Functional Specification of EURent

Rieks Joosten (rieks.joosten@tno.nl)

13 June 2014

Contents

Chapter 1

Introduction

This document specifies automated support for the EU-Rent example as described in 'DEMO-3 Way of Working (version 3, 1 September 2009)' by Jan L.G. Dietz. The purpose of the effort that resulted in this document is to provide case material to support statements regarding the extent that the DEMO approach and the Ampersand approach interfere and/or support one another.

We use the notation 'slide n' to refer to a specific slide in the DEMO-3 document mentioned above. In this notation, n is the slide number that can be found at the bottom of the slide. We use 'Slide n,m' to refer to slides n and m.

We use the notation 'Px:y', to refer to a specific sentence in the EU-Rent description of slide 3. In this notation, x identifies the paragraph number, and y identifies the sentence in that paragraph. Occasionally, the letter 'a' or 'b' may be appended to indicate the first or second part of (long) sentences. The notation 'Px:y-z' is used to refer to sentences y through z of paragraph x.

Issue: P2:1 states: "A car may be rented by a reservation in advance or by a 'walk-in' customer on the day of renting". The Note on slide 10 says that there is no difference between these two. We will follow this idea so as not to digress too much from the case. The consequence of this is that making a reservation in advance does not mean that there is a higher chance that a car of the requested type will be available.

Issue: A consequence of P3.4 is that an issue may arise when a branch is the pick-up branch for multiple promised (but not yet started) rentals, each of which has requested the same car type. The issue arises when the branch has fewer cars of that type available than it has promised rentals for cars of that type. This consequence should be accepted (and dealt with manually at the branc offices when it happens), or specified in a better way.

Issue: Slide 26 states that the rental ends after the rental has been paid. According to slide 4, P4:2, the renter has the right to make use of the rented car between the start and end of a rental. However, when rental payment is stated, it must be checked that 'everything is ok' (slide 30), which takes time. In that time, according to Slide 4, P4:2, the renter still has the right to make use of the rented car, and if he does so, it is undefined what will happen.

This document[1] defines the functionality of an information system called 'EU-Rent'. It defines the database and the business services of EURent by means of business rules[2]. Those rules are listed in chapter 2, ordered by theme. , ordered by theme.

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

[1]This document was generated at 13-6-2014 on 16:51:55, using Ampersand v3.0.2.1362, build time: 11-Jun-14 03:41:07 UTC.

[2]Rule based design 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 'EURent' can be discussed and expressed. The purpose of this chapter is to create shared understanding among stakeholders. The language of 'EURent' 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 'EURent', they share precisely enough language to have meaningful discussions about functional requirements. All definitions have been numbered for the sake of traceability.

2.1 EU-Rent

This section models the organizational structure of rental companies (limited to EU-Rent), as well as company-wide policies such as the maximum rental duration and rental and penalty tariffs.

At this point, the definitions of *branch*, *carRentalCompany*, *rentalCase*, *location*, *carType*, *brand*, *model*, and *amount* are given.

This system is designed for companies that rent cars according to the business essence as described in the DEMO document.

Definition 1: a company whose business is renting cars. *CarRentalCompany*

Car rental companies operate from branch offices at different geographical locations, each of which must be identifiable.

Definition 2: an office of a car rental company at a specific location. *Branch*

Branch offices are at different geographical locations. In order to compute penalties for dropping off cars at another branch than contractually agreed, the locations of such branches must be known.

Definition 3: a city (at which a branch office is located). *Location*

Rental charges (and penalties) depend on the type of a car.

Definition 4: the brand and model of a car. *CarType*

Car types are composed of a brand and a model. Examples of brands are: 'Volkswagen', 'Audi'.

Definition 5: the brand of a car. *Brand*

Car types are composed of a brand and a model. Examples of models are: 'Polo' or 'Beetle'.

Definition 6: the model of a car. *Model*

Tariffs, charges etc. are amounts of money. It is necessary to be specific about the nature of amounts, such as the sum and the currency.

Definition 7: a sum of money, expressed in 'Euro'. *Amount*

A common practice in case management is to define an anchorpoint for everything whose life cycle has to be managed, monitored, etc. To this end, we introduce such an anchorpoint for rentals, and call it a 'RentalCase'.

Definition 8: an information object that contains all information about a rental, including contractual items, rental items, billing items etc. *RentalCase*

EU-Rent is a company that rents cars to persons, operating from geographically dispersed braches. Therefore, we must know what branches exist with EU-Rent. *P1:1*

Agreement 9: Every branch is part of a car rental company.

Phrases that can be made are for instance:

AMS is a branch of EU-Rent.

DHG is a branch of EU-Rent.

RTD is a branch of EU-Rent.

EU-Rent operates from geographically dispersed braches. We need to know where such locations are in order to compute penalty charges for drivers that drop off their car at a location other than is contracted, because such charges depend on the distance between the actual and the contracted drop-off branch. *P1:1, P4:5*

Agreement 10: Every branch operates from a geographical location.

Phrases that can be made are for instance:

AMS is located in Amsterdam.

DHG is located in Den Haag.

RTD is located in Rotterdam.

Since only cars that are available at the pick-up branch may be rented, the availability of these cars at the branches must be known. *P3.4*

Agreement 11: It is known which cars are available at a branch.

Phrases that can be made are for instance:

Car with license plate 1-AMS-11 is available at EU-Rent branch AMS.

Car with license plate 1-AMS-13 is available at EU-Rent branch AMS.

Car with license plate 2-DHG-14 is available at EU-Rent branch DHG.

In order for the renter/driver to specify the car (s)he wants to rent, but also to correctly compute rental charges, the type of every car must be known.

Agreement 12: Every car is of a specific type (brand, model).

Phrases that can be made are for instance:

Car with license plate 1-AMS-11 is a VW Polo.

Car with license plate 1-AMS-12 is a VW Polo.

Car with license plate 1-AMS-13 is a VW Passat.

The cars of EU-Rent are divided in car types (brands and models). *P1:2a*

Agreement 13: A cartype has a specific brand.

Phrases that can be made are for instance:

The brand of Audi A4 is Audi.

The brand of VW Beetle is Volkswagen.

The brand of VW Passat is Volkswagen.

The cars of EU-Rent are divided in car types (brands and models). *P1:2a*

Agreement 14: A cartype has a specific model.

Phrases that can be made are for instance:

The model of Audi A4 is A4.

The model of VW Beetle is Beetle.

The model of VW Passat is Passat.

For every car type there is a particular rental tariff per day. *P1:2b*

Agreement 15: All car types have a specified rental tariff (Euros/day).

Phrases that can be made are for instance:

The rental tariff for Audi A4 is 103 Euros/day.

The rental tariff for VW Beetle is 60 Euros/day.

The rental tariff for VW Passat is 34 Euros/day.

In order to compute the penalty charge for exceeding the contracted rental duration, for each type of car it is specified what the excess charge per day will be.

Agreement 16: All car types have a specified excess tariff (Euro/day)

Phrases that can be made are for instance:

For cars of type Audi A4 the extra charge for a late drop-off is 56 Euro/day.

For cars of type VW Beetle the extra charge for a late drop-off is 38 Euro/day.

For cars of type VW Passat the extra charge for a late drop-off is 19 Euro/day.

Rentals have a maximum duration (P2:3), which is defined (as a poplicy constant) by EU-Rent (slide 7). *P2:3, slide 7*

Agreement 17: Rental companies must have specified the maximum duration of a rental.

A phrase that can be formed is for instance:

EU-Rent has set the maximum duration of a rental to 60 days.

In order to keep track of the cars that EU-Rent owns, every case must specify the car that is being rented.

Agreement 18: Rental contracts specify the car that is (to be) issued to the driver.

Phrases that can be made are for instance:

The car that will be, or has been issued under RC_AMS_123 has license plate 1-AMS-12.

The car that will be, or has been issued under RC_RTD_262 has license plate 3-RTD-18.

The transaction result B-R01 ([rental] has been started) must be modeled. This result is produced (stated) when the following rules are satisfied: *Result B-R01, slide 18*

1. the rental case has the property 'rental has been promised'.
2. a car (of the type as listed in the contract) has been assigned to the rental case;
3. the driver has promised to pick up the car at the contracted start date;
4. the driver has promised to drop off the car at or before the contracted end date, at the contracted drop-off branch.

Agreement 19: Rental cases may have the property 'rental has been started'.

Phrases that can be made are for instance:

RC_AMS_123 has the property 'rental has started', meaning that the rental associated with RC_AMS_123 has started.

RC_RTD_262 has the property 'rental has started', meaning that the rental associated with RC_RTD_262 has started.

Rentals that have been ended satisfy the following rules: *result B-R02, slides 26,30*

1. the rental case has the property 'rental has been dropped off'.
2. the rental case has the property 'rental has been paid'.

Agreement 20: Rental cases may have the property 'rental has been ended'.

A phrase that can be formed is for instance:

RC_RTD_262 has the property 'rental has ended', meaning that the rental associated with RC_RTD_262 has ended..

While our scope is limited to EU-Rent, we need to explicitly model it as a company in order to be able to define company policy that holds for all branches. An example of this would be the maximum rental period. *P2:3*

Agreement 21: The system is limited to branches that are part of EU-Rent.

In order to ensure that cars are not lost 'administratively', every car must be accounted for.

Agreement 22: All cars must either be rented, or in stock at one of the branches.

In order to prevent errors from occurring when Yes/No answers are answered differently, it is necessary to check whether such answers are either 'Yes' or 'No'.

Agreement 23: A Yes/No answer may only take the values 'Yes' or 'No'.

2.2 Rental Contracts

This section defines the contents of rental contracts and any constraints that must apply. It was decided not to introduce a specific concept 'RentalContract' because such an information object was also not mentioned in the slides.

The sequel introduces the language of Rental Contracts.

In order to be sure that a driver has a valid driving license, an identification number of the driving license must be known.

Definition 24: the identification number of a (valid) driving license. *DrivingLicense*

In order to compute the correct charge for renting a car, the start date must be known. Note that the meaning of this date depends on whether or not the rental has already started. If the rental has not yet started, it is the date that the rental is foreseen to start. If the rental has started, it is the date on which the rental actually started. *P2:2*

Agreement 25: Rental contracts may specify the actual (and contractual) start date of the rental.

Phrases that can be made are for instance:

The contractual and/or actual starting date of the rental of RC_AMS_123 is 01-07-2014.

The contractual and/or actual starting date of the rental of RC_RTD_262 is 01-06-2014.

In order to determine whether or not a penalty has to be paid for a late drop-off, the end date before which the car will be dropped off must be contractually administrated. *P2:2*

Agreement 26: Rental contracts may specify the (contractual) end date of the rental.

Phrases that can be made are for instance:

The contractual end date of the rental of RC_AMS_123 is 10-07-2014.

The contractual end date of the rental of RC_RTD_262 is 07-06-2014.

Since the daily charges depend on the car type, the contract must mention what type of car is (going to be) rented. *P2:2*

Agreement 27: Rental contracts may specify the car type of the rental.

Phrases that can be made are for instance:

The contractual type of the car being rented under RC_AMS_123 is VW Polo.

The contractual type of the car being rented under RC_RTD_262 is VW Polo.

Drivers can only rent cars that are available at the pick-up branch. Therefore, it must be known which branch this is. *P2:2*

Agreement 28: Rental contracts may specify the branch where the rental starts (i.e.: the car is picked up).

Phrases that can be made are for instance:

The contractual and/or actual pick-up branch for the rental of RC_AMS_123 is AMS.

The contractual and/or actual pick-up branch for the rental of RC_RTD_262 is RTD.

In order to allow branches to plan their stock of available cars, it helps to know what cars will be dropped off at what branch. *P2:2*

Agreement 29: Rental contracts may specify the branch where the rental supposedly ends (i.e.: the car is dropped off).

Phrases that can be made are for instance:

The contractual drop-off branch for the rental of RC_AMS_123 is DHG.

The contractual drop-off branch for the rental of RC_RTD_262 is UTR.

The person that will be held accountable for the rent, in particular for the payment thereof, must be administered. *P3.1*

Agreement 30: The person who rents the car is called the renter.

Phrases that can be made are for instance:

The renter for RC_AMS_123 is Richard Enter.

The renter for RC_RTD_262 is Richard Enter.

The person that will be driving the rented car, must be administered, allowing amongst others that his driving license is checked. *P3.2*

Agreement 31: The person who is going to drive is called the driver.

Phrases that can be made are for instance:

The driver for RC_AMS_123 is Dick River.

The driver for RC_RTD_262 is Dick River.

Since rentals may only be started if the driver has a valid driving license, the number of such a license will be registered. It is assumed that a driving license will only be registered if its expiration date is later than the contracted end date of the rental. The system does not check this. *P3.3*

Agreement 32: Rental cases register the driving license of the driver.

Phrases that can be made are for instance:

The driver for RC_AMS_123 has a valid driving license, with number DL01235467.

The driver for RC_RTD_262 has a valid driving license, with number DL01235467.

In order to ensure that the information contents of the cases are valid, it must be checked whether the car that is issued is of the type that is mentioned in the contract.

Agreement 33: The type of a rented car must be the same as the type mentioned in the contract.

2.3 Promising Rentals

This process describes the interaction between a renter and/or branch office employee as they prepare a request for obtaining a car rental. The bulk of the work consists of filling in most parts of the contract. The result of the process is that the rental has been promised (B-T01). *B-T01 promised*

The sequel introduces the language of Promising Rentals.

Some questions should only be answered with 'Yes' or 'No'. For automated reasoning it is necessary to be certain that no other answers can be given.

Definition 34: the answer to a question that must be 'Yes' or 'No'. *YesNoAnswer*

In order to account for the fact that the contracted rental period does not exceed the maximum rental duration (in particular when this maximum rental duration, which is company policy, is changed), the maximum rental duration must be made part of the contract. *P2:3*

Agreement 35: Rental contracts may specify the maximum rental duration.

Rentals that have been promised satisfy the following rules: *Slide 18*

1. it has been ascertained that the driver has a valid driving license;
2. the drop-off branch has a car available of the type specified in the contract;
3. the end date is no later than the start date plus the maximum allowed duration of rentals.
4. the following contractual information is known:
    - the pick-up branch;
    - the drop-off branch;
    - the start date;
    - the end date;
    - the car type;
    - the driver;
    - the driving license;
    - the renter.

Agreement 36: Rental cases may have the property 'rental has been promised'

Phrases that can be made are for instance:

RC_AMS_123 corresponds to RC_AMS_123 in relation rentalHasBeenPromised.

RC_RTD_262 corresponds to RC_RTD_262 in relation rentalHasBeenPromised.

Because rentals have a maximum duration (P2:3), it must be assessed whether or not the duration of the time interval of the requested rental, that starts with the contracted start date and ends with the contracted end date, is less than or equal to this maximum duration (slide 11). *P2:3, slide 11*

Agreement 37: the date interval (e.g.: [start date,end date]) is within the maximum rental duration as specified by EURent.

Phrases that can be made are for instance:

The period between 01-06-2014 and 07-06-2014 does not exceed the maximum allowed rental duration.

The period between 01-07-2014 and 10-07-2014 does not exceed the maximum allowed rental duration.

Promising a rental request consists of checking that all associated conditions have been met. This is done as follows: *Slide 11*

1. Ascertainment of the validity of the driving license, as well as that it belongs to the driver, is a manual check. The system assumes that this check has succeeded when the rental has registered a driver and a driving license.
2. The rule "Rentable cars" ensures that there is a car of the requested type available at the pick-up branch.
3. The rule "Enforcing maximum rental duration" guarantees that the maximum rental duration is not exceeded.
4. The rule "Default renter" ensures that if there is a driver, there is also a renter (because the driver is the renter by default).

Completeness of the rental request means that the following fields have been filled in:

- the pick-up branch;
- the drop-off branch;
- the start date;
- the end date;
- the car type;
- the driver;
- the driving license;

Agreement 38: A rental will be promised when all information from the rental request is complete.

The information that led to the decision of a rental having been promised, may not be lost or modified.

Agreement 39: When a rental has been promised, the request form must remain completely filled in.

Users should not be required to fill in duplicate information, e.g. in the case wehre the driver and renter are the same person.

Agreement 40: For submitted rental requests that specify the driver but not the renter, the driver is considered to be the renter.

Because the maximum rental duration as set by the car rental company may change over time, its value is copied into a rental case as soon as its value can be determined.

Agreement 41: A rental must record the maximum rental duration that the car rental company has specified.

For every rental contract, it must be checked (computed) whether or not the (proposed) rental period does or does not exceed the maximum allowed duration for that rental. *P2:3*

Agreement 42: The difference between the contracted end date and start date may bot exceed the maximum duration for rentals.

2.4 Starting Rentals

This process describes the work for the car rental company employee, starting with a filled in rental request and leading up to the result that the car of a rental has been picked up (B-R03) and the rental has started (B-R01). *Results: B-R01, B-R03*

Note that since the transactional parts as stated in slides 11 and 18 are manual, they are not modeled here.

The transaction result B-R03 (the car of [rental] has been picked up) must be modeled. *Slides 12-13*

Rentals for which the car has been picked up satisfy the following rules:

1. the rental case has the property 'rental has been promised'.
2. the car is available at the pick-up location;
3. a car (of the type as listed in the contract) has been assigned to the rental case;
4. keys of that car are handed to the driver, which we assume to imply that

- the driver has picked up the car at the contracted start date;
- the driver has promised to drop off the car according to the contractual constraints.

Agreement 43: Rental cases may have the property 'rental has been started'.

Phrases that can be made are for instance:

RC_AMS_123 has the property 'car of rental has been picked up', meaning that the keys of the car associated with RC_AMS_123 have been handed over to the driver.

RC_RTD_262 has the property 'car of rental has been picked up', meaning that the keys of the car associated with RC_RTD_262 have been handed over to the driver.

A rental starts when a driver has been handed the car keys. In order for the system to keep track of its cars (amongst other things), this (manual) action must be registered. Registration of this action presupposes that the information as registered in the rental contract is in accordance with reality, which the issuer of the keys must check. Note that when a rental is started, the car is no longer available for rent.

Agreement 44: Branches must register the handover of car keys (i.e. the responsibility for the car).

Phrases that can be made are for instance:

The answer to the question 'have the keys of the car rented under RC_AMS_123 been handed over to the designated driver?' is Yes.

The answer to the question 'have the keys of the car rented under RC_RTD_262 been handed over to the designated driver?' is Yes.

Starting a rental consists of checking that all associated conditions have been met. This is done as follows: *Slides 4-5,18*

1. the rental case must have the property 'rental has been promised'.
2. a car (of the type as listed in the contract) has been assigned to the rental case;
3. the keys of that car are handed to the driver, which we assume to imply that
   - the driver has picked up the car at the contracted start date;
   - the driver has promised to drop off the car according to the contractual constraints.

Agreement 45: A rental starts when the rental has been promised, a car of the correct type has been assigned and the driver has received the keys for this car.

The information that led to the decision of starting a rental, may not be lost or modified.

Agreement 46: When a rental has been started, a car of the correct type has been and remains assigned and the driver has received the keys for this car.

The type of car that is requested can only be one for which the pick-up branch has cars available. *P3.4*

Agreement 47: Rentals may only be promised if a car of the type specified in the contract is available at the pick-up branch.

For sanity reasons, the question of whether or not the keys are handed over can only be answered if the driver is known.

Agreement 48: Keys may only be handed over to the driver that is mentioned in the contract.

When the keys are handed to the driver, and the renter is not specified, we may assume that the driver also fulfills the role of renter, and fill this in the contract.

Agreement 49: When keys are handed over to the driver and the renter is not yet known, the driver is considered to be the renter.

2.5 Dropping off Cars

This process describes the work for the car rental company employee when a car is being dropped off and leading up to the results where the car of the rental has been dropped off (B-R04). *Result: B-R04*

The transaction result B-R04 (the car of [rental] has been dropped-off) must be modeled. *Slides 4-5*

Agreement 50: Rental cases may have the property 'car has been dropped off'.

A phrase that can be formed is for instance:

RC_RTD_262 has the property 'car has been dropped off', meaning that the car associated with RC_RTD_262 (and its keys) have been returned to a branch..

In order to allow checking whether or not the dropped off car is the same car as was rented, the dropped off car must be identified. *P4.1*

Agreement 51: Rental cases may specify the car that has actually been dropped off.

A phrase that can be formed is for instance:

The car that has been dropped-off for RC_RTD_262 is 3-RTD-18.

In order to make up the bill for the rental, the date at which the rented car is dropped off must be known.

Agreement 52: Rented cars are dropped off on specific dates.

A phrase that can be formed is for instance:

The car rented under RC_RTD_262 has been dropped off on 14-06-2014.

In order to make up the bill for the rental, the branch at which the rented car is dropped off must be known.

Agreement 53: Rental cases may specify the branch that the drop-off has taken place.

A phrase that can be formed is for instance:

The car rented under RC_RTD_262 has been dropped off at AMS.

The rules that need to be satisfied in order for a rental case to have the property 'rental car has been dropped-off', are as follows:

- the (license plate of the) dropped-off car must be administratd;
- the date of the drop-off must be administratd;
- the actual drop-off must be administrated.

Agreement 54: Dropping off a car means: identifying the dropped off car, and registering the branch and date of the drop-off.

Whenever a car has been dropped-off (in the context of a specific rental), it must be ensured that it remains dropped-off (for that rental).

Agreement 55: When a car has been dropped off, the car is identified, the drop-off date is known, and the branch where the drop-off took place is known.

In order to counter possible fraud, it must be checked that the car that is dropped-off is the same as that has been issued to the driver.

Agreement 56: The car that is dropped off must be the one that has been issued.

2.6 Billing Rentals

This process describes the work for the car rental company, starting when the car has been dropped off, and leading up to the result that the bill is made. This (fully automated) process consists of the following parts: *Result: Bill presented*

1. Computing the basic charge;
2. Computing the penalty charge for the use of the car beyond the contractual end date;
3. Computing the penalty charge in case the car is dropped off at a location other than contractually agreed;
4. Computing the total of these charged.

In order to compute the basic rental charge, the period of the actual rental must be known. *P4:3*

Agreement 57: A rental may specify the number of days that the rental has lasted.

The first component of the rental charge is the rental basic charge. *P4.3*

Agreement 58: Rental contracts may specify the basic charge.

In order to compute the penalty charge for exceeding the contracted rental duration, the period of the actual rental must be known. *P4:4*

Agreement 59:

The second component of the rental charge is the penalty charge (for exeeding the contracted rental duration). *P4.4*

Agreement 60: Rental contracts may specify a penalty charge for late drop-offs.

In order to compute the penalty charge for dropping of a car at another location than was contractually agreed, the amount that will be charged as a penalty for this must be known. *P4.5*

Agreement 61: There is a location penalty charge for cars that are dropped-off at another branch than agreed.

Phrases that can be made are for instance:

The penalty charge for dropping off a car at a branch that is AMS-DHG km away from the contracted drop-off branch, is 61 Euro..

The penalty charge for dropping off a car at a branch that is AMS-RTD km away from the contracted drop-off branch, is 67 Euro..

The penalty charge for dropping off a car at a branch that is AMS-UTR km away from the contracted drop-off branch, is 38 Euro..

The third component of the rental charge is the penalty for dropping off a rented car another location than was contractually agreed. *P4.5*

Agreement 62: Rental contracts may specify a location penalty charge, i.e. a penalty for dropping off the car at a location that differs from the contracted drop-off branch.

In order for a renter/driver to pay for a rental, the total amount (rental charge) must be known.

Agreement 63: The rental charge is the total amount to be paid for a rental.

The period of the actual rental is the difference between the date of the drop-off and the date of the pick-up of the rented car, plus one (so that if the drop-off date and the pick-up date are the same, the period is 1 day). *P4.3*

Agreement 64: The number of days that a rental has lasted is one more than the difference between the date that the rented car has been dropped off, and the date that the rented car was picked up.

The basic rental charge is the product of the period of the actual rental times the daily tariff that is valid for the type of car that was rented. *P4.3*

Agreement 65: The basic charge for a rental is the number of days the rental has lasted multiplied with the daily tariff for the type of car that was rented.

The excess period of the rental is zero, unless the drop-off date exceeds the contracted end date, in which case the period is the number of days between these two. *P4.4*

Agreement 66: The number of days in the excess period of a rental is zero, or the difference between the date that the rented car has been dropped off, and the contracted end date, whichever is more.

The penalty charge (for exeeding the contracted rental duration) is basic rental charge is the product of the excess period of the rental times the excess charge per day for the type of car that was rented. *P4.4*

Agreement 67: The penalty charge for a rental is the number of days in the excess period of the rental, multiplied with the excess tariff.

The penalty charge for dropping off a rented car another location than was contractually agreed is an amount that depends on the distance between the branches. *P4.5*

Agreement 68: The location penalty charge is the number of kilometres between the actual and contracted drop-off locations, multiplied with the location penalty tariff.

In order for a rental case to have the property 'rental has been promised', the total amount that the renter has to pay must be computed. This total amount consists of three parts: *P4:2-5*

1. the basic rental charge,
2. the penalty charge when the car is returned after the contracted drop-off date, and
3. a penalty charge in case the car is dropped off at a different branch than contractually agreed.

When the car has been dropped-off and the total charge is computed, payment must be requested.

Agreement 69: The rental charge is the sum of the basic charge, the penalty charge and the location penalty charge.

2.7 Paying Rentals

This process describes the work for the car rental company, starting when the rental charge is computed (the renter is presented the bill), and leading up to the result that the rental has ended (B-R05). *Result: B-R05*

Before a payment may be requested, it must be known that the corresponding rules are satisfied. Rental cases that have the property that payment has been requested satisfy these rules. *Slide 30*

Agreement 70: Rental cases may have the property 'payment has been requested'.

A phrase that can be formed is for instance:

RC_RTD_262 has the property 'payment has been requested', meaning that the amount that the renter has to pay is computed. RC_RTD_262.

In order to be able to terminate the rental, it must be known that payment is received.

Agreement 71: Payments for rental contracts need to be accepted (or declined).

A phrase that can be formed is for instance:

The answer to the question: 'Has the rental charge for RC_RTD_262 been received?' is Yes.

Rentals can only be paid after payment has been requested, implying that the total charge is known.

Agreement 72: Payment for a rental may only be accepted after payment is requested.

2.8 Ending Rentals

This process describes the work for the car rental company employee when a car is being dropped off and leading up to the results where the car of the rental has been dropped off (B-R04) and the rental has ended (B-R02). *Results: B-R02, B-R04*

Agreement 73: Ending rentals means: checking that the rented car has been dropped off and that the rental charge has been paid.

Whenever a rental has been ended, it must be ensured that it remains ended.

Agreement 74: When a rental has ended, the rented car has been dropped off and the rental has been paid.

2.9 Session Initialization

The interfaces provided by this system provide for user interaction with (parts of) the system. This section describes the automated functionality necessary to initialize the system to engate in such user interaction.

Since some computations depend on today's date, we need to ensure such a value is available. However, since this system is only for prototyping purposes, we need a rule that ensures there is a (reasonable) value for today's date, but it is not enforced to be the actual date of today: this allows us to run prototype sessions and change this date if necessary.

Agreement 75: Every session must have a value for 'today'

2.10 Branch Interface: Handling New Rentals and Pickups

The interfaces provided for branch offices, related to handling new rentals and pickups, provide some automated functionality. This section describes the features for filling in or changing the contents of forms that are presented in such interfaces. The assumption is that this interface is only provided within branch offices, allowing EU-Rent employees to create new rental applications for 'walk in customers' (see P2:1).

When a rental request is filled in by a branch, this branch will play the role of pick-up branch.

When a contract is being created, cars and/or branches may only be selected if such cars are available at these branches.

Agreement 77: When creating a new rental contract, car types and pick-up branch must be selected such that the branch has cars of that type available.

When a rental request in a branch is filled in, and they keys have already been handed over, the request is considered to be submitted.

Branch offices may only assign a car to a (new or existing) rental if this car is available at that branch.

Agreement 79: A branch office may only assign cars that are available at that location.

When a branch office has assigned a car to a (new or existing) rental, the keys must be handed to the contracted driver.

2.11 Branch Interface: Handling Drop-offs and Payment

The interfaces provided for branch offices, related to handling drop-offs, bill presentment and receiving payment, provide some automated functionality. This section describes the features for filling in or changing the contents of forms that are presented in such interfaces. The assumption is that this interface is only provided within branch offices, allowing EU-Rent employees to handle the dropping off of cars and obtaining rental payments.

When the keys of a car are returned (and the branch employee has checked that the car has been returned is in good order), the car's license plate must be entered to complete the drop-off.

In order to be sure that the car that is presented for a drop-off should be processed, it must be verified that there is a rental contract for this car that says that the car has been picked-up but not yet dropped-off.

Agreement 82: A car can only be returned if it is in the possession of the renter or driver.

When a car is returned to a branch, this branch will play the role of drop-off branch.

When a car is returned to a branch, that date is the drop-off date.

Handling a dropped-off car means that payment for the associated rental is to be obtained.

Chapter 3

Diagnosis

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

EURent does not specify which roles may change the contents of which relations.

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

rule	ExecEngine	Developer	Branch
Promising rental requests	×		
Default renter	×		
Compute max rental duration	×		
Starting the rental	×		
Auto fill in renter in rental contract	×		
Dropping off Cars	×		
Rental period computation	×		
Basic charge computation	×		
Excess period computation	×		
Excess charge computation	×		
Location penalty computation	×		
Requesting payment	×		
Requesting payment	×		
Ending Rentals	×		
Initialize today's date	×		
The branch that fills in the request is the pick-up branch	×		

Auto submit new branch request	×	
Car key handover to the driver		×
Accepting dropped-off car	×	
Cars are returned to the drop-off branch	×	
Cars are returned on the drop-off date	×	
Car drop-off handling		×

Concepts Car, Integer, Date, Person, and DistanceBetweenLocations remain without a purpose.

The purpose of relations *rcUserRequestedQ*, *rcBranchRequestedQ*, *arg1*, *arg2*, *arg3*, *computedRentalCharge*, *earliestDate*, *latestDate*, *computedRentalPeriod*, *ctcNrOfDays*, *ctcDailyAmount*, *computedTariffedCharge*, *firstDate*, *lastDate*, *computedNrOfExcessDays*, *distbranch*, *sessionToday*, *sessionBranch*, *sessionNewBranchRC*, *sessionPickupPerson*, and *sessionDroppedOffCar* is not documented.

Relations *branchLocation*, *brand*, and *model* are not used in any rule.

Figure 3.1 shows a conceptual diagram with all relations declared in 'EU-Rent'.

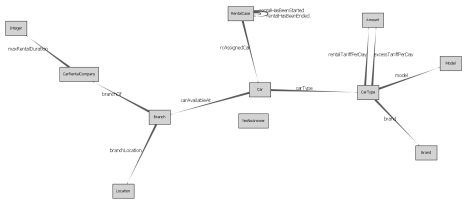

Figure 3.1: Concept diagram of relations in EU-Rent

Figure 3.2 shows a conceptual diagram with all relations declared in 'Rental Contracts'.

The purpose of rule *Ending Rentals* on line 416 of file .\EURent Ontology.adl is not documented. On line numbers 161, 174, 193, 340, 352, 359, and 366 of file .\EURent BRANCH interface.adl rules are defined, the meaning of which is documented by means of computer generated language.

The table below shows for each theme (i.e. process or pattern) the number of relations and rules, followed by the number and percentage that have a reference. Relations declared in multiple themes are counted multiple times.

Theme	Relations	With reference	%	Rules	With
EU-Rent	9	7	77%	3	
Rental Contracts	9	8	88%	1	
Promising Rentals	4	2	50%	5	

Starting Rentals	3	2	66%	5
Dropping off Cars	4	2	50%	3
Cost Projections	2	0	0%	2
Billing Rentals	7	6	85%	6
Paying Rentals	2	1	50%	2
Ending Rentals	1	1	100%	2
Enforcing maximum rental duration	2	1	50%	1
Compute total rental charge	4	0	0%	3
Compute number of regular days (period)	3	0	0%	4
Compute tariffed (regular or excess) charge	3	0	0%	5
Compute number of excess days (period)	3	0	0%	3
Distance computations	2	0	0%	1
Developer rules	0	0	-	1
Session Initialization	0	0	-	1
EU-Rent website: New Rental Requests	0	0	-	2
Branch Interface: Handling New Rentals and Pickups	0	0	-	5
Branch Interface: Handling Drop-offs and Payment	0	0	-	5
Entire context	68	30	44%	60

The following table shows which rules are not linked to a role within a particular process. This has as consequence that these rule(s) will be maintained by the computer.

process	rule
Promising Rentals	Promised rental requests, Enforcing maximum renta
Starting Rentals	Started rentals, Rentable cars, Keys must be hande
Dropping off Cars	Dropped off Cars, Dropped-off car type integrity, U
Billing Rentals	UNI rentalPeriod::RentalCase*Integer, UNI rentalB
Paying Rentals	Rental payment amount is known
Ending Rentals	Ended Rentals
Branch Interface: Handling New Rentals and Pickups	Car availability at branch, Assigning a car to a rent
Branch Interface: Handling Drop-offs and Payment	Dropped off car sanity check

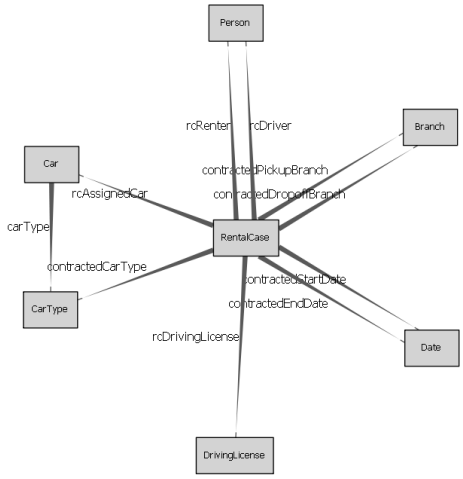

Figure 3.2: Concept diagram of relations in Rental Contracts

The role-rule assignments in any of the described processes have been assigned to rules within that same process.

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 defines the formal language, in which functional requirements of 'EURent' can be analysed and expressed.The purpose of this formalisation is to obtain a buildable specification. This chapter allows an independent professional with sufficient background to check whether the agreements made correspond to the formal rules and definitions.

This document specifies automated support for the EU-Rent example as described in 'DEMO-3 Way of Working (version 3, 1 September 2009)' by Jan L.G. Dietz. The purpose of the effort that resulted in this document is to provide case material to support statements regarding the extent that the DEMO approach and the Ampersand approach interfere and/or support one another.

We use the notation 'slide n' to refer to a specific slide in the DEMO-3 document mentioned above. In this notation, n is the slide number that can be found at the bottom of the slide. We use 'Slide n,m' to refer to slides n and m.

We use the notation 'Px:y', to refer to a specific sentence in the EU-Rent description of slide 3. In this notation, x identifies the paragraph number, and y identifies the sentence in that paragraph. Occasionally, the letter 'a' or 'b' may be appended to indicate the first or second part of (long) sentences. The notation 'Px:y-z' is used to refer to sentences y through z of paragraph x.

Issue: P2:1 states: "A car may be rented by a reservation in advance or by a 'walk-in' customer on the day of renting". The Note on slide 10 says that there is no difference between these two. We will follow this idea so as not to digress too much from the case. The consequence of this is that making a reservation in advance does not mean that there is a higher chance that a car of the requested type will be available.

Issue: A consequence of P3.4 is that an issue may arise when a branch is the pick-up branch for multiple promised (but not yet started) rentals, each of which has requested the same car type. The issue arises when the branch has fewer cars of that type available than it has promised rentals for cars of that type. This consequence should be accepted (and dealt with manually at the branc offices when it happens), or specified in a better way. *P3.4*

Issue: Slide 26 states that the rental ends after the rental has been paid. Ac- *Slides 26, 30*

cording to slide 4, P4:2, the renter has the right to make use of the rented car between the start and end of a rental. However, when rental payment is stated, it must be checked that 'everything is ok' (slide 30), which takes time. In that time, according to Slide 4, P4:2, the renter still has the right to make use of the rented car, and if he does so, it is undefined what will happen.

4.1 EU-Rent

This section models the organizational structure of rental companies (limited to EU-Rent), as well as company-wide policies such as the maximum rental duration and rental and penalty tariffs.

Figure 4.1 shows a conceptual diagram of this pattern.

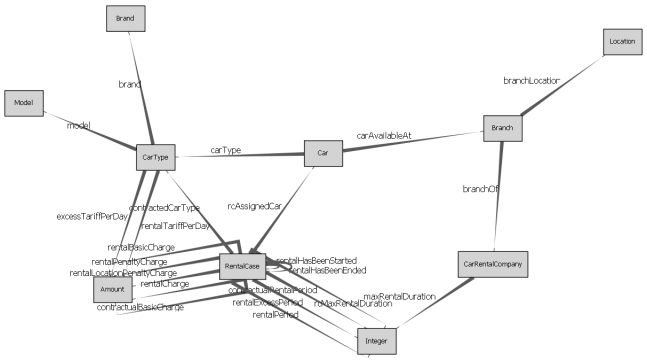

Figure 4.1: Concept diagram of the rules in EU-Rent

The definitions of concepts can be found in the glossary.

4.1.1 Declared relations

This section itemizes the declared relations with properties and purpose.

EU-Rent is a company that rents cars to persons, operating from geographically dispersed braches. Therefore, we must know what branches exist with EU-Rent. *P1:1*

For this purpose, the following function has been defined

$$branchOf \quad : \quad Branch \rightarrow CarRentalCompany \tag{4.1}$$

Every branch is part of a car rental company.

EU-Rent operates from geographically dispersed braches. We need to know where such locations are in order to compute penalty charges for drivers *P1:1, P4:5*

that drop off their car at a location other than is contracted, because such charges depend on the distance between the actual and the contracted drop-off branch.

For this purpose, the following function has been defined

$$branchLocation \quad : \quad Branch \rightarrow Location \tag{4.2}$$

Every branch operates from a geographical location.

Since only cars that are available at the pick-up branch may be rented, the availability of these cars at the branches must be known. *P3.4*

For this purpose, the following univalent relation has been defined

$$carAvailableAt \quad : \quad Car \times Branch \tag{4.3}$$

It is known which cars are available at a branch.

In order for the renter/driver to specify the car (s)he wants to rent, but also to correctly compute rental charges, the type of every car must be known.

For this purpose, the following function has been defined

$$carType \quad : \quad Car \rightarrow CarType \tag{4.4}$$

Every car is of a specific type (brand, model).

The cars of EU-Rent are divided in car types (brands and models). *P1:2a*

For this purpose, the following function has been defined

$$brand \quad : \quad CarType \rightarrow Brand \tag{4.5}$$

A cartype has a specific brand.

The cars of EU-Rent are divided in car types (brands and models). *P1:2a*

For this purpose, the following function has been defined

$$model \quad : \quad CarType \rightarrow Model \tag{4.6}$$

A cartype has a specific model.

For every car type there is a particular rental tariff per day. *P1:2b*

For this purpose, the following function has been defined

$$rentalTariffPerDay \quad : \quad CarType \rightarrow Amount \tag{4.7}$$

All car types have a specified rental tariff (Euros/day).

In order to compute the penalty charge for exceeding the contracted rental duration, for each type of car it is specified what the excess charge per day will be.

For this purpose, the following function has been defined

$$excessTariffPerDay \quad : \quad CarType \rightarrow Amount \tag{4.8}$$

All car types have a specified excess tariff (Euro/day)

Rentals have a maximum duration (P2:3), which is defined (as a poplicy constant) by EU-Rent (slide 7). *P2:3, slide 7*

For this purpose, the following relation has been defined

$$maxRentalDuration \quad : \quad CarRentalCompany \times Integer \tag{4.9}$$

Rental companies must have specified the maximum duration of a rental.

In order to keep track of the cars that EU-Rent owns, every case must specify the car that is being rented.

For this purpose, the following univalent relation has been defined

$$rcAssignedCar \quad : \quad RentalCase \times Car \tag{4.10}$$

Rental contracts specify the car that is (to be) issued to the driver.

The transaction result B-R01 ([rental] has been started) must be modeled. This result is produced (stated) when the following rules are satisfied: *Result B-R01, slide 18*

1. the rental case has the property 'rental has been promised'.
2. a car (of the type as listed in the contract) has been assigned to the rental case;
3. the driver has promised to pick up the car at the contracted start date;
4. the driver has promised to drop off the car at or before the contracted end date, at the contracted drop-off branch.

For this purpose, the following relation has been defined

$$rentalHasBeenStarted \quad : \quad RentalCase \times RentalCase \tag{4.11}$$

Rental cases may have the property 'rental has been started'.

Rentals that have been ended satisfy the following rules: *result B-R02, slides 26,30*

1. the rental case has the property 'rental has been dropped off'.

2. the rental case has the property 'rental has been paid'.

For this purpose, the following relation has been defined

$$rentalHasBeenEnded \quad : \quad RentalCase \times RentalCase \tag{4.12}$$

Rental cases may have the property 'rental has been ended'.

4.1.2 Formal rules

This section itemizes the formal rules with a reference to the shared language of stakeholders for the sake of traceability.

While our scope is limited to EU-Rent, we need to explicitly model it as a company in order to be able to define company policy that holds for all branches. An example of this would be the maximum rental period. Therefore the following requirement has been defined in section 2.1 p. 11: The system is limited to branches that are part of EU-Rent. This is formalized - using relations 5.12 - as *P2:3*

$$branchOf \vdash branchOf;' tEU - Rent' \tag{4.13}$$

Figure 4.2 shows a conceptual diagram of this rule.

Figure 4.2: Concept diagram of rule EURent branches

In order to ensure that cars are not lost 'administratively', every car must be accounted for. Therefore the following requirement has been defined in section 2.1 p. 11: All cars must either be rented, or in stock at one of the branches. This is formalized - using relations 5.19, 5.16, 4.11, 4.12 - as

$$I_{Car} \vdash (I_{Car} \cap \overline{(carAvailableAt; carAvailableAt^{\smile})} \cap rcAssignedCar^{\smile}; (rentalHasBeenStarted \cap \overline{rentalHasB} \tag{4.14}$$

Figure 4.3 shows a conceptual diagram of this rule.

Figure 4.3: Concept diagram of rule Car accountability

In order to prevent errors from occurring when Yes/No answers are answered differently, it is necessary to check whether such answers are either 'Yes' or 'No'.
Therefore the following requirement has been defined in section 2.1 p. 11:
A Yes/No answer may only take the values 'Yes' or 'No'.
This is formalized - using relations - as

$$I_{YesNoAnswer} \vdash' tYes' \cup' tNo' \tag{4.15}$$

Figure 4.4 shows a conceptual diagram of this rule.

Figure 4.4: Concept diagram of rule YesNoAnswer validity

4.2 Rental Contracts

This section defines the contents of rental contracts and any constraints that must apply. It was decided not to introduce a specific concept 'RentalContract' because such an information object was also not mentioned in the slides.

Figure 4.5 shows a conceptual diagram of this pattern.

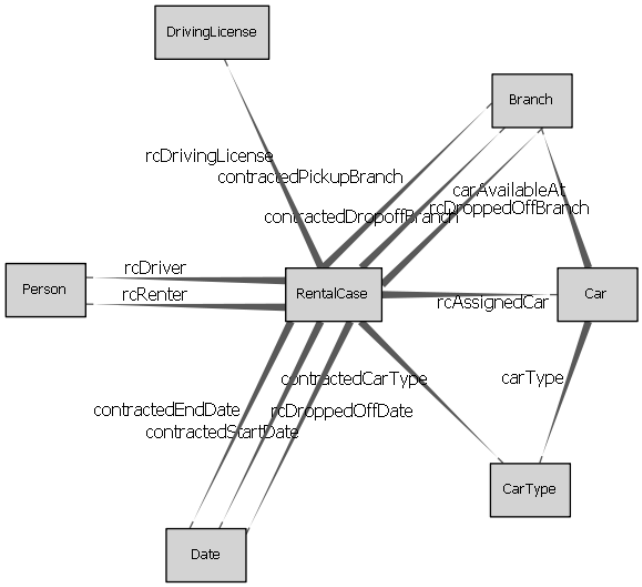

Figure 4.5: Concept diagram of the rules in Rental Contracts

The definitions of concepts can be found in the glossary.

4.2.1 Declared relations

This section itemizes the declared relations with properties and purpose.

In order to compute the correct charge for renting a car, the start date must be known. Note that the meaning of this date depends on whether or not the rental has already started. If the rental has not yet started, it is the date that the rental is foreseen to start. If the rental has started, it is the date on which the rental actually started. *P2:2*
For this purpose, the following univalent relation has been defined

$$\mathit{contractedStartDate} \quad : \quad \mathit{RentalCase} \times \mathit{Date} \tag{4.16}$$

Rental contracts may specify the actual (and contractual) start date of the rental.

In order to determine whether or not a penalty has to be paid for a late drop-off, the end date before which the car will be dropped off must be contractually administrated. *P2:2*
For this purpose, the following univalent relation has been defined

$$\mathit{contractedEndDate} \quad : \quad \mathit{RentalCase} \times \mathit{Date} \tag{4.17}$$

Rental contracts may specify the (contractual) end date of the rental.

Since the daily charges depend on the car type, the contract must mention what type of car is (going to be) rented. *P2:2*
For this purpose, the following univalent relation has been defined

$$\mathit{contractedCarType} \quad : \quad \mathit{RentalCase} \times \mathit{CarType} \tag{4.18}$$

Rental contracts may specify the car type of the rental.

Drivers can only rent cars that are available at the pick-up branch. Therefore, it must be known which branch this is. *P2:2*
For this purpose, the following univalent relation has been defined

$$\mathit{contractedPickupBranch} \quad : \quad \mathit{RentalCase} \times \mathit{Branch} \tag{4.19}$$

Rental contracts may specify the branch where the rental starts (i.e.: the car is picked up).

In order to allow branches to plan their stock of available cars, it helps to know what cars will be dropped off at what branch. *P2:2*
For this purpose, the following univalent relation has been defined

$$\mathit{contractedDropoffBranch} \quad : \quad \mathit{RentalCase} \times \mathit{Branch} \tag{4.20}$$

Rental contracts may specify the branch where the rental supposedly ends (i.e.: the car is dropped off).

The person that will be held accountable for the rent, in particular for the payment thereof, must be administered. *P3.1*
For this purpose, the following univalent relation has been defined

$$rcRenter \quad : \quad RentalCase \times Person \tag{4.21}$$

The person who rents the car is called the renter.

The person that will be driving the rented car, must be administered, allowing amongst others that his driving license is checked. *P3.2*
For this purpose, the following univalent relation has been defined

$$rcDriver \quad : \quad RentalCase \times Person \tag{4.22}$$

The person who is going to drive is called the driver.

Since rentals may only be started if the driver has a valid driving license, the number of such a license will be registered. It is assumed that a driving license will only be registered if its expiration date is later than the contracted end date of the rental. The system does not check this. *P3.3*
For this purpose, the following relation has been defined

$$rcDrivingLicense \quad : \quad RentalCase \times DrivingLicense \tag{4.23}$$

Rental cases register the driving license of the driver.

In order to keep track of the cars that EU-Rent owns, every case must specify the car that is being rented.
For this purpose, the following univalent relation has been defined

$$rcAssignedCar \quad : \quad RentalCase \times Car \tag{4.24}$$

Rental contracts specify the car that is (to be) issued to the driver.

In order for the renter/driver to specify the car (s)he wants to rent, but also to correctly compute rental charges, the type of every car must be known.
For this purpose, the following function has been defined

$$carType \quad : \quad Car \rightarrow CarType \tag{4.25}$$

Every car is of a specific type (brand, model).

4.2.2 Formal rules

This section itemizes the formal rules with a reference to the shared language of stakeholders for the sake of traceability.

In order to ensure that the information contents of the cases are valid, it must be checked whether the car that is issued is of the type that is mentioned in the contract.

Therefore the following requirement has been defined in section 2.2 p. 13:
The type of a rented car must be the same as the type mentioned in the contract.
This is formalized - using relations 5.16, 5.5, 5.20 - as

$$rcAssignedCar \vdash contractedCarType; carType^{\smile} \tag{4.26}$$

Figure 4.6 shows a conceptual diagram of this rule.

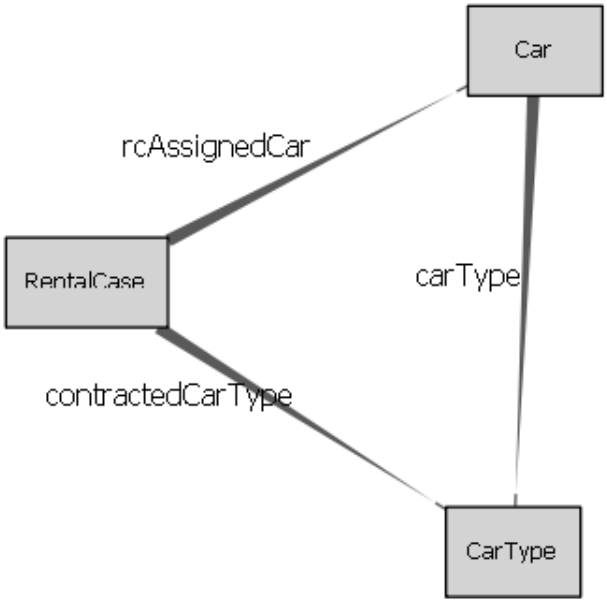

Figure 4.6: Concept diagram of rule Rented car type integrity

Chapter 5

Process Analysis

This document specifies automated support for the EU-Rent example as described in 'DEMO-3 Way of Working (version 3, 1 September 2009)' by Jan L.G. Dietz. The purpose of the effort that resulted in this document is to provide case material to support statements regarding the extent that the DEMO approach and the Ampersand approach interfere and/or support one another.

We use the notation 'slide n' to refer to a specific slide in the DEMO-3 document mentioned above. In this notation, n is the slide number that can be found at the bottom of the slide. We use 'Slide n,m' to refer to slides n and m.

We use the notation 'Px:y', to refer to a specific sentence in the EU-Rent description of slide 3. In this notation, x identifies the paragraph number, and y identifies the sentence in that paragraph. Occasionally, the letter 'a' or 'b' may be appended to indicate the first or second part of (long) sentences. The notation 'Px:y-z' is used to refer to sentences y through z of paragraph x.

Issue: P2:1 states: "A car may be rented by a reservation in advance or by a 'walk-in' customer on the day of renting". The Note on slide 10 says that there is no difference between these two. We will follow this idea so as not to digress too much from the case. The consequence of this is that making a reservation in advance does not mean that there is a higher chance that a car of the requested type will be available.

Issue: A consequence of P3.4 is that an issue may arise when a branch is the pick-up branch for multiple promised (but not yet started) rentals, each of which has requested the same car type. The issue arises when the branch has fewer cars of that type available than it has promised rentals for cars of that type. This consequence should be accepted (and dealt with manually at the branc offices when it happens), or specified in a better way. *P3.4*

Issue: Slide 26 states that the rental ends after the rental has been paid. According to slide 4, P4:2, the renter has the right to make use of the rented car between the start and end of a rental. However, when rental payment is stated, it must be checked that 'everything is ok' (slide 30), which takes time. In that time, according to Slide 4, P4:2, the renter still has the right to make use of the rented car, and if he does so, it is undefined what will happen. *Slides 26, 30*

EURent does not specify which roles may change the contents of which relations.

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

Role	Rule
ExecEngine	Promising rental requests Default renter Compute max rental duration Starting the rental Auto fill in renter in rental contract Dropping off Cars contractualRentalPeriod computation contractualBasicCharge computation Rental period computation Basic charge computation Excess period computation Excess charge computation Location penalty computation Requesting payment Requesting payment Ending Rentals Trigger interval computation Trigger rental charge computation Compute rental charge Trigger projected rental period computation Trigger rental period computation Compute number of days in period Trigger projected basic charge computation Trigger regular charge computation Trigger excess charge computation Compute charge based on number of days Trigger excess period computation Compute number of excess period days Initialize today's date Submitting user rental requests The branch that fills in the request is the pick-up branch Auto submit new branch request Accepting dropped-off car Cars are returned to the drop-off branch Cars are returned on the drop-off date
Developer	Dummy rule
Branch	Car key handover to the driver Car drop-off handling

5.1 Promising Rentals

This process describes the interaction between a renter and/or branch office employee as they prepare a request for obtaining a car rental. The bulk of the *B-T01 promised*

work consists of filling in most parts of the contract. The result of the process is that the rental has been promised (B-T01).

Figure 5.1 shows the process model.

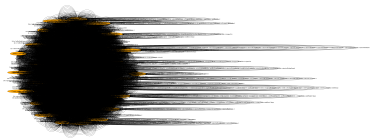

Figure 5.1: Process model of Promising Rentals

Promising rental requests Promising a rental request consists of checking that all associated conditions have been met. This is done as follows: *Slide 11*

1. Ascertainment of the validity of the driving license, as well as that it belongs to the driver, is a manual check. The system assumes that this check has succeeded when the rental has registered a driver and a driving license.
2. The rule "Rentable cars" ensures that there is a car of the requested type available at the pick-up branch.
3. The rule "Enforcing maximum rental duration" guarantees that the maximum rental duration is not exceeded.
4. The rule "Default renter" ensures that if there is a driver, there is also a renter (because the driver is the renter by default).

Completeness of the rental request means that the following fields have been filled in:

- the pick-up branch;
- the drop-off branch;
- the start date;
- the end date;
- the car type;
- the driver;
- the driving license;

Drivers can only rent cars that are available at the pick-up branch. Therefore, it must be known which branch this is. *P2:2*

In order to allow branches to plan their stock of available cars, it helps to know what cars will be dropped off at what branch. *P2:2*

In order to compute the correct charge for renting a car, the start date must be known. Note that the meaning of this date depends on whether or not the rental has already started. If the rental has not yet started, it is the date that the rental is foreseen to start. If the rental has started, it is the date on which the rental actually started. *P2:2*

In order to determine whether or not a penalty has to be paid for a late drop-off, the end date before which the car will be dropped off must be contractually administrated. *P2:2*

Since the daily charges depend on the car type, the contract must mention what type of car is (going to be) rented. *P2:2*

The person that will be driving the rented car, must be administered, allowing amongst others that his driving license is checked. *P3.2*

Since rentals may only be started if the driver has a valid driving license, the number of such a license will be registered. It is assumed that a driving license will only be registered if its expiration date is later than the contracted end date of the rental. The system does not check this. *P3.3*

To arrive at the formalization in equation 5.8, the following 7 relations are introduced.

$$\begin{aligned}
\mathit{contractedPickupBranch} &: \mathit{RentalCase} \times \mathit{Branch} && (5.1)\\
\mathit{contractedDropoffBranch} &: \mathit{RentalCase} \times \mathit{Branch} && (5.2)\\
\mathit{contractedStartDate} &: \mathit{RentalCase} \times \mathit{Date} && (5.3)\\
\mathit{contractedEndDate} &: \mathit{RentalCase} \times \mathit{Date} && (5.4)\\
\mathit{contractedCarType} &: \mathit{RentalCase} \times \mathit{CarType} && (5.5)\\
\mathit{rcDriver} &: \mathit{RentalCase} \times \mathit{Person} && (5.6)\\
\mathit{rcDrivingLicense} &: \mathit{RentalCase} \times \mathit{DrivingLicense} && (5.7)
\end{aligned}$$

We also use definitions **??** (*rentalHasBeenPromised*), **??** (*rcUserRequestedQ*), and **??** (*rcBranchRequestedQ*).

Activities that are defined by this rule are finished when:

$$I_{\mathit{RentalCase}} \cap (\mathit{rcUserRequestedQ};' \mathit{tYes}'; \mathit{rcUserRequestedQ}^{\smile} \cup \mathit{rcBranchRequestedQ};' \mathit{tYes}'; \mathit{rcBranchRe} \quad (5.8)$$

This corresponds to 'Promising rental requests' (2.3 op pg. 15).

Promised rental requests The information that led to the decision of a rental having been promised, may not be lost or modified.

We use definitions 5.1 (*contractedPickupBranch*), 5.2 (*contractedDropoffBranch*), 5.3 (*contractedStartDate*), 5.4 (*contractedEndDate*), 5.5 (*contractedCarType*), 5.6 (*rcDriver*), 5.7 (*rcDrivingLicense*), **??** (*rentalHasBeenPromised*), **??** (*rcUserRequestedQ*), and **??** (*rcBranchRequestedQ*).

This means:

$$\mathit{rentalHasBeenPromised} \vdash (\mathit{rcUserRequestedQ};' \mathit{tYes}'; \mathit{rcUserRequestedQ}^{\smile} \cup \mathit{rcBranchRequestedQ};' \mathit{tYes}' \quad (5.9)$$

Default renter Users should not be required to fill in duplicate information, e.g. in the case wehre the driver and renter are the same person.

The person that will be held accountable for the rent, in particular for the payment thereof, must be administered. *P3.1*

In order to formalize this, a relation rcRenter is introduced (5.10):

$$rcRenter \quad : \quad RentalCase \times Person \tag{5.10}$$

We also use definitions 5.6 (*rcDriver*), **??** (*rcUserRequestedQ*), and **??** (*rcBranchRequestedQ*) to formalize requirement 2.3 (page 16):

Activities that are defined by this rule are finished when:

$$(I_{RentalCase} \cap \overline{(rcRenter; rcRenter^{\smile})} \cap rcDriver; rcDriver^{\smile} \cap (rcUserRequestedQ \cup rcBranchRequestedQ);' \tag{5.11}$$

Compute max rental duration Because the maximum rental duration as set by the car rental company may change over time, its value is copied into a rental case as soon as its value can be determined.

EU-Rent is a company that rents cars to persons, operating from geographically dispersed braches. Therefore, we must know what branches exist with EU-Rent. *P1:1*

Rentals have a maximum duration (P2:3), which is defined (as a poplicy constant) by EU-Rent (slide 7). *P2:3, slide 7*

To arrive at the formalization in equation 5.14, the following two relations are introduced.

$$branchOf \quad : \quad Branch \to CarRentalCompany \tag{5.12}$$

$$maxRentalDuration \quad : \quad CarRentalCompany \times Integer \tag{5.13}$$

We also use definitions 5.1 (*contractedPickupBranch*) and **??** (*rcMaxRentalDuration*).

Activities that are defined by this rule are finished when:

$$I_{RentalCase} \cap contractedPickupBranch; branchOf; maxRentalDuration; V_{Integer_{imes}RentalCase} \vdash rcMax \tag{5.14}$$

This corresponds to 'Compute max rental duration' (2.3 op pg. 16).

Enforcing maximum rental duration For every rental contract, it must be checked (computed) whether or not the (proposed) rental period does or does not exceed the maximum allowed duration for that rental. *P2:3*

We use definitions 5.3 (*contractedStartDate*), 5.4 (*contractedEndDate*), **??** (*dateIntervalIsWithinMaxRentalDuration*), and **??** (*rcMaxRentalDuration*).

This means:

$$I_{RentalCase} \cap contractedStartDate; contractedStartDate^{\smile} \cap contractedEndDate; contractedEndDate^{\smile} \vdash co \tag{5.15}$$

5.2 Starting Rentals

This process describes the work for the car rental company employee, starting with a filled in rental request and leading up to the result that the car of a rental has been picked up (B-R03) and the rental has started (B-R01).

Results: B-R01, B-R03

Note that since the transactional parts as stated in slides 11 and 18 are manual, they are not modeled here.

Figure 5.2 shows the process model.

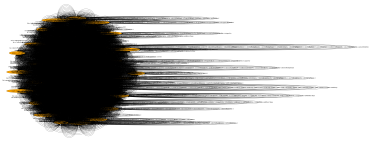

Figure 5.2: Process model of Starting Rentals

Starting the rental Starting a rental consists of checking that all associated conditions have been met. This is done as follows:

Slides 4-5,18

1. the rental case must have the property 'rental has been promised'.
2. a car (of the type as listed in the contract) has been assigned to the rental case;
3. the keys of that car are handed to the driver, which we assume to imply that
   - the driver has picked up the car at the contracted start date;
   - the driver has promised to drop off the car according to the contractual constraints.

In order to keep track of the cars that EU-Rent owns, every case must specify the car that is being rented.

In order to formalize this, a relation rcAssignedCar is introduced (5.16):

$$rcAssignedCar \quad : \quad RentalCase \times Car \tag{5.16}$$

We also use definitions 5.1 (*contractedPickupBranch*), **??** (*rentalHasBeenPromised*), 4.11 (*rentalHasBeenStarted*), and **??** (*rcKeysHandedOverQ*) to formalize requirement 2.4 (page 18):

Activities that are defined by this rule are finished when:

$$I_{RentalCase} \cap rentalHasBeenPromised \cap rcAssignedCar; rcAssignedCar^{\smile} \cap rcKeysHandedOverQ;' tYes'; r \tag{5.17}$$

Started rentals The information that led to the decision of starting a rental, may not be lost or modified.

We use definitions 5.16 (*rcAssignedCar*), **??** (*rentalHasBeenPromised*), 4.11 (*rentalHasBeenStarted*), and **??** (*rcKeysHandedOverQ*).

This means:

$$rentalHasBeenStarted \vdash rentalHasBeenPromised \cap rcAssignedCar ; rcAssignedCar^{\smile} \cap rcKeysHandedOver \quad (5.18)$$

Rentable cars The type of car that is requested can only be one for which the pick-up branch has cars available. *P3.4*

Since only cars that are available at the pick-up branch may be rented, the availability of these cars at the branches must be known. *P3.4*

In order for the renter/driver to specify the car (s)he wants to rent, but also to correctly compute rental charges, the type of every car must be known.

To arrive at the formalization in equation 5.21, the following two relations are introduced.

$$carAvailableAt \quad : \quad Car \times Branch \quad (5.19)$$

$$carType \quad : \quad Car \to CarType \quad (5.20)$$

We also use definitions 5.1 (*contractedPickupBranch*), 5.5 (*contractedCarType*), **??** (*rentalHasBeenPromised*), and **??** (*rcKeysHandedOverQ*).

This means:

$$contractedPickupBranch^{\smile} ; (I_{RentalCase} \cap rentalHasBeenPromised \cap \overline{(rcKeysHandedOverQ ;' tYes' ; rcKey} \quad (5.21)$$

This corresponds to the requirement on page 18:

Rentals may only be promised if a car of the type specified in the contract is available at the pick-up branch.

Keys must be handed over to driver For sanity reasons, the question of whether or not the keys are handed over can only be answered if the driver is known.

We use definitions 5.6 (*rcDriver*) and **??** (*rcKeysHandedOverQ*).

This means:

$$I_{RentalCase} \cap rcKeysHandedOverQ ;' tYes' ; rcKeysHandedOverQ^{\smile} \vdash rcDriver ; rcDriver^{\smile} \quad (5.22)$$

Auto fill in renter in rental contract When the keys are handed to the driver, and the renter is not specified, we may assume that the driver also fulfills the role of renter, and fill this in the contract.

We use definitions 5.10 (*rcRenter*), 5.6 (*rcDriver*), and **??** (*rcKeysHandedOverQ*).

Activities that are defined by this rule are finished when:

$$I_{RentalCase} \cap rcKeysHandedOverQ;' tYes'; rcKeysHandedOverQ^{\smile} \vdash rcRenter; rcRenter^{\smile} \tag{5.23}$$

5.3 Dropping off Cars

This process describes the work for the car rental company employee when a car is being dropped off and leading up to the results where the car of the rental has been dropped off (B-R04). *Result: B-R04*

Figure 5.3 shows the process model.

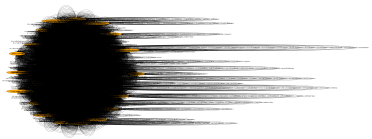

Figure 5.3: Process model of Dropping off Cars

Dropping off Cars The rules that need to be satisfied in order for a rental case to have the property 'rental car has been dropped-off', are as follows:

- the (license plate of the) dropped-off car must be administratd;
- the date of the drop-off must be administratd;
- the actual drop-off must be administrated.

We use definitions 4.11 (*rentalHasBeenStarted*), **??** (*rcCarHasBeenDroppedOff*), **??** (*rcDroppedOffCar*), **??** (*rcDroppedOffDate*), and **??** (*rcDroppedOffBranch*).

Activities that are defined by this rule are finished when:

$$I_{RentalCase} \cap rentalHasBeenStarted \cap rcDroppedOffCar; rcDroppedOffCar^{\smile} \cap rcDroppedOffDate; rcDropp \tag{5.24}$$

Dropped off Cars Whenever a car has been dropped-off (in the context of a specific rental), it must be ensured that it remains dropped-off (for that rental).

We use definitions 4.11 (*rentalHasBeenStarted*), **??** (*rcCarHasBeenDroppedOff*), **??** (*rcDroppedOffCar*), **??** (*rcDroppedOffDate*), and **??** (*rcDroppedOffBranch*).

This means:

$$rcCarHasBeenDroppedOff \vdash rentalHasBeenStarted \cap rcDroppedOffCar; rcDroppedOffCar^{\smile} \cap rcDroppedO$$ (5.25)

Dropped-off car type integrity In order to counter possible fraud, it must be checked that the car that is dropped-off is the same as that has been issued to the driver.

We use definitions 5.16 (*rcAssignedCar*) and **??** (*rcDroppedOffCar*).

This means:

$$rcDroppedOffCar \vdash rcAssignedCar$$ (5.26)

5.4 Billing Rentals

This process describes the work for the car rental company, starting when the car has been dropped off, and leading up to the result that the bill is made. This (fully automated) process consists of the following parts: *Result: Bill presented*

1. Computing the basic charge;
2. Computing the penalty charge for the use of the car beyond the contractual end date;
3. Computing the penalty charge in case the car is dropped off at a location other than contractually agreed;
4. Computing the total of these charged.

Figure 5.4 shows the process model.

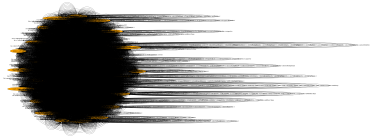

Figure 5.4: Process model of Billing Rentals

Rental period computation The period of the actual rental is the difference between the date of the drop-off and the date of the pick-up of the rented car, plus one (so that if the drop-off date and the pick-up date are the same, the period is 1 day). *P4.3*

We use definitions 5.3 (*contractedStartDate*), **??** (*rcDroppedOffDate*), **??** (*rentalPeriod*), **??** (*earliestDate*), **??** (*latestDate*), and **??** (*computedRentalPeriod*).

Activities that are defined by this rule are finished when:

$$(contractedStartDate; earliestDate^{\smile} \cap rcDroppedOffDate; latestDate^{\smile}); computedRentalPeriod \vdash rentalPe \tag{5.27}$$

Basic charge computation The basic rental charge is the product of the period of the actual rental times the daily tariff that is valid for the type of car that was rented. *P4.3*

For every car type there is a particular rental tariff per day. *P1:2b*

In order to formalize this, a function rentalTariffPerDay is introduced (5.28):

$$rentalTariffPerDay \quad : \quad CarType \rightarrow Amount \tag{5.28}$$

We also use definitions 5.20 (*carType*), 5.16 (*rcAssignedCar*), **??** (*rentalPeriod*), **??** (*rentalBasicCharge*), **??** (*ctcNrOfDays*), **??** (*ctcDailyAmount*), and **??** (*computedTariffedCharge*) to formalize requirement 2.6 (page 22):

Activities that are defined by this rule are finished when:

$$(rentalPeriod; ctcNrOfDays^{\smile} \cap rcAssignedCar; carType; rentalTariffPerDay; ctcDailyAmount^{\smile}); compute \tag{5.29}$$

Excess period computation The excess period of the rental is zero, unless the drop-off date exceeds the contracted end date, in which case the period is the number of days between these two. *P4.4*

We use definitions 5.4 (*contractedEndDate*), **??** (*rcDroppedOffDate*), **??** (*rentalExcessPeriod*), **??** (*firstDate*), **??** (*lastDate*), and **??** (*computedNrOfExcessDays*).

Activities that are defined by this rule are finished when:

$$(rcDroppedOffDate; lastDate^{\smile} \cap contractedEndDate; firstDate^{\smile}); computedNrOfExcessDays \vdash rentalExce \tag{5.30}$$

Excess charge computation The penalty charge (for exeeding the contracted rental duration) is basic rental charge is the product of the excess period of the rental times the excess charge per day for the type of car that was rented. *P4.4*

In order to compute the penalty charge for exceeding the contracted rental duration, for each type of car it is specified what the excess charge per day will be.

In order to formalize this, a function excessTariffPerDay is introduced (5.31):

$$excessTariffPerDay \quad : \quad CarType \rightarrow Amount \tag{5.31}$$

We also use definitions 5.20 (*carType*), 5.16 (*rcAssignedCar*), **??** (*rentalExcessPeriod*), **??** (*rentalPenaltyCharge*), **??** (*ctcNrOfDays*), **??** (*ctcDailyAmount*), and **??** (*computedTariffedCharge*) to formalize requirement 2.6 (page 22):

Activities that are defined by this rule are finished when:

$$(rentalExcessPeriod; ctcNrOfDays^{\smile} \cap rcAssignedCar; carType; excessTariffPerDay; ctcDailyAmount^{\smile}); c \tag{5.32}$$

Location penalty computation The penalty charge for dropping off a rented car another location than was contractually agreed is an amount that depends on the distance between the branches. *P4.5*

We use definitions 5.2 (*contractedDropoffBranch*), **??** (*rcDroppedOffBranch*), **??** (*computedLocationPenaltyCharge*), **??** (*rentalLocationPenaltyCharge*), and **??** (*distbranch*).

Activities that are defined by this rule are finished when:

$$(rcDroppedOffBranch; distbranch^{\smile} \cap contractedDropoffBranch; distbranch^{\smile}); computedLocationPenaltyCl \tag{5.33}$$

Requesting payment In order for a rental case to have the property 'rental has been promised', the total amount that the renter has to pay must be computed. This total amount consists of three parts: *P4:2-5*

1. the basic rental charge,
2. the penalty charge when the car is returned after the contracted drop-off date, and
3. a penalty charge in case the car is dropped off at a different branch than contractually agreed.

When the car has been dropped-off and the total charge is computed, payment must be requested.

We use definitions **??** (*rentalBasicCharge*), **??** (*rentalPenaltyCharge*), **??** (*rentalLocationPenaltyCharge*), **??** (*rentalCharge*), **??** (*arg1*), **??** (*arg2*), **??** (*arg3*), and **??** (*computedRentalCharge*).

Activities that are defined by this rule are finished when:

$$(rentalBasicCharge; arg1^{\smile} \cap rentalPenaltyCharge; arg2^{\smile} \cap rentalLocationPenaltyCharge; arg3^{\smile}); compute \tag{5.34}$$

5.5 Paying Rentals

This process describes the work for the car rental company, starting when the rental charge is computed (the renter is presented the bill), and leading up to the result that the rental has ended (B-R05). *Result: B-R05*

Figure 5.5 shows the process model.

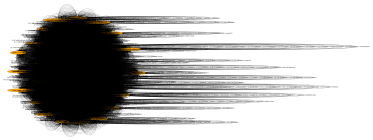

Figure 5.5: Process model of Paying Rentals

Requesting payment In order for a rental case to have the property 'rental has been promised', the total amount that the renter has to pay must be computed. This total amount consists of three parts: *P4:2-5*

1. the basic rental charge,
2. the penalty charge when the car is returned after the contracted drop-off date, and
3. a penalty charge in case the car is dropped off at a different branch than contractually agreed.

When the car has been dropped-off and the total charge is computed, payment must be requested.

We use definitions **??** (*rcCarHasBeenDroppedOff*), **??** (*rentalCharge*), and **??** (*paymentHasBeenRequested*).

Activities that are defined by this rule are finished when:

$$I_{RentalCase} \cap rcCarHasBeenDroppedOff \cap rentalCharge; rentalCharge^{\smile} \vdash paymentHasBeenRequested \tag{5.35}$$

Rental payment amount is known Rentals can only be paid after payment has been requested, implying that the total charge is known.

We use definitions **??** (*paymentHasBeenRequested*) and **??** (*rentalIsPaidQ*).

This means:

$$I_{RentalCase} \cap rentalIsPaidQ; {}'tYes'; rentalIsPaidQ^{\smile} \vdash paymentHasBeenRequested \tag{5.36}$$

5.6 Ending Rentals

This process describes the work for the car rental company employee when a car is being dropped off and leading up to the results where the car of the rental has been dropped off (B-R04) and the rental has ended (B-R02).

Results: B-R02, B-R04

Figure 5.6 shows the process model.

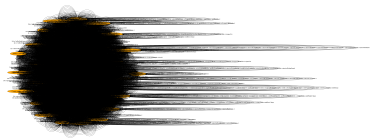

Figure 5.6: Process model of Ending Rentals

Ending Rentals We use definitions **??** (*rcCarHasBeenDroppedOff*), **??** (*rentalIsPaidQ*), and 4.12 (*rentalHasBeenEnded*).
Activities that are defined by this rule are finished when:

$$I_{RentalCase} \cap rcCarHasBeenDroppedOff \cap rentalIsPaidQ;' tYes'; rentalIsPaidQ^{\smile} \vdash rentalHasBeenEnded \tag{5.37}$$

Ended Rentals Whenever a rental has been ended, it must be ensured that it remains ended.
We use definitions **??** (*rcCarHasBeenDroppedOff*), **??** (*rentalIsPaidQ*), and 4.12 (*rentalHasBeenEnded*).
This means:

$$rentalHasBeenEnded \vdash rcCarHasBeenDroppedOff \cap rentalIsPaidQ;' tYes'; rentalIsPaidQ^{\smile} \tag{5.38}$$

5.7 Session Initialization

The interfaces provided by this system provide for user interaction with (parts of) the system. This section describes the automated functionality necessary to initialize the system to engate in such user interaction.

Figure 5.7 shows the process model.

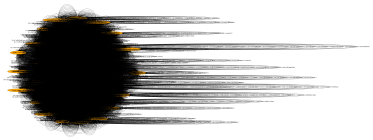

Figure 5.7: Process model of Session Initialization

Initialize today's date Since some computations depend on today's date, we need to ensure such a value is available. However, since this system is only for prototyping purposes, we need a rule that ensures there is a (reasonable) value for today's date, but it is not enforced to be the actual date of today: this allows us to run prototype sessions and change this date if necessary. In order to formalize this, a relation sessionToday is introduced (5.39):

$$sessionToday \quad : \quad SESSION \times Date \tag{5.39}$$

Activities that are defined by this rule are finished when:

$$I_{SESSION} \vdash sessionToday; sessionToday^{\smile} \tag{5.40}$$

5.8 Branch Interface: Handling New Rentals and Pickups

The interfaces provided for branch offices, related to handling new rentals and pickups, provide some automated functionality. This section describes the features for filling in or changing the contents of forms that are presented in such interfaces. The assumption is that this interface is only provided within branch offices, allowing EU-Rent employees to create new rental applications for 'walk in customers' (see P2:1).

Figure 5.8 shows the process model.

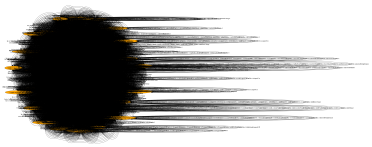

Figure 5.8: Process model of Branch Interface: Handling New Rentals and Pickups

The branch that fills in the request is the pick-up branch When a rental request is filled in by a branch, this branch will play the role of pick-up branch.
To arrive at the formalization in equation 5.45, the following three relations are introduced.

$$sessionNewBranchRC \quad : \quad SESSION \times RentalCase \tag{5.41}$$
$$sessionPickupPerson \quad : \quad SESSION \times Person \tag{5.42}$$
$$sessionBranch \quad : \quad SESSION \times Branch \tag{5.43}$$

We also use definitions 5.10 (*rcRenter*), 5.1 (*contractedPickupBranch*), 5.6 (*rcDriver*), **??** (*rentalHasBeenPromised*), and 4.11 (*rentalHasBeenStarted*).

Activities that are defined by this rule are finished when:

$$(I_{RentalCase} \cap \overline{rentalHasBeenStarted}); (sessionNewBranchRC^{\smile} \cup rentalHasBeenPromised; (rcRenter \cup rc \tag{5.44}$$

This corresponds to 'The branch that fills in the request is the pick-up branch' (?? op pg. ??).

Car availability at branch When a contract is being created, cars and/or branches may only be selected if such cars are available at these branches.
We use definitions 5.41 (*sessionNewBranchRC*), 5.19 (*carAvailableAt*), 5.20 (*carType*), 5.1 (*contractedPickupBranch*), and 5.5 (*contractedCarType*).
This means:

$$contractedCarType^{\smile}; (I_{RentalCase} \cap sessionNewBranchRC^{\smile}; {}' t_S ESSION'; sessionNewBranchRC); con \tag{5.45}$$

Auto submit new branch request When a rental request in a branch is filled in, and they keys have already been handed over, the request is considered to be submitted.
We use definitions 5.41 (*sessionNewBranchRC*), 5.16 (*rcAssignedCar*), ?? (*rcBranchRequestedQ*), and ?? (*rcKeysHandedOverQ*).
Activities that are defined by this rule are finished when:

$$'t_S ESSION'; sessionNewBranchRC; (I_{RentalCase} \cap rcAssignedCar; rcAssignedCar^{\smile}); rcKeysHandedOu \tag{5.46}$$

Assigning a car to a rental Branch offices may only assign a car to a (new or existing) rental if this car is available at that branch.
We use definitions 5.41 (*sessionNewBranchRC*), 5.42 (*sessionPickupPerson*), 5.44 (*sessionBranch*), 5.19 (*carAvailableAt*), 5.16 (*rcAssignedCar*), 5.10 (*rcRenter*), 5.1 (*contractedPickupBranch*), 5.6 (*rcDriver*), ?? (*rentalHasBeenPromised*), and ?? (*rcCarHasBeenPickedUp*).
This means:

$$'t_S ESSION'; (sessionNewBranchRC \cup (sessionBranch; contractedPickupBranch^{\smile} \cap sessionPickupPerson \tag{5.47}$$

Car key handover to the driver When a branch office has assigned a car to a (new or existing) rental, the keys must be handed to the contracted driver.
We use definitions 5.41 (*sessionNewBranchRC*), 5.42 (*sessionPickupPerson*), 5.44 (*sessionBranch*), 5.19 (*carAvailableAt*), 5.16 (*rcAssignedCar*), 5.10 (*rcRenter*), 5.1 (*contractedPickupBranch*), 5.6 (*rcDriver*), ?? (*rentalHasBeenPromised*), ?? (*rcCarHasBeenPickedUp*), and ?? (*rcKeysHandedOverQ*).
Activities that are defined by this rule are finished when:

$$'t_S ESSION'; ((sessionNewBranchRC \cup (sessionBranch; contractedPickupBranch^{\smile} \cap sessionPickupPerso \tag{5.48}$$

5.9 Branch Interface: Handling Drop-offs and Payment

The interfaces provided for branch offices, related to handling drop-offs, bill presentment and receiving payment, provide some automated functionality. This section describes the features for filling in or changing the contents of forms that are presented in such interfaces. The assumption is that this interface is only provided within branch offices, allowing EU-Rent employees to handle the dropping off of cars and obtaining rental payments.

Figure 5.9 shows the process model.

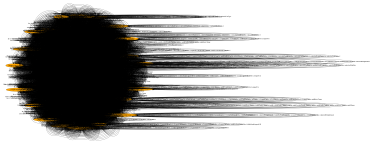

Figure 5.9: Process model of Branch Interface: Handling Drop-offs and Payment

Accepting dropped-off car When the keys of a car are returned (and the branch employee has checked that the car has been returned is in good order), the car's license plate must be entered to complete the drop-off.
In order to formalize this, a relation sessionDroppedOffCar is introduced (5.50):

$$sessionDroppedOffCar \quad : \quad SESSION \times Car \tag{5.49}$$

We also use definitions 5.16 (*rcAssignedCar*), **??** (*rcCarHasBeenPickedUp*), **??** (*rcCarHasBeenDroppedOff*), and **??** (*rcDroppedOffCar*) to formalize requirement **??** (page **??**):
Activities that are defined by this rule are finished when:

$$(I_{RentalCase} \cap rcCarHasBeenPickedUp \cap \overline{rcCarHasBeenDroppedOff}); rcAssignedCar; (I_{Car} \cap sessionDrop \tag{5.50}$$

Dropped off car sanity check In order to be sure that the car that is presented for a drop-off should be processed, it must be verified that there is a rental contract for this car that says that the car has been picked-up but not yet dropped-off.
We use definitions 5.50 (*sessionDroppedOffCar*), 5.19 (*carAvailableAt*), 5.16 (*rcAssignedCar*), **??** (*rcCarHasBeenPickedUp*), and **??** (*rcCarHasBeenDroppedOff*).
This means:

$$I_{Car} \cap sessionDroppedOffCar^{\smile};' t_S ESSION'; sessionDroppedOffCar \vdash rcAssignedCar^{\smile}; (I_{RentalCase} \cap \tag{5.51}$$

Cars are returned to the drop-off branch When a car is returned to a branch, this branch will play the role of drop-off branch.
We use definitions 5.50 (*sessionDroppedOffCar*), 5.44 (*sessionBranch*), **??** (*rcCarHasBeenDroppedOff*), **??** (*rcDroppedOffCar*), and **??** (*rcDroppedOffBranch*).
Activities that are defined by this rule are finished when:

$$(I_{RentalCase} \cap \overline{rcCarHasBeenDroppedOff}); rcDroppedOffCar; sessionDroppedOffCar^{\smile}; {}' t_S ESSION'; se \tag{5.52}$$

Cars are returned on the drop-off date When a car is returned to a branch, that date is the drop-off date.
We use definitions 5.50 (*sessionDroppedOffCar*), 5.39 (*sessionToday*), **??** (*rcCarHasBeenDroppedOff*), **??** (*rcDroppedOffCar*), and **??** (*rcDroppedOffDate*).
Activities that are defined by this rule are finished when:

$$(I_{RentalCase} \cap \overline{rcCarHasBeenDroppedOff}); rcDroppedOffCar; sessionDroppedOffCar^{\smile}; {}' t_S ESSION'; se \tag{5.53}$$

Car drop-off handling Handling a dropped-off car means that payment for the associated rental is to be obtained.
We use definitions 5.50 (*sessionDroppedOffCar*), 5.16 (*rcAssignedCar*), **??** (*rcCarHasBeenDroppedOff*), **??** (*rentalIsPaidQ*), and 4.12 (*rentalHasBeenEnded*).
Activities that are defined by this rule are finished when:

$${}' t_S ESSION'; sessionDroppedOffCar; rcAssignedCar^{\smile}; (I_{RentalCase} \cap rcCarHasBeenDroppedOff \cap \overline{renta} \tag{5.54}$$

Chapter 6

Data structure

This chapter contains the result of the data analisys. It is structured as follows:

We start with the classification model, followed by a list of all relations, that are the foundation of the rest of the analisys. Finally, the logical and technical data model are discussed.

6.1 Classifications

No classifications have been defined

6.2 Fact types

This section enumerates the fact types, that have been used in the design of the datastructure. For each fact type its name, the source and target concept and the properties are documented.

branchOf : ***Branch*** × ***CarRentalCompany*** Every branch is part of a car rental company.

Properties: UNI, TOT

branchLocation : ***Branch*** × ***Location*** Every branch operates from a geographical location.

Properties: UNI, TOT

carAvailableAt : ***Car*** × ***Branch*** It is known which cars are available at a branch.

Properties: UNI, TOT

carType : ***Car*** × ***CarType*** Every car is of a specific type (brand, model).

Properties: UNI, TOT

brand : ***CarType*** × ***Brand*** A cartype has a specific brand.

Properties: UNI, TOT

model : ***CarType*** × ***Model*** A cartype has a specific model.

Properties: UNI, TOT

rentalTariffPerDay : ***CarType*** × ***Amount*** All car types have a specified rental tariff (Euros/day).

Properties: UNI, TOT

excessTariffPerDay : ***CarType*** × ***Amount*** All car types have a specified excess tariff (Euro/day)

Properties: UNI, TOT

maxRentalDuration : ***CarRentalCompany*** × ***Integer*** Rental companies must have specified the maximum duration of a rental.

Properties: --

contractedStartDate : ***RentalCase*** × ***Date*** Rental contracts may specify the actual (and contractual) start date of the rental.

Properties: UNI

contractedEndDate : ***RentalCase*** × ***Date*** Rental contracts may specify the (contractual) end date of the rental.

Properties: UNI

contractedCarType : ***RentalCase*** × ***CarType*** Rental contracts may specify the car type of the rental.

Properties: UNI

contractedPickupBranch : ***RentalCase*** × ***Branch*** Rental contracts may specify the branch where the rental starts (i.e.: the car is picked up).

Properties: UNI

contractedDropoffBranch : ***RentalCase*** × ***Branch*** Rental contracts may specify the branch where the rental supposedly ends (i.e.: the car is dropped off).

Properties: UNI

rcRenter : ***RentalCase*** × ***Person*** The person who rents the car is called the renter.

Properties: UNI

rcDriver : ***RentalCase*** × ***Person*** The person who is going to drive is called the driver.

Properties: UNI

rcDrivingLicense : ***RentalCase*** × ***DrivingLicense*** Rental cases register the driving license of the driver.

Properties: --

rcAssignedCar : ***RentalCase*** × ***Car*** Rental contracts specify the car that is (to be) issued to the driver.

Properties: UNI, SUR

rentalHasBeenPromised : ***RentalCase*** × ***RentalCase*** Rental cases may have the property 'rental has been promised'

Properties: --

rcUserRequestedQ : ***RentalCase*** × ***YesNoAnswer*** A user has requested a new rental to be started, and has provided all necessary information for that.

Properties: --

rcBranchRequestedQ : ***RentalCase*** × ***YesNoAnswer*** A branch office has requested a new rental to be started, and has provided all necessary information for that.

Properties: --

dateIntervalIsWithinMaxRentalDuration : ***Date*** × ***Date*** the date interval (e.g.: [start date,end date]) is within the maximum rental duration as specified by EURent.

Properties: --

rcCarHasBeenPickedUp : ***RentalCase*** × ***RentalCase*** Rental cases may have the property 'rental has been started'.

Properties: --

rentalHasBeenStarted : ***RentalCase*** × ***RentalCase*** Rental cases may have the property 'rental has been started'.

Properties: --

rcKeysHandedOverQ : ***RentalCase*** × ***YesNoAnswer*** Branches must register the handover of car keys (i.e. the responsibility for the car).

Properties: --

rcCarHasBeenDroppedOff : ***RentalCase*** × ***RentalCase*** Rental cases may have the property 'car has been dropped off'.

Properties: --

rcDroppedOffCar : ***RentalCase*** × ***Car*** Rental cases may specify the car that has actually been dropped off.

Properties: UNI

rcDroppedOffDate : ***RentalCase*** × ***Date*** Rented cars are dropped off on specific dates.

Properties: UNI

rcDroppedOffBranch : ***RentalCase*** × ***Branch*** Rental cases may specify the branch that the drop-off has taken place.

Properties: UNI

rentalPeriod : ***RentalCase*** × ***Integer*** A rental may specify the number of days that the rental has lasted.

Properties: UNI

rentalBasicCharge : ***RentalCase*** × ***Amount*** Rental contracts may specify the basic charge.

Properties: UNI

rentalExcessPeriod : ***RentalCase*** × ***Integer*** **Properties**: UNI

rentalPenaltyCharge : ***RentalCase*** × ***Amount*** Rental contracts may specify a penalty charge for late drop-offs.

Properties: UNI

computedLocationPenaltyCharge : ***DistanceBetweenLocations*** × ***Amount*** There is a location penalty charge for cars that are dropped-off at another branch than agreed.

Properties: UNI, TOT

rentalLocationPenaltyCharge : ***RentalCase*** × ***Amount*** Rental contracts may specify a location penalty charge, i.e. a penalty for dropping off the car at a location that differs from the contracted drop-off branch.

Properties: UNI

rentalCharge : ***RentalCase*** × ***Amount*** The rental charge is the total amount to be paid for a rental.

Properties: UNI

paymentHasBeenRequested : ***RentalCase*** × ***RentalCase*** Rental cases may have the property 'payment has been requested'.

Properties: --

rentalIsPaidQ : ***RentalCase*** × ***YesNoAnswer*** Payments for rental contracts need to be accepted (or declined).

Properties: --

rentalHasBeenEnded : ***RentalCase*** × ***RentalCase*** Rental cases may have the property 'rental has been ended'.

Properties: --

rcMaxRentalDuration : ***RentalCase*** × ***Integer*** Rental contracts may specify the maximum rental duration.

Properties: UNI

arg1 : ***CompRentalCharge*** × ***Amount*** **Properties**: UNI, TOT

arg2 : ***CompRentalCharge*** × ***Amount*** **Properties**: UNI, TOT

arg3 : ***CompRentalCharge*** × ***Amount*** **Properties**: UNI, TOT

computedRentalCharge : ***CompRentalCharge*** × ***Amount*** **Properties**: UNI

earliestDate : ***DateDifferencePlusOne*** × ***Date*** **Properties**: UNI, TOT

latestDate : ***DateDifferencePlusOne*** × ***Date*** **Properties**: UNI, TOT

computedRentalPeriod : ***DateDifferencePlusOne*** × ***Integer*** **Properties**: UNI

ctcNrOfDays : ***CompTariffedCharge*** × ***Integer*** **Properties**: UNI, TOT

ctcDailyAmount : ***CompTariffedCharge*** × ***Amount*** **Properties**: UNI, TOT

computedTariffedCharge : ***CompTariffedCharge*** × ***Amount*** **Properties**: UNI

firstDate : ***DateDifference*** × ***Date*** **Properties**: UNI, TOT

lastDate : ***DateDifference*** × ***Date*** **Properties**: UNI, TOT

computedNrOfExcessDays : ***DateDifference*** × ***Integer*** **Properties**: UNI

distbranch : ***DistanceBetweenLocations*** × ***Branch*** A distance is computed relative to a branch.

Properties: TOT, SUR

sessionToday : ***SESSION*** × ***Date*** **Properties**: UNI

sessionBranch : ***SESSION*** × ***Branch*** **Properties**: UNI

sessionNewBranchRC : ***SESSION*** × ***RentalCase*** **Properties**: UNI

sessionPickupPerson : ***SESSION*** × ***Person*** **Properties**: UNI

sessionDroppedOffCar : ***SESSION*** × ***Car*** **Properties**: UNI

6.3 Logical datamodel

The functional requirements have been translated into a data model. This model is shown by figure 6.1.

There are 10 entity types.The details of each entity type are described (in alfabetical order) in the following paragraphs:

6.3.1 Entity type: *Branch*

This entity type has the following attributes:

Attribute	Type	
Id	Branch	Primary key
branchOf	CarRentalCompany	Mandatory
branchLocation	Location	Mandatory

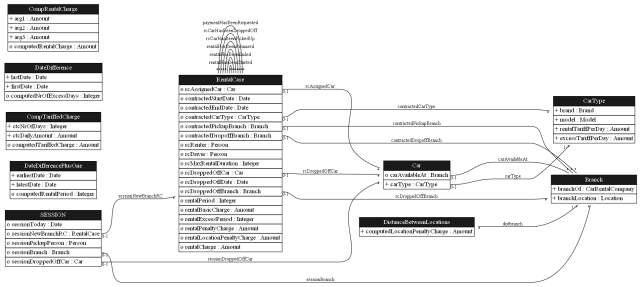

Figure 6.1: Logical data model of EURent

Branch has the following associations:

1. Every *Car* 'carAvailableAt' zero or more *Branch*. For the other way round, for this relation holds that each *Branch* at most one *Car*.
2. Every *RentalCase* 'contractedPickupBranch' zero or more *Branch*. For the other way round, for this relation holds that each *Branch* at most one *RentalCase*.
3. Every *RentalCase* 'contractedDropoffBranch' zero or more *Branch*. For the other way round, for this relation holds that each *Branch* at most one *RentalCase*.
4. Every *RentalCase* 'rcDroppedOffBranch' zero or more *Branch*. For the other way round, for this relation holds that each *Branch* at most one *RentalCase*.
5. Every *DistanceBetweenLocations* must 'distbranch' at least one *Branch*. For the other way round, for this relation holds that each *Branch* zero or more *DistanceBetweenLocations*.
6. Every *SESSION* 'sessionBranch' zero or more *Branch*. For the other way round, for this relation holds that each *Branch* at most one *SESSION*.

6.3.2 Entity type: *Car*

This entity type has the following attributes:

Attribute	Type	
Id	Car	Primary key
carAvailableAt	Branch	Optional
carType	CarType	Mandatory

Car has the following associations:

1. Every *Car* 'carAvailableAt' zero or more *Branch*. For the other way round, for this relation holds that each *Branch* at most one *Car*.
2. Every *Car* must 'carType' at least one *CarType*. For the other way round, for this relation holds that each *CarType* at most one *Car*.
3. Every *RentalCase* 'rcAssignedCar' zero or more *Car*. For the other way round, for this relation holds that each *Car* at most one *RentalCase*.
4. Every *RentalCase* 'rcDroppedOffCar' zero or more *Car*. For the other way round, for this relation holds that each *Car* at most one *RentalCase*.

6.3.3 Entity type: *CarType*

This entity type has the following attributes:

Attribute	Type	
Id	CarType	Primary key
brand	Brand	Mandatory
model	Model	Mandatory
rentalTariffPerDay	Amount	Mandatory
excessTariffPerDay	Amount	Mandatory

CarType has the following associations:

1. Every *Car* must 'carType' at least one *CarType*. For the other way round, for this relation holds that each *CarType* at most one *Car*.
2. Every *RentalCase* 'contractedCarType' zero or more *CarType*. For the other way round, for this relation holds that each *CarType* at most one *RentalCase*.

6.3.4 Entity type: *CompRentalCharge*

This entity type has the following attributes:

Attribute	Type	
Id	CompRentalCharge	Primary key
arg1	Amount	Mandatory
arg2	Amount	Mandatory
arg3	Amount	Mandatory
computedRentalCharge	Amount	Optional

CompRentalCharge has the following associations:

6.3.5 Entity type: *CompTariffedCharge*

This entity type has the following attributes:

Attribute	Type

Id	CompTariffedCharge	Primary key
ctcNrOfDays	Integer	Mandatory
ctcDailyAmount	Amount	Mandatory
computedTariffedCharge	Amount	Optional

CompTariffedCharge has the following associations:

6.3.6 Entity type: *DateDifference*

This entity type has the following attributes:

Attribute	Type	
Id	DateDifference	Primary key
lastDate	Date	Mandatory
firstDate	Date	Mandatory
computedNrOfExcessDays	Integer	Optional

DateDifference has the following associations:

6.3.7 Entity type: *DateDifferencePlusOne*

This entity type has the following attributes:

Attribute	Type	
Id	DateDifferencePlusOne	Primary key
earliestDate	Date	Mandatory
latestDate	Date	Mandatory
computedRentalPeriod	Integer	Optional

DateDifferencePlusOne has the following associations:

6.3.8 Entity type: *DistanceBetweenLocations*

This entity type has the following attributes:

Attribute	Type	
Id	DistanceBetweenLocations	Primary key
computedLocationPenaltyCharge	Amount	Mandatory

DistanceBetweenLocations has the following associations:

1. Every *DistanceBetweenLocations* must 'distbranch' at least one *Branch*. For the other way round, for this relation holds that each *Branch* zero or more *DistanceBetweenLocations*.

6.3.9 Entity type: *RentalCase*

This entity type has the following attributes:

Attribute	Type	
Id	RentalCase	Primary key
rcAssignedCar	Car	Optional
contractedStartDate	Date	Optional
contractedEndDate	Date	Optional
contractedCarType	CarType	Optional
contractedPickupBranch	Branch	Optional
contractedDropoffBranch	Branch	Optional
rcRenter	Person	Optional
rcDriver	Person	Optional
rcMaxRentalDuration	Integer	Optional
rcDroppedOffCar	Car	Optional
rcDroppedOffDate	Date	Optional
rcDroppedOffBranch	Branch	Optional
rentalPeriod	Integer	Optional
rentalBasicCharge	Amount	Optional
rentalExcessPeriod	Integer	Optional
rentalPenaltyCharge	Amount	Optional
rentalLocationPenaltyCharge	Amount	Optional
rentalCharge	Amount	Optional

RentalCase has the following associations:

1. Every *RentalCase* 'rcAssignedCar' zero or more *Car*. For the other way round, for this relation holds that each *Car* at most one *RentalCase*.
2. Every *RentalCase* 'rentalHasBeenStarted' zero or more *RentalCase*. For the other way round, for this relation holds that each *RentalCase* zero or more *RentalCase*.
3. Every *RentalCase* 'rentalHasBeenEnded' zero or more *RentalCase*. For the other way round, for this relation holds that each *RentalCase* zero or more *RentalCase*.
4. Every *RentalCase* 'contractedCarType' zero or more *CarType*. For the other way round, for this relation holds that each *CarType* at most one *RentalCase*.
5. Every *RentalCase* 'contractedPickupBranch' zero or more *Branch*. For the other way round, for this relation holds that each *Branch* at most one *RentalCase*.
6. Every *RentalCase* 'contractedDropoffBranch' zero or more *Branch*. For the other way round, for this relation holds that each *Branch* at most one *RentalCase*.
7. Every *RentalCase* 'rentalHasBeenPromised' zero or more *RentalCase*. For the other way round, for this relation holds that each *RentalCase* zero or more *RentalCase*.
8. Every *RentalCase* 'rcCarHasBeenPickedUp' zero or more *RentalCase*. For the other way round, for this relation holds that each *RentalCase* zero or more *RentalCase*.
9. Every *RentalCase* 'rcCarHasBeenDroppedOff' zero or more *RentalCase*. For the other way round, for this relation holds that each *RentalCase* zero or more *RentalCase*.
10. Every *RentalCase* 'rcDroppedOffCar' zero or more *Car*. For the other way round, for this relation holds that each *Car* at most one *RentalCase*.
11. Every *RentalCase* 'rcDroppedOffBranch' zero or more *Branch*. For the other way round, for this relation holds that each *Branch* at most one *RentalCase*.
12. Every *RentalCase* 'paymentHasBeenRequested' zero or more *RentalCase*. For the other way round, for this relation holds that each *RentalCase* zero or more *RentalCase*.
13. Every *SESSION* 'sessionNewBranchRC' zero or more *RentalCase*. For the other way round, for this relation holds that each *RentalCase* at most one *SESSION*.

6.3.10 Entity type: *SESSION*

This entity type has the following attributes:

Attribute	Type	
Id	SESSION	Primary key
sessionToday	Date	Optional
sessionNewBranchRC	RentalCase	Optional
sessionPickupPerson	Person	Optional
sessionBranch	Branch	Optional
sessionDroppedOffCar	Car	Optional

SESSION has the following associations:

1. Every *SESSION* 'sessionNewBranchRC' zero or more *RentalCase*. For the other way round, for this relation holds that each *RentalCase* at most one *SESSION*.
2. Every *SESSION* 'sessionBranch' zero or more *Branch*. For the other way round, for this relation holds that each *Branch* at most one *SESSION*.
3. Every *SESSION* 'sessionDroppedOffCar' zero or more *Car*. For the other way round, for this relation holds that each *Car* at most one *SESSION*.

6.4 Technical datamodel

The functional requirements have been translated into a technical data model. This model is shown by figure 6.2.

The technical datamodel consists of the following 36tables:

6.4.1 Table: Amount

This table has the following 1 fields:

- **Amount**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

6.4.2 Table: Branch

This table has the following 3 fields:

- **Branch**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

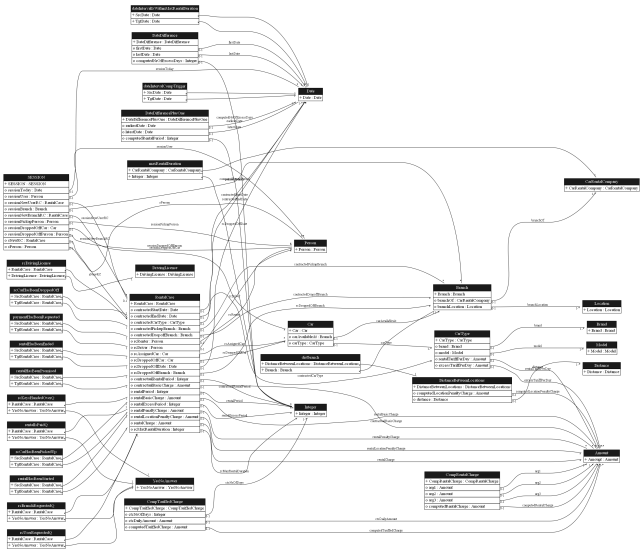

Figure 6.2: Technical data model of EURent

- **branchOf**
  This attribute implements the relation $Branch \xrightarrow{branchOf} CarRentalCompany$.
  `SQLVarchar 255`, Optional.

- **branchLocation**
  This attribute implements the relation $Branch \xrightarrow{branchLocation} Location$.
  `SQLVarchar 255`, Optional.

6.4.3 Table: Brand

This table has the following 1 fields:

- **Brand**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

6.4.4 Table: Car

This table has the following 3 fields:

- **Car**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

- **carAvailableAt**
  This attribute implements the relation $Car \xrightarrow{carAvailableAt} Branch$.
  `SQLVarchar 255`, Optional.

- **carType**
  This attribute implements the relation $Car \xrightarrow{carType} CarType$.
  `SQLVarchar 255`, Optional.

6.4.5 Table: CarRentalCompany

This table has the following 1 fields:

- **CarRentalCompany**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

6.4.6 Table: CarType

This table has the following 5 fields:

- **CarType**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.
- **brand**
  This attribute implements the relation $CarType \xrightarrow{brand} Brand$.
  `SQLVarchar 255`, Optional.
- **model**
  This attribute implements the relation $CarType \xrightarrow{model} Model$.
  `SQLVarchar 255`, Optional.
- **rentalTariffPerDay**
  This attribute implements the relation $CarType \xrightarrow{rentalTariffPerDay} Amount$.
  `SQLVarchar 255`, Optional.
- **excessTariffPerDay**
  This attribute implements the relation $CarType \xrightarrow{excessTariffPerDay} Amount$.
  `SQLVarchar 255`, Optional.

6.4.7 Table: CompRentalCharge

This table has the following 5 fields:

- **CompRentalCharge**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.
- **arg1**
  This attribute implements the relation $CompRentalCharge \xrightarrow{arg1} Amount$.
  `SQLVarchar 255`, Optional.

- **arg2**
  This attribute implements the relation $CompRentalCharge \xrightarrow{arg2} Amount$.
  `SQLVarchar 255`, Optional.
- **arg3**
  This attribute implements the relation $CompRentalCharge \xrightarrow{arg3} Amount$.
  `SQLVarchar 255`, Optional.
- **computedRentalCharge**
  This attribute implements the relation $CompRentalCharge \xrightarrow{computedRentalCharge} Amount$.
  `SQLVarchar 255`, Optional.

6.4.8 Table: CompTariffedCharge

This table has the following 4 fields:

- **CompTariffedCharge**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.
- **ctcNrOfDays**
  This attribute implements the relation $CompTariffedCharge \xrightarrow{ctcNrOfDays} Integer$.
  `SQLVarchar 255`, Optional.
- **ctcDailyAmount**
  This attribute implements the relation $CompTariffedCharge \xrightarrow{ctcDailyAmount} Amount$.
  `SQLVarchar 255`, Optional.
- **computedTariffedCharge**
  This attribute implements the relation $CompTariffedCharge \xrightarrow{computedTariffedCharge} Amount$.
  `SQLVarchar 255`, Optional.

6.4.9 Table: Date

This table has the following 1 fields:

- **Date**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

6.4.10 Table: DateDifference

This table has the following 4 fields:

- **DateDifference**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

- **firstDate**
  This attribute implements the relation $DateDifference \xrightarrow{firstDate} Date$.
  `SQLVarchar 255`, Optional.

- **lastDate**
  This attribute implements the relation $DateDifference \xrightarrow{lastDate} Date$.
  `SQLVarchar 255`, Optional.

- **computedNrOfExcessDays**
  This attribute implements the relation $DateDifference \xrightarrow{computedNrOfExcessDays} Integer$.
  `SQLVarchar 255`, Optional.

6.4.11 Table: DateDifferencePlusOne

This table has the following 4 fields:

- **DateDifferencePlusOne**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

- **earliestDate**
  This attribute implements the relation $DateDifferencePlusOne \xrightarrow{earliestDate} Date$.
  `SQLVarchar 255`, Optional.

- **latestDate**
  This attribute implements the relation $DateDifferencePlusOne \xrightarrow{latestDate} Date$.
  `SQLVarchar 255`, Optional.

- **computedRentalPeriod**
  This attribute implements the relation $DateDifferencePlusOne \xrightarrow{computedRentalPeriod} Integer$.
  `SQLVarchar 255`, Optional.

6.4.12 Table: Distance

This table has the following 1 fields:

- **Distance**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

6.4.13 Table: DistanceBetweenLocations

This table has the following 3 fields:

- **DistanceBetweenLocations**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

- **computedLocationPenaltyCharge**
  This attribute implements the relation $DistanceBetweenLocations \xrightarrow{computedLocationPenaltyCharge} Amou$
  `SQLVarchar 255`, Optional.
- **distance**
  This attribute implements the relation $DistanceBetweenLocations \xrightarrow{distance} Distance$.
  `SQLVarchar 255`, Optional.

6.4.14 Table: DrivingLicense

This table has the following 1 fields:

- **DrivingLicense**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

6.4.15 Table: Integer

This table has the following 1 fields:

- **Integer**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

6.4.16 Table: Location

This table has the following 1 fields:

- **Location**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

6.4.17 Table: Model

This table has the following 1 fields:

- **Model**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

6.4.18 Table: Person

This table has the following 1 fields:

- **Person**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

6.4.19 Table: RentalCase

This table has the following 21 fields:

- **RentalCase**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.
- **contractedStartDate**
  This attribute implements the relation $RentalCase \xrightarrow{contractedStartDate} Date.$
  `SQLVarchar 255`, Optional.
- **contractedEndDate**
  This attribute implements the relation $RentalCase \xrightarrow{contractedEndDate} Date.$
  `SQLVarchar 255`, Optional.
- **contractedCarType**
  This attribute implements the relation $RentalCase \xrightarrow{contractedCarType} CarType.$
  `SQLVarchar 255`, Optional.
- **contractedPickupBranch**
  This attribute implements the relation $RentalCase \xrightarrow{contractedPickupBranch} Branch.$
  `SQLVarchar 255`, Optional.
- **contractedDropoffBranch**
  This attribute implements the relation $RentalCase \xrightarrow{contractedDropoffBranch} Branch.$
  `SQLVarchar 255`, Optional.
- **rcRenter**
  This attribute implements the relation $RentalCase \xrightarrow{rcRenter} Person.$
  `SQLVarchar 255`, Optional.
- **rcDriver**
  This attribute implements the relation $RentalCase \xrightarrow{rcDriver} Person.$
  `SQLVarchar 255`, Optional.
- **rcAssignedCar**
  This attribute implements the relation $RentalCase \xrightarrow{rcAssignedCar} Car.$
  `SQLVarchar 255`, Optional.
- **rcDroppedOffCar**
  This attribute implements the relation $RentalCase \xrightarrow{rcDroppedOffCar} Car.$
  `SQLVarchar 255`, Optional.
- **rcDroppedOffDate**
  This attribute implements the relation $RentalCase \xrightarrow{rcDroppedOffDate} Date.$
  `SQLVarchar 255`, Optional.
- **rcDroppedOffBranch**
  This attribute implements the relation $RentalCase \xrightarrow{rcDroppedOffBranch} Branch.$
  `SQLVarchar 255`, Optional.

- **contractualRentalPeriod**
  This attribute implements the relation $RentalCase \xrightarrow{contractualRentalPeriod} Integer$.
  `SQLVarchar 255`, Optional.

- **contractualBasicCharge**
  This attribute implements the relation $RentalCase \xrightarrow{contractualBasicCharge} Amount$.
  `SQLVarchar 255`, Optional.

- **rentalPeriod**
  This attribute implements the relation $RentalCase \xrightarrow{rentalPeriod} Integer$.
  `SQLVarchar 255`, Optional.

- **rentalBasicCharge**
  This attribute implements the relation $RentalCase \xrightarrow{rentalBasicCharge} Amount$.
  `SQLVarchar 255`, Optional.

- **rentalExcessPeriod**
  This attribute implements the relation $RentalCase \xrightarrow{rentalExcessPeriod} Integer$.
  `SQLVarchar 255`, Optional.

- **rentalPenaltyCharge**
  This attribute implements the relation $RentalCase \xrightarrow{rentalPenaltyCharge} Amount$.
  `SQLVarchar 255`, Optional.

- **rentalLocationPenaltyCharge**
  This attribute implements the relation $RentalCase \xrightarrow{rentalLocationPenaltyCharge} Amount$.
  `SQLVarchar 255`, Optional.

- **rentalCharge**
  This attribute implements the relation $RentalCase \xrightarrow{rentalCharge} Amount$.
  `SQLVarchar 255`, Optional.

- **rcMaxRentalDuration**
  This attribute implements the relation $RentalCase \xrightarrow{rcMaxRentalDuration} Integer$.
  `SQLVarchar 255`, Optional.

6.4.20 Table: SESSION

This table has the following 11 fields:

- **SESSION**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

- **sessionToday**
  This attribute implements the relation $SESSION \xrightarrow{sessionToday} Date$.
  `SQLVarchar 255`, Optional.

- **sessionUser**
  This attribute implements the relation $SESSION \xrightarrow{sessionUser} Person$.
  `SQLVarchar 255`, Optional.

- **sessionNewUserRC**
  This attribute implements the relation $SESSION \xrightarrow{sessionNewUserRC} RentalCase$.
  `SQLVarchar 255`, Optional, Unique.

- **sessionBranch**
  This attribute implements the relation $SESSION \xrightarrow{sessionBranch} Branch$.
  `SQLVarchar 255`, Optional.

- **sessionNewBranchRC**
  This attribute implements the relation $SESSION \xrightarrow{sessionNewBranchRC} RentalCase$.
  `SQLVarchar 255`, Optional.

- **sessionPickupPerson**
  This attribute implements the relation $SESSION \xrightarrow{sessionPickupPerson} Person$.
  `SQLVarchar 255`, Optional.

- **sessionDroppedOffCar**
  This attribute implements the relation $SESSION \xrightarrow{sessionDroppedOffCar} Car$.
  `SQLVarchar 255`, Optional.

- **sessionDroppedOffPerson**
  This attribute implements the relation $SESSION \xrightarrow{sessionDroppedOffPerson} Person$.
  `SQLVarchar 255`, Optional.

- **sNewRC**
  This attribute implements the relation $SESSION \xrightarrow{sNewRC} RentalCase$.
  `SQLVarchar 255`, Optional.

- **sPerson**
  This attribute implements the relation $SESSION \xrightarrow{sPerson} Person$.
  `SQLVarchar 255`, Optional.

6.4.21 Table: YesNoAnswer

This table has the following 1 fields:

- **YesNoAnswer**
  This attribute is the primary key.
  `SQLVarchar 255`, Mandatory, Unique.

6.4.22 Table: dateIntervalCompTrigger

This is a link-table, implementing the relation $Date \xrightarrow{dateIntervalCompTrigger} Date$.
It contains the following columns:

- **SrcDate**
  This attribute is a foreign key to Date
  `SQLVarchar 255`, Mandatory.

- **TgtDate**
  This attribute implements the relation $Date \xrightarrow{dateIntervalCompTrigger} Date$.
  `SQLVarchar 255`, Mandatory.

6.4.23 Table: dateIntervalIsWithinMaxRentalDuration

This is a link-table, implementing the relation $Date \xrightarrow{dateIntervalIsWithinMaxRentalDuration} Date$.
It contains the following columns:

- **SrcDate**
  This attribute is a foreign key to Date
  `SQLVarchar 255`, Mandatory.
- **TgtDate**
  This attribute implements the relation $Date \xrightarrow{dateIntervalIsWithinMaxRentalDuration} Date$.
  `SQLVarchar 255`, Mandatory.

6.4.24 Table: distbranch

This is a link-table, implementing the relation $DistanceBetweenLocations \xrightarrow{distbranch} Branch$.
It contains the following columns:

- **DistanceBetweenLocations**
  This attribute is the primary key.
  `SQLVarchar 255`, Optional.
- **Branch**
  This attribute implements the relation $DistanceBetweenLocations \xrightarrow{distbranch} Branch$.
  `SQLVarchar 255`, Optional.

6.4.25 Table: maxRentalDuration

This is a link-table, implementing the relation $CarRentalCompany \xrightarrow{maxRentalDuration} Integer$.
It contains the following columns:

- **CarRentalCompany**
  This attribute is a foreign key to CarRentalCompany
  `SQLVarchar 255`, Mandatory.
- **Integer**
  This attribute implements the relation $CarRentalCompany \xrightarrow{maxRentalDuration} Integer$.
  `SQLVarchar 255`, Mandatory.

6.4.26 Table: paymentHasBeenRequested

This is a link-table, implementing the relation $RentalCase \xrightarrow{paymentHasBeenRequested} RentalCase$.
It contains the following columns:

- **SrcRentalCase**
  This attribute is a foreign key to RentalCase
  `SQLVarchar 255`, Mandatory.
- **TgtRentalCase**
  This attribute implements the relation $RentalCase \xrightarrow{paymentHasBeenRequested} RentalCase$.
  `SQLVarchar 255`, Mandatory.

6.4.27 Table: rcBranchRequestedQ

This is a link-table, implementing the relation $RentalCase \xrightarrow{rcBranchRequestedQ} YesNoAnswer$.
It contains the following columns:

- **RentalCase**
  This attribute is a foreign key to RentalCase
  `SQLVarchar 255`, Mandatory.
- **YesNoAnswer**
  This attribute implements the relation $RentalCase \xrightarrow{rcBranchRequestedQ} YesNoAnswer$.
  `SQLVarchar 255`, Mandatory.

6.4.28 Table: rcCarHasBeenDroppedOff

This is a link-table, implementing the relation $RentalCase \xrightarrow{rcCarHasBeenDroppedOff} RentalCase$.
It contains the following columns:

- **SrcRentalCase**
  This attribute is a foreign key to RentalCase
  `SQLVarchar 255`, Mandatory.
- **TgtRentalCase**
  This attribute implements the relation $RentalCase \xrightarrow{rcCarHasBeenDroppedOff} RentalCase$.
  `SQLVarchar 255`, Mandatory.

6.4.29 Table: rcCarHasBeenPickedUp

This is a link-table, implementing the relation $RentalCase \xrightarrow{rcCarHasBeenPickedUp} RentalCase$.
It contains the following columns:

- **SrcRentalCase**
  This attribute is a foreign key to RentalCase
  `SQLVarchar 255`, Mandatory.

- **TgtRentalCase**
  This attribute implements the relation $RentalCase \xrightarrow{rcCarHasBeenPickedUp} RentalCase$.
  `SQLVarchar 255`, Mandatory.

6.4.30 Table: rcDrivingLicense

This is a link-table, implementing the relation $RentalCase \xrightarrow{rcDrivingLicense} DrivingLicense$.
It contains the following columns:

- **RentalCase**
  This attribute is a foreign key to RentalCase
  `SQLVarchar 255`, Mandatory.
- **DrivingLicense**
  This attribute implements the relation $RentalCase \xrightarrow{rcDrivingLicense} DrivingLicense$.
  `SQLVarchar 255`, Mandatory.

6.4.31 Table: rcKeysHandedOverQ

This is a link-table, implementing the relation $RentalCase \xrightarrow{rcKeysHandedOverQ} YesNoAnswer$.
It contains the following columns:

- **RentalCase**
  This attribute is a foreign key to RentalCase
  `SQLVarchar 255`, Mandatory.
- **YesNoAnswer**
  This attribute implements the relation $RentalCase \xrightarrow{rcKeysHandedOverQ} YesNoAnswer$.
  `SQLVarchar 255`, Mandatory.

6.4.32 Table: rcUserRequestedQ

This is a link-table, implementing the relation $RentalCase \xrightarrow{rcUserRequestedQ} YesNoAnswer$.
It contains the following columns:

- **RentalCase**
  This attribute is a foreign key to RentalCase
  `SQLVarchar 255`, Mandatory.
- **YesNoAnswer**
  This attribute implements the relation $RentalCase \xrightarrow{rcUserRequestedQ} YesNoAnswer$.
  `SQLVarchar 255`, Mandatory.

6.4.33 Table: rentalHasBeenEnded

This is a link-table, implementing the relation $RentalCase \xrightarrow{rentalHasBeenEnded} RentalCase$.
It contains the following columns:

- **SrcRentalCase**
  This attribute is a foreign key to RentalCase
  `SQLVarchar 255`, Mandatory.
- **TgtRentalCase**
  This attribute implements the relation $RentalCase \xrightarrow{rentalHasBeenEnded} RentalCase$.
  `SQLVarchar 255`, Mandatory.

6.4.34 Table: rentalHasBeenPromised

This is a link-table, implementing the relation $RentalCase \xrightarrow{rentalHasBeenPromised} RentalCase$.
It contains the following columns:

- **SrcRentalCase**
  This attribute is a foreign key to RentalCase
  `SQLVarchar 255`, Mandatory.
- **TgtRentalCase**
  This attribute implements the relation $RentalCase \xrightarrow{rentalHasBeenPromised} RentalCase$.
  `SQLVarchar 255`, Mandatory.

6.4.35 Table: rentalHasBeenStarted

This is a link-table, implementing the relation $RentalCase \xrightarrow{rentalHasBeenStarted} RentalCase$.
It contains the following columns:

- **SrcRentalCase**
  This attribute is a foreign key to RentalCase
  `SQLVarchar 255`, Mandatory.
- **TgtRentalCase**
  This attribute implements the relation $RentalCase \xrightarrow{rentalHasBeenStarted} RentalCase$.
  `SQLVarchar 255`, Mandatory.

6.4.36 Table: rentalIsPaidQ

This is a link-table, implementing the relation $RentalCase \xrightarrow{rentalIsPaidQ} YesNoAnswer$.
It contains the following columns:

- **RentalCase**
  This attribute is a foreign key to RentalCase
  `SQLVarchar 255`, Mandatory.

- **YesNoAnswer**
  This attribute implements the relation $RentalCase \xrightarrow{rentalIsPaidQ} YesNoAnswer$.
  `SQLVarchar 255`, Mandatory.

Glossary

Amount a sum of money, expressed in 'Euro'.. 7

Branch an office of a car rental company at a specific location.. 6

Brand the brand of a car.. 7

CarRentalCompany a company whose business is renting cars.. 6

CarType the brand and model of a car.. 7

DrivingLicense the identification number of a (valid) driving license.. 11

Location a city (at which a branch office is located).. 7

Model the model of a car.. 7

RentalCase an information object that contains all information about a rental, including contractual items, rental items, billing items etc.. 7

YesNoAnswer the answer to a question that must be 'Yes' or 'No'.. 14