goods The ordered goods must be delivered at some point in time to the client. This is done in one delivery.

We use definitions 4.15 (*from*), 4.2 (*of*), and **??** (*deliveredTo*). Activities that are defined by this rule are finished when:

$$of; from \vdash deliveredTo \tag{5.11}$$

These activities are signalled by:

$$of; from \vdash deliveredTo \tag{5.12}$$

send invoices In order to induce payment, a provider sends an invoice for deliveries made.

We use definitions 4.15 (*from*), 4.2 (*of*), 4.12 (*sentTo*), and 4.16 (*delivery*). Activities that are defined by this rule are finished when:

$$delivery; of; from \vdash sentTo \tag{5.13}$$

These activities are signalled by:

$$delivery; of; from \vdash sentTo \tag{5.14}$$

Chapter 6

Function Point Analysis

The specification of 'Delivery' has been analysed by counting function points[**?**]. This has resulted in an estimated total of 49 function points.

data set	analysis	FP
Invoice	ILGV Eenvoudig	7
Order	ILGV Eenvoudig	7
Delivery	ILGV Eenvoudig	7
Session	ILGV Eenvoudig	7
Provider	ILGV Eenvoudig	7
Product	ILGV Eenvoudig	7
Client	ILGV Eenvoudig	7

interface	analysis	FP
login	NO	0
create orders	NO	0
accept orders	NO	0
ship orders	NO	0
pay invoices	NO	0
receive goods	NO	0
send invoices	NO	0

Chapter 7

Data structure

The requirements, which are listed in chapter 2, have been translated into the data model in figure 7.1. There are two data sets, two associations and no aggregations. Delivery has a total of 7 concepts.

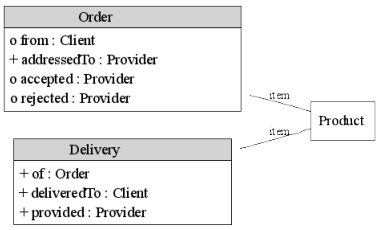

Figure 7.1: Data model of Delivery

Relation	Description	Properties
$from: \ Order \times Client$		UNI
$item: \ Order \times Product$		
$item: \ Delivery \times Product$		
$of: \ Delivery \times Order$		UNI, TOT
$provided: \ Provider \times Delivery$		SUR, INJ
$accepted: \ Provider \times Order$		INJ
$rejected: \ Provider \times Order$		INJ
$addressedTo: \ Order \times Provider$		UNI, TOT
$deliveredTo: \ Delivery \times Client$		UNI, TOT
$sentTo: \ Invoice \times Client$		UNI, TOT
$delivery: \ Invoice \times Delivery$		UNI, TOT
$from: \ Invoice \times Provider$		UNI, TOT
$paid: \ Client \times Invoice$		INJ
$sProvider: \ Session \times Provider$		UNI
$sClient: \ Session \times Client$		UNI

7.1 Invoice

The attributes in Invoice have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Invoice	√	√
sentTo	Client	√	
delivery	Delivery	√	
from	Provider	√	
paid	Client		

Within this data set, the following integrity rule shall be true at all times.

$$paid \vdash sentTo^{\smile}$$

This data set generates one process rule.

$$sentTo^{\smile} \vdash paid$$

7.2 Order

The attributes in Order have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Order	√	√
from	Client		
addressedTo	Provider	√	
accepted	Provider		
rejected	Provider		

Within this data set, the following integrity rule shall be true at all times.

$$accepted \cup rejected \vdash addressedTo^{\smile}$$

This data set generates the following process rules.

- $$I_{[Order]} \vdash from; from^{\smile}$$
- $$addressedTo^{\smile} \vdash accepted \cup rejected$$

The following rules define the integrity of data within this data set. They must remain true at all times.

- $$\overline{I} \cup from; from^{\smile}$$
- $$\overline{I} \cup I^{\smile}_{[Order]}; from; from^{\smile}$$
- $$\overline{I} \cup from; from^{\smile}; I^{\smile}_{[Order]}$$

7.3 Delivery

The attributes in Delivery have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Delivery	√	√
of	Order	√	
deliveredTo	Client	√	
provided	Provider		

7.4 Session

The attributes in Session have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Session	√	√
sProvider	Provider		
sClient	Client		

The following rules define the integrity of data within this data set. They must remain true at all times.

- $$\overline{I} \cup (sProvider; V \cap \overline{sClient}; V) \cup sClient; V \cup \overline{sProvider}; V$$
- $$\overline{I} \cup I^{\smile}_{[Session]}; (sProvider; V \cap \overline{sClient}; V) \cup I^{\smile}_{[Session]}; sClient; V \cup I^{\smile}_{[Session]}; \overline{sProvider}; V$$
- $$\overline{I} \cup (sProvider; V \cap \overline{sClient}; V); I^{\smile}_{[Session]} \cup sClient; V; I^{\smile}_{[Session]} \cup \overline{sProvider}; V; I^{\smile}_{[Session]}$$

7.5 item1

The attributes in item1 have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Order	√	
Product	Product	√	

7.6 item2

The attributes in item2 have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Delivery	√	
Product	Product	√	

Chapter 8

login

Activity ‘login’ must be performed by a user with role Client or Provider. Figure 8.1 shows the knowledge graph of this interface.

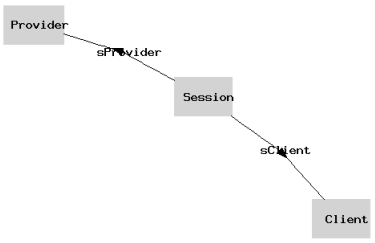

Figure 8.1: Language diagram of login

Every section in this chapter describes one activity. While performing an activity, users will insert or delete population in various relations. This may potentially violate invariants. (An invariant is a business rule rules that must remain true at all times.) The software to maintain the truth of invariant rules is generated automatically. The structure of that software is illustrated by a so called switchboard diagram, the first of which you will find in figure X. Each switchboard diagram consists of three columns: Invariant rules are drawn in the middle, and relations occur on the (right and left hand) sides. An arrow on the left hand side points from a relation that may be edited to a rule that may be violated as a consequence thereof. Each arrow on the right hand side of a rule represents an edit action that is required to restore its truth. It points to the relation that is edited for that purpose. If that arrow is labeled ’+’, it involves an insert event; if labeled ’-’ it refers to a delete event. This yields an accurate perspective on the way in which invariants are maintained.

Figure 8.2 shows the switchboard diagram of this interface.

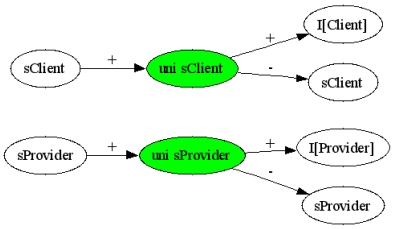

Figure 8.2: Switchboard of login

Chapter 9

create orders

Activity ‘create orders’ must be performed by a user with role Client. During that activity, rule ‘correct invoices’ will be maintained without intervention of a user. Figure 9.1 shows the knowledge graph of this interface.

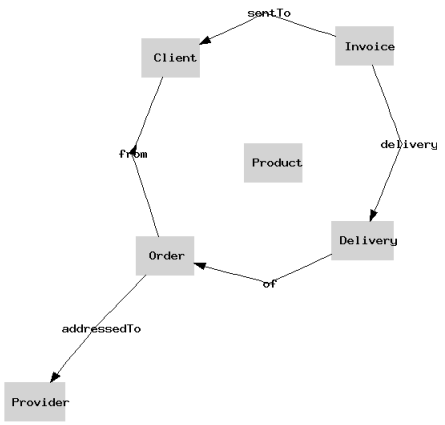

Figure 9.1: Language diagram of create orders

Figure 9.2 shows the switchboard diagram of this interface.

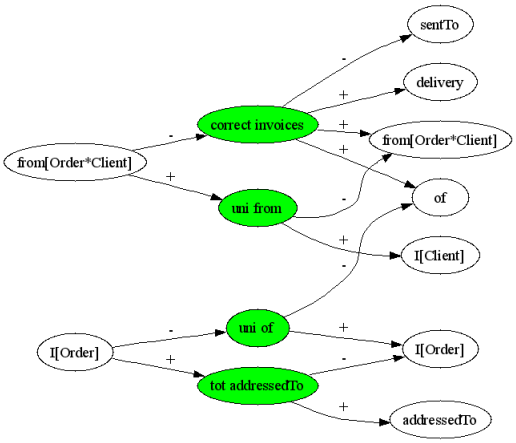

Figure 9.2: Switchboard of create orders

Chapter 10

accept orders

Activity ‘accept orders’ must be performed by a user with role Provider. During that activity, rules ‘proper address’ and ‘order based delivery’ will be maintained without intervention of a user. Figure 10.1 shows the knowledge graph of this interface.

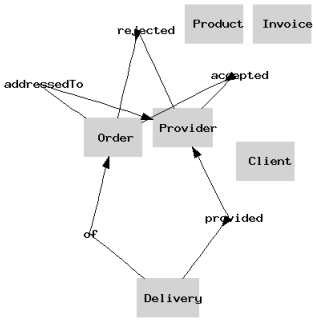

Figure 10.1: Language diagram of accept orders

Figure 10.2 shows the switchboard diagram of this interface.

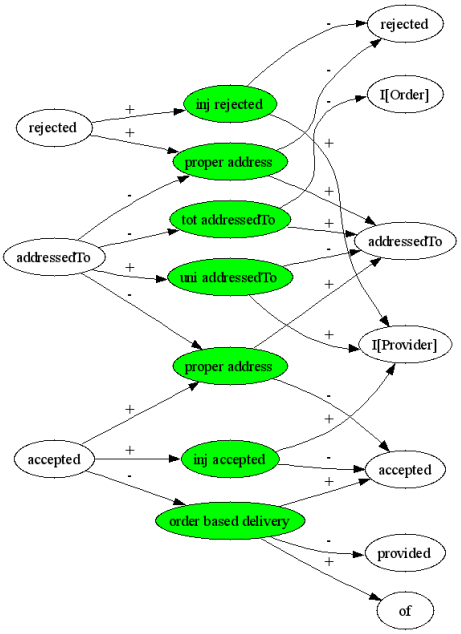

Figure 10.2: Switchboard of accept orders

Chapter 11

ship orders

Activity ‘ship orders’ must be performed by a user with role Provider. During that activity, rules ‘correct delivery’, ‘complete delivery’, ‘proper address’, ‘order based delivery’, and ‘correct invoices’ will be maintained without intervention of a user. Figure 11.1 shows the knowledge graph of this interface.

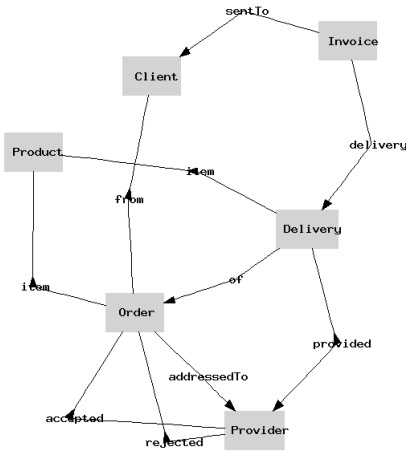

Figure 11.1: Language diagram of ship orders

Figure 11.2 shows the switchboard diagram of this interface.

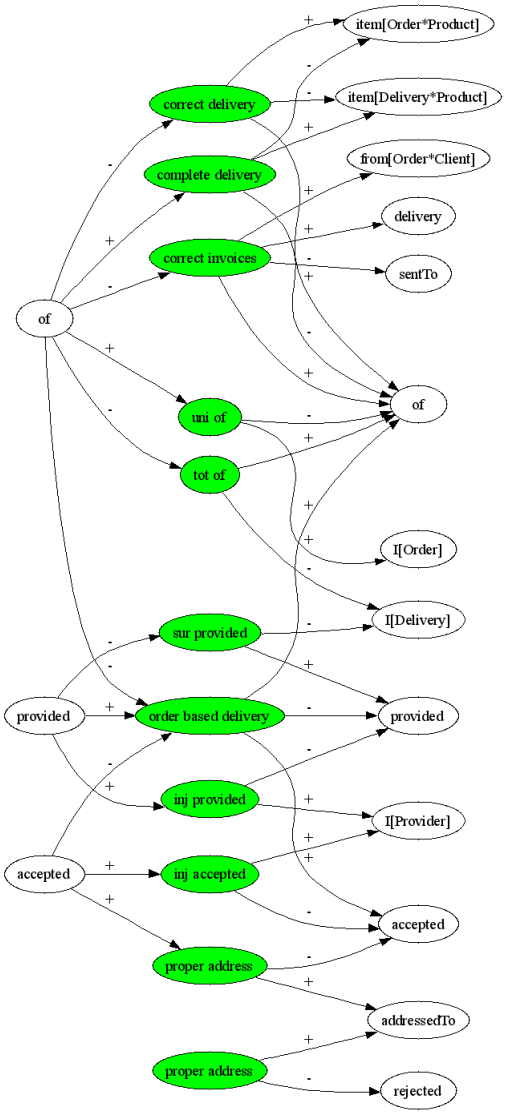

Figure 11.2: Switchboard of ship orders

Chapter 12

pay invoices

Activity ‘pay invoices’ must be performed by a user with role Client. During that activity, rules ‘correct payments’ and ‘correct invoices’ will be maintained without intervention of a user. Figure 12.1 shows the knowledge graph of this interface.

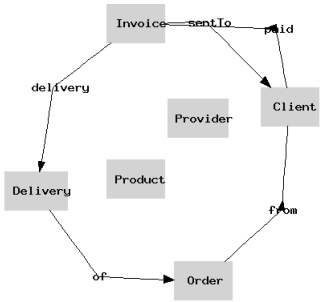

Figure 12.1: Language diagram of pay invoices

Figure 12.2 shows the switchboard diagram of this interface.

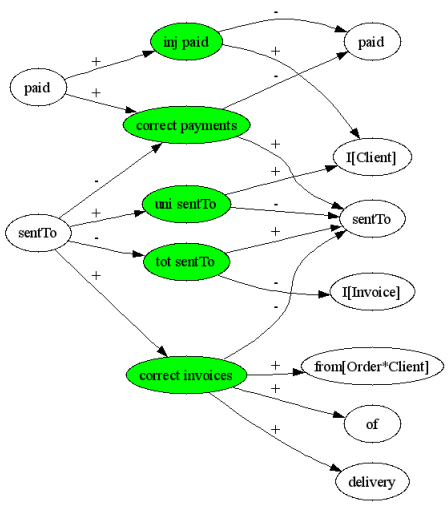

Figure 12.2: Switchboard of pay invoices

Chapter 13

receive goods

Activity ‘receive goods’ must be performed by a user with role Client. During that activity, rules ‘correct delivery’, ‘complete delivery’, ‘order based delivery’, and ‘correct invoices’ will be maintained without intervention of a user. Figure 13.1 shows the knowledge graph of this interface.

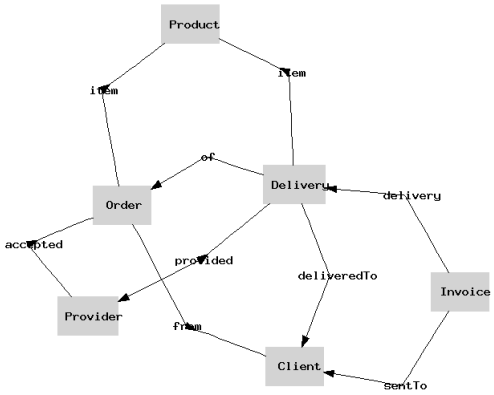

Figure 13.1: Language diagram of receive goods

Figure 13.2 shows the switchboard diagram of this interface.

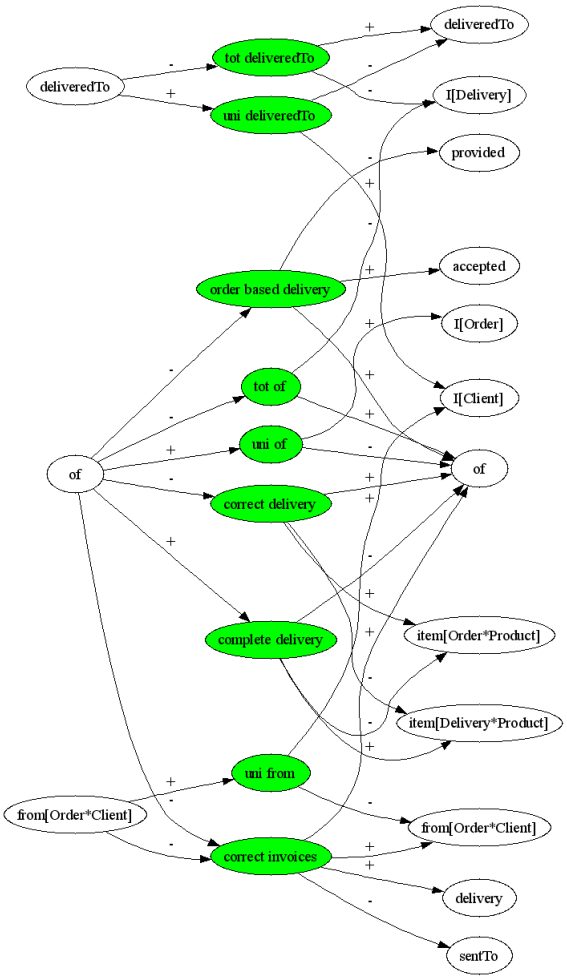

Figure 13.2: Switchboard of receive goods

Chapter 14

send invoices

Activity ‘send invoices’ must be performed by a user with role Provider. During that activity, rules ‘correct delivery’, ‘complete delivery’, ‘order based delivery’, ‘correct payments’, and ‘correct invoices’ will be maintained without intervention of a user. Figure 14.1 shows the knowledge graph of this interface.

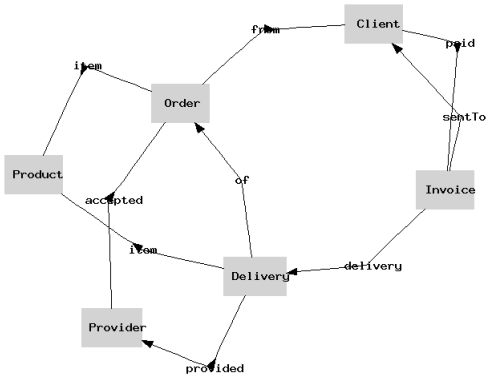

Figure 14.1: Language diagram of send invoices

Figure 14.2 shows the switchboard diagram of this interface.

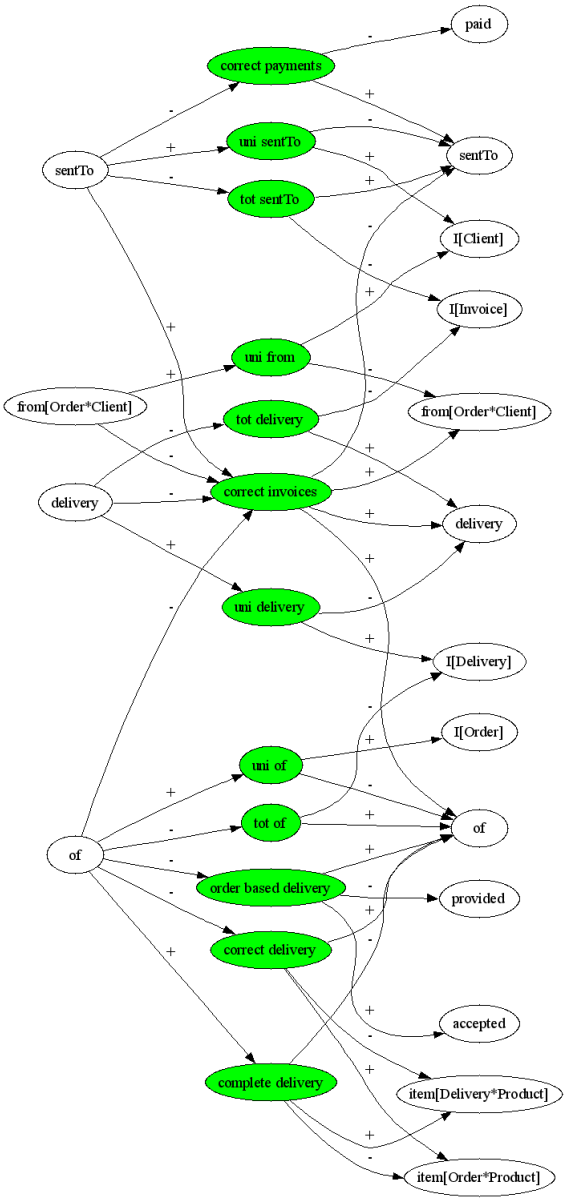

Figure 14.2: Switchboard of send invoices