Danmarks Tekniske Universitet

02291 System Integration

Examination Project

Toll System

Mikkel Riber Bojsen
s093255

Johan van Beusekom
s093251

Andreas Foldager
s093285

Kasper Aaquist Johansen
s112461

Martin Kasban Tange
s093280

May 16th 2013

Contents

1 Introduction

[Kasper, Mikkel] We have tried to work with an agile approach in this project. First we made a component diagram and a domain diagram and to tried to look at the business processes mentioned in the project description, this was done as an attempt to understand the project as a whole. After this was done we focused on the use cases that could cover the business processes in the project. We ordered them by importance and assigned them to development pairs that worked with one of the functionalities each (as we are an odd number, this was not done strictly as pairs).

We have used domain-driven design as an approach to break the task down into manageable tasks/functionalities that could be developed independently without having to complete the project with a waterfall process.

For the class diagram and object life cycle state machines we did exploratory modelling. Not explicitly with CRC cards, but by acting as classes while drawing diagrams and refactoring before creating final diagrams.

During the development phase we have come back to the parts we have modelled and corrected the design if we have changed it on another level in the domain. We have also done reviews in teams of cross-cutting concepts.

2 Requirements

In this chapter we describe the entire requirement analysis of the project. We start with analysing the business processes. We use this to create a domain analysis where we model the domain. We then provide the functional requirements of the project in the form of use case diagrams and provide details of the use cases we have implemented. Finally we provide a short list of some non-functional requirements we have identified.

2.1 Business Processes

In this section we provide a short overview of the business processes of the company.

2.1.1 Buying and using toll tags

[Andreas, Mikkel] In Figure 2.1 we see how a driver can order a toll tag and repeatedly use the motorway. He is billed for his use every month.

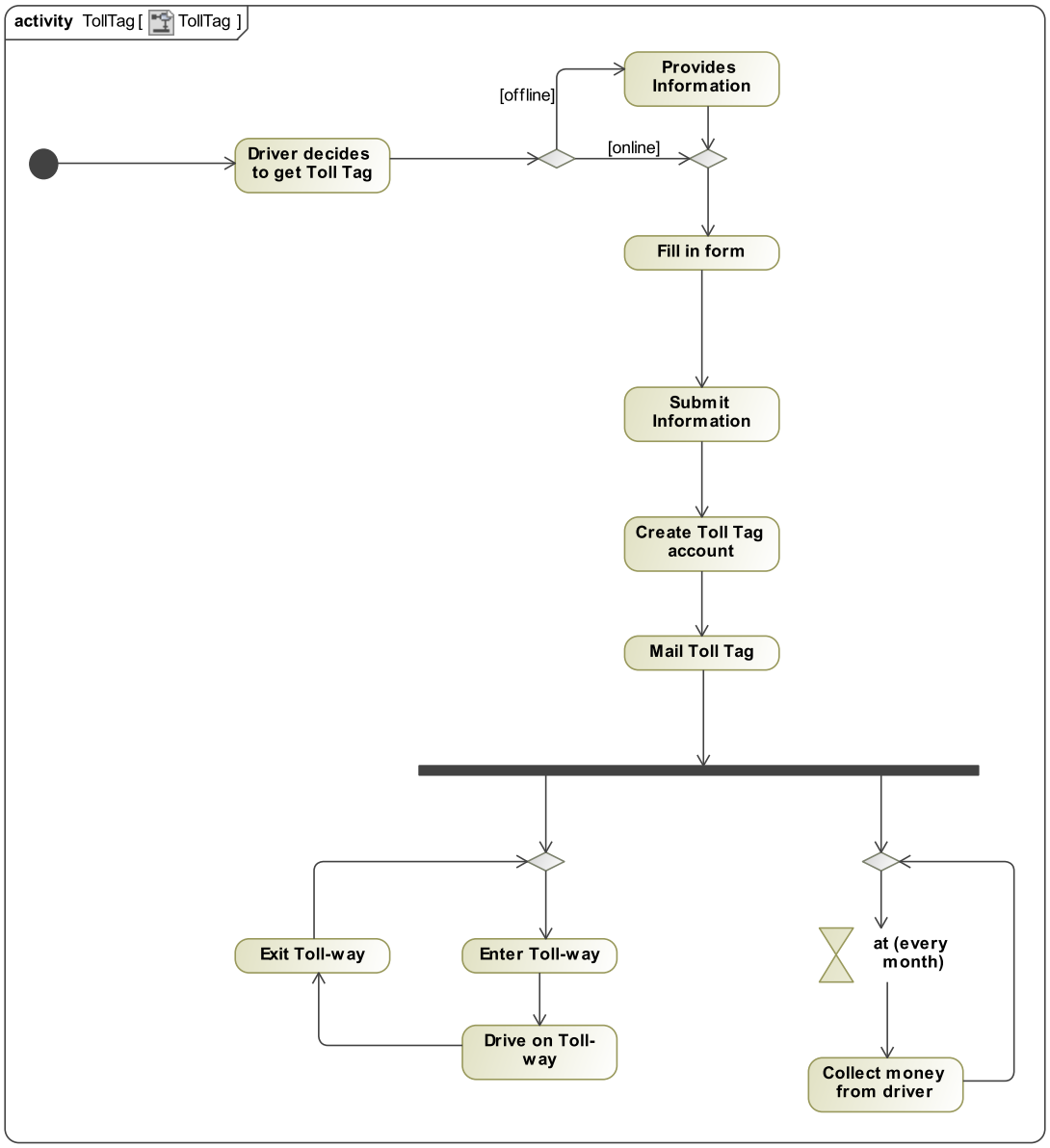

Figure 2.1: Activity diagram for buying and using toll tags. *[Andreas, Mikkel]*

2.1.2 Using tickets

[Johan, Martin] In Figure 2.2 we see how the process of using tickets to enter and exit the motorway is modelled. It is possible that the checkout fails due to an expired ticket, in which case a cashier is summoned to resolve the issue.

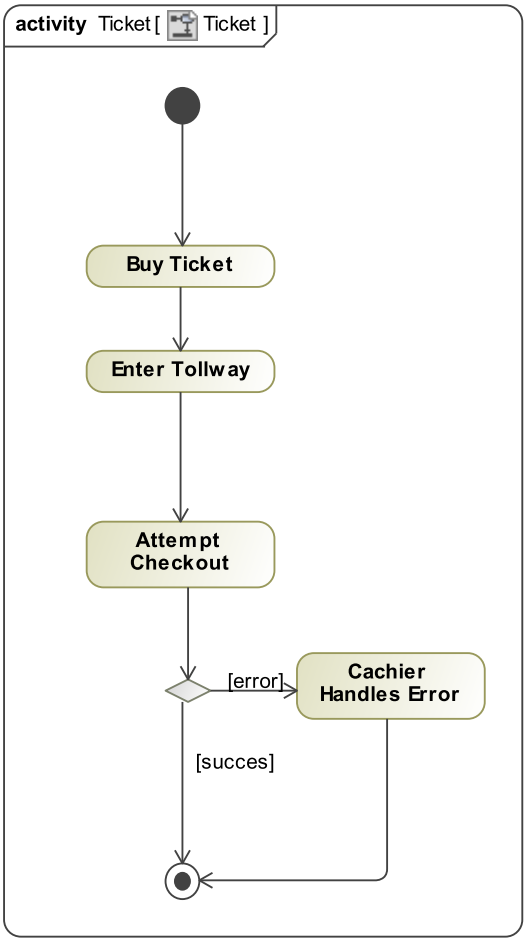

Figure 2.2: Activity diagram for buying and using tickets. *[Johan, Martin]*

2.1.3 Manager tasks

[Andreas, Kasper] In Figure 2.3 we see how the different types of managers uses the system. Each manager has their own set of tasks they can do and some of them happens in different orders.

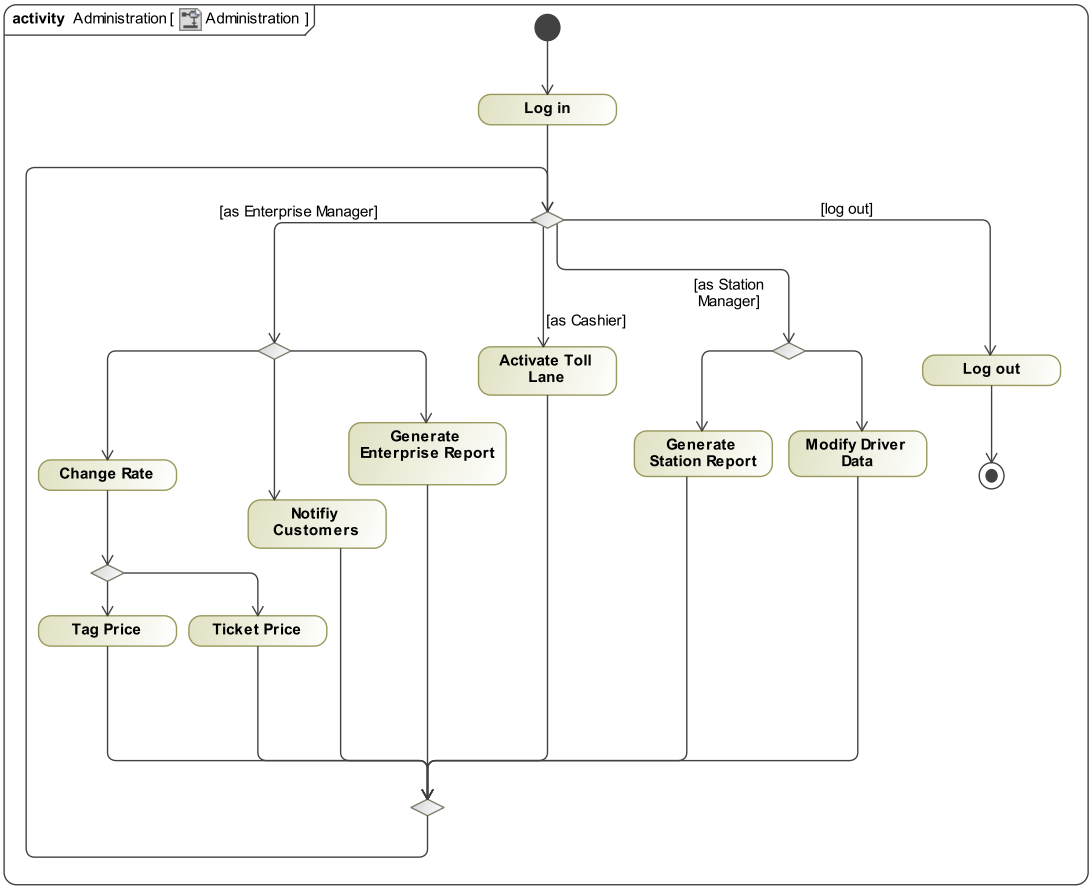

Figure 2.3: Activity diagram for managing the system. *[Andreas, Kasper]*

2.2 Domain Analysis

In this section we provide a glossary of important terms in the domain along with descriptions of the individual terms. We provide a class diagram showing a model of the domain and finally we provide acceptance tests for the system.

2.2.1 Glossary

[Mikkel, Andreas]

Term	Synonyms	Description
Toll Lane		A single lane of a bigger road that either enters or exits a pay-to-use motorway.
Toll Ticket		A ticket which indicates that you are allowed to be driving on a given pay-to-use motorway. It is used in the check-out process to validate the usage of it.
Normal Toll Lane		A Toll lane that support payment with credit card and the possibility of paying with cash
Express Toll Lane		A specialised Toll Lane which accepts using Tag, which can be read wireless by an antenna when a vehicle approaches. These lanes make the process of accessing and leaving the motorway smoother and faster.
Toll Tag		A wireless tag which is used in Express Toll Lanes. The toll on these tags are paid monthly.
Payment		Vehicles can pay for a trip either using a Toll Tag or a Toll Ticket.
Vehicle		The cost of a Payment varies according to the type of the vehicle. Vehicle types are motorcycle, car and truck.
Payment Method		When a Toll Ticket is being bought, it can be be paid for by both using cash or credit.
Trip		Ongoing trips consists of a Check-In and a Payment. Trips that have been completed also consist of a Check-out.
Cashier		An employee that normally sits in a normal check-in, and ensures that all lanes function properly, i.e. no cars get stuck in their payment or check-out
Driver		The person responsible for driving the car, and thereby the one responsible for paying for a trip
Lane Pass	Pass	Each time a driver either check-in or check-out of the motorway
Check-in		The driver is registered as being on the the motorway either by the use of a ticket or his toll tag
Check-out		The driver ends his trip by returning his ticket or using his toll tag
Check-out Failure		The driver ends his trip by getting help from a cashier to check-out
Check-in Failure		The driver start his trip by getting help from a cashier to check-in
Trip		The period of time the driver spends on the road between check-in and check-out

2.2.2 Domain model

[Kasper, Martin] The Toll System is composed of one single Enterprise Server that is the core of the system, which has access to several Stations that is placed throughout the motorway that we monitor. It will also contain all Toll Tags registered in the Toll System, which enables a Driver to ease his registration when he wishes to use the motorway. The Toll Tag will contain what vehicle is assigned to it, as well as what Trips it has taken on the motorway, identified by what stations the Trip started and ended on.

The Stations which is contained within the system, has knowledge of several local Toll Lanes, that allow Drivers to pass onto or exit the motorway. These Toll Lanes can take on two roles, either they are Express Lanes, only accepting Toll Tags, or they are a Normal Lane accepting

credit cards or cash in case a Cashier is assigned to the Toll Lane. The Toll Lanes will likewise have knowledge of when a Lane Pass has occurred, i.e. when a driver either enter or exit the motorway, and how it was performed, with a Ticket or with the use of a Toll Tag.

An illustration of these ideas has been comprised into the Domain Model seen in Figure 2.4

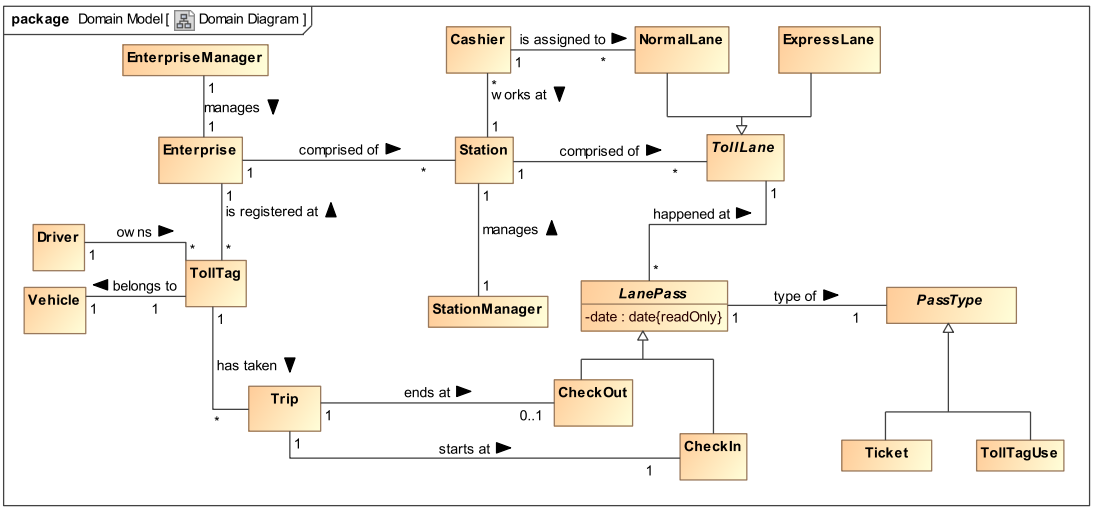

Figure 2.4: Domain model of the Toll System. *[Johan, Martin]*

System Users

[Mikkel, Andreas] If we look into which users the Toll System will encounter during runtime, we have two distinct groups, the Drivers which are the users that will use the motorway and thereby pay for the maintenance cost of the Toll System. Beside the Drivers, we have the Employees, which can be simple cashiers that can handle a single Toll Lane on the Station at a time. Another type of employee, are the Managers that will be able to generate reports, change the price for travelling etc. The Station Manager, responsible for a single Station can only generate report, whereas the Enterprise Manager is a specialization of the Station Manager, and has full access to all task a manager can do. The Toll System in itself will only contains a single Enterprise Manager. An overview of the taxonomy for the System Users can be seen in Figure 2.5.

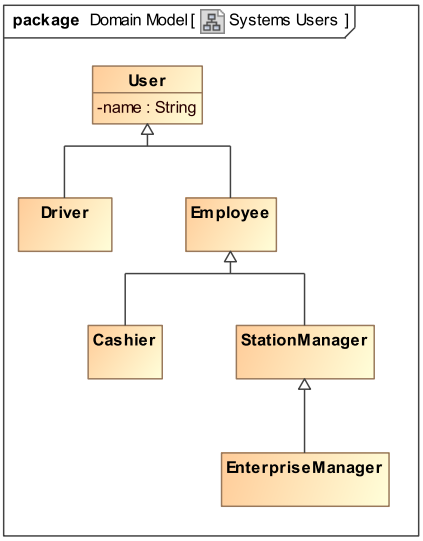

Figure 2.5: Systems Users. *[Kasper, Johan]*

2.3 Functional requirements

In this section, we provide an overview of the functional requirements for the system. We first show a series of use case diagrams where the use cases are grouped with similar use cases. Afterwards we provide five detailed use cases.

2.3.1 Use Case Diagrams

Log-in

[Andreas, Johan] We have that every Employee in the system can log-in. The specific actions varies depending on the level of authorisation that the actor who performed the log-in is assigned to.

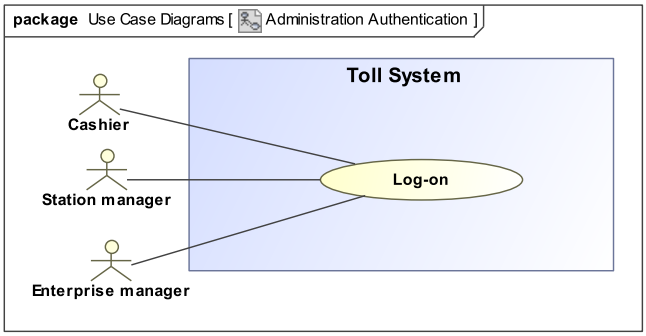

Figure 2.6: Log-in use case. *[Martin, Johan]*

Administration

[Mikkel, Kasper] If we look at the tasks that the managers of the Toll System can do. We have that both the Station and Enterprise manager who can generate reports according to their level of authorisation. Beside this the Enterprise manager is able to change the pricing of the toll way, for both ticket and toll tags. Finally the Enterprise manager can notify all customers registered in the system, about upcoming changes that will affect the customer. E.g. be a change in the price for using the toll way. An overview of the use cases related to administration if shown in Figure 2.7.

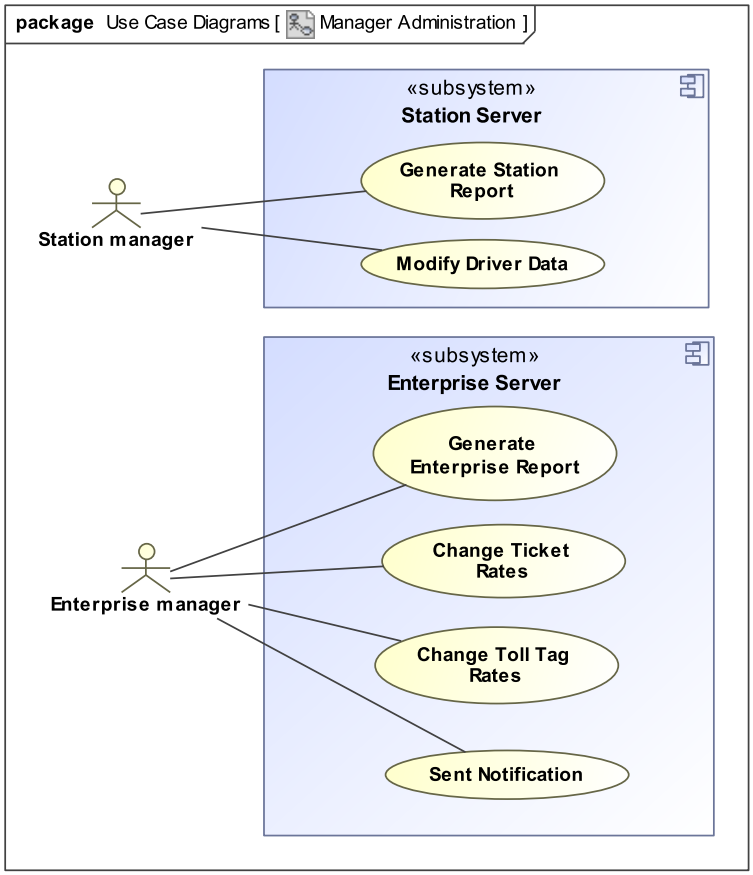

Figure 2.7: Use cases related to the administration that managers can perform on the Toll System. *[Johan, Mikkel]*

Buy Toll Tag

[Kasper, Martin] The process of buying a Toll Tag is divided into several steps, in which the Driver can manually do all parts of the process. A normal procedure is that the Driver will request a form, that he needs to fill out and return. When the form has been filled out and processed the Driver will receive a Toll Tag for future use. The cashier can likewise fill in the form, in case the Driver shows up at the station. The use cases related for buying a Toll Tag can

be seen in Figure 2.8

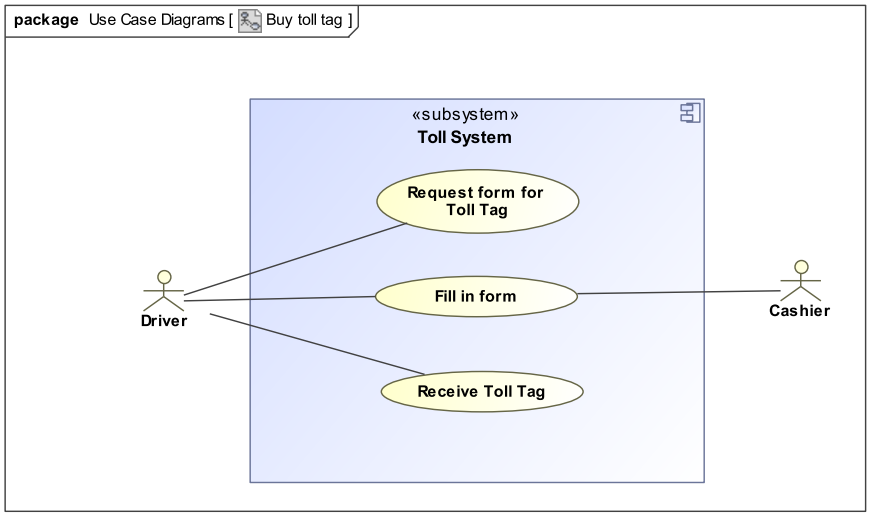

Figure 2.8: Use cases related to buying a toll tag. *[Mikkel, Andreas]*

Check-in

[Mikkel, Andreas] The use cases related to checking-in to the system, i.e. the Driver enters the toll way, can be seen in Figure 2.9. The idea is that two payment methods can fail, either by the Toll tag not being recognised or a credit card payment has failed. In this case a manual override will take place, with the help from a Cashier that is summoned to assist in the situation. In case the Driver decides to pay with cash, no problems can occur as a cashier is already there to resolve any problems they might experience during the money transfer.

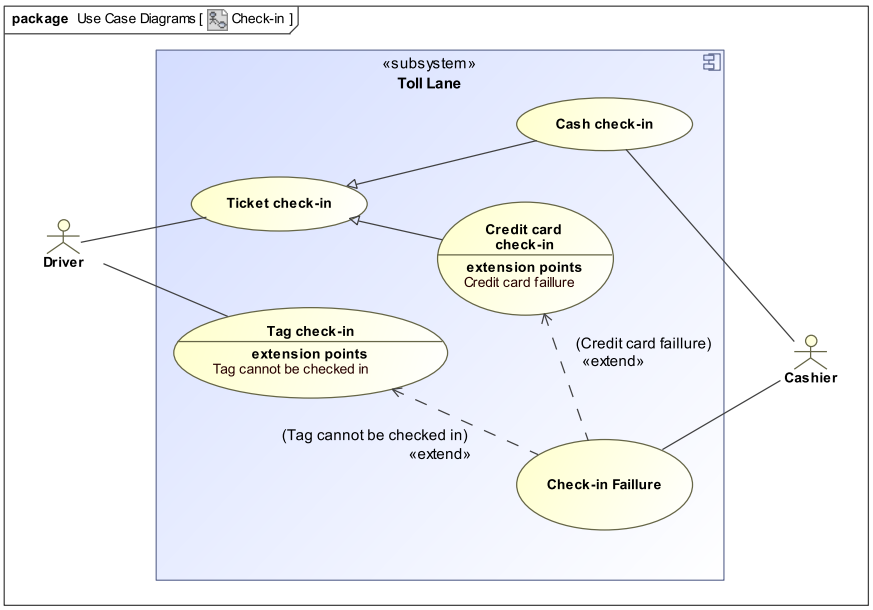

Figure 2.9: Use cases related to checking-in to the toll way. *[Johan, Martin]*

Check-out

[Martin, Kasper] The use cases related to checking-out of the system, i.e. the Driver exits the toll way can be seen in Figure 2.10. As shown there are two ways for a Driver to check-out from the toll way, by Toll Tag or by Ticket. In both cases the system may report a failure, either if the Toll Tag is not recognised or the Ticket is not valid any longer. So by having extension points on both use cases, check-out failure will react on the failure, and a Cashier will resolve the issue as seen fit.

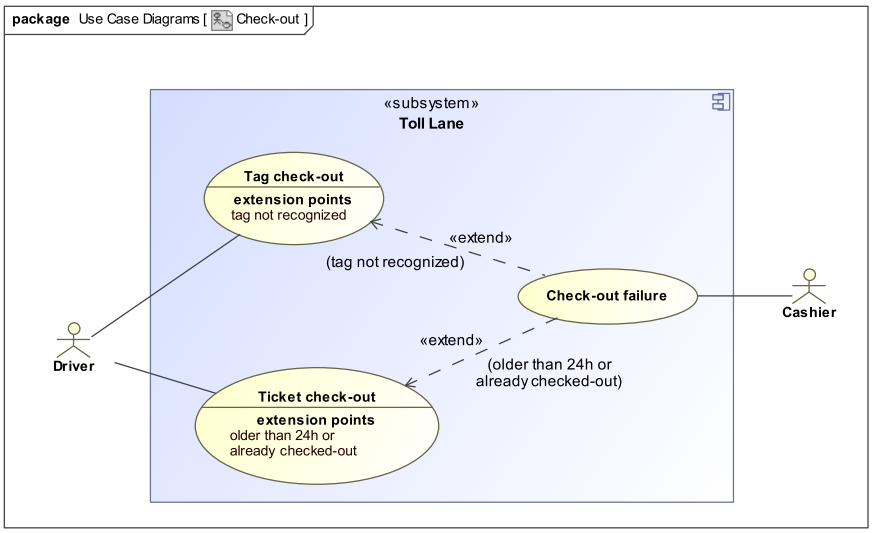

Figure 2.10: Use cases related to checking-out of the toll way. *[Mikkel, Andreas]*

2.3.2 Detailed Use Cases

We will in the remainder of the report focus primarily on 5 of the many uses cases displayed in subsection 2.3.1. We choose to focus on the main purpose of the Toll System, i.e. the use cases that make it possible for a Driver to actually use the motorway. This means we have chosen **Checkin-tag**, **Check-in Ticket**, **Check-out Ticket** and **Check-out Tag** as out primary focus. Beside these we also chose to focus on **Generate Enterprise report** in order to get some administration aspect into our system design.

Use Case:	**Check-in tag*[Martin, Johan]***
Description:	A driver wants to check-in with his vehicle with a toll tag.
Actor:	Driver, Cashier
Preconditions:	The driver has a toll tag in his vehicle.

Main Scenario:

1. The driver enters the express lane
2. The lane antenna recognises the toll tag.
3. The toll lane reports a check-in to the toll station.
4. The toll station replies with a check-in success.
5. The toll lane lifts the barrier and the driver enters the motorway.

Alternative Scenario:

2.a The antenna does not recognise the toll tag:

1. The check-in fails and a cashier is summoned.
2. The cashier resolves the issue as it sees fit.

4.a The toll station replies with a check-in failure.

1. The check-in fails and a cashier is summoned.
2. The cashier resolves the issue as it sees fit.

Postconditions:	The driver has entered the motorway.

Use Case:	**Check-in with ticket*[Kasper, Martin]***
Description:	A driver wants to check-in with his vehicle by buying a ticket.
Actor:	Driver, Cashier
Preconditions:	The driver is at the point of sale at a normal check-in lane.

Main Scenario:

1. The driver enters information about the vehicle into the toll lane system.
2. The toll lane system informs the driver about the price.
3. The driver chooses to pay with credit card.
4. The driver uses his credit card to pay for the ticket.
5. The ticket printer prints the ticket.
6. The driver takes his ticket from the ticket printer.
7. The toll barrier raises and the driver enters the motor way.

Alternative Scenario:

3.a The driver chooses to pay with cash and has the correct amount of money.

1. The driver pays the cashier the correct amount of money.
2. The cashier a issues a ticket.
3. Go to main scenario 5.

3.b The driver chooses to pay with cash and has too much money.

1. The driver pays the cashier.
2. The cashier gives the driver the correct amount of change.
3. The cashier issues a issues a ticket.
4. Go to main scenario 5.

4.a The credit card payment fails:

1. If not present, a cashier is summoned to handle the error.
2. If error is not resolvable, go to alternative scenario 3.a or 3.b.

Postconditions:	The driver has entered the motorway with a ticket that is valid for 24 hours.

Use Case:	**Ticket checkout *[Johan, Andreas]***
Description:	A driver wants to check-out his vehicle using a ticket.
Actor:	Driver, Cashier
Preconditions:	The driver has a ticket.

Main Scenario:

1. The driver approaches the toll lane exit and inserts the ticket.
2. The toll lane system lifts the barrier.
3. The driver drives past the barrier and the barrier closes.

Alternative Scenario:

2.a Ticket older than 24h or already checked out:

1. The check-out fails and a cashier is summoned.
2. The cashier resolves the issue as it sees fit.
3. Go to main Scenario step 2.

Postconditions:	The vehicle has exited the motorway and the checkout has been stored in the lane computer.

Use Case:	**Toll tag checkout *[Mikkel, Andreas]***
Description:	A driver wants to check-out his vehicle using a toll tag.
Actor:	Driver, Cashier
Preconditions:	The driver has a toll tag in his vehicle.

Main Scenario:

1. The driver approaches a toll lane with an antenna.
2. The toll lane antenna recognises the toll tag.
3. The toll lane reports the check-out to the toll station.
4. The toll station replies with a check-out success.
5. The toll lane lifts the barrier and the driver exits the motorway.

Alternative Scenario:

2.a The antenna does not recognise the toll tag:

1. The check-out fails and a cashier is summoned.
2. The cashier resolves the issue as it sees fit.
3. Go to Main Scenario step 5.

4.a The toll station replies with check-out failure.

1. The check-out fails and a cashier is summoned.
2. The cashier resolves the issue as it sees fit.
3. Go to Main Scenario step 5.

Postconditions:	The vehicle has exited the motorway and the checkout has been stored in the lane computer.

Use Case:	**Generate Reports*[Mikkel, Kasper]***
Description:	An Enterprise Manager wants to generate reports with statistics on, types of vehicles that have made check-in/check-out, single ticket usages, toll tag usages in a freely definable time period.
Actor:	Enterprise Manager
Preconditions:	The Enterprise Manager is logged on the enterprise client.

Main Scenario:

1. The Enterprise Manager query a report in a given time period.
2. The Enterprise client gathers information for the report from station clients.
3. The Enterprise Manager can see the report on his/her computer screen.

Alternative Scenario:

2.a Connection fail to Station clients while requesting query from all Station Servers.:

1. The Enterprise Manager query a report in a given time period.
2. The Enterprise Manager can see the report on his/her computer screen, the report dose not include data from Station Servers that could not be reached.
3. The Enterprise Manager can see on his/her computer screen which Station Server that where not included in the statistics.

Postconditions:

2.4 Non functional requirements

[Johan, Mikkel] This section provides a list of non-functional requirements we have derived based on the description of the Toll system combined with some assumptions about the business case for creating the Toll System. The system contains three tiers: Toll lanes, toll stations and a enterprise tier.

2.4.1 Toll lane

Each toll lane consist of a number of hardware components based on which type of toll lane it is.

All lanes

All lanes consists of the following hardware components:

- A toll lane computer
- A barrier

Check-in Lane

A normal check-in lane must have the following hardware components:

- Touchscreen for drivers to interact with the system
- A cash register for accepting cash payments
- A ticket printer
- A credit card reader for accepting credit card payments

Check-out Lane

A normal check-out lane must have the following hardware components.

- Ticket reader to validate check-outs

Express Lane

An express lane must have the following components.

- An Antenna to wirelessly read toll tags

2.4.2 Toll station

Each toll station consists of a number of check-in and check-out lanes. These are either express lanes or normal lanes.

2.4.3 Enterprise

The enterprise consists of a number of toll stations which are connected to the enterprise. The enterprise server is connected to the internet via a web server.

3 Acceptance tests

This chapter outline the acceptance test for the five detailed use cases described in Section 2.3.2. Fit[1] tests is used to test the implementation of the use cases. To see our FitNesse test server with the fit test goto the web address[2].

Use case "Check-in Tag" success scenario *[Martin, Johan]*

ActionFixture		
start	Driver	
check	Tag Recognized	true
check	Check-in created	true
check	Barrier Open	true

Use case "Check-in Tag" Tag not recognized scenario *[Martin, Johan]*

ActionFixture		
start	Driver	
check	Tag Recognized	false
press	Notify Cashier	
check	Problem Resolved	true
check	Barrier Open	true

Use case "Check-in Tag" Check-in failure scenario *[Martin, Johan]*

ActionFixture		
start	Driver	
check	Tag Recognized	true
check	Check-in created	false
press	Notify Cashier	
check	Problem Resolved	true
check	Barrier Open	true

Use case "Check-in Buy Ticket Cash" success scenario *[Kasper, Martin]*

[1] `http://fit.c2.com/`
[2] `http://goejl.dk:9020/TollSystem`

ActionFixture		
start	Driver	
press	Car	
check	Price	10 kr.
press	Cash	
enter	Cash	10 kr.
check	Ticket Received	true
check	Barrier Open	true

Use case "Check-in Buy Ticket Credit Card" success scenario *[Kasper, Martin]*

ActionFixture		
start	Driver	
press	Car	
check	Price	10 kr.
press	Credit Card	
enter	Credit Card Pin Code	1234
check	Credit Card verified	true
check	Ticket Received	true
check	Barrier Open	true

Alternative scenario: Use case "Check-in Buy Ticket Credit Card" verify failure *[Kasper, Martin]*

ActionFixture		
start	Driver	
press	Car	
check	Price	10 kr.
press	Credit Card	
enter	Credit Card Pin Code	41225
check	Credit Card verified	false
press	Notify Cashier	
check	Problem Resolved	true
check	Ticket Received	true
check	Barrier Open	true

Use case "Check-out Ticket" success scenario *[Johan, Andreas]*

ActionFixture		
start	Driver	
enter	Ticket	ticket
check	Valid ticket	true
check	Barrier Open	true

Use case "Check-out Ticket" Invalid ticket scenario *[Johan, Andreas]*

ActionFixture		
start	Driver	
enter	Ticket	ticket
check	Valid ticket	false
press	Notify cashier	
check	Problem resolved	true
check	Barrier Open	true

Use case "Check-out Tag" success scenario *[Mikkel, Andreas]*

ActionFixture		
start	Driver	
check	Tag recognized	true
check	Checked-out	true
check	Barrier Open	true

Use case "Check-out Tag" Tag not recognized scenario *[Mikkel, Andreas]*

ActionFixture		
start	Driver	
check	Tag recognized	false
press	Notify Cashier	
check	Problem resolved	true
check	Barrier Open	true

Use case "Check-out Tag" Check-out failed scenario *[Mikkel, Andreas]*

ActionFixture		
start	Driver	
check	Tag recognized	true
check	Checked-out	false
press	Notify Cashier	
check	Problem resolved	true
check	Barrier Open	true

Use case "Generate Enterprise Report" success scenario *[Mikkel, Kasper]*

ActionFixture		
start	Enterprise Manager	
press	Generated Enterprise Report	
enter	start	11/12/2012
enter	end	11/04/2013
check	Complete report	true

Use case "Generate Enterprise Report" Missing station scenario *[Mikkel, Kasper]*

ActionFixture		
start	Enterprise Manager	
press	Generated Enterprise Report	
enter	start	11/12/2012
enter	end	11/04/2013
check	Complete report	false

4 Design

This chapter will outline the design of the Toll System. We will provide a system design using a component diagram to describe the interfaces used between the etc. We also provide state machines outlining the protocols used to communicate between the components. Finally we will provide a detailed Class diagram and behaviour diagrams of the flow in the program.

4.1 Assumptions

[Andreas, Mikkel] When producing this project we have made some assumptions, both on the state of the system, on the project description and on the hardware used.

4.1.1 State of system

We have made the following assumptions about the state of the system.

Buy toll tag: We have assumed that infrastructure for signing up for toll tags is already in place. This is so we can focus on generating reports which spans more components instead.

Security: We have also assumed that there is no need for security in terms of passwords or other means of authentication.

4.1.2 Hardware components

We have made the following assumptions on the hardware used for the system.

Barrier: We have assumed that a barrier contains sensors so it will close automatically once a vehicle has passed.

Credit Card Reader: We have assumed that part of the credit card reader is communicating with banks.

Touchscreen: We have assumed that the touchscreen is used by the drivers, meanwhile the cashier will interact with a normal computer when he/she is logged-in on the normal lanes

4.1.3 Project Description

The following is assumptions we have made on the contents of the project description

Cashier resolution: We have assumed that when cashiers resolves errors at check-in and check-out they perform the transaction of money directly with the customer and does a manual override to open the barrier.

Printer: We have assumed that the printer attached to all normal check in lanes acts as a ticket printer.

4.2 Design Decisions

[Kasper, Johan] We have made some decisions on the design and to which extent we model things.

Malformed Antenna data: We have decided that any corrupt antenna data sent to the system triggers an immediate notification to the station, and have not modelled this.

Manuel Override: We have decided not to explicitly model the manual override done at check-in and check-out failures, and instead just leave it up to implementation.

Toll tag costs: We have not modelled how toll tags are charged as we see this as a separate use case. We have however made sure that the current system contains data needed for this calculation.

Edge cases: We have not modelled strange edge cases in our uses cases. Things such that the barrier closing on top of a vehicle or drivers not having enough money to pay for tickets that turn around etc.

TollLanes: We have decided to split toll lanes into check-in lanes and check-out lanes. We then use the components attached to the lane to differentiate express lanes from normal lanes. This was done as the functionality of toll lanes is mainly split on whether it is a check-in or a check-out.

Station Client: As we have not done any use case that required the station client, we have not modelled this in our project.

4.3 Component Design

[Mikkel, Martin] In this section we describe the design of our system at the component level. We will first give an overview of how the different components are connected, followed by a more detailed view of which interfaces they provide and require. Finally we provide protocol state machines for all interfaces.

4.3.1 System Components

[Martin, Mikkel] In Figure 4.1 we provide the overall layout of our components. We can see that every toll lane is modelled as a component that connects to multiple hardware components. The number of hardware components depends on the type of lane as pr the non-functional requirements in Section 2.4. Stations are modelled as a component which connects to all its lanes. The enterprise is modelled as a component which connects to all its stations.

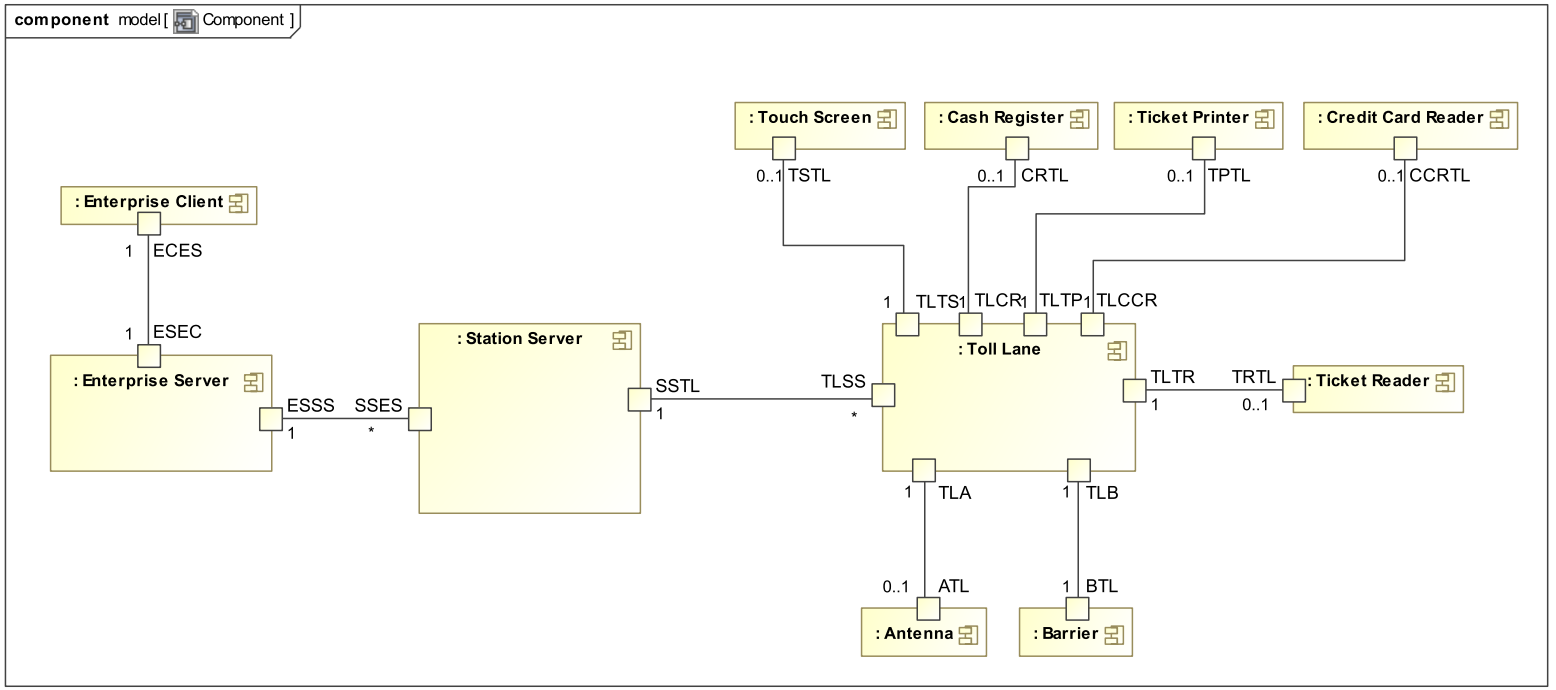

Figure 4.1: Overview of components and their connections. *[Kasper, Andreas]*

In Figure 4.2 we see an overview of which interfaces are used to communicate between the major components. We see that the Enterprise Client has an interface to the Enterprise server which is implemented on a webserver. All other components can communicate in both directions.

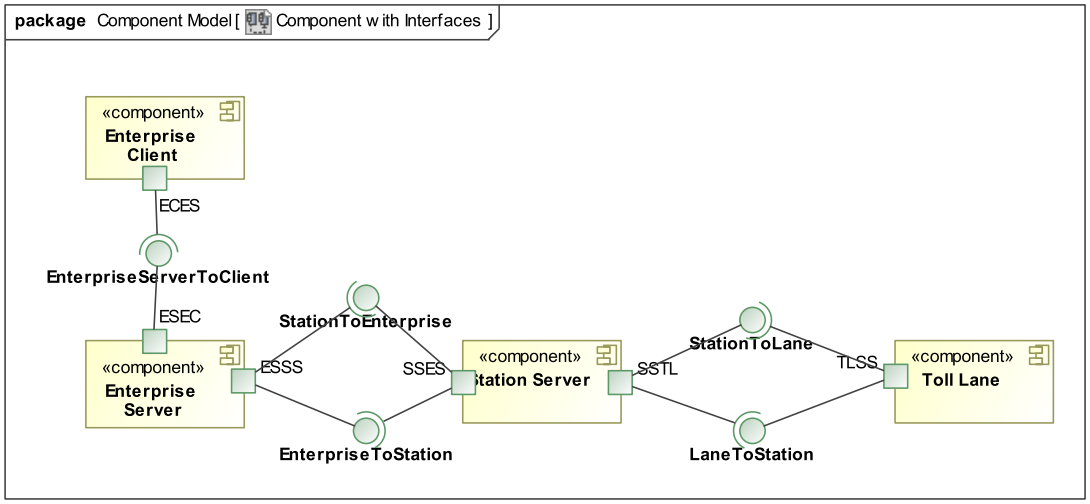

Figure 4.2: Overview of interfaces between major components. *[Johan, Andreas]*

In Figure 4.3 we see how the toll lane is connected to hardware. Most connections are one-way except the connection between the lane and a touch screen.

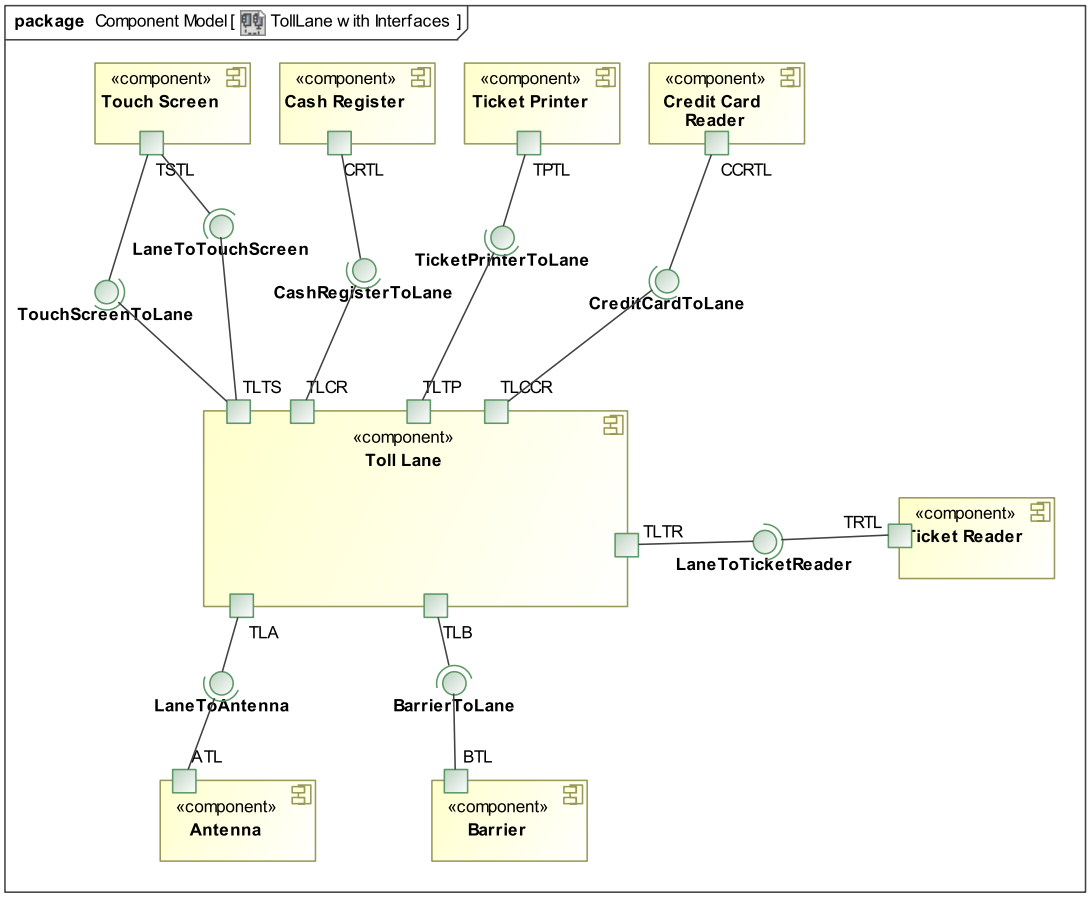

Figure 4.3: Overview of interfaces between a toll lane and its hardware. *[Mikkel, Kasper]*

4.3.2 Protocol State Machines

[Johan, Martin] In this section we show all protocol state machines. We first show the protocols between a lane and its hardware components, then we show the protocols for the interfaces between major components.

One-way communication

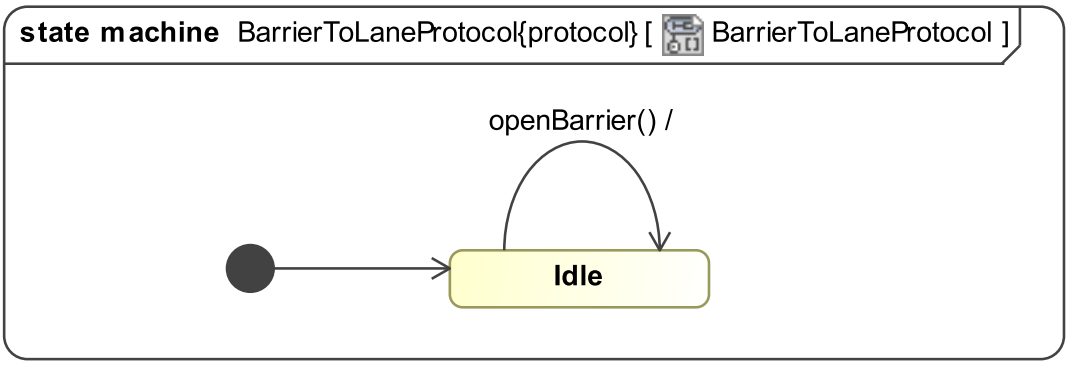

Figure 4.4: PSM for communicating with a barrier. *[Kasper, Mikkel]*

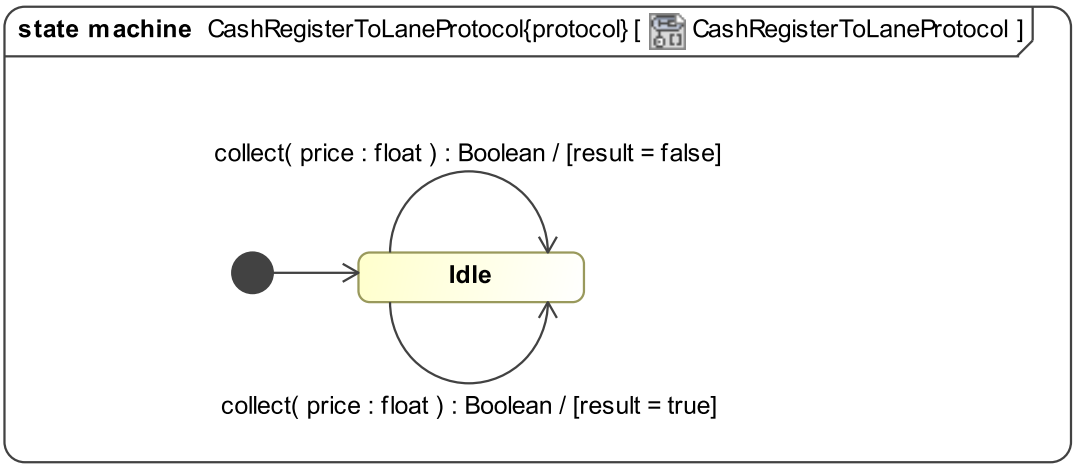

Figure 4.5: PSM for communicating with a cash register. *[Andreas, Martin]*

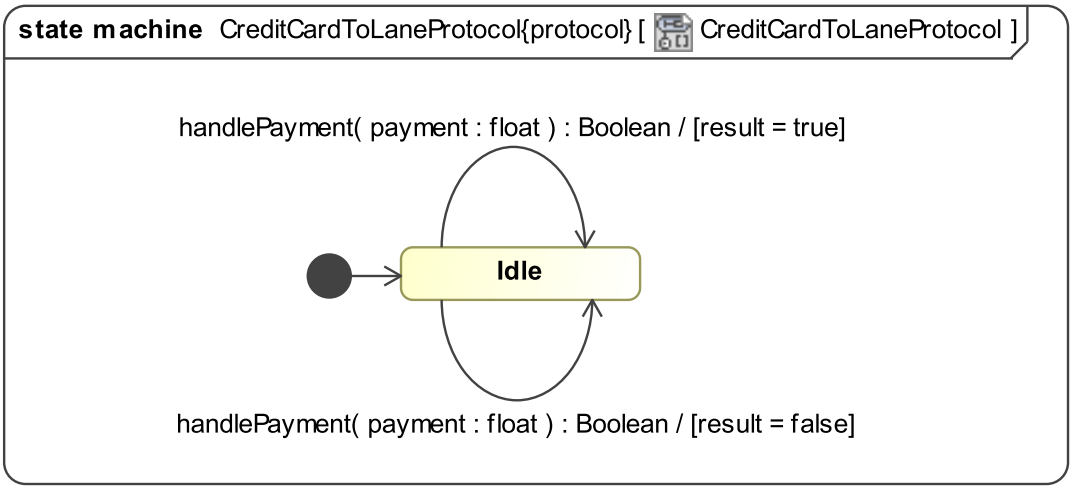

Figure 4.6: PSM for communicating with a credit card reader. *[Johan, Martin]*

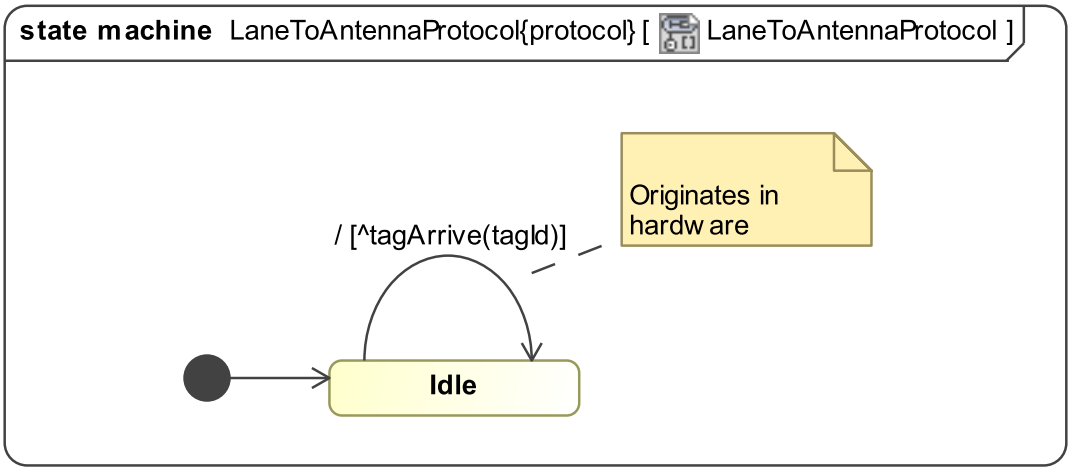

Figure 4.7: PSM showing how the antenna communicates with a lane.*[Andreas, Kasper]*

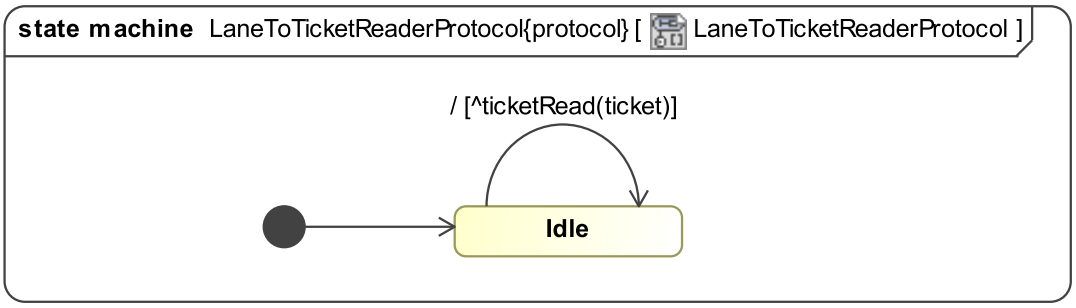

Figure 4.8: PSM showing how the ticket reader communicates with a lane. *[Mikkel, Martin]*

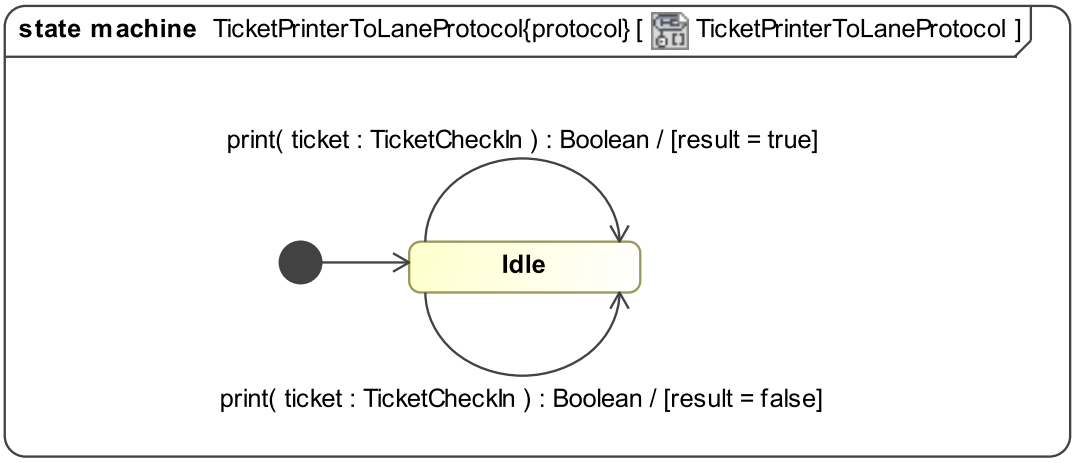

Figure 4.9: PSM showing how the ticket printer communicates with a lane. *[Johan, Andreas]*

Two-way communication

[Andreas, Johan] In the communication between the touch screen and the lane as shown in Figure 4.10 and Figure 4.11, we see that messages sent from the touch screen, originates from user interaction with the screen and that the lane answers with signals.

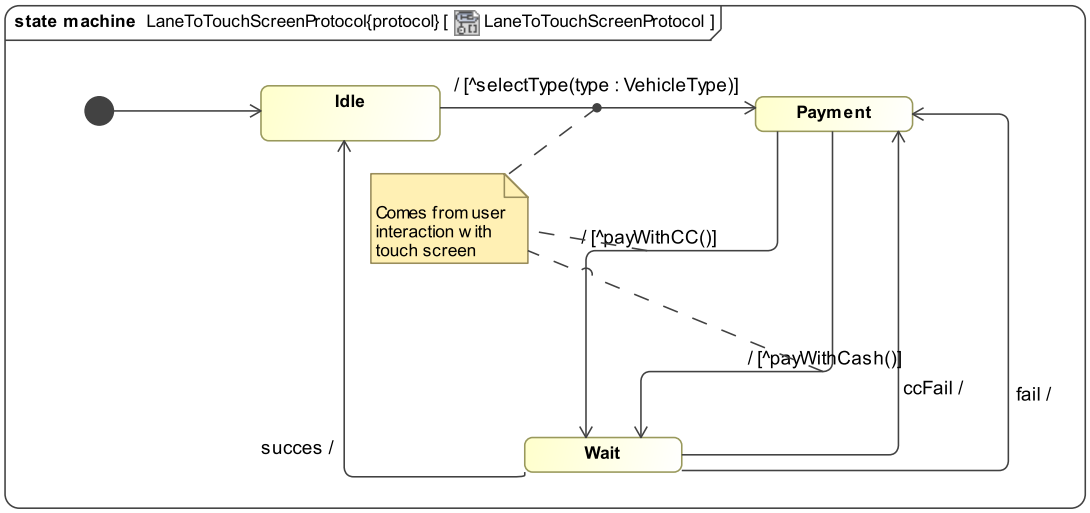

Figure 4.10: PSM showing how the touch screen communicates with the lane. *[Kasper, Johan]*

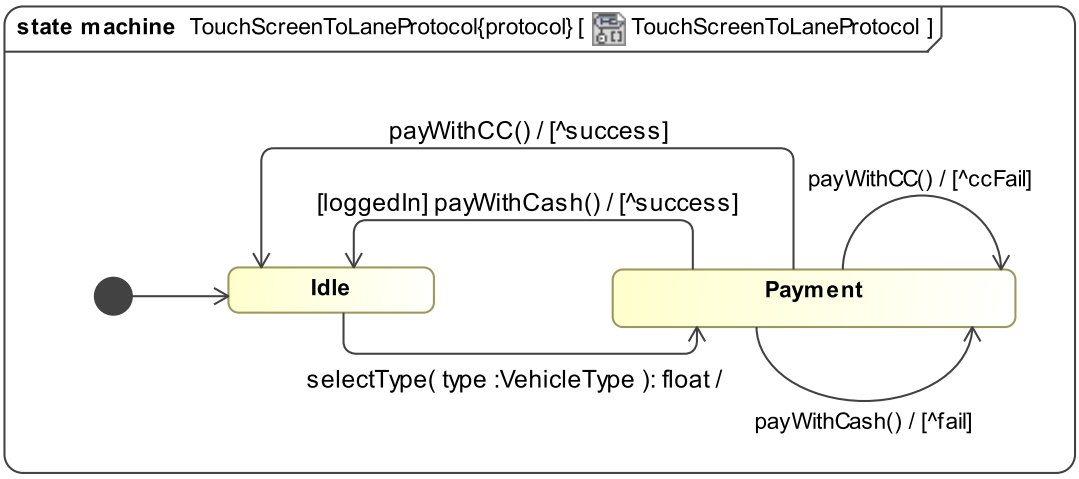

Figure 4.11: PSM showing how the lane communicates with the touch screen. *[Mikkel, Martin]*

Communication between major components

Here we show the protocols for communication between the major components.

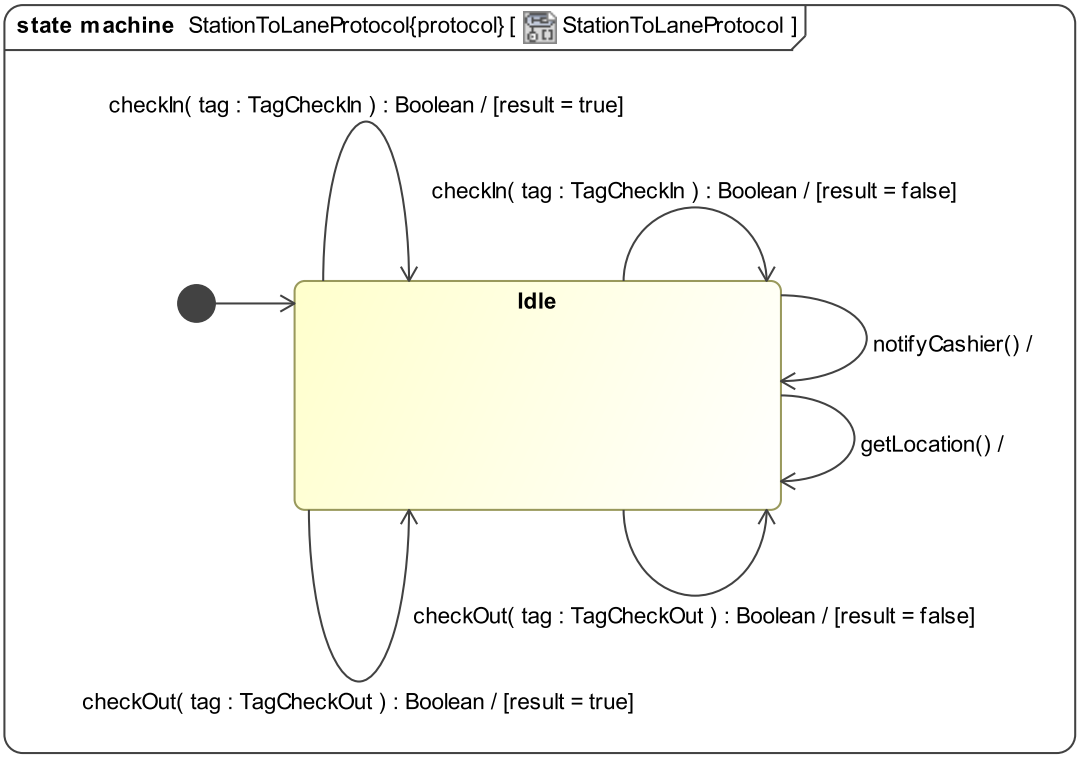

Figure 4.12: Communication from a toll lane to a station. *[Martin, Andreas]*

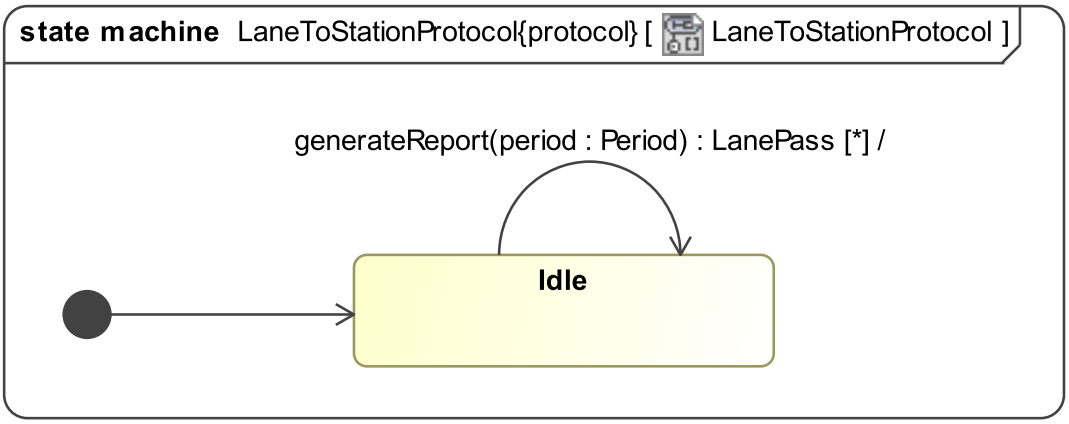

Figure 4.13: Communication from a station to a toll lane. *[Kasper, Johan]*

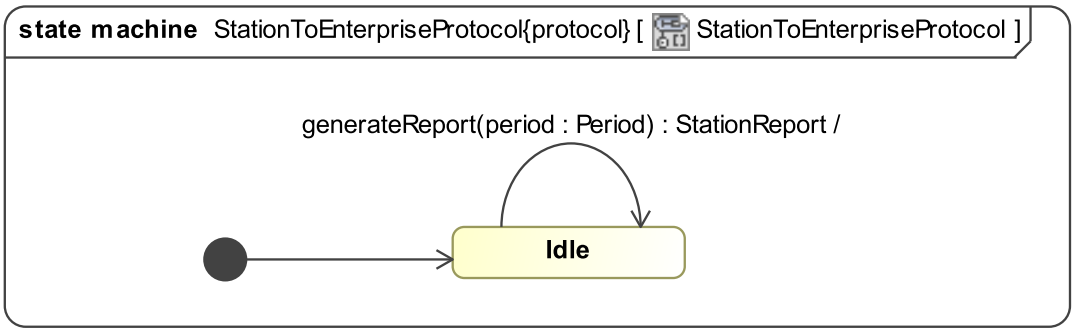

Figure 4.14: Communication from the enterprise server to the stations. *[Mikkel, Martin]*

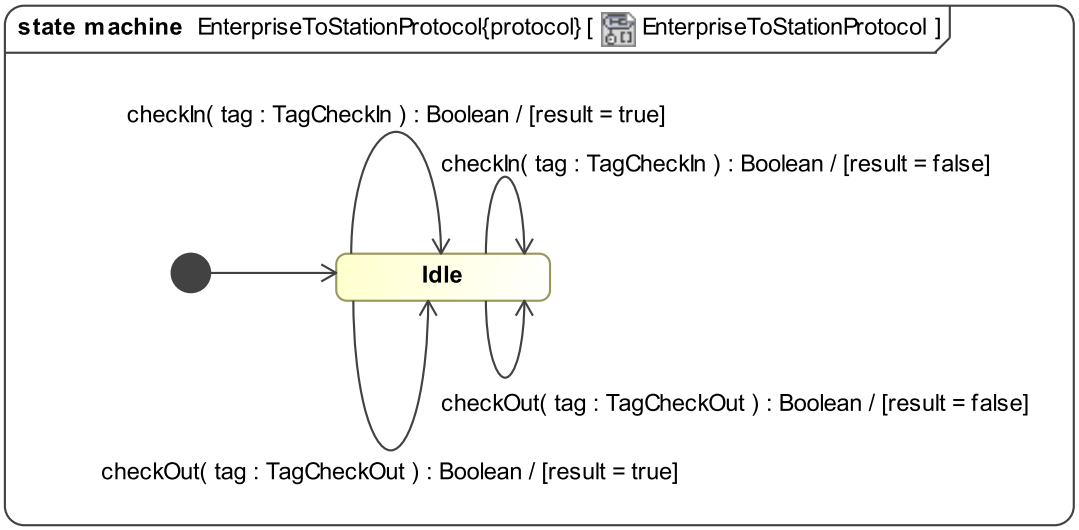

Figure 4.15: Communication from the stations to the enterprise server. *[Andreas, Kasper]*

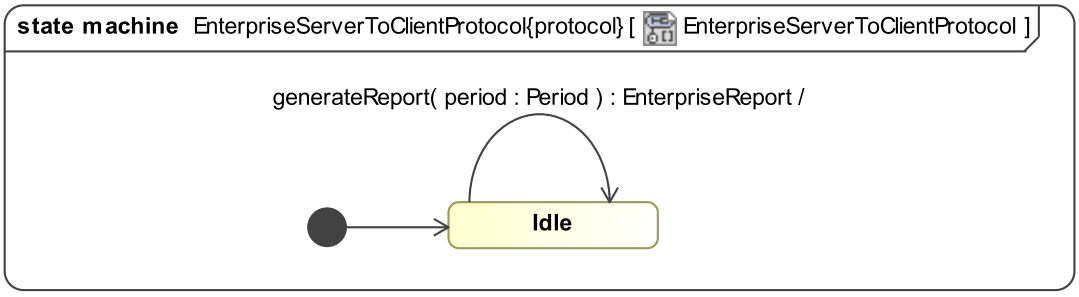

Figure 4.16: Communication from the enterprise client to the enterprise server. *[Johan, Mikkel]*

4.4 Class Design

[Johan, Martin] This section describes in details how the components are realised in classes as well as the behaviour of these classes. We do not explicitly implement interfaces to hardware that is not needed in other contexts, such as the antenna.

4.4.1 Class Diagrams

This section shows how components are realised as classes and how they implement the required interfaces.

Major components

Figure 4.17 shows how the major components are implemented. Tolltag is refined in Figure 4.20. The Toll lane is refined in Figure 4.18.

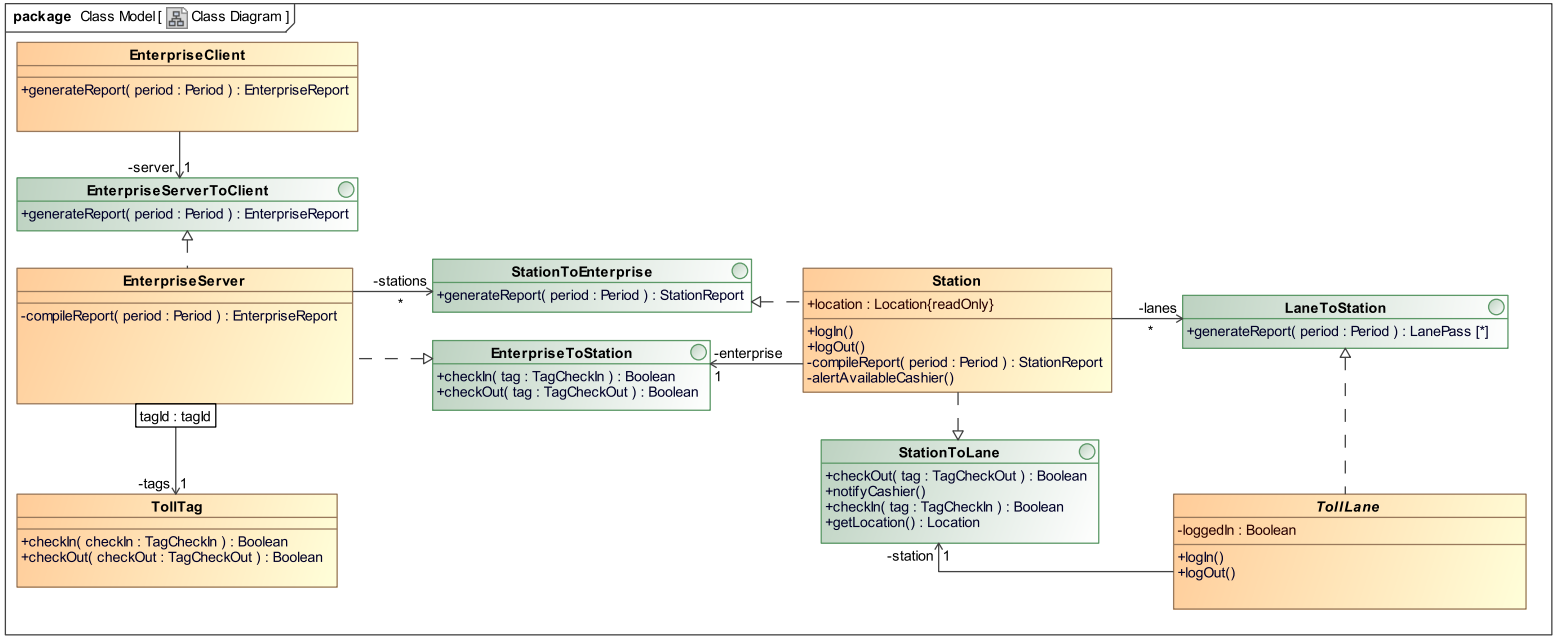

Figure 4.17: Class Model of major components.*[Andreas, Mikkel]*

TollLane

In Figure 4.18 we show how toll lanes implement interfaces as well as the hierarchy of lanes. We also show how lanes refer to their required interfaces. Lanepasses are refined in Figure 4.19.

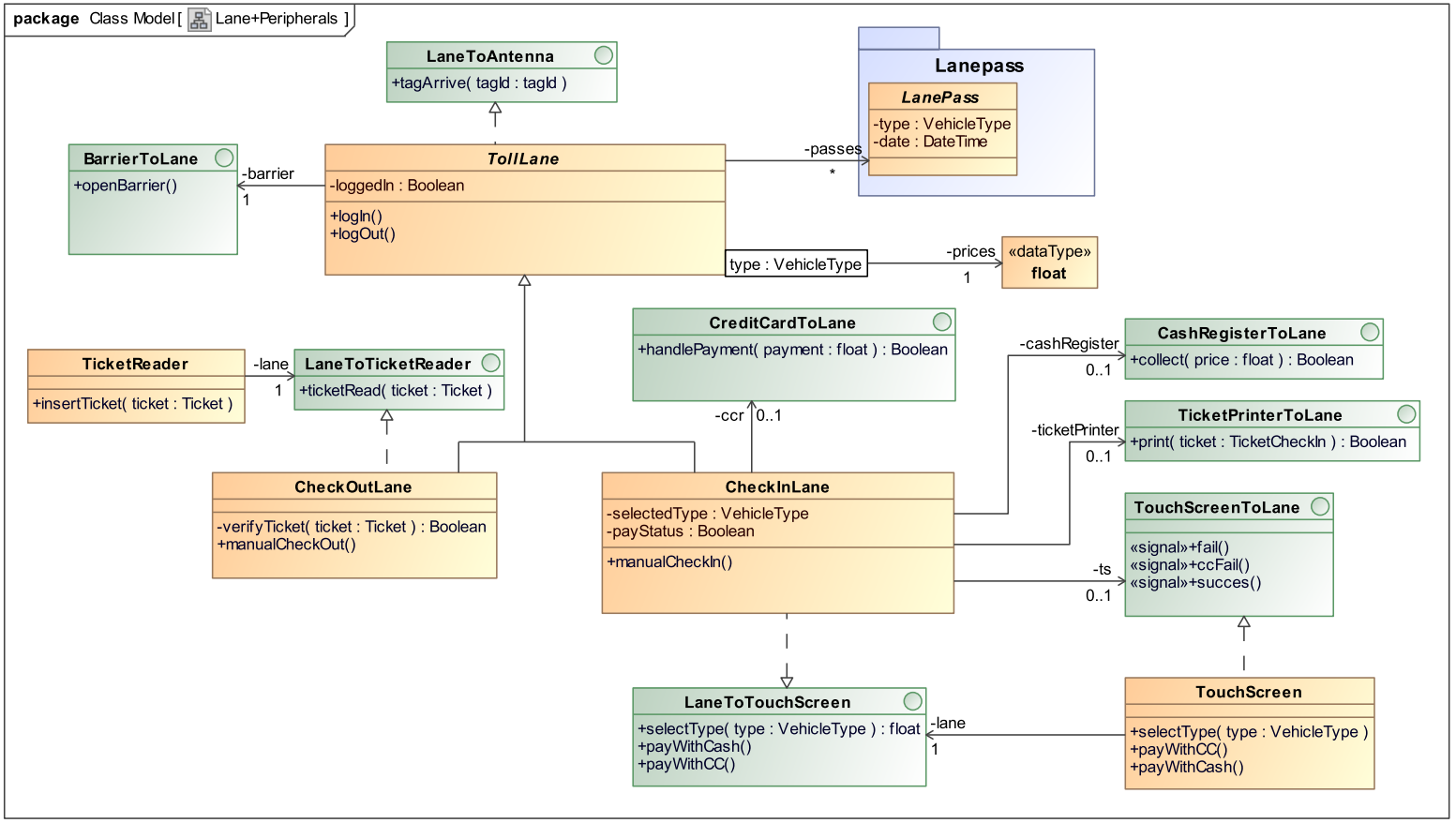

Figure 4.18: Class Model of the TollLane. *[Andreas, Kasper]*

Lanepass

In Figure 4.19 we show how lanepasses are modelled. This heirarchy is used in Toll tags and Toll lanes.

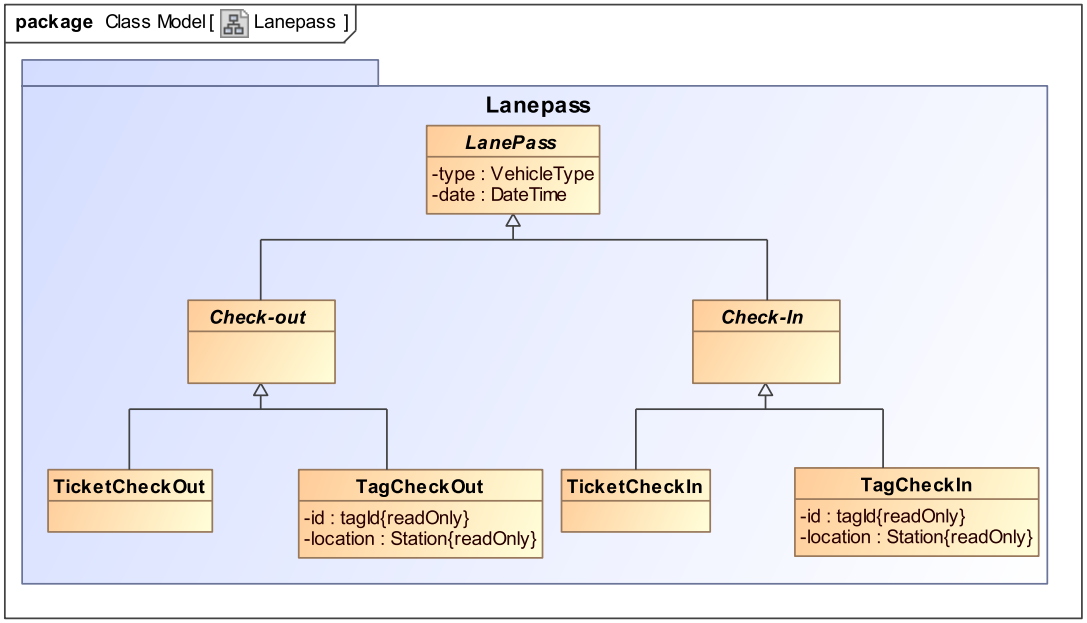

Figure 4.19: Class Model of the Lanepass.*[Johan, Mikkel]*

TollTag

In Figure 4.20 we show how the TollTag is modelled.

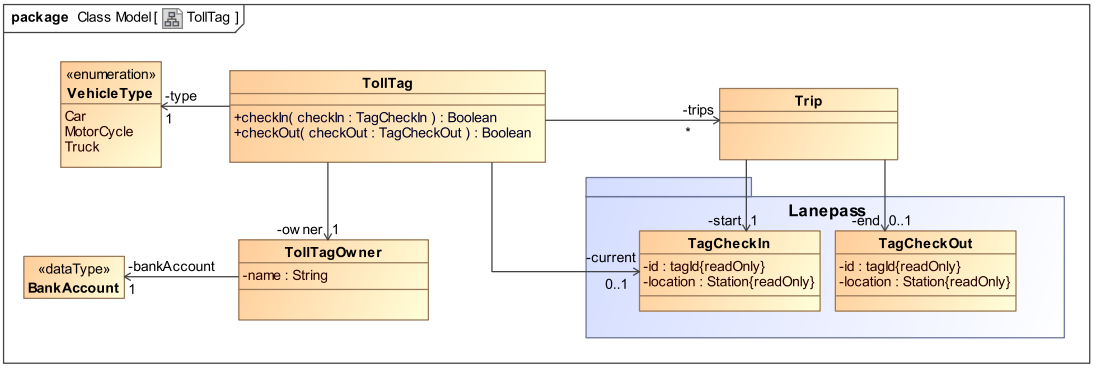

Figure 4.20: Class Model of TollTag*[Mikkel, Martin]*

DataTypes

In Figure 4.21 we show how the datatypes that does not fit anywhere else are modelled.

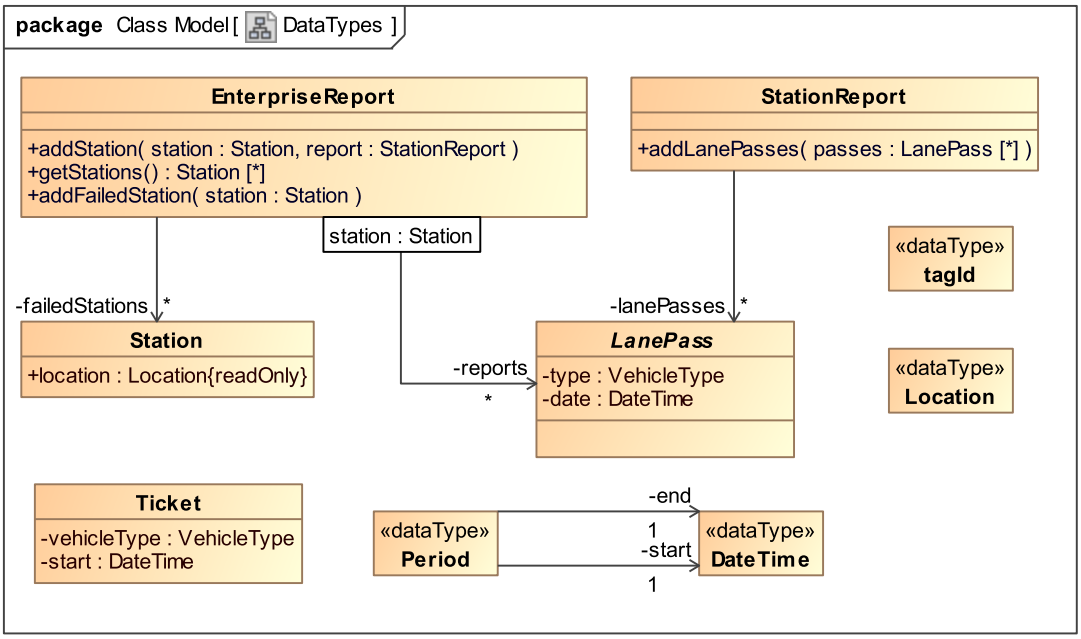

Figure 4.21: Class Model of Datatypes*[Andreas, Kasper]*

4.4.2 Object Constraint Language

[Andreas, Mikkel] Figure 4.22 shows class invariants and non-trivial operations for the CheckIn-Lane using OCL constraints.

checkIn: If a check-in with a tag is performed, when the tag is already checked in, the check in fails and no check-in gets associated with the Toll Tag. If the check-in is performed and the tag is not already checked in, the check-in succeeds and gets assosiated with the Toll Tag.

checkOut: If a check-out with a tag is performed, but the tag was not checked in, the check-out fails and no changes are made. If a check-in is performed with a tag and the tag was checked in, the check-out succeeds and a trip is stored by TollTag.

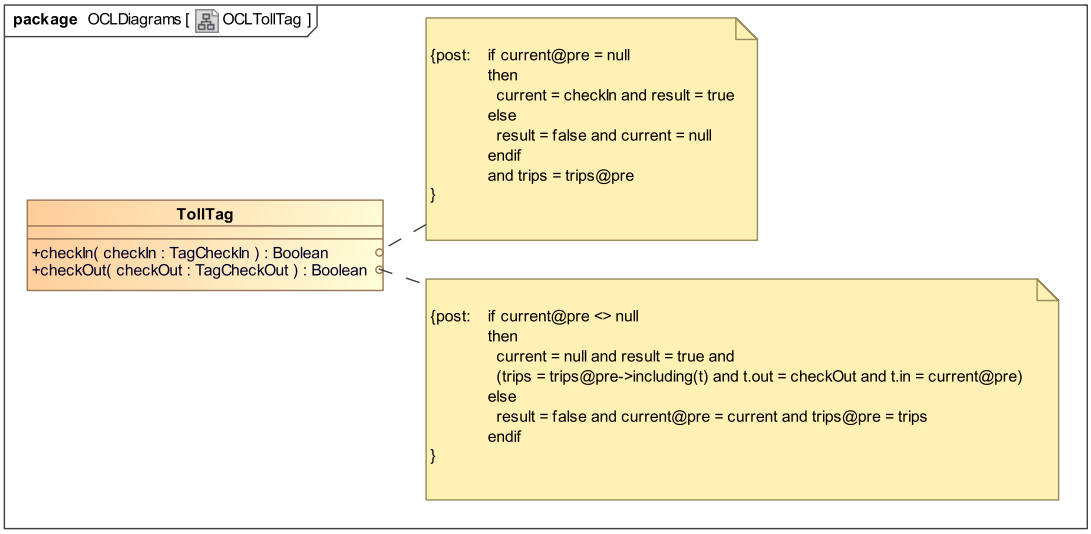

Figure 4.22: OCL for TollTag*[Kasper, Mikkel]*

Figure 4.23 shows class invariants and non-trivial operations for the CheckInLane using OCL constraints.

checkIn and checkOut When checkIn or checkOut with a tag gets called on the EnterpriseServer, it finds the tag from it's stored tags and performs a checkIn or a checkOut respectively.

compileReport: The enterpriseReport must contain some result from all stations. It must also be the case that if a report exists for a station, it must be identical to the result of generating a report from that station. If no report from the station exists, it must be contained in the list of failed stations.

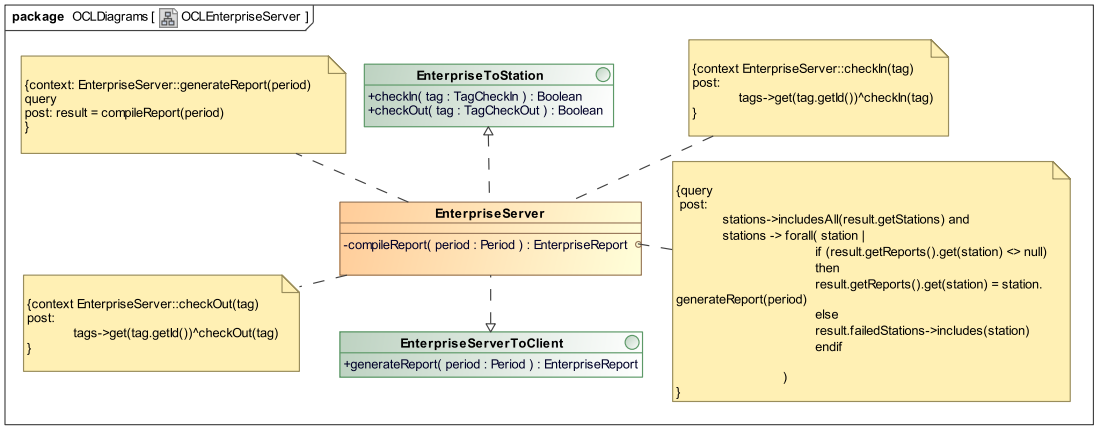

Figure 4.23: OCL for EnterpriseServer*[Andreas, Martin]*

Figure 4.24 shows class invariants and non-trivial operations for the CheckInLane using OCL constraints.

checkIn and checkOut: When a checkOut or a checkIn with a tag is performed on the Station, it sends the checkOut or checkIn to EnterpriseServer.

compileReport: When a station compiles a report it must be the case that the result of getting a report from all lanes must be contained in the station result. It must also be the case that all passes in the station report must exist in some lane.

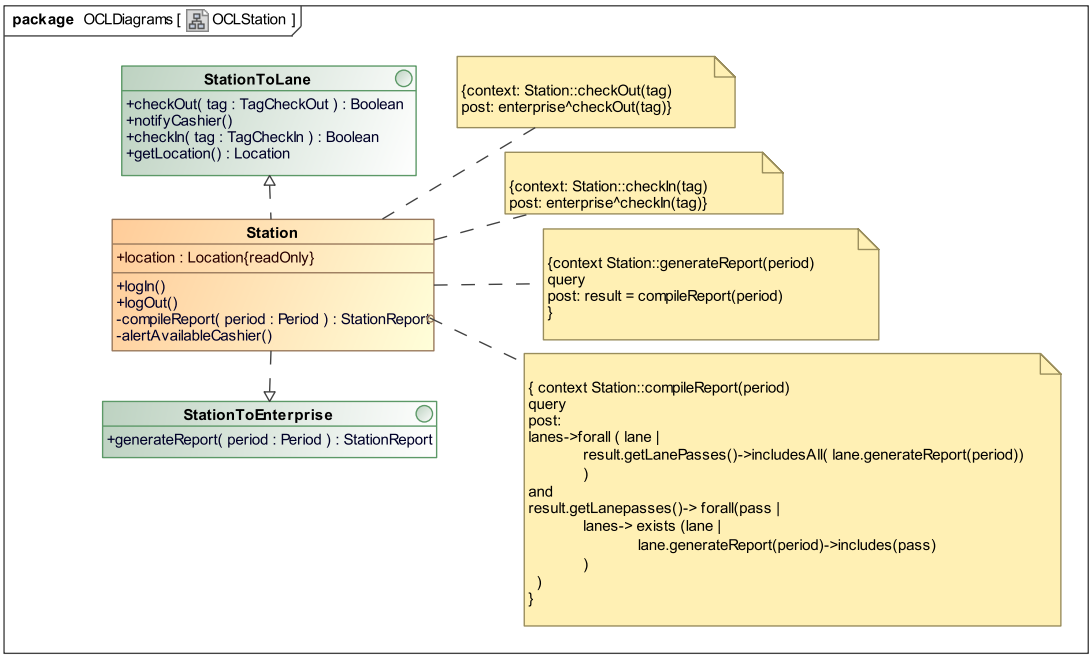

Figure 4.24: OCL for Station*[Johan, Mikkel]*

Figure 4.25 shows class invariants and non-trivial operations for the CheckInLane using OCL constraints.

logIn and logOut: To be able to log in on the TollLane, the user should not already be logged in, the opposite goes for users who want to log out.

generateReport: When a TollLane is queried to generate a report, all Lane Passes that are inside the given time interval are included in the result, and all Lane Passes in the result are actual Lane Passes known by the Toll Lane.

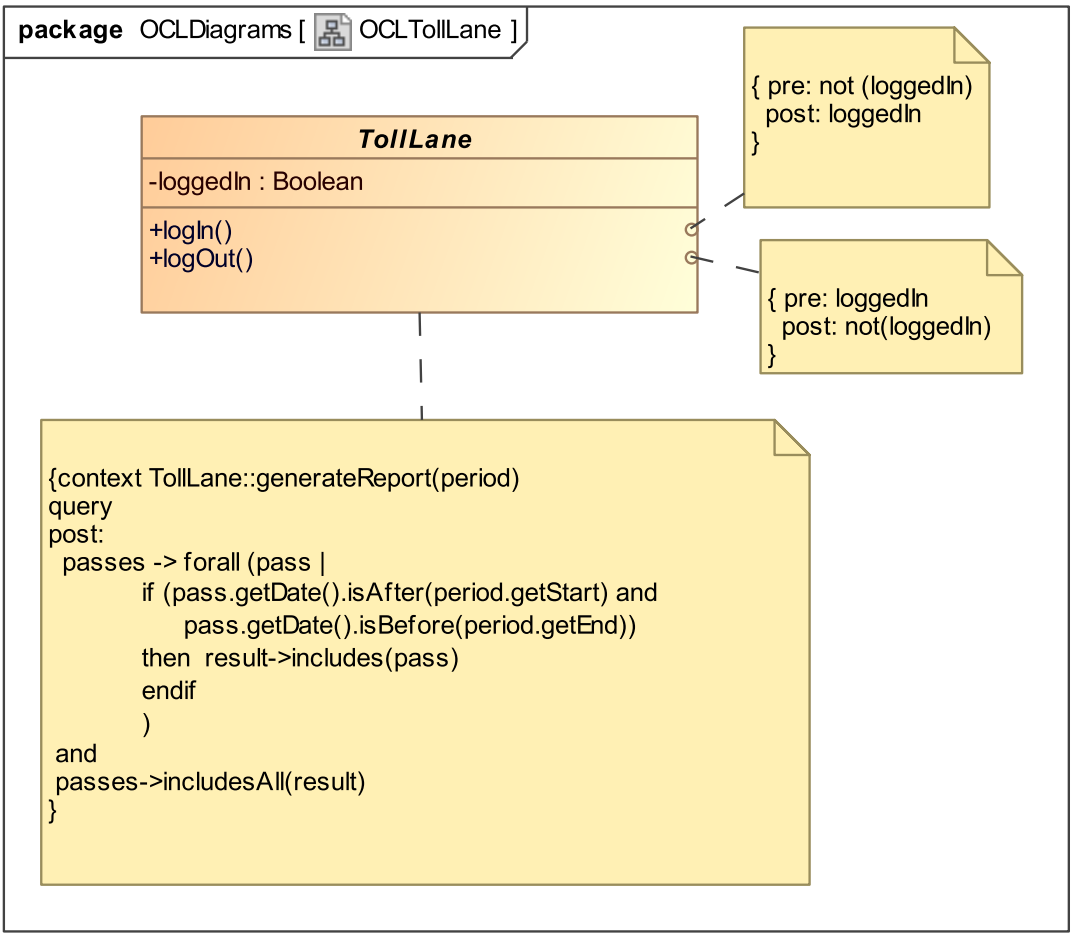

Figure 4.25: OCL for TollLane*[Andreas, Kasper]*

Figure 4.26 shows class invariants and non-trivial operations for the CheckInLane using OCL constraints.

payWithCC: When a driver pays with credit card, the credit card reader handles the payment, and if this is succesful, a new LanePass is instantiated and stored by the Lane.

payWithCash: When a driver pays with cash and has enough money and a new LanePass is instantiated and stored by the Lane.

selectType: The selected type of vehicle that the CheckInLane stores, is selected on the touch screen that is assosiated with the lane.

tagArrives When a tag arrives, the post condition states that if a lanepass was added all previous lane passes must still be present. It must always be the case that a request to check in a tag has been sent to the station.

manualCheckIn When a manual check-in is performed, the size of the set of Lane Passes stored by the CheckInLane is incremented by one and includes all the previous elements and a new Lane Pass.

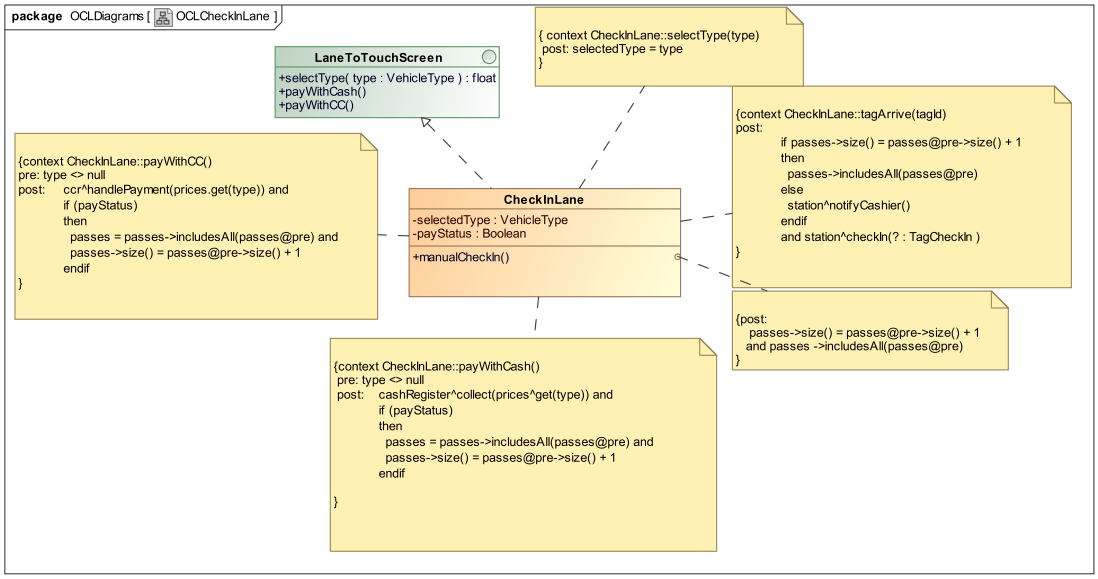

Figure 4.26: OCL for CheckInLane *[Johan, Martin]*

Figure 4.27 shows class invariants and non-trivial operations for the CheckInLane using OCL constraints.

ticketRead: When a ticket check-out is performed, the CheckOutLane verifies that ticket is valid. If that is the case, the barrier is opened and a new Lane Pass is added to the Lanes set of Lane Passes. If the ticket is not valid, a cashier is notified.

tagArrives: When a check-out with a tag is performed, the size of the stored Lane Passes is checked to have incremented by one in size. If this check fails, a cashier must have been notified otherwise it must be the case that all previous passes are still stored along with the new one.

manualCheckOut When a manual check-out is performed, the size of the set of Lane Passes stored by the CheckOutLane is incremented by one and includes all the previous elements and a new Lane Pass.

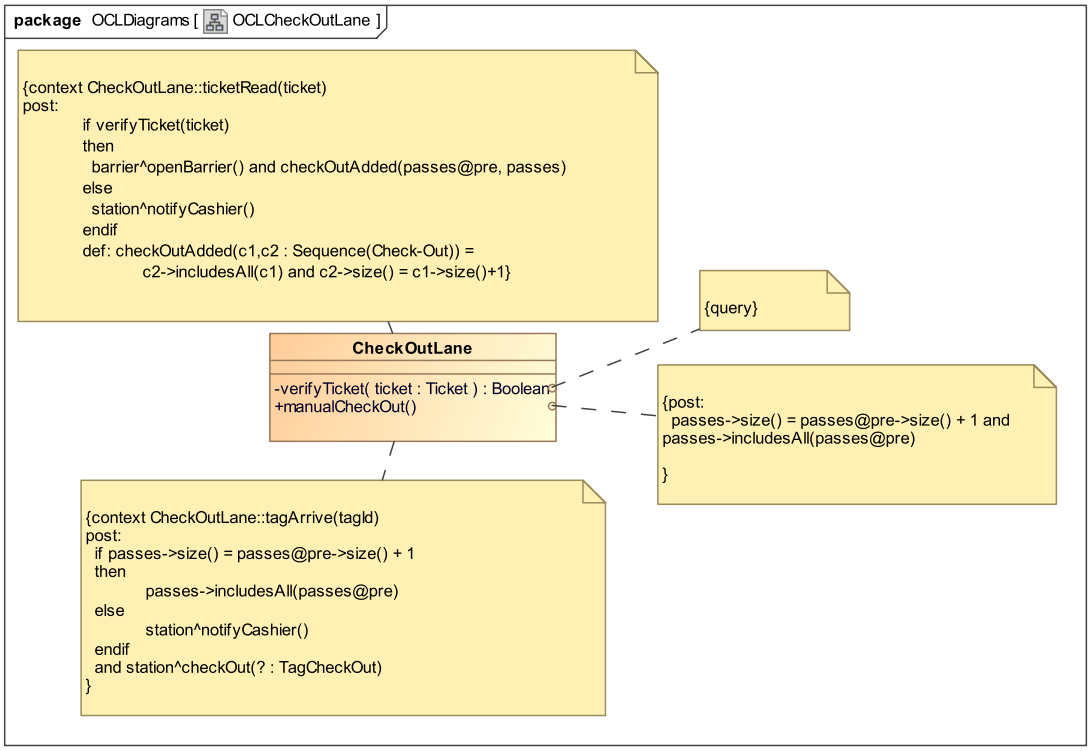

Figure 4.27: OCL for CheckOutLane *[Andreas, Mikkel]*

Figure 4.28 shows post-conditions for non-trivial operations for the EnterpriseReport using OCL constraints.

addStation: When a station and a report is added, it must be the case that the station and report is linked in the map of the enterprisereport.

getStations: The result of this operation must be the union of all failed stations and report stations added.

addFailedStation: The result of this operation must be that the station is added to the list of failed stations without removing any of the existing stations.

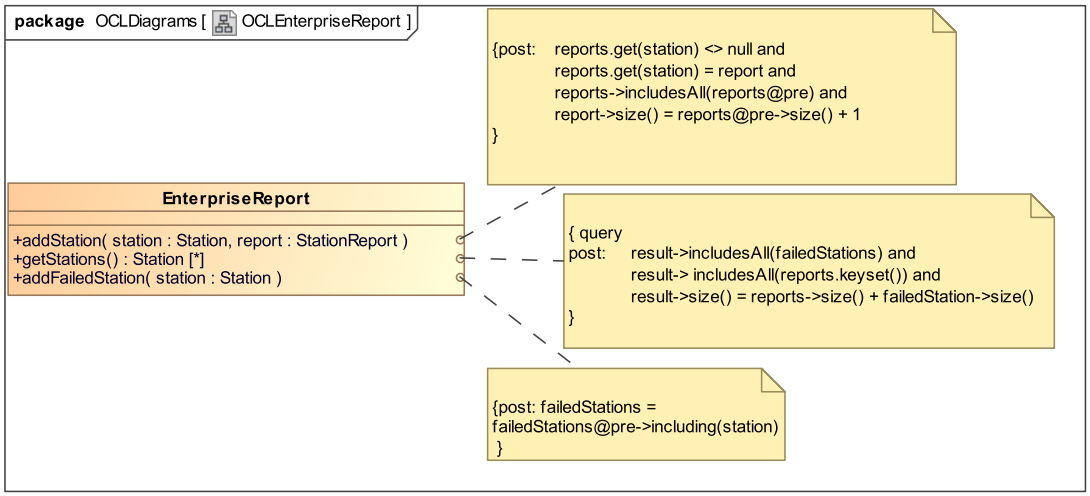

Figure 4.28: OCL for EnterpriseReport*[Kasper, Martin]*

Figure 4.29 shows post-conditions for non-trivial operations for the StationReport using OCL constraints.

addLanePasses: It must be the case that all lane passes in the argument is added to the list of lanepasses without removing any of the existing lane passes.

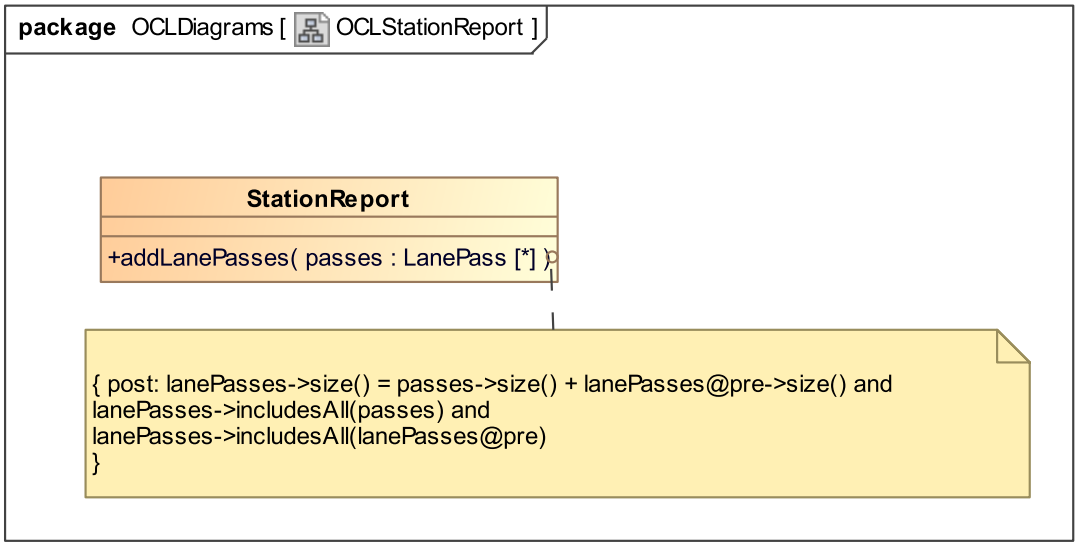

Figure 4.29: OCL for StationReport*[Mikkel, Martin]*

Figure 4.30 shows the OCL constraints for the Period data type.

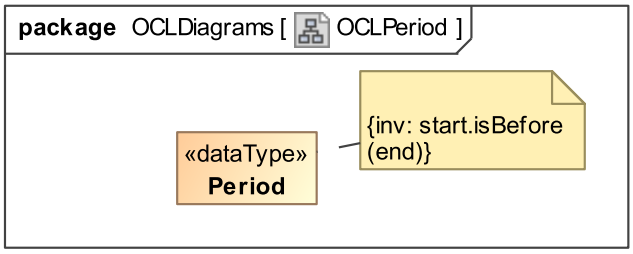

Figure 4.30: OCL for Period *[Kasper, Johan]*

4.4.3 Class Descriptions

TollTag			
This object represents a Toll Tag purchased by a Driver, and can retrieve all relevant data needed in order to charge the Driver every month. The TollTag will likewise hold information on whether the TollTag is currently logged in, i.e. the Driver is driving on the Motorway			
Interaction	**Class**	**Description**	
trips	Trip	Stores every Trip that the given TollTag has performed from the time it was purchased by the Driver. Will create a new Trip whenever a corresponding TagCheckOut occurs when it is checked-in.	
owner	TollTagOwner	Used to provide any information related to the Driver that owns the TollTag	
current	TagCheckIn	Is set whenever the Driver is currently checked-in with his TollTag	
type	VehicleType	The type of vehicle assigned for the TollTag	
Operation	**In**	**Out**	**Description**
checkIn	TagCheckIn	Boolean	Attempts to check-in the TollTag, by setting the `current` to become the given input. In case something goes wrong during this process a **false** will be returned disabling the Driver from entering the Motorway until the problem is resolved. If a **true** is returned the check-in is registered.
checkOut	TagCheckOut	Boolean	Returns whether the check-out of the TollTag was successful, i.e. the TollTag needs to be checked-in. In case something goes wrong, **false** is return, and the Driver is not granted to exit the motorway, until the problem is resolved.

Table 4.1: Description of the class TollTag *[Mikkel, Martin]*

Trip			
Is responsible for holding the information related to one Trip that occurred with the use of any TollTag.			
Interaction	**Class**	**Description**	
start	TagCheckIn	The TagCheckIn that started the Trip, will need to be initialised when the Trip is created	
end	TagCheckOut	A TagCheckOut that represents the location in which the Trip ended. The Driver will be charged according to the distance between the TagCheckIn and TagCheckOut of the given Trip.	
Operation	**In**	**Out**	**Description**

Table 4.2: Description of the class Trip *[Johan, Mikkel]*

Enterprise Server			
The server highest in the hierarchy, enabling it to communicate with all `Stations` contained in the Toll System. It will contain all Toll Tags that has been registered in the system, and is responsible for handling all LanePass which a Toll Tag is involved in.			
Interaction	**Class**	**Description**	
tags	TollTags	A map containing the TollTag corresponding to every tagId registered in the system. Enable the system to get a fast response time whenever a TollTag tries to either check-in or check-out.	
stations	EnterpriseToStation	The interface responsible for all communication from the Enterprise to any Station	
Operation	**In**	**Out**	**Description**
generateReport	Period	Enterprise-Project	Is realised from the implemented interface, and will extract all information needed from the stations in the Toll System
checkIn	TagCheckIn	Boolean	Attempts to to create a newly started Trip with the given TagCheckIn. A trip will be created in case it violates no rules, and **true** will be returned when it is successfully created
checkOut	TagCheckOut	Boolean	Will try to end a Trip for a given TagCheckOut. If the trip is successfully stopped, **true** will be returned. Otherwise **false** if any problems occur during check-out.
compileReport	Period	Enterprise-Project	Will compile a final report composed of the results for all the Stations in the Toll System. Any stations that communication failed with will be reported

Table 4.3: Description of the class Enterprise Server *[Mikkel, Martin]*

Station			
The station is acting as an layer between the Toll Lanes and the rest of the Toll System, and is using the communication originating from the Toll Lane either to issue action locally or transfer the request towards the Enterprise Server.			
Interaction	**Class**	**Description**	
lanes	TollLane	All TollLanes situated at the Station	
location	Location	The exact location of the Station	
enterprise	Enterprise Server	The Enterprise Server in the Toll System	
Operation	**In**	**Out**	**Description**
logIn			Will enable a Station Manager to log-in to the Station
logOut			Logs-out the Station Manager from the Station
compileReport	Period	Station-Report	Will compile a StationReport composed of the information retrieved from the Toll Lanes placed on the Station
alertAvailableCashier			Will alert any available cashier at the Station if a problem occurs in a Toll Lane
checkIn	TagCheckIn	Boolean	Will try to make a check-in with the use of a toll tag
checkOut	TagCheckOut	Boolean	Attempts to check-out the given tag
notifyCashier			Will attempt to notify a cashier at the station as a problem has occurred on the station
getLocation generateReport	Period	StationReport	Returns the location of the station Will generate a report containing all passes that has occurred throughout the given period

Table 4.4: Description of class Station *[Andreas, Johan]*

TollLane			
Responsible for allowing people to pass a Toll Lane, with the use of a TollTag, credit-card or cash depending on the setting of the the TollLane. This is possible through the use of several peripherals assigned to each Toll Lane.			
Interaction	**Class**	**Description**	
barrier	Barrier	The barrier assigned to the Toll Lane	
prices	float	A map composed of the prices assigned to each VehicleType registered in the system	
passes	LanePass	All LanePasses that has occurred on the TollLane, i.e. either check-ins or check-outs.	
station	Station	The station which the TollLane is assigned to.	
Operation	**In**	**Out**	**Description**
logIn			Will enable a Cashier to log-in to the TollLane, to enable cash payment or resolve issues
logOut			Log-outs the Cashier from the TollLane, disabling cash payment
generateReport	Period	LanePass[*]	Returns all LanePass that has occurred on the TollLane within the given period
tagArrive	tagId		Is called whenever a toll tag is being registered by the antenna

Table 4.5: Description of class TollLane *[Andreas, Mikkel]*

TollTagOwner			
Is responsible for the containment of all information related to the owner of a toll tag			
Interaction	**Class**	**Description**	
bankAccount	BankAccount	The bank account assigned to the toll tag owner, used for monthly payment	
Operation	**In**	**Out**	**Description**

Table 4.6: Description of class TollTagOwner *[Andreas, Martin]*

VehicleType			
All types of vehicles that may occur in the system, meaning they have a price.			
Interaction	**Class**	**Description**	
Operation	**In**	**Out**	**Description**

Table 4.7: Description of class VehicleType *[Mikkel, Martin]*

LanePass			
Is responsible for linking a date and time together with a VehicleType, registering that a vehicle has crossed a TollLane, either by checking-in or -out			
Interaction	**Class**	**Description**	
type	VehicleType	The type of vehicle that has performed the LanePass	
date	DateTime	The date and time in which the LanePass took place.	
Operation	**In**	**Out**	**Description**

Table 4.8: Description of class LanePass *[Kasper, Martin]*

CheckInLane extends *TollLane*			
The lanes responsible for checking-in the Driver so he can enter the motorway.			
Interaction	**Class**	**Description**	
selectedType	VehicleType	The type of vehicle selected whenever a check-in the result in a Ticket being printed is made at the Lane.	
ts	TouchScreen	Used to report how well the system has dealt with the check-in. This could be that it has successfully dealt with the check-in or an error occurred.	
ccr	CreditCardReader	Will transfer the payment to the credit card reader, the will handle the payment. The credit card reader will return **true** whenever a payment is successful.	
cashRegistrer	CashRegister	Is used to handle the payment, when cash has been selected. The cash register is responsible for collecting the correct amount.	
ticketPrinter	TicketPrinter	The printer which is responsible for printing a ticket for the Driver whenever he pays with credit card or cash.	
Operation	**In**	**Out**	**Description**
manualCheckIn			Will enable a cashier to make a manual check-in in case errors happen while checking-in.
selectType	VehicleType	float	Select the type of vehicle currently located at the CheckInLane
payWithCash			Set the check-in lane to handle the payment using the cash register in case a cashier is available
payWithCC			Set the check-in lane to handle the payment with the use of the credit card reader

Table 4.9: Description of class CheckInLane *[Andreas, Kasper]*

CheckOutLane extends *TollLane*			
The lanes responsible for checking-out the Driver from the motorway, whenever he decides to leave it.			
Interaction	**Class**	**Description**	
Operation	**In**	**Out**	**Description**
verifyTicket	Ticket	Boolean	Will check that a ticket is no older than 24 hours. In case of a valid ticket, **true** will be returned.
manualCheckOut			Enables the cashier to check-out a driver manual, in case errors happen during the normal check-out procedure.
ticketRead	Ticket		Is called when a Ticket is read at a normal lane, and will try to verify the ticket that is read

Table 4.10: Description of class CheckOutLane *[Johan, Mikkel]*

TagCheckIn			
Is tied to all completed and active Trips. A TollTag has a TagCheckIn, if it is currently used for an active Trip. The TagCheckIn holds information of which tag it is tied to and at which station it was checked in.			
Interaction	**Class**	**Description**	
id	tagId	The ID of the TollTag that the TagCheckIn is tied to.	
location	Station	The Station that the TagCheckIn came from.	
Operation	**In**	**Out**	**Description**

Table 4.11: Description of class TagCheckIn *[Andreas, Mikkel]*

TagCheckOut			
Is tied to all completed Trips. The TagCheckOut holds information of which tag it is tied to and at which station it was checked out.			
Interaction	**Class**	**Description**	
id	tagId	The ID of the TollTag that the TagCheckOut is tied to.	
location	Station	The Station that the TagCheckOut came from.	
Operation	**In**	**Out**	**Description**

Table 4.12: Description of class TagCheckOut *[Johan, Martin]*

EnterpriseReport			
A report containing all relevant information needed from all stations contained within the system			
Interaction	**Class**	**Description**	
reports	LanePass	A map to all stations that we received an answer from while requesting a StationReport. For each station we have all LanePass made on that station for the requested period	
failedStations	Station	All stations which communication could not be established	
Operation	**In**	**Out**	**Description**
addStation	Station,StationReport		Adds a station report to the given station
getStation		Station[*]	Returns all stations that is contained in the system with their corresponding report, or if the communication failed
addFailedStation	Station		Adds a station that communication couldn't be established with

Table 4.13: Description of class EnterpriseReport *[Andreas, Martin]*

StationReport			
A report containing all LanePass that was performed during any given Period			
Interaction	**Class**	**Description**	
lanePasses	LanePass	All the lane passes that has occurred on all lanes at the station	
Operation	**In**	**Out**	**Description**
addLanePasses	LanePass[*]		Adds all the given lane passes to the report

Table 4.14: Description of class StationReport *[Kasper, Johan]*

Location			
Describes the exact location of a station. Will be used to calculate the price for a trip using toll tag			
Interaction	**Class**	**Description**	
Operation	**In**	**Out**	**Description**

Table 4.15: Description of class Location *[Kasper, Mikkel]*

Ticket			
Used to hold the VehicleType selected at a normal check-in in order for the reports to extract what vehicles perform check-outs at each station. Also holds the time which the Ticket was purchased, used for validating if a ticket is valid			
Interaction	**Class**	**Description**	
vehicleType	VehicleType	The vehicle type selected during check-in	
start	DateTime	The time for which the ticket was purchased	
Operation	**In**	**Out**	**Description**

Table 4.16: Description of class Ticket *[Kasper, Johan]*

Period			
A period of time, defined by two DateTimes			
Interaction	**Class**	**Description**	
start	DateTime	The start time of the period	
end	DateTime	The end time of the period	
Operation	**In**	**Out**	**Description**

Table 4.17: Description of class Period *[Johan, Martin]*

4.5 Behaviour Design

This section describes the object life cycle state machines for non-trivial objects. For each state machine we provide an overview of any orthogonal states as well as an informal description of how it works.

4.5.1 EnterpriseServer

[Mikkel, Andreas] Figure 4.31 shows the life-cycle state machine of the Enterprise object. The state machine has two orthogonal sub-states.

Handle tags This sub-state handles check-ins and check-outs. The state machine is idle until a check-in or a check-out request arrives. When either of these transitions is fired, the state machine goes to a state where it checks the check-in or check-out and returns to the idle state by either retuning a success if the check-in/check-out succeed or a failure if the check-in/check-out failed.

Generate report This state is idle until a generate report event triggers the transition that starts the compilation of a report before going back to being idle.

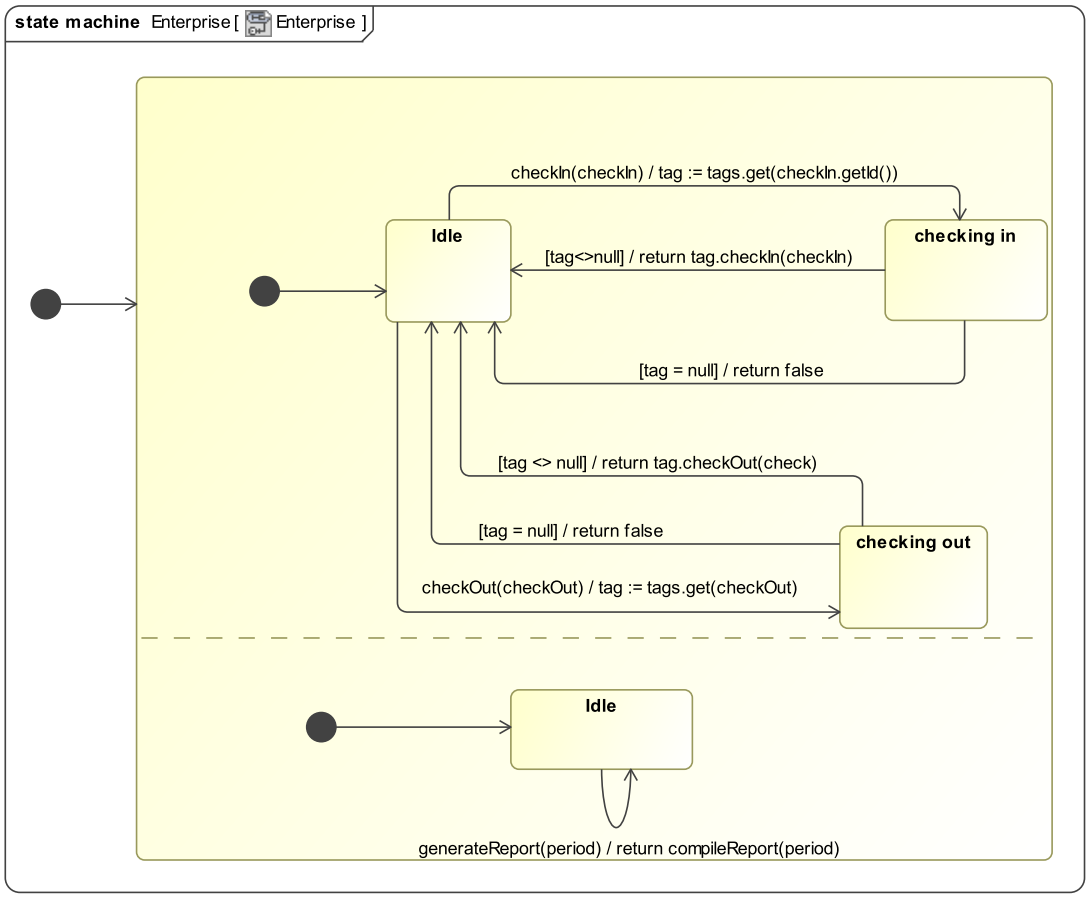

Figure 4.31: Object Life Cycle State Machine Enterprise. *[Andreas, Mikkel]*

4.5.2 Station

[Johan, Martin] Figure 4.32 shows the life-cycle state machine of the Station object. The state machine only has one state, from which there are four transitions to itself.

Check-in and check-out can be made and returns the value of the correponding call to the enterprise.

Generate report generates report event triggers the transition that starts the compilation of a report before going back to being idle.

Notify cashier sends a notification to a cashier when something is wrong.

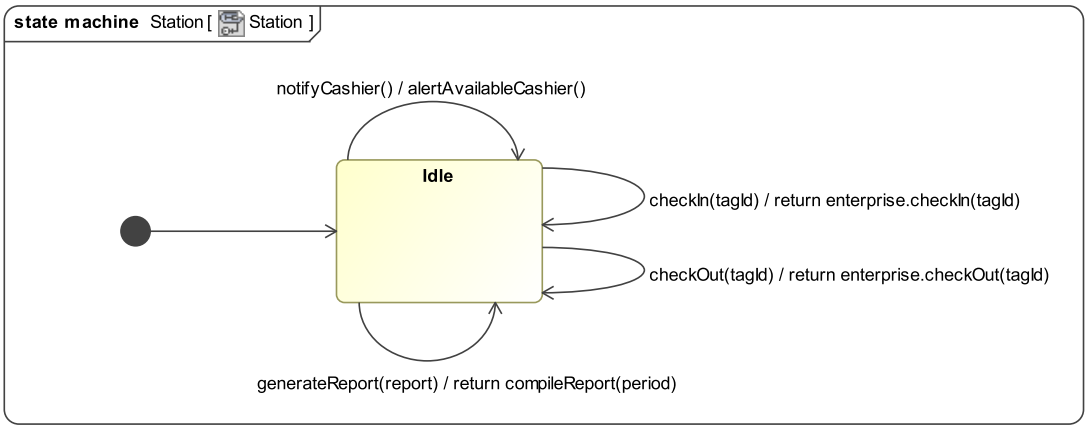

Figure 4.32: Object Life Cycle State Machine Station. *[Johan, Martin]*

4.5.3 CheckInLane

[Johan, Martin] Figure 4.33 shows the life-cycle state machine of the CheckInLane object. The state machine has two orthogonal sub-states.

Driver triggered :

Ticket check-in When the state machine is in the ready state, and a driver enters his vehicle type on the touch-screen, he is brought to a state where he is informed about the price and can choose to pay with either cash or credit. After paying, the state machine goes to a state where it checks the payment. Failing from this state brings the state machine back to the previous state where the driver can try to pay once again.
If a cashier is not logged in to the lane and a credit card payment fails, a cashier is notified, and the state machine goes to a state where it waits for the cashier to log in, before proceeding with the payment.
Once the payment has succeeded, the state machine goes to a state where the payment is ok. After this, the printer will try to print a ticket, and if this succeeds, a ticket is issued and the barrier opens. If it fails, a cashier is summoned and performs a manual check-in.

Tag check-in When a tag arrives, the state machine goes to a state where it verifies the tag. If this succeeds, the barrier opens and the state machine return to the ready state. If it fails, a cashier is summoned and performs a manual check-in.

Cashier triggered :

Login A cashier can log in, if he is not already logged in.

Logout A cashier can log out, if he is not already logged out.

Generate report returns the lane passes of the lane for the given time period and goes back to the idle state.

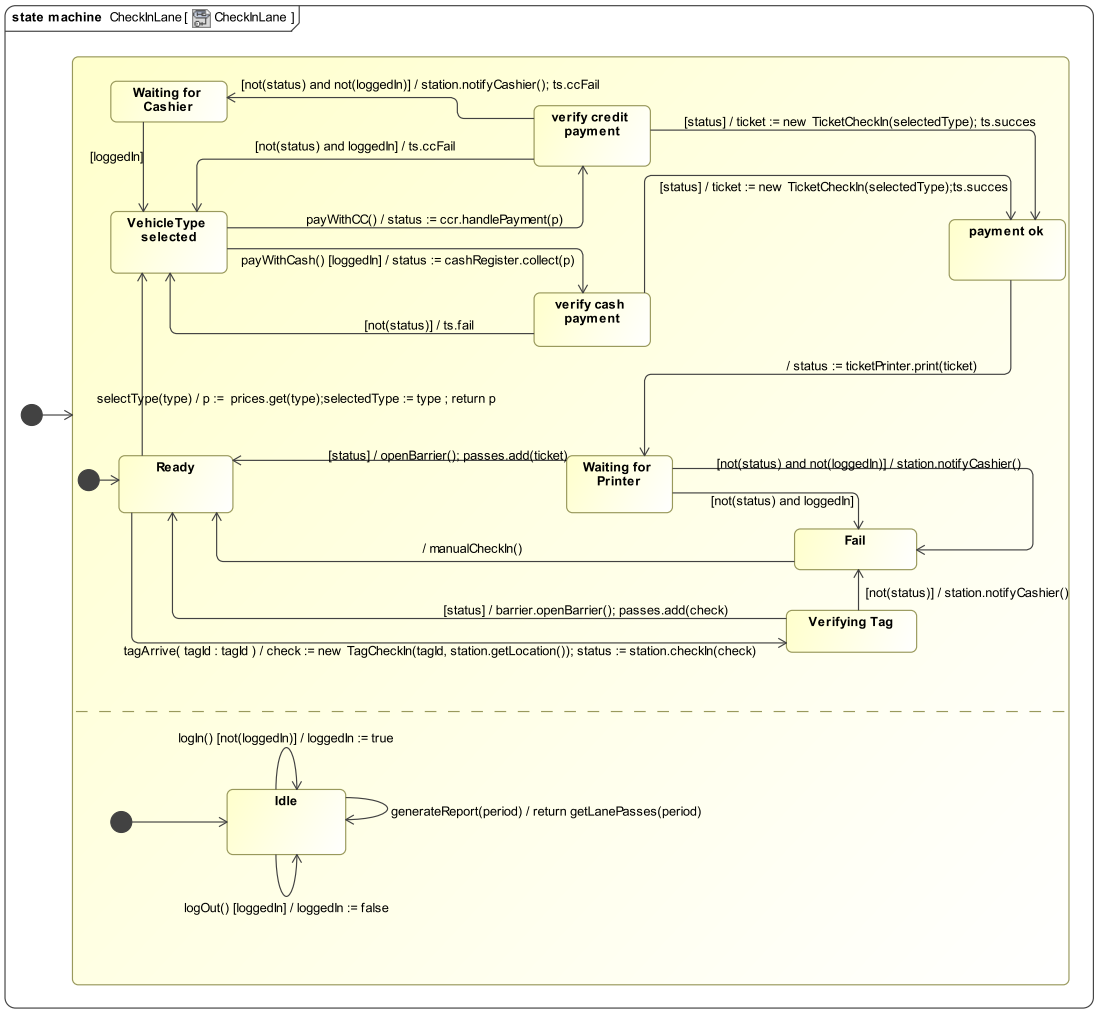

Figure 4.33: Object Life Cycle State Machine CheckInLane. *[Johan, Martin]*

4.5.4 CheckOutLane

[Kasper, Andreas] Figure 4.34 shows the life-cycle state machine of the CheckInLane object. The state machine has two orthogonal sub-states.

Tag arrive when a tag or ticket arrives, the state machine goes to verification state. If the verification is succesful, the barrier is opened and the state machine returns to initial state. If the verification is unsuccesful, a notifyCashier transition brings the state machine to a state where it waits for a cashier. After the cashier has arrived, it performs a manual check-out, thereby bringing the state machine back to the initial state.

Generate report generates report event triggers the transition that returns the lane passes of the check-out lane before going back to being idle.

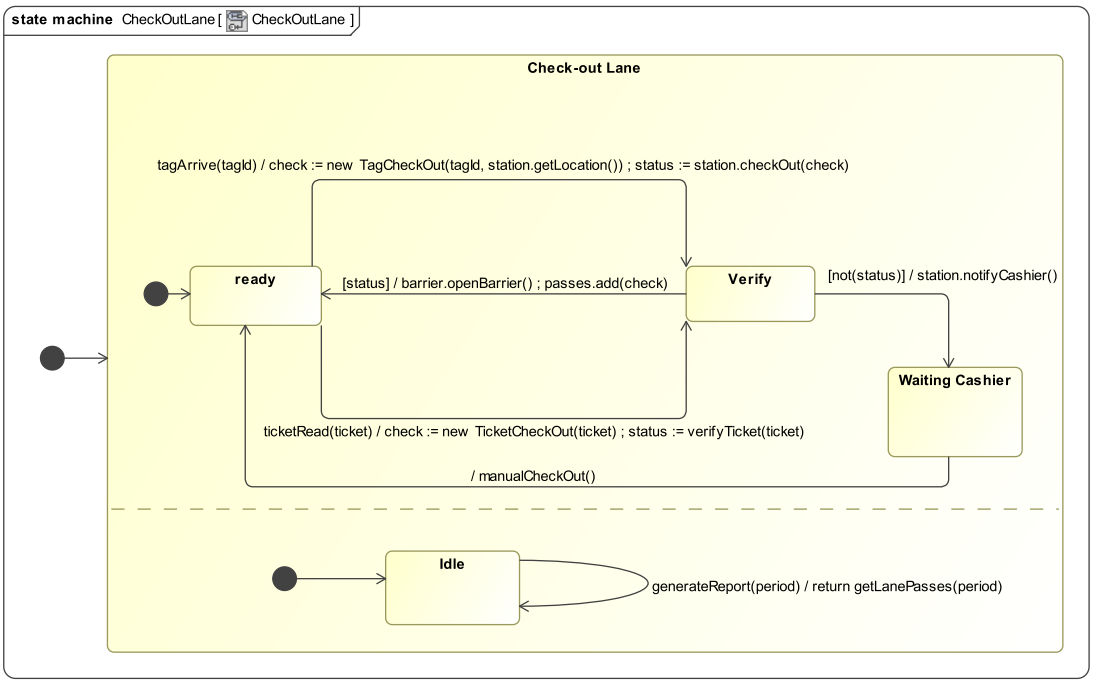

Figure 4.34: Object Life Cycle State Machine CheckOutLane. *[Kasper, Andreas]*

4.5.5 TollTag

[Mikkel] Figure 4.35 shows the life-cycle state machine of the TollTag object.

Check-in From the idle state, a check-in transition brings the state machine to a state where it checks in the tag. Depending on the result of the check-in, the state machines returns to the idle state with a succesful or unsuccesful check-in.

Check-out Similarly to the check-in, a check-out transition brings the state machine to a state where it checks out the tag. Depending on the result of the check-out, the state machines returns to the idle state with a succesful or unsuccesful check-out.

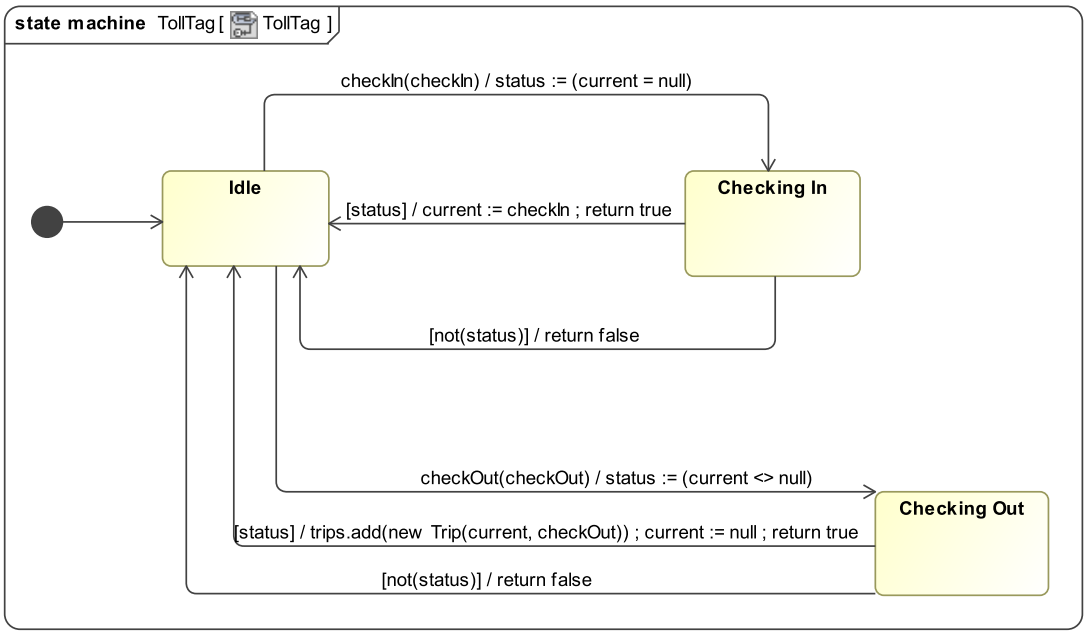

Figure 4.35: Object Life Cycle State Machine TollTag. *[Mikkel]*

5 Validation

In this chapter we use sequence diagrams to validate that the use cases from Section 2.3.2 can be done with the modelled system. Some diagrams contains just one scenario, while others show several scenarios.

5.1 Use Case Realisation

5.1.1 Tag Check-in

[Martin, Johan] The Tag check-in use case main scenario is realised in Figure 5.1. Figure 5.2 shows two alternative scenarios. Optionally the tag id is recognised by the enterprise server, but is not checked out from the last trip, or the tag id is not recognised at all. Either way, it results in a failure to check in.

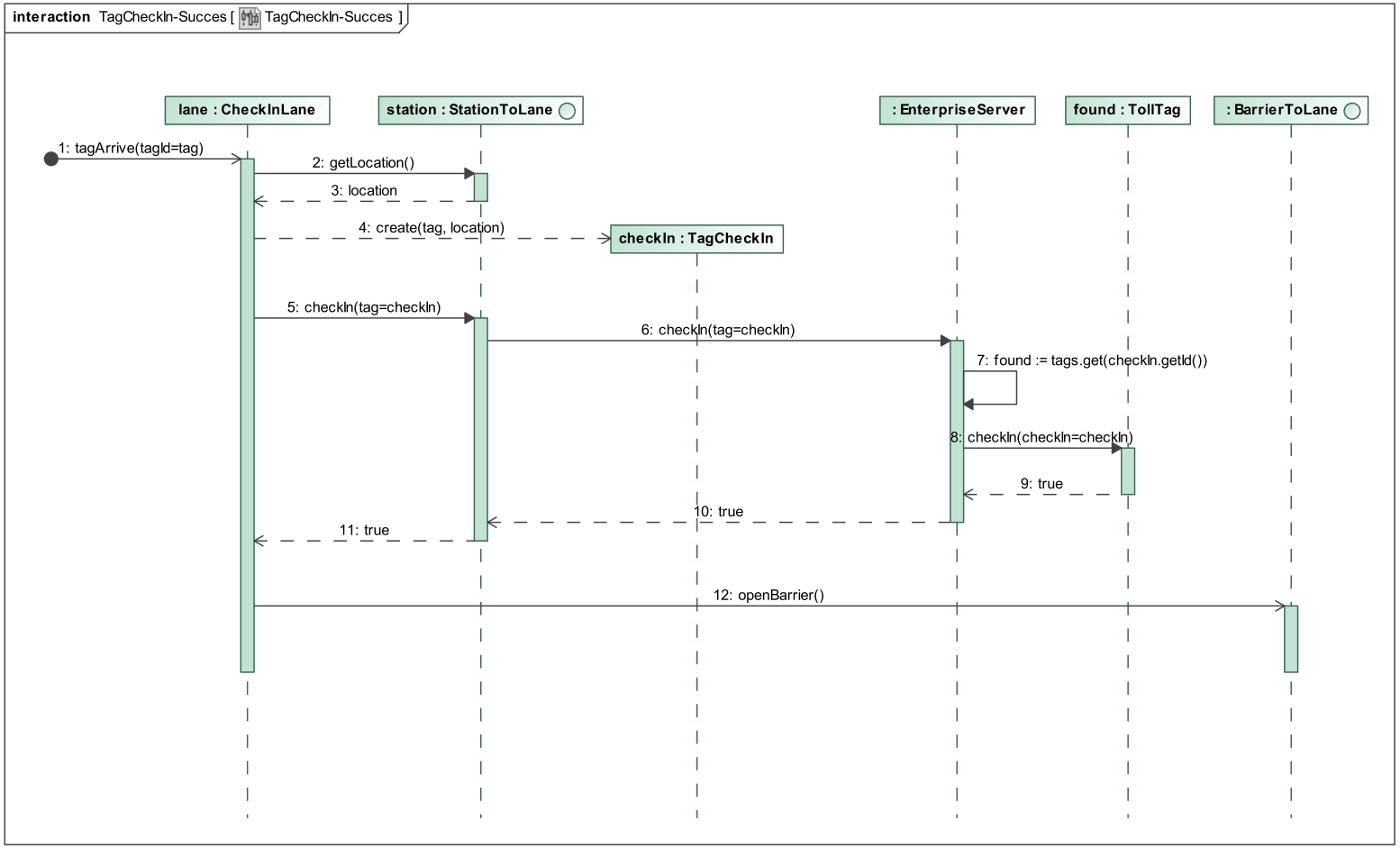

Figure 5.1: Success scenario for the Tag Check-in use case. *[Martin, Johan]*

Mikkel Riber Bojsen
Andreas Foldager
Martin Kasban Tange
Johan van Beusekom
Kasper Aaquist Johansen

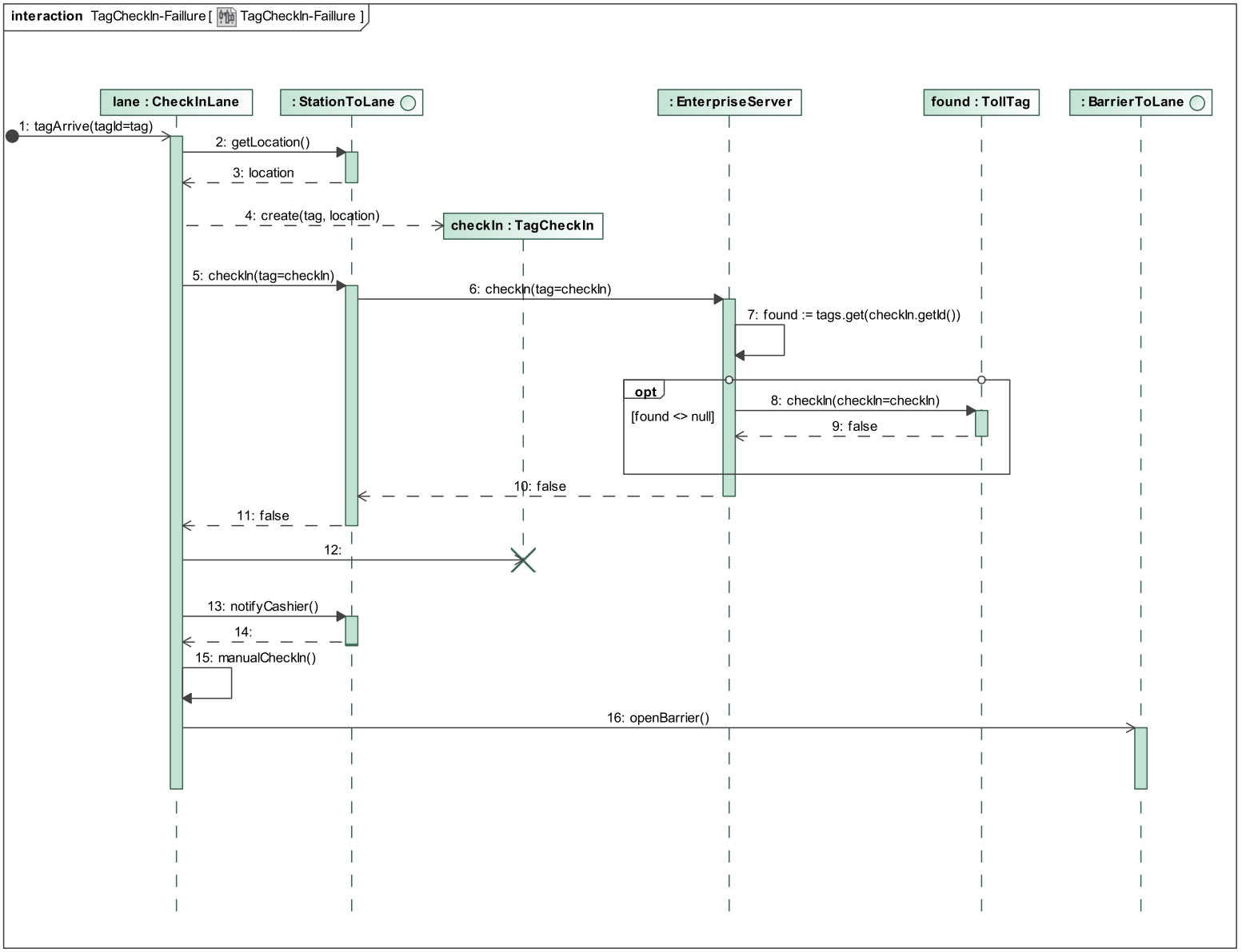

Figure 5.2: Alternative scenario for the Tag Check-in use case. *[Martin, Johan]*

Mikkel Riber Bojsen
Andreas Foldager
Martin Kasban Tange
Johan van Beusekom
Kasper Aaquist Johansen

5.1.2 Tag Check-out

[Mikkel, Andreas] The Tag Check-out use case main scenario is realised in the sequence diagram in Figure 5.3. The two alternative scenarios are shown in Figure 5.4. One scenario where the check-out fails at the enterprise level because the tag id is not recognized and one scenario where it fails at the toll tag level because the tag is not already checked in somewhere.

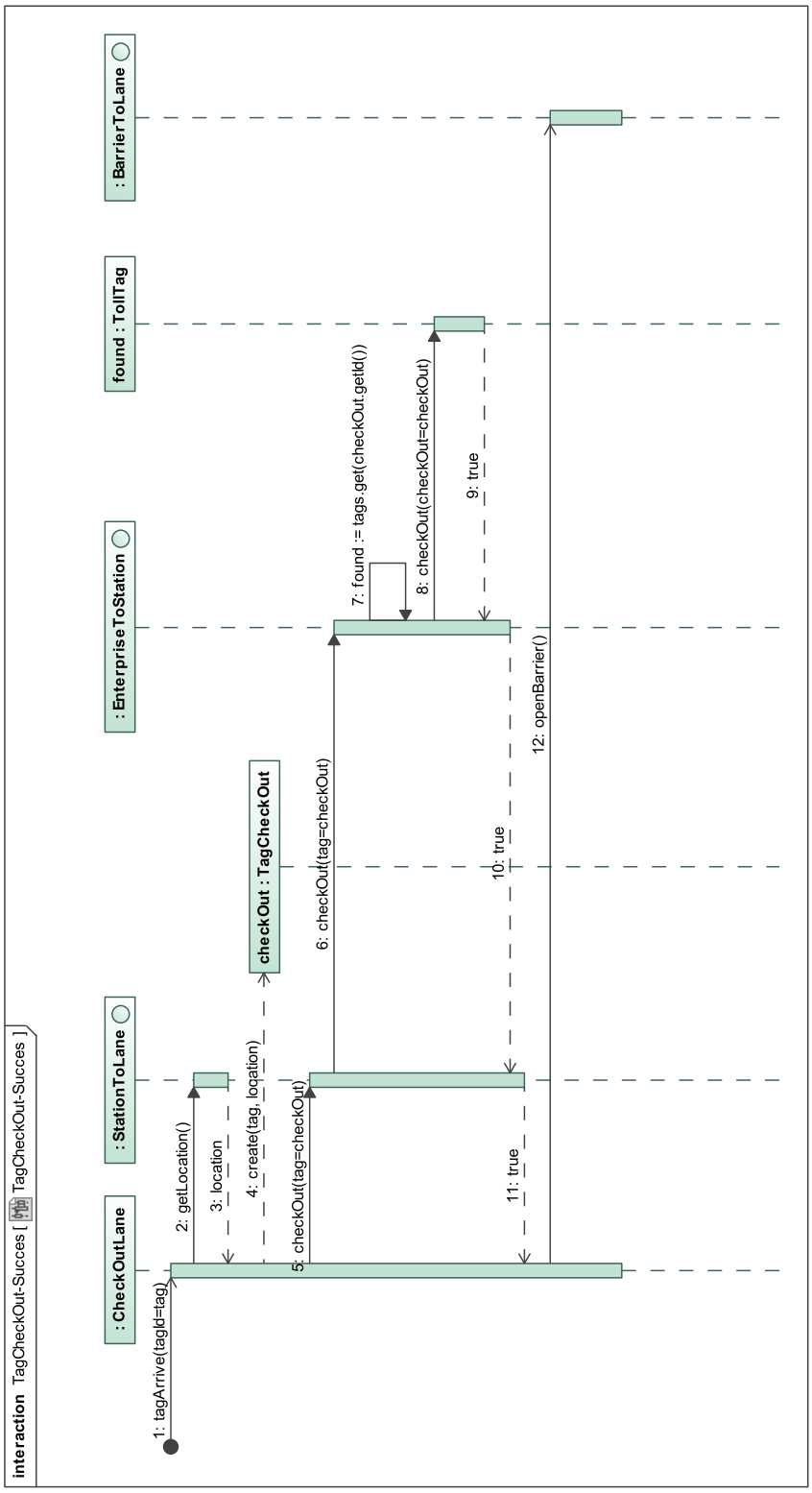

Figure 5.3: Success scenario for the Tag Check-out use case. *[Mikkel, Andreas]*

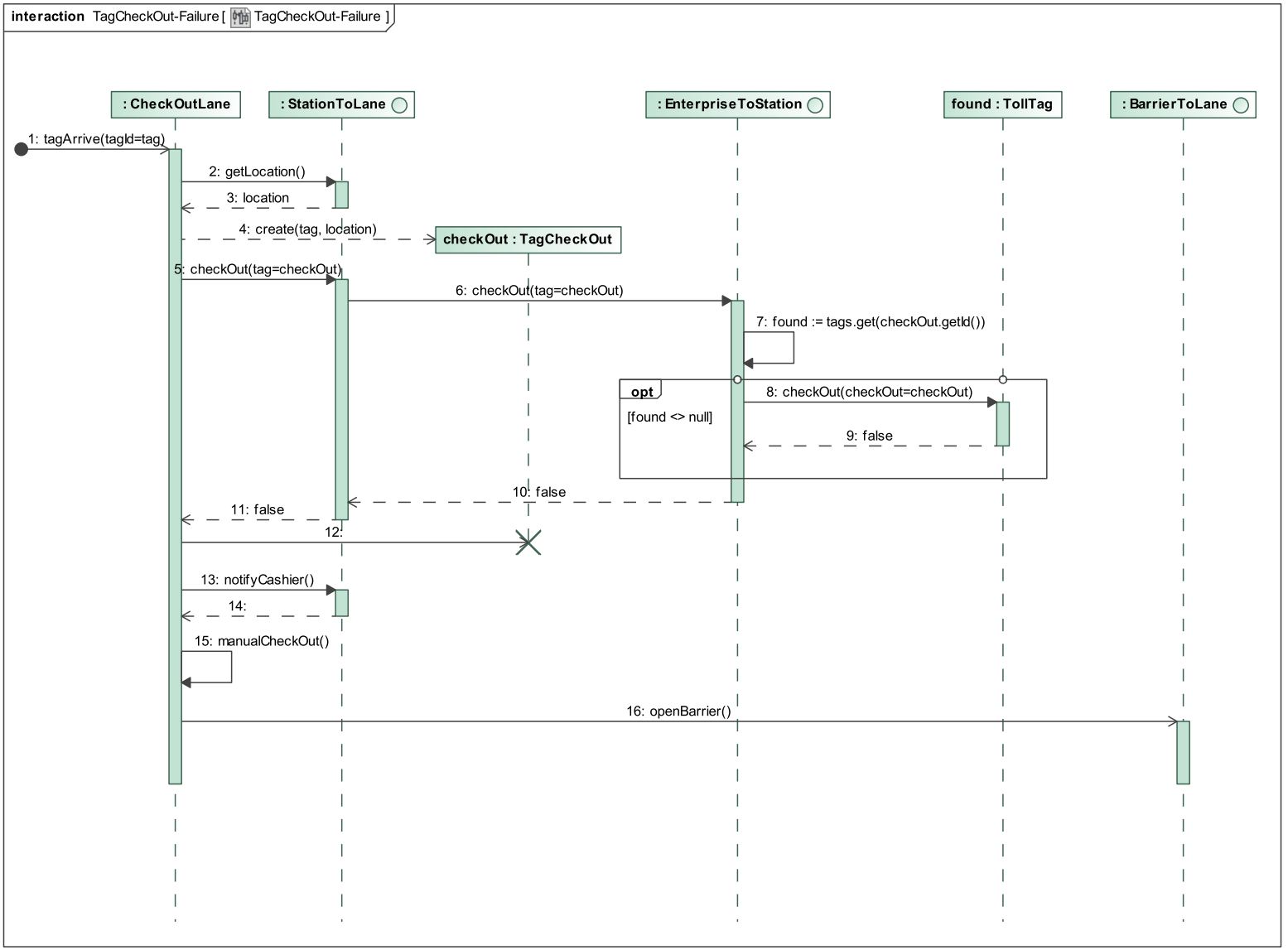

Figure 5.4: Alternative scenarios for the Tag Check-out use case. *[Mikkel, Andreas]*

5.1.3 Check-in with Ticket

[Kasper, Martin] The Ticket Check-in use case main scenario as well as one alternative scenario is realised in the sequence diagram in Figure 5.5. The two scenarios is one where the driver wants to pay with credit card and one where he wants to pay with cash. The alternative scenario where a driver wants to pay with credit card but the payment fails is shown in Figure 5.6

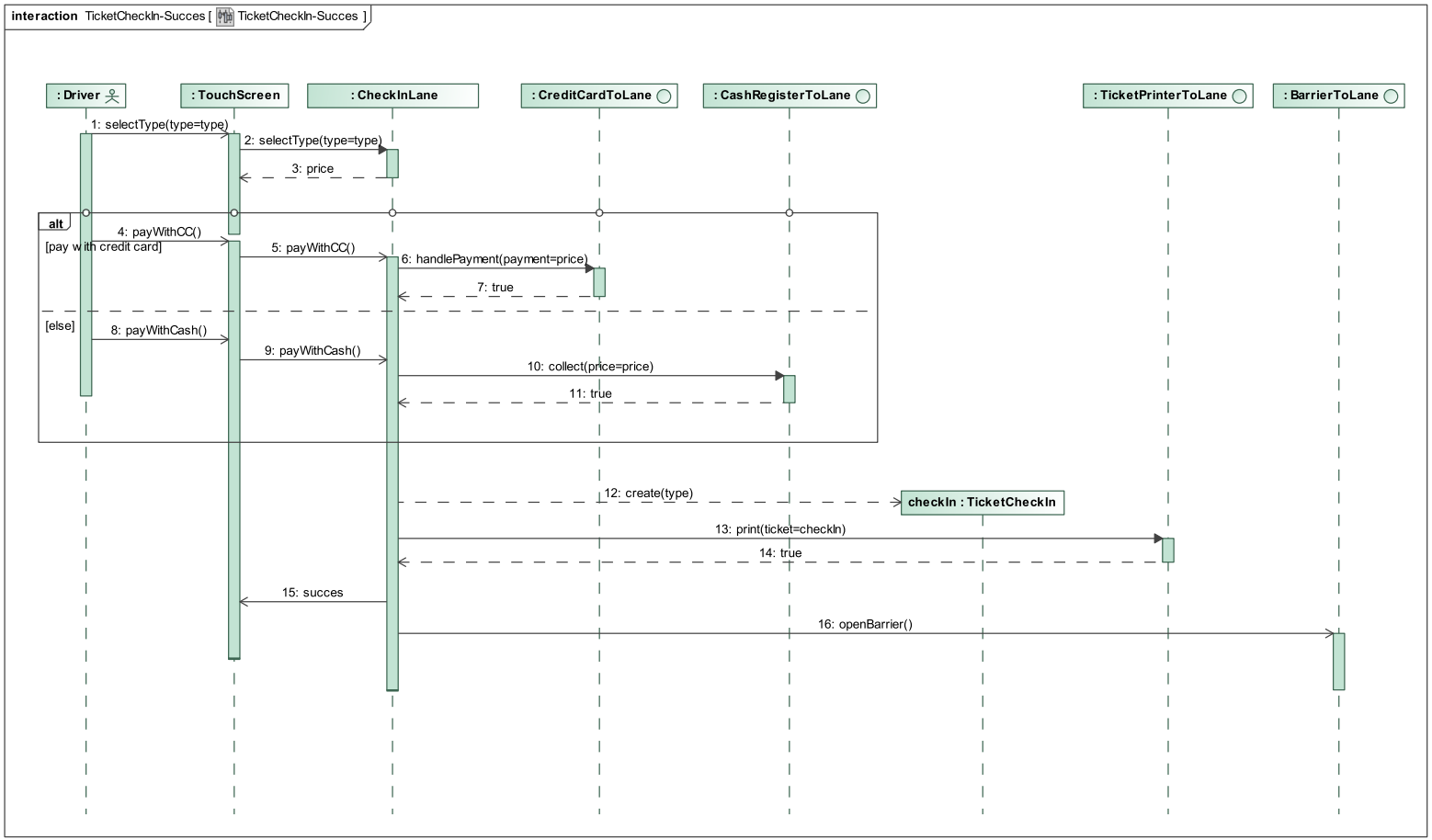

Figure 5.5: Success scenario for the Ticket Check-in use case. *[Kasper, Martin]*

Mikkel Riber Bojsen
Andreas Foldager
Martin Kasban Tange
Johan van Beusekom
Kasper Aaquist Johansen

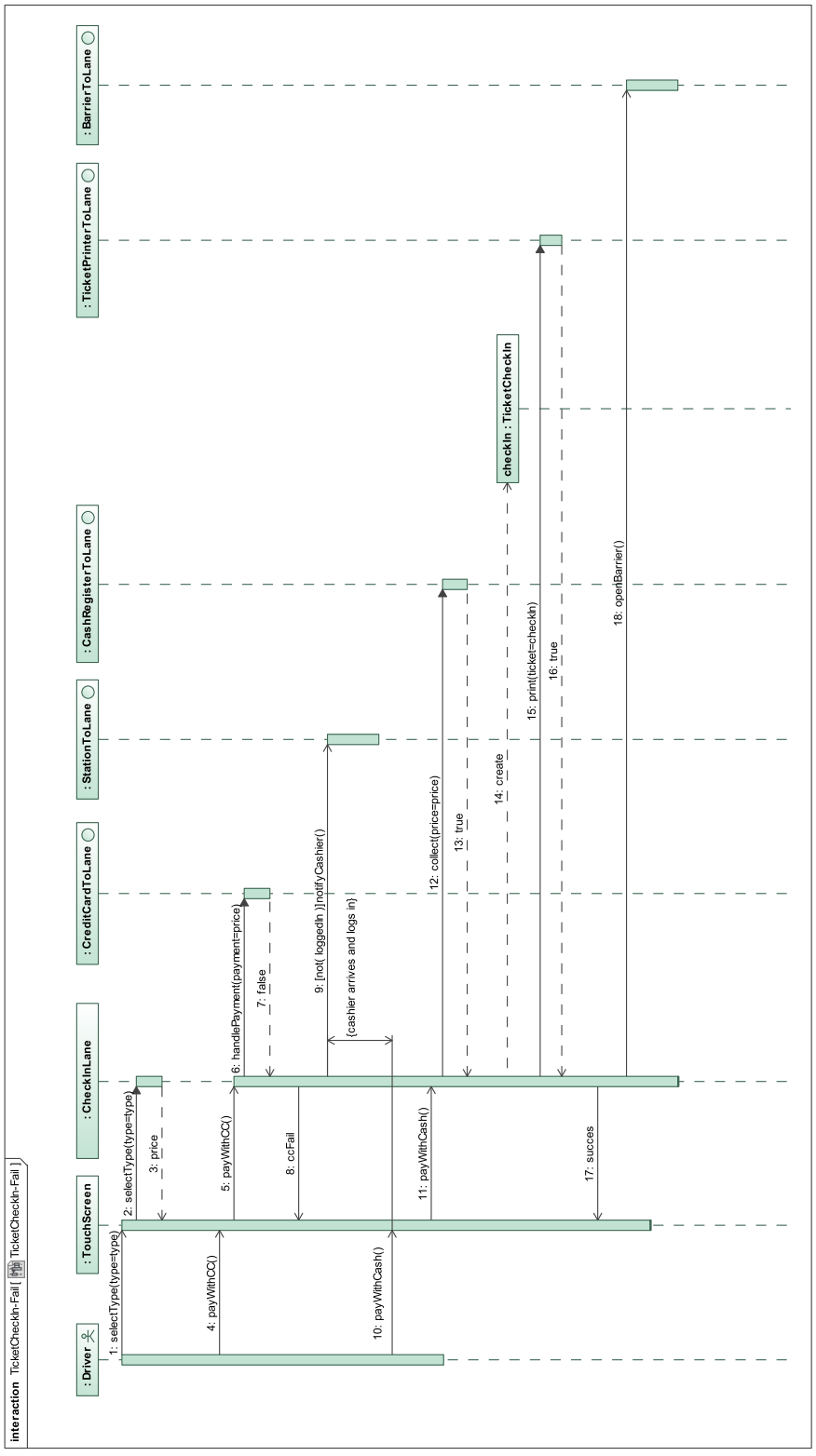

Figure 5.6: Alternative scenario for the Ticket Check-in use case. *[Kasper, Martin]*

5.1.4 Check-out with Ticket

[Johan, Andreas] The Ticket Check-out use case main scenario is shown in Figure 5.7. An alternative scenario where the ticket is invalid is shown in Figure 5.8

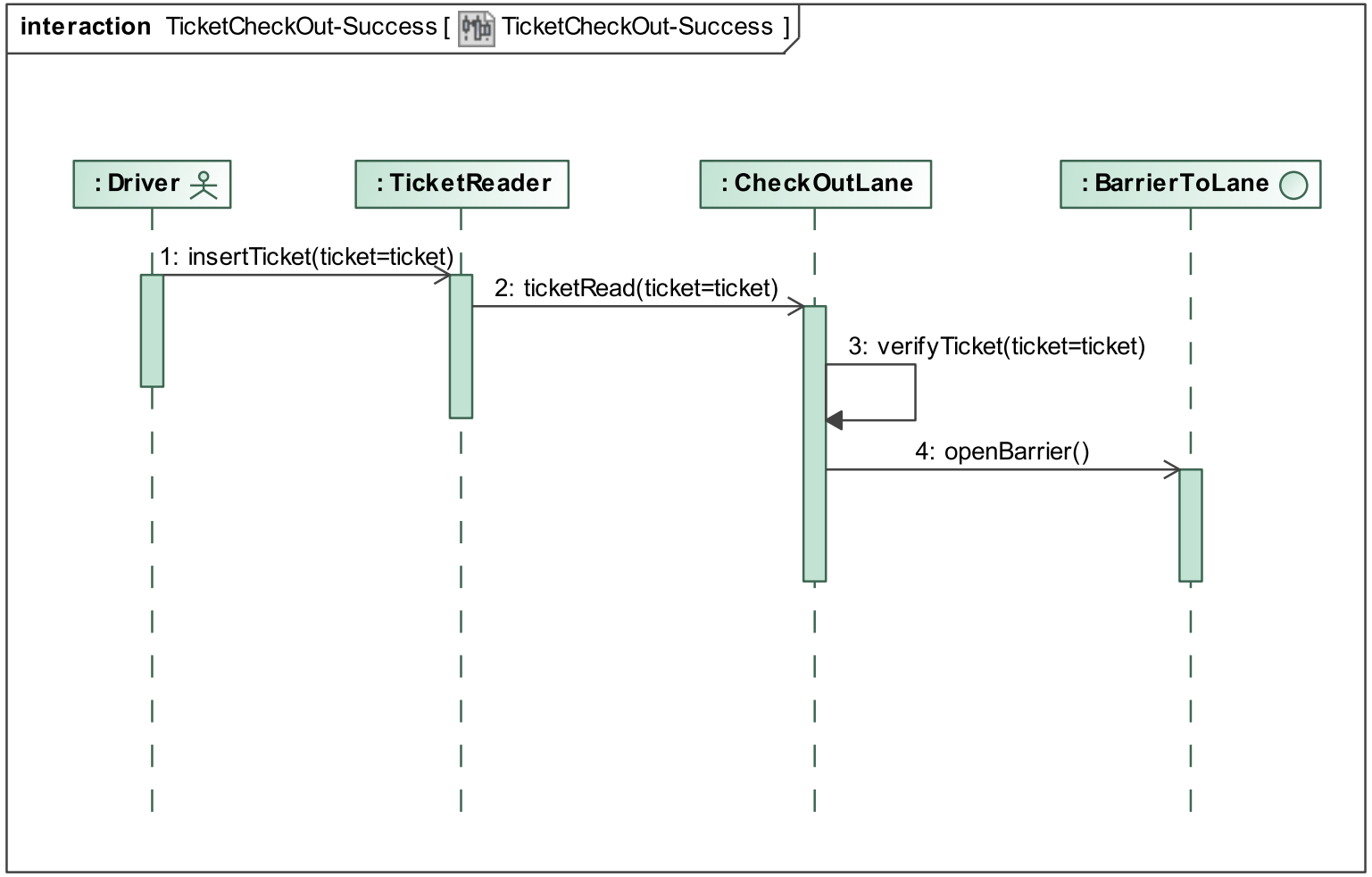

Figure 5.7: Main scenario for the Ticket Check-out use case. *[Johan, Andreas]*

Mikkel Riber Bojsen
Andreas Foldager
Martin Kasban Tange
Johan van Beusekom
Kasper Aaquist Johansen

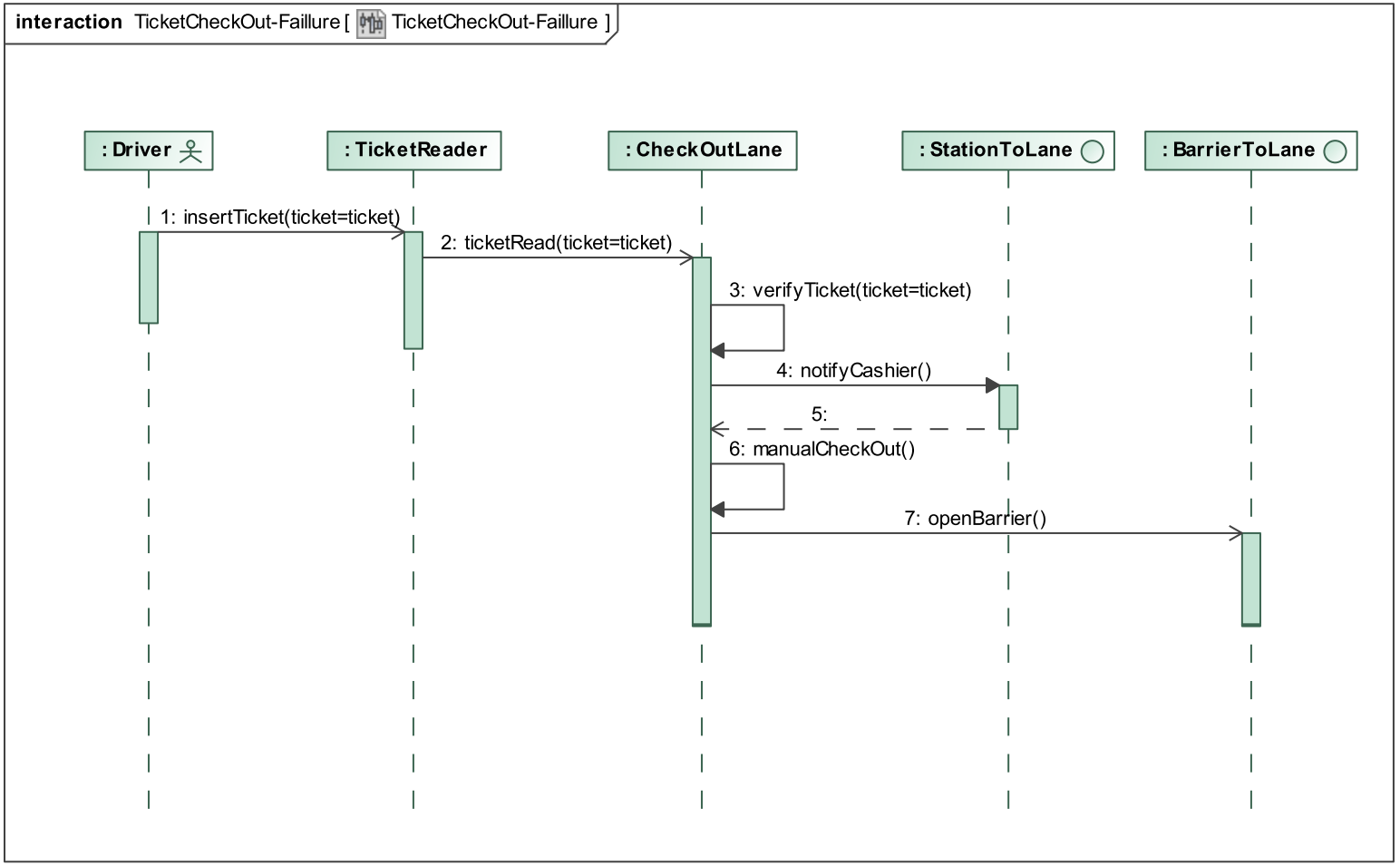

Figure 5.8: Alternative scenario for the Ticket Check-out use case. *[Johan, Andreas]*

Mikkel Riber Bojsen
Andreas Foldager
Martin Kasban Tange
Johan van Beusekom
Kasper Aaquist Johansen

5.1.5 Generate Enterprise Report

[Mikkel, Kasper]

The Generate Enterprise Report use case is realised in Figure 5.9 and Figure 5.10. The alternative scenario happens in Figure 5.9 where it is impossible to reach some stations, so a failure entry is added to the enterprise report.

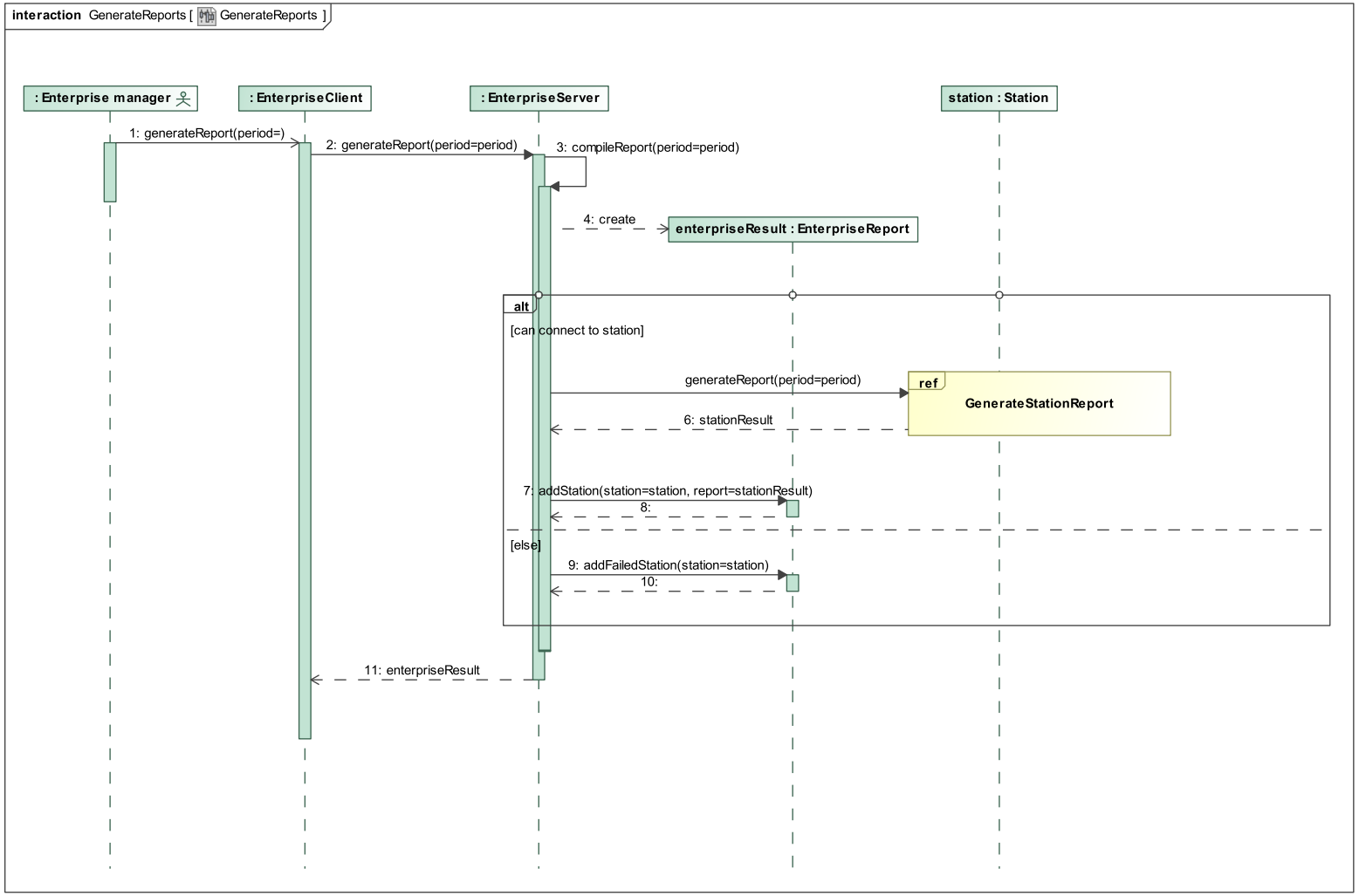

Figure 5.9: Generation of enterprise report use case. Refers to Figure 5.10. *[Mikkel, Kasper]*

Mikkel Riber Bojsen
Andreas Foldager
Martin Kasban Tange
Johan van Beusekom
Kasper Aaquist Johansen

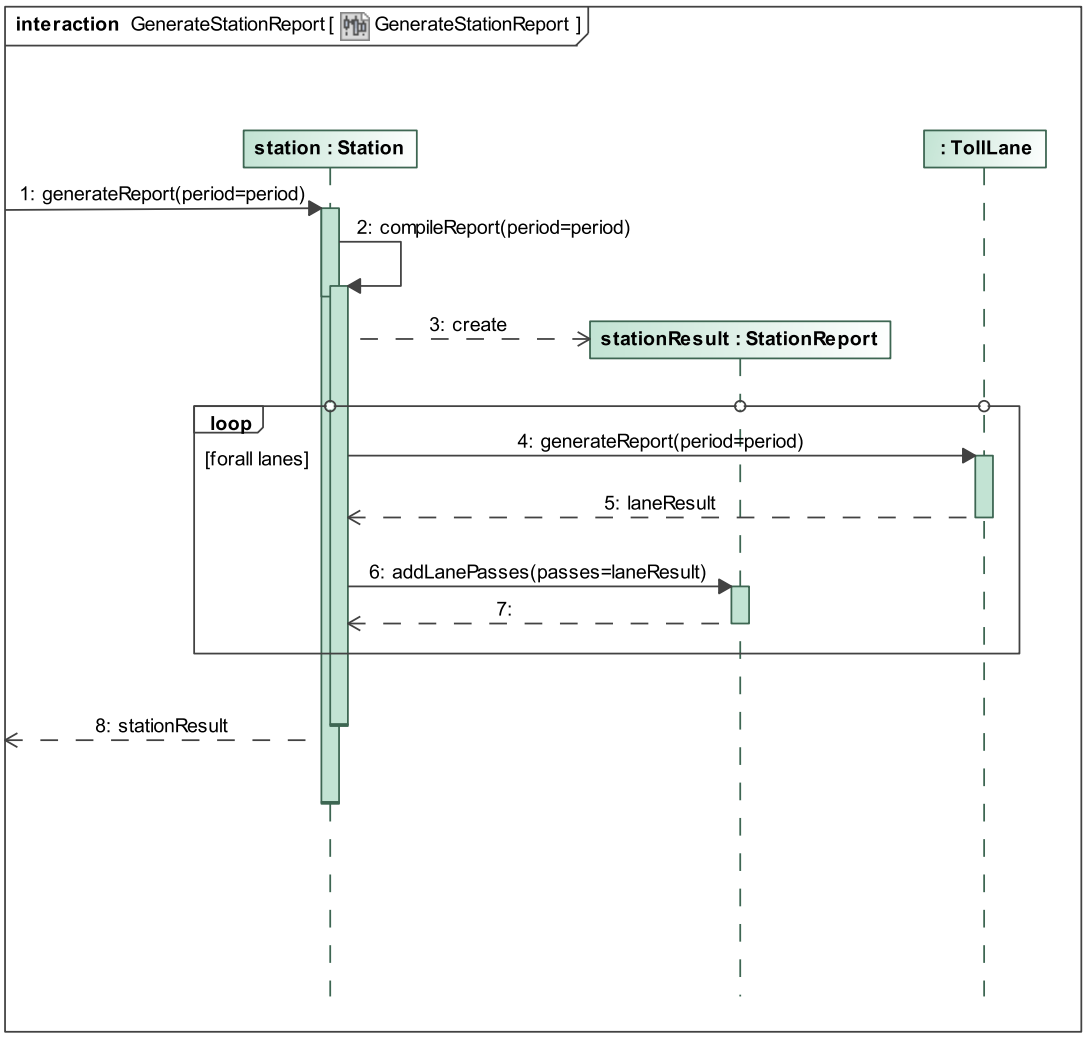

Figure 5.10: Generation of reports of individual stations. Referred to from Figure 5.9. *[Mikkel, Kasper]*

6 Conclusion

In this project we have gotten some valuable experience. When multiple people model many aspects of the same system, consistency becomes a major issue. Using the correct tool can help some of the way, but it still requires discipline from all participants to keep things consistent. When the tool is also unable to merge changes, further care must be taken. We decided to have a physical lock in the form of a hat when the group was gathered and use social media to claim the resource when working apart.

Working on models in a pair helps a lot to get something good done, as someone to spar with can help explore the possibilities better. Reviewing the model with the whole team present looking at the same screen, e.g. with a beamer can help find some of the more sneaky mistakes.