Tallinn University of Technology

Faculty of Information Technology

Department of Informatics

Agent-Oriented Modelling and Multiagent Systems (IDY0303)

Project

Smart Loan Provider Service

Tallinn 2016

Table of Contents

Introduction

The idea of this mini-project is to model a smart loan provider service. Banks are already using similar solutions but our purpose is to make the processes more automated by using different agents, which may be different systems or persons.

The purpose is to make client-friendly loan applying system with quick process so that client should give just his identification number and a sum he/she wants to loan (in some cases some extra info is asked if necessary). By getting client's personal identification number Profiler makes client's profile by getting most (if not all) of the necessary information (income, education etc.) about the client from different agents and so the Analyst can make the final decision about providing loan. The system is meant for smaller loans (without collateral).

This service could also be rather general so that not only banks but everyone (individuals, companies etc.) could use it for calculating lending capacity and providing loan, but the problem is that for getting this kind of personal data about the client some kind of rights and licenses are needed that only banks can get. Individuals could use this service but in this case agents should ask the information straight from the client.

Requirements analysis

Goal model

Goal model is a part of agent organization behaviour analysis. It contains functional goals and also quality goals for agent roles. The goals are described hierarchically. The goal model for current project is shown in Figure 1. In current project the main goal is to provide loan with agents to help with automated decisions and the whole process will be more fluent for the client. In order to achieve this goal providing loan is defined to sub- goal - Calculate lending capacity.

Sub-goals to calculate lending capacity is to get client profile and make decision upon profile. Analyst will make decision only with complex cases or where loan is no longer "small-loan". As risk for loan provider rises, the more personalized the process will get. To gather all necessary information about client's personal data, requests to several registers to get marriage info, age info, supported persons and education info.

Second and important information for profile is client's credit data. Information will be gathered with the same logic – requests to registers and loan provider own credit rating.

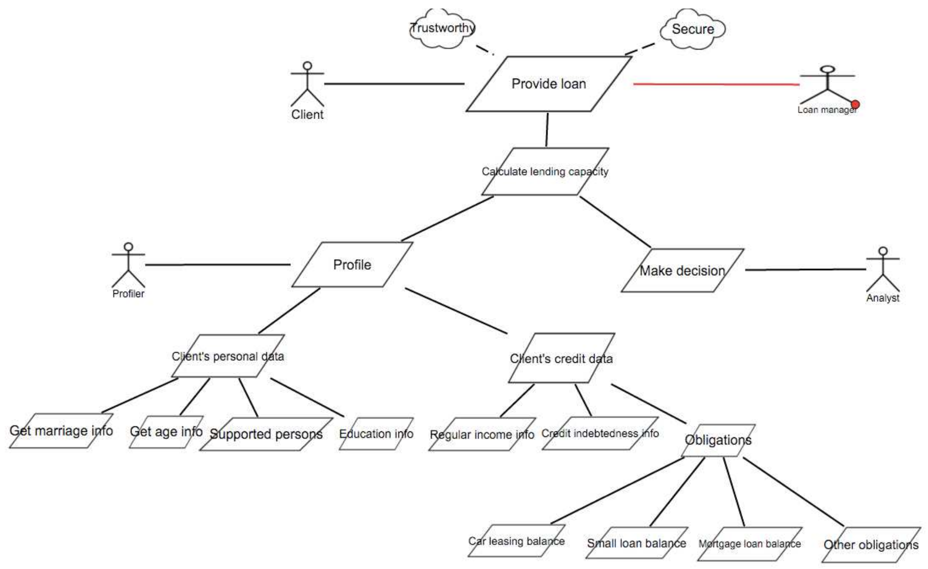
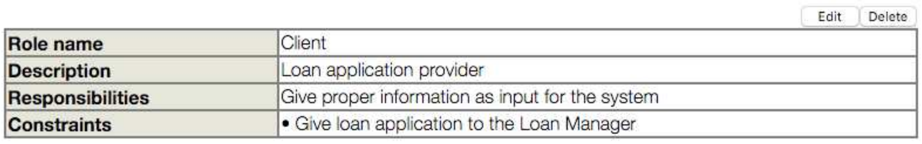


Figure 1 Goal model

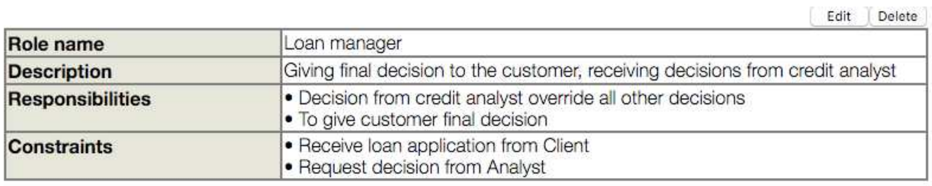
Role model

There are 4 different roles in the system: Client, Loan manager, Analyst and Profiler. These roles are described in detail below.


Edit
Delete

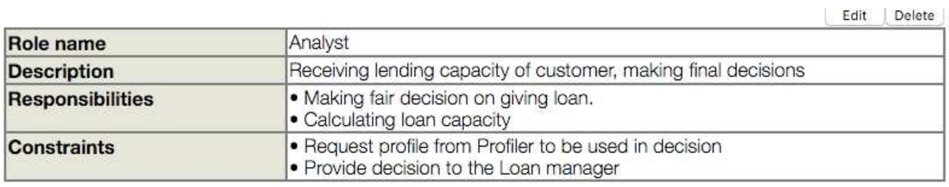
Role name	Client
Description	Loan application provider
Responsibilities	Give proper information as input for the system
Constraints	• Give loan application to the Loan Manager

Table 1 Client role


Edit
Delete

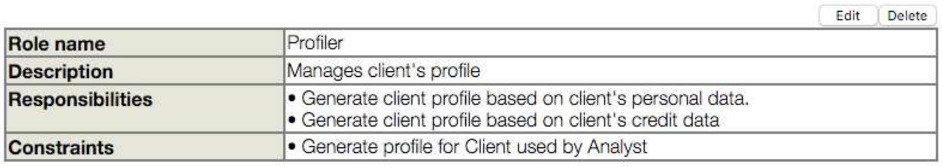
Role name	Loan manager
Description	Giving final decision to the customer, receiving decisions from credit analyst
Responsibilities	• Decision from credit analyst override all other decisions • To give customer final decision
Constraints	• Receive loan application from Client • Request decision from Analyst

Table 2 Loan manager role


Edit
Delete

Role name	Analyst
Description	Receiving lending capacity of customer, making final decisions
Responsibilities	• Making fair decision on giving loan. • Calculating loan capacity
Constraints	• Request profile from Profiler to be used in decision • Provide decision to the Loan manager

Table 3 Analyst role


Edit
Delete

Role name	Profiler
Description	Manages client's profile
Responsibilities	• Generate client profile based on client's personal data. • Generate client profile based on client's credit data
Constraints	• Generate profile for Client used by Analyst

Table 4 Profiler role

Organisation model

Figure 2 shows the organisation model which captures types of interactions between different roles. The model is rather simple, but it describes the interactions very easily.

Loan Manager first receives the loan application from the Client, so the Loan Manager **has to be benevolent to** the Client, Loan Manager requests decision from Analyst, so he **is controlled by** Analyst. Analyst requests the full profile from Profiler, who then generates profile about the client so that Analyst could use the profile in making the decision, so the Profiler **is** also **controlled by** the Analyst, because after getting the profile Analyst makes the decision and provides the decision to the Loan manager who then gives the decision to the Client.

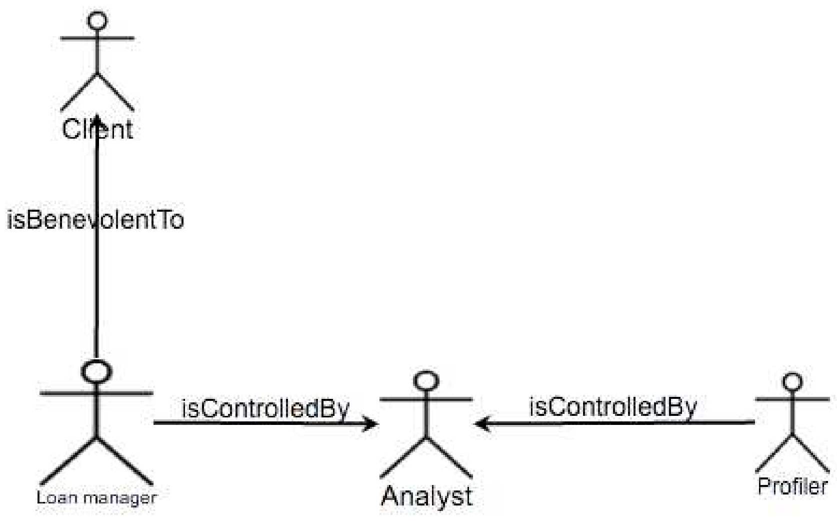


Figure 2 Organisation model

Domain model

Figure 3 shows the domain model. Its purpose is to describe which roles use which resources. There are four different domain entities in the current project: Loan offer, Profile, Lending capacity and Decision.

Client asks Loan offer by filling loan application. Loan manager generates initial Profile with the loan application information from client so that Profiler would have input to collect all the other needed data.

Profiler manages Profile, which is the basis of calculating Lending capacity. Analyst receives Lending capacity and makes decision, after that Loan manager receives the Decision and approves the Loan offer.

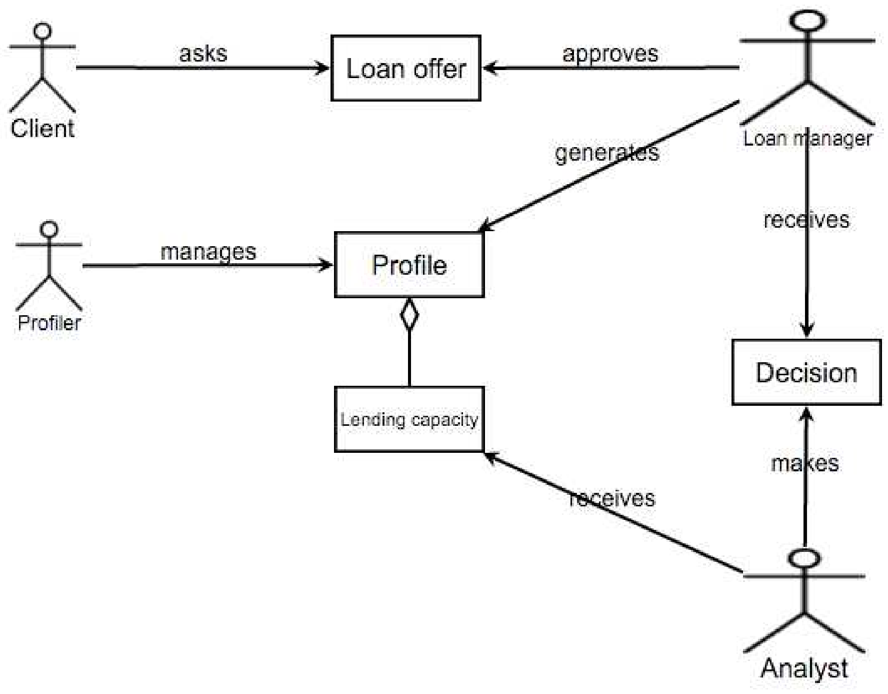


Figure 3 Domain model

System design layer

Agent and acquaintance model

The agent and acquaintance model is a part in agent organization interaction design. The agent and acquaintance model for the current project is shown in Figure 4 and describes relationships between agent roles and agent types.

The Client role is mapped to Applicant Agent type.
The Loan manager role is mapped to Loan Manager Agent.
Roles Analyst and Profiler are both mapped to Credit Analyst Agent.
The model on Figure 4 also shows the interaction pathways between the agent types:
Loan Manager Agent interacts with both Applicant and Credit Analyst Agent but Applicant Agent and Credit Analyst Agent doesn't interact with each other.

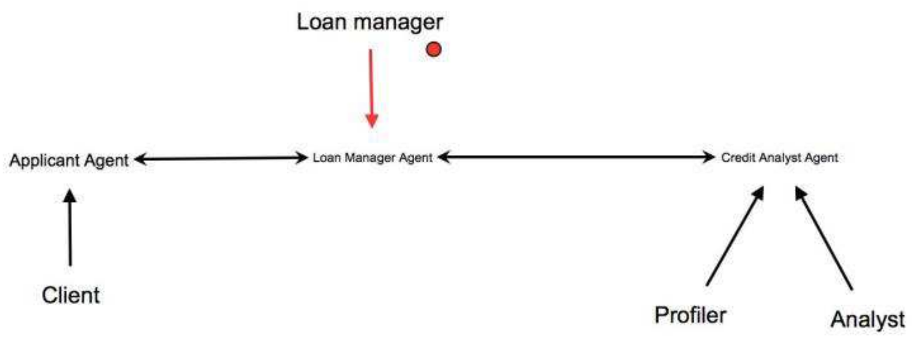
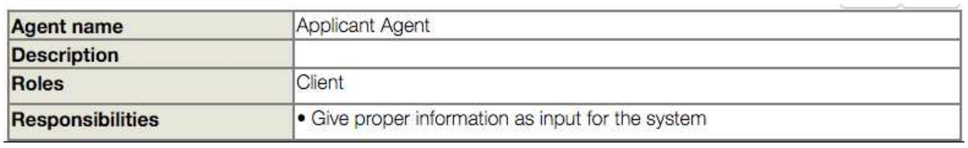


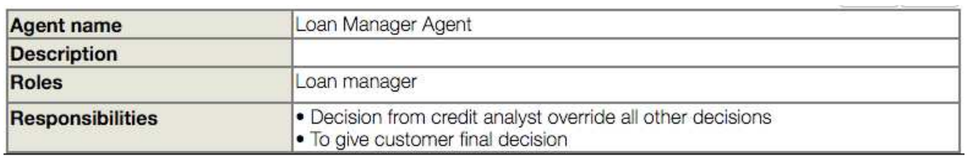
Figure 4 Agent and acquaintance model

Below is also shortly described the responsibilities of the Agents.



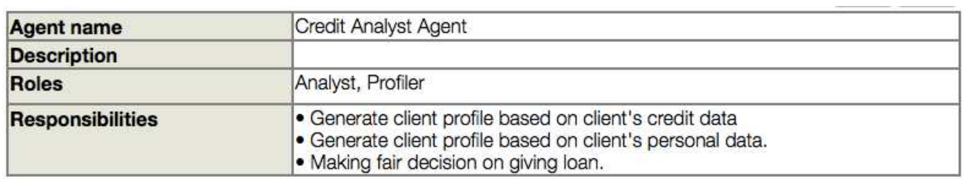
Agent name	Applicant Agent
Description	
Roles	Client
Responsibilities	• Give proper information as input for the system

Table 5 Applicant Agent



Agent name	Loan Manager Agent
Description	
Roles	Loan manager
Responsibilities	• Decision from credit analyst override all other decisions • To give customer final decision

Table 6 Loan Manager Agent



Agent name	Credit Analyst Agent
Description	
Roles	Analyst, Profiler
Responsibilities	• Generate client profile based on client's credit data • Generate client profile based on client's personal data. • Making fair decision on giving loan.

Table 7 Credit Analyst Agent

Knowledge model

Knowledge model is a part in agent organization information design. It represents the knowledge that agent types have about their environments and about themselves. The knowledge model for current project is shown in Figure 5. The knowledge model is already a more elaborate model of information available for agent types, it has the same relationship framework as the domain model but here the information is represented in a more structured way to explain the basis of providing loan.

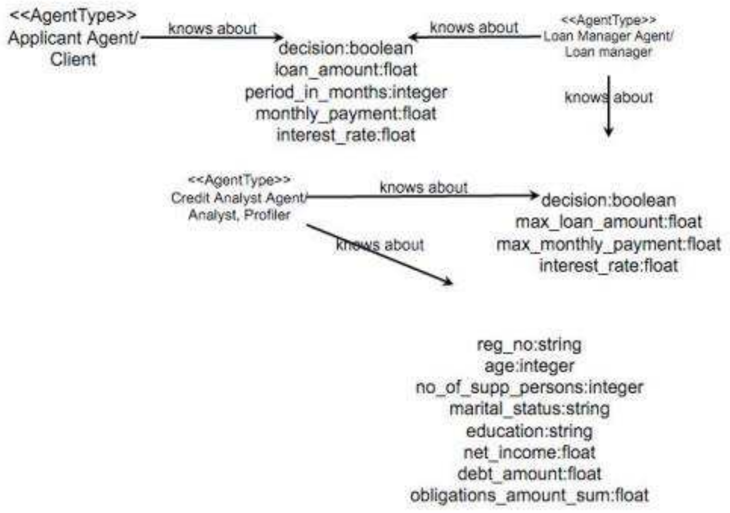


Figure 5 Knowledge model

Interaction models

The basic interactions required by the agents are shown in Figure 6.

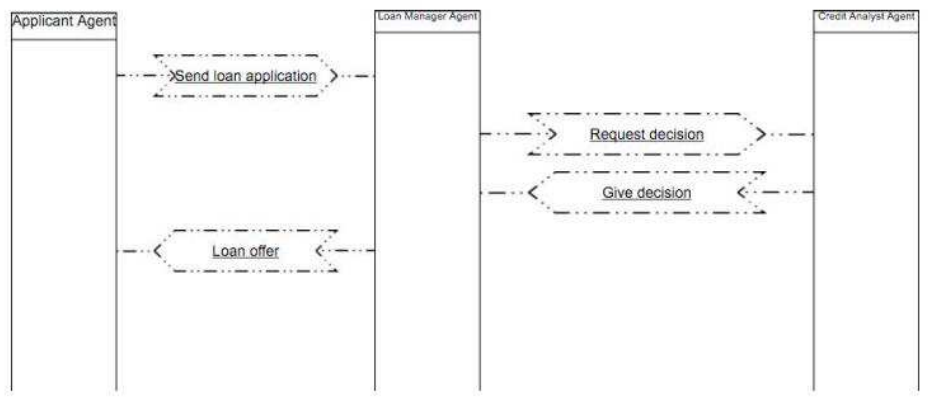


Figure 6 Interaction models

Behaviour models

Behaviour model is shown in Figure 7.

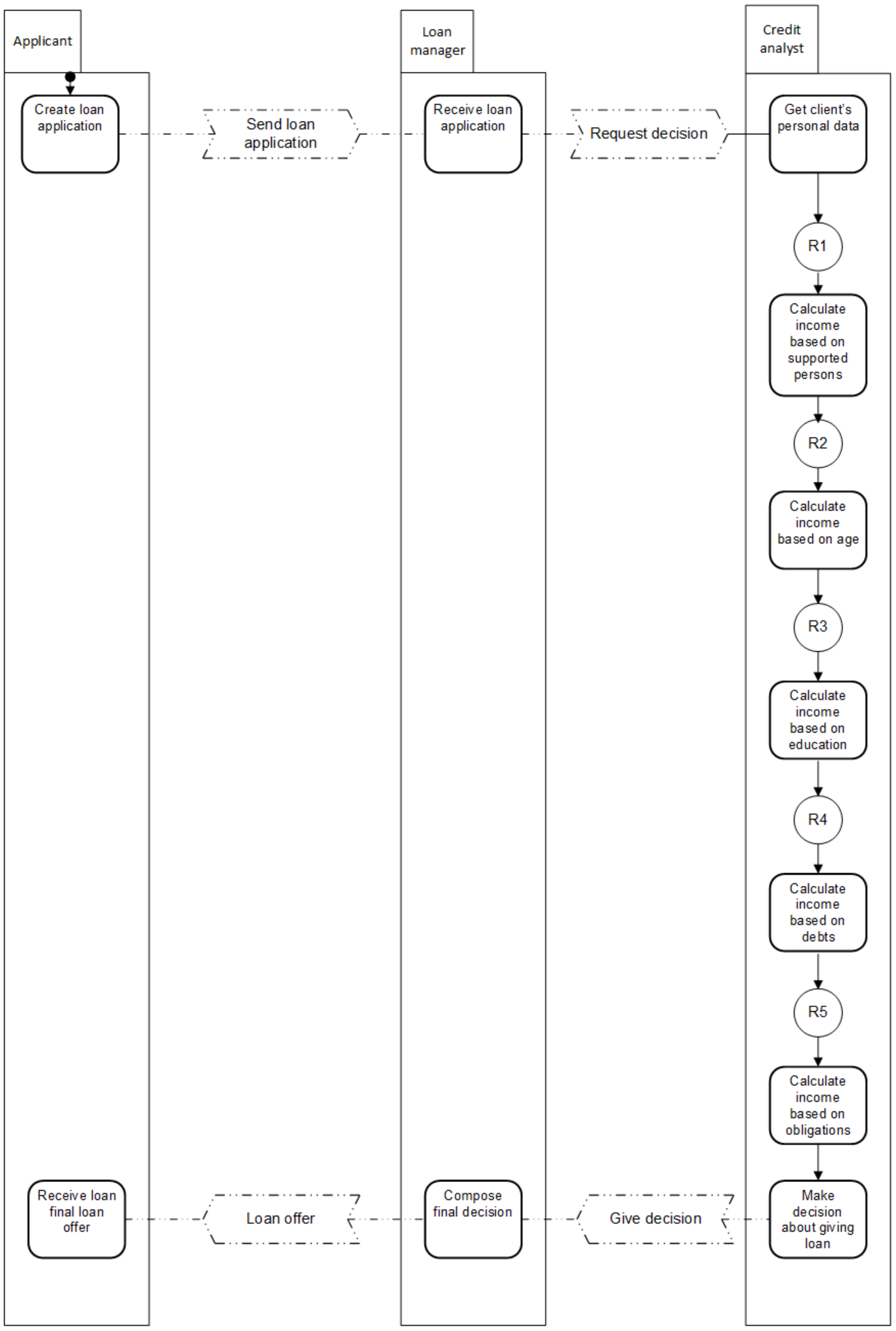

Applicant
Loan manager
Credit analyst
Create loan application
Send loan application
Receive loan application
Request decision
Get client's personal data
R1
Calculate income based on supported persons
R2
Calculate income based on age
R3
Calculate income based on education
R4
Calculate income based on debts
R5
Calculate income based on obligations
Make decision about giving loan
Give decision
Compose final decision
Loan offer
Receive loan final loan offer

Figure 7 Behaviour models

Analysis in CPN Tools

Analysis in CPN Tools, focused on validation and rules are under this chapter.

Validation

In this section we are using CPN tools to validate our models. This tool helps us to visualize and validate our proposed models and run rules against it.

In order to do this, we have described the following scenarios:

1. A positive result on loan request
    a. The loan request is satisfied to the fullest
2. A negative result on loan request
    a. The loan request is rejected fully

CPN model is in Figure 8. It is showing our flow in high-level. All kind of different rules (relations between applicant's profile and decision) are not in the place in our current model. It's mainly because we have many complicated rules in our system and it was hard to implement that logic in CPN Tools. So in Figure 8 and 9 we have basic working version of CPN model.

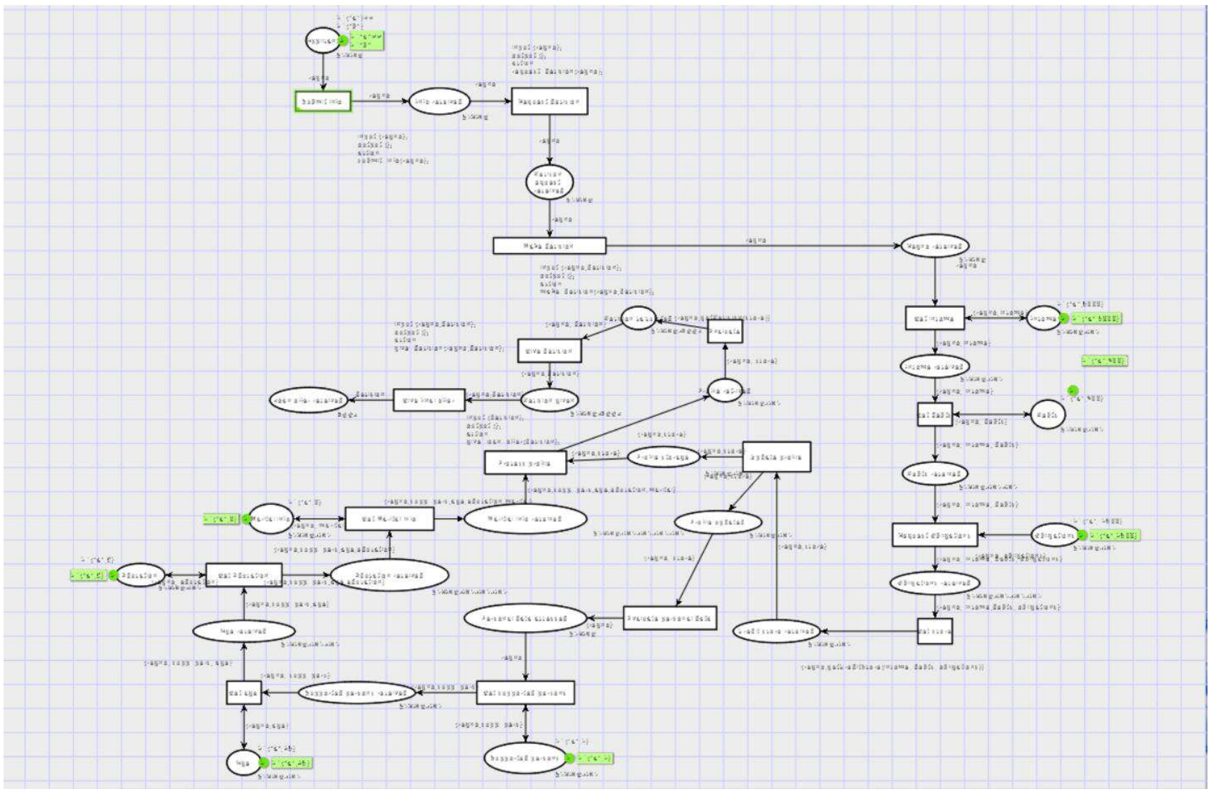
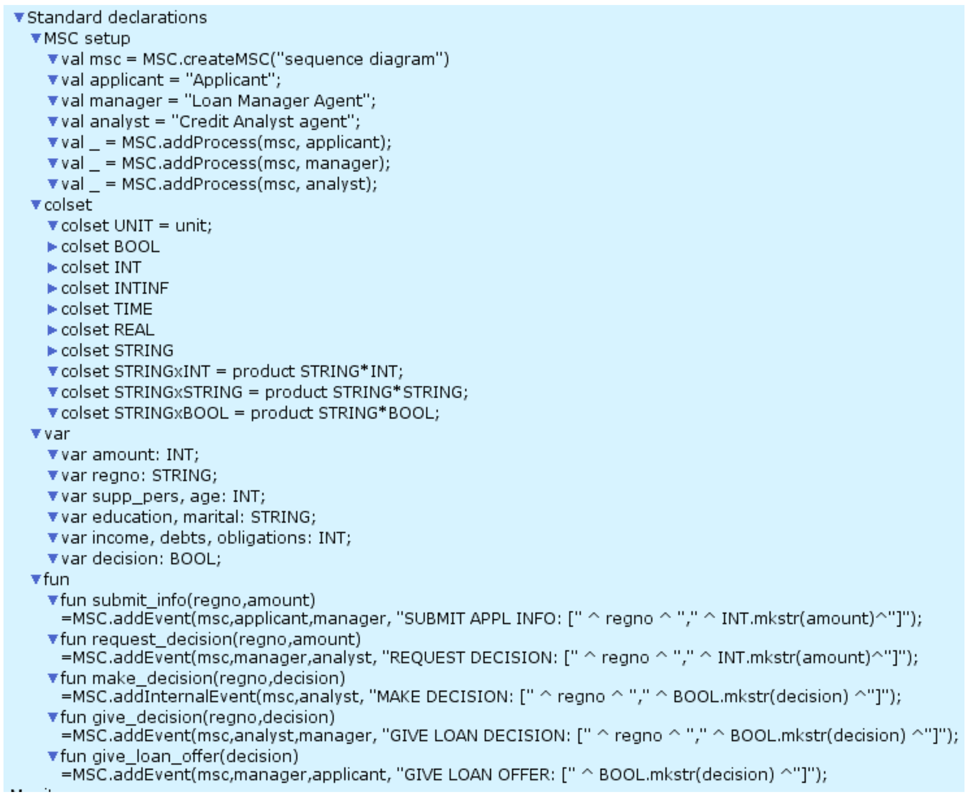


Figure 8 CPN model



```
▼Standard declarations
  ▼MSC setup
    ▼val msc = MSC.createMSC("sequence diagram")
    ▼val applicant = "Applicant";
    ▼val manager = "Loan Manager Agent";
    ▼val analyst = "Credit Analyst agent";
    ▼val _ = MSC.addProcess(msc, applicant);
    ▼val _ = MSC.addProcess(msc, manager);
    ▼val _ = MSC.addProcess(msc, analyst);
  ▼colset
    ▼colset UNIT = unit;
    ▶colset BOOL
    ▶colset INT
    ▶colset INTINF
    ▶colset TIME
    ▶colset REAL
    ▶colset STRING
    ▼colset STRINGxINT = product STRING*INT;
    ▼colset STRINGxSTRING = product STRING*STRING;
    ▼colset STRINGxBOOL = product STRING*BOOL;
  ▼var
    ▼var amount: INT;
    ▼var regno: STRING;
    ▼var supp_pers, age: INT;
    ▼var education, marital: STRING;
    ▼var income, debts, obligations: INT;
    ▼var decision: BOOL;
  ▼fun
    ▼fun submit_info(regno,amount)
      =MSC.addEvent(msc,applicant,manager, "SUBMIT APPL INFO: [" ^ regno ^ "," ^ INT.mkstr(amount)^"]");
    ▼fun request_decision(regno,amount)
      =MSC.addEvent(msc,manager,analyst, "REQUEST DECISION: [" ^ regno ^ "," ^ INT.mkstr(amount)^"]");
    ▼fun make_decision(regno,decision)
      =MSC.addInternalEvent(msc,analyst, "MAKE DECISION: [" ^ regno ^ "," ^ BOOL.mkstr(decision) ^"]");
    ▼fun give_decision(regno,decision)
      =MSC.addEvent(msc,analyst,manager, "GIVE LOAN DECISION: [" ^ regno ^ "," ^ BOOL.mkstr(decision) ^"]");
    ▼fun give_loan_offer(decision)
      =MSC.addEvent(msc,manager,applicant, "GIVE LOAN OFFER: [" ^ BOOL.mkstr(decision) ^"]");
```

Figure 9 CPN model - declared variables

Simulation were done based on Figure 8. We had two examples, one person who gets loan and other don't. So we have simulated two possible scenarios. They are visible in generated message sequence chart (Figure 10)

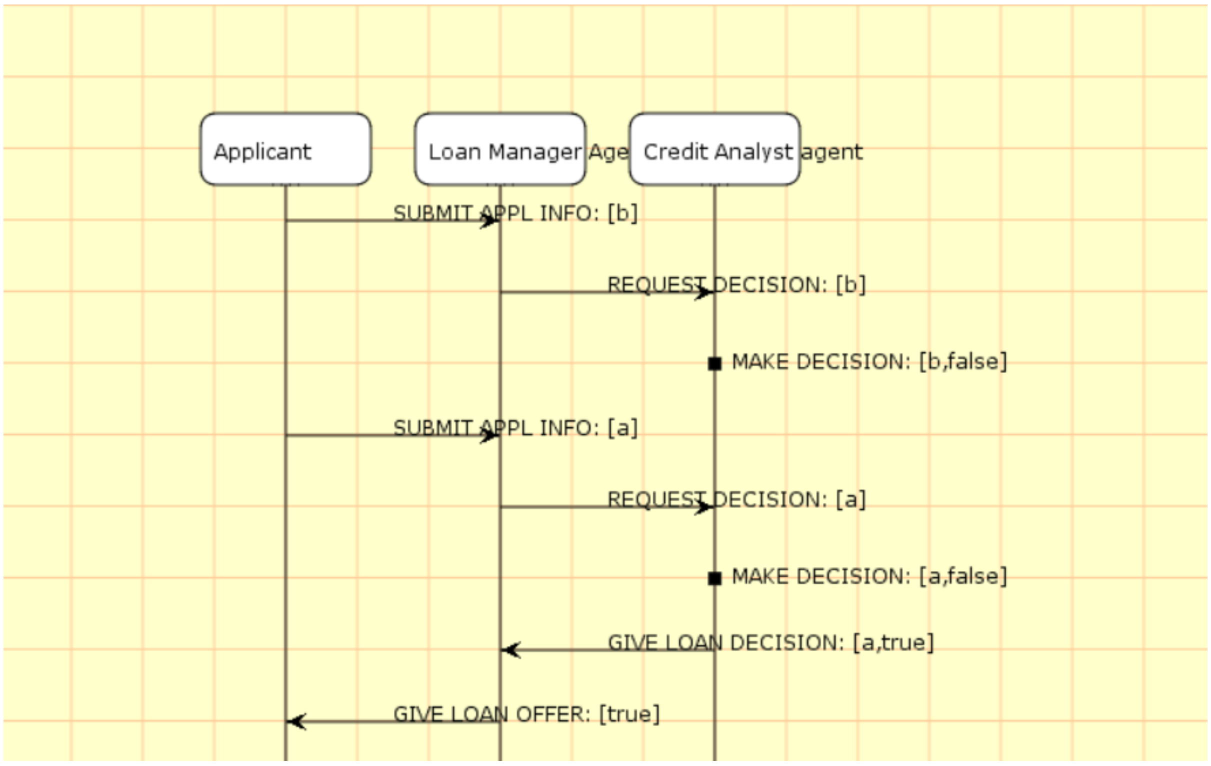


Figure 10 Message sequence chart

Rules

In this section are described rules which are missing from CPN model, most of them should have under "MAKE DECISION".

We need to make decision (if we can give loan in requested amount or not) based on applicant's profile. Biggest relation with amount is applicant's income. So we are getting applicant's income as initial amount and we are changing it based on other info which belongs to applicant.

Here are described some examples of rules, they are showing how initial income is impacted:

- Supported persons
  - If applicant has more than 3 children, then we are decreasing income amount by some %
- Age
  - If applicant is too old (going soon to retirement), then we are decreasing income amount by some %
- Education
  - If applicant got higher education (probably can make career and get higher salary in future), then we are increasing income amount by some %
- Debts
  - If applicant has some debts (getting overall amount), then depends on amount we probably cannot give a loan at all
- Obligations
  - If applicant has some obligations (getting overall amount), then we are dividing amount by average loan period and can decrease initial income by that amount

Then we need to take into account that applicants cannot use all his income for loan payment, then we are taking some % of final income (average that how much usually people are spending their income on their household etc.). After that we got final income amount which can be used for loan payments in every month.

Finally, we calculate see how much loan (and for how long) we can give to applicant based on this monthly payment.

Verification

Verification results from Coloured Petri nets application is in Appendix 1.

Conclusion

It was interesting to investigate and deal with our smart loan providining system, but it seemed to be a little bit complicated for this mini-project. We understood that when we tried to create different scenarios and rules for them (especially when tried to visualize them).

It was hard to implement complicated system in CPN Tools. Is seemed to be more suitable for smaller systems - to visualize process, simulate, verify and validate them. Otherwise it seemed to be too time-consuming and hard to use for fancy things. Even MSC creation process was complicated (too much writing, copy-paste etc.). In positive side, the simulation part was really good and it can be very helpful to see bottlenecks in new systems.
In conclusion, visualizing process flow in CPN Tools can be very useful but it is too time-consuming if you are not CPN-expert.

Table of figures

Appendix 1

```
CPN Tools state space report for:
/cygdrive/D/Kool/IDY0303 - Agentorienteeritud/loan_draft (2).cpn
Report generated: Mon May 23 00:03:57 2016


 Statistics
------------------------------------------------------------------------

  State Space
     Nodes:  72
     Arcs:   144
     Secs:   2
     Status: Full

  Scc Graph
     Nodes:  72
     Arcs:   144
     Secs:   0


 Boundedness Properties
------------------------------------------------------------------------

  Best Integer Bounds
                             Upper      Lower
     New_Page'Age 1          1          1
     New_Page'Age_received 1 1          0
     New_Page'Applicant 1    2          0
     New_Page'Credit_score_received 1
                             1          0
     New_Page'Debts 1        1          1
     New_Page'Debts_received 1
                             1          0
     New_Page'Decision_calculated 1
                             1          0
     New_Page'Decision_given 1
                             1          0
     New_Page'Decison_equest_received 1
                             2          0
     New_Page'Education 1    1          1
     New_Page'Education_received 1
                             1          0
     New_Page'Income 1      1         1
     New_Page'Income_received 1
                         1         0
     New_Page'Info_received 1
                         2         0
     New_Page'Loan_offer_received 1
                         1         0
     New_Page'Marital_info 1 1         1
     New_Page'Maritial_info_received 1
                         1         0
     New_Page'Obligations 1  1         0
     New_Page'Obligations_received 1
                         1         0
     New_Page'Personal_data_accessed 1
                         1         0
     New_Page'Profile_retrived 1
                         1         0
     New_Page'Profile_storage 1
                         1         0
     New_Page'Profile_updated 1
                         1         0
     New_Page'Regno_received 1
                         2         0
     New_Page'Supported_persons 1
                         1         1
     New_Page'Supported_persons_received 1
                         1         0

  Best Upper Multi-set Bounds
     New_Page'Age 1      1`("a",25)
     New_Page'Age_received 1
                     1`("a",1,25)
     New_Page'Applicant 1
                     1`"a"++
1`"b"
     New_Page'Credit_score_received 1
                     1`("a",3100)
     New_Page'Debts 1    1`("a",400)
     New_Page'Debts_received 1
                     1`("a",5000,400)
     New_Page'Decision_calculated 1
                     1`("a",true)
     New_Page'Decision_given 1
                     1`("a",true)
     New_Page'Decison_equest_received 1
                     1`"a"++
1`"b"
     New_Page'Education 1
                     1`("a",3)
     New_Page'Education_received 1
                     1`("a",1,25,3)
     New_Page'Income 1   1`("a",5000)
     New_Page'Income_received 1
                     1`("a",5000)
     New_Page'Info_received 1
                     1`"a"++
1`"b"
     New_Page'Loan_offer_received 1
                     1`true
     New_Page'Marital_info 1
                     1`("a",0)
     New_Page'Maritial_info_received 1
                     1`("a",1,25,3,0)
     New_Page'Obligations 1
                     1`("a",1500)
     New_Page'Obligations_received 1
```

```
                    1`("a",5000,400,1500)
    New_Page'Personal_data_accessed 1
                    1`"a"
    New_Page'Profile_retrived 1
                    1`("a",3100)
    New_Page'Profile_storage 1
                    1`("a",3100)
    New_Page'Profile_updated 1
                    1`("a",3100)
    New_Page'Regno_received 1
                    1`"a"++
1`"b"
    New_Page'Supported_persons 1
                    1`("a",1)
    New_Page'Supported_persons_received 1
                    1`("a",1)

  Best Lower Multi-set Bounds
    New_Page'Age 1      1`("a",25)
    New_Page'Age_received 1
                    empty
    New_Page'Applicant 1
                    empty
    New_Page'Credit_score_received 1
                    empty
    New_Page'Debts 1    1`("a",400)
    New_Page'Debts_received 1
                    empty
    New_Page'Decision_calculated 1
                    empty
    New_Page'Decision_given 1
                    empty
    New_Page'Decison_equest_received 1
                    empty
    New_Page'Education 1
                    1`("a",3)
    New_Page'Education_received 1
                    empty
    New_Page'Income 1   1`("a",5000)
    New_Page'Income_received 1
                    empty
    New_Page'Info_received 1
                    empty
    New_Page'Loan_offer_received 1
                    empty
    New_Page'Marital_info 1
                    1`("a",0)
    New_Page'Maritial_info_received 1
                    empty
    New_Page'Obligations 1
                    empty
    New_Page'Obligations_received 1
                    empty
    New_Page'Personal_data_accessed 1
                    empty
    New_Page'Profile_retrived 1
                    empty
    New_Page'Profile_storage 1
                    empty
    New_Page'Profile_updated 1
                    empty
    New_Page'Regno_received 1
                    empty
    New_Page'Supported_persons 1
                    1`("a",1)
    New_Page'Supported_persons_received 1
```

```
                  empty


 Home Properties
------------------------------------------------------------------------

  Home Markings
     [72]


 Liveness Properties
------------------------------------------------------------------------

  Dead Markings
     [72]

  Dead Transition Instances
     None

  Live Transition Instances
     None


 Fairness Properties
------------------------------------------------------------------------
     No infinite occurrence sequences.
```

SAVE THE

Project Report

May 2016

A mini-project in the course

„Agent-Oriented modelling and Multiagent Systems“

TALLINN UNIVERSITY OF TECHNOLOGY

Table of content

Introduction

Our purpose is to create a sociotechnical system that would allow unused and leftover food to be provided to those in need of nutritious food…

The end objective goal is to mitigate food waste, provide a helping service and overall improve the communities we live in.

We are taking on an issue within society's complex infrastructure in relation to human behavior.

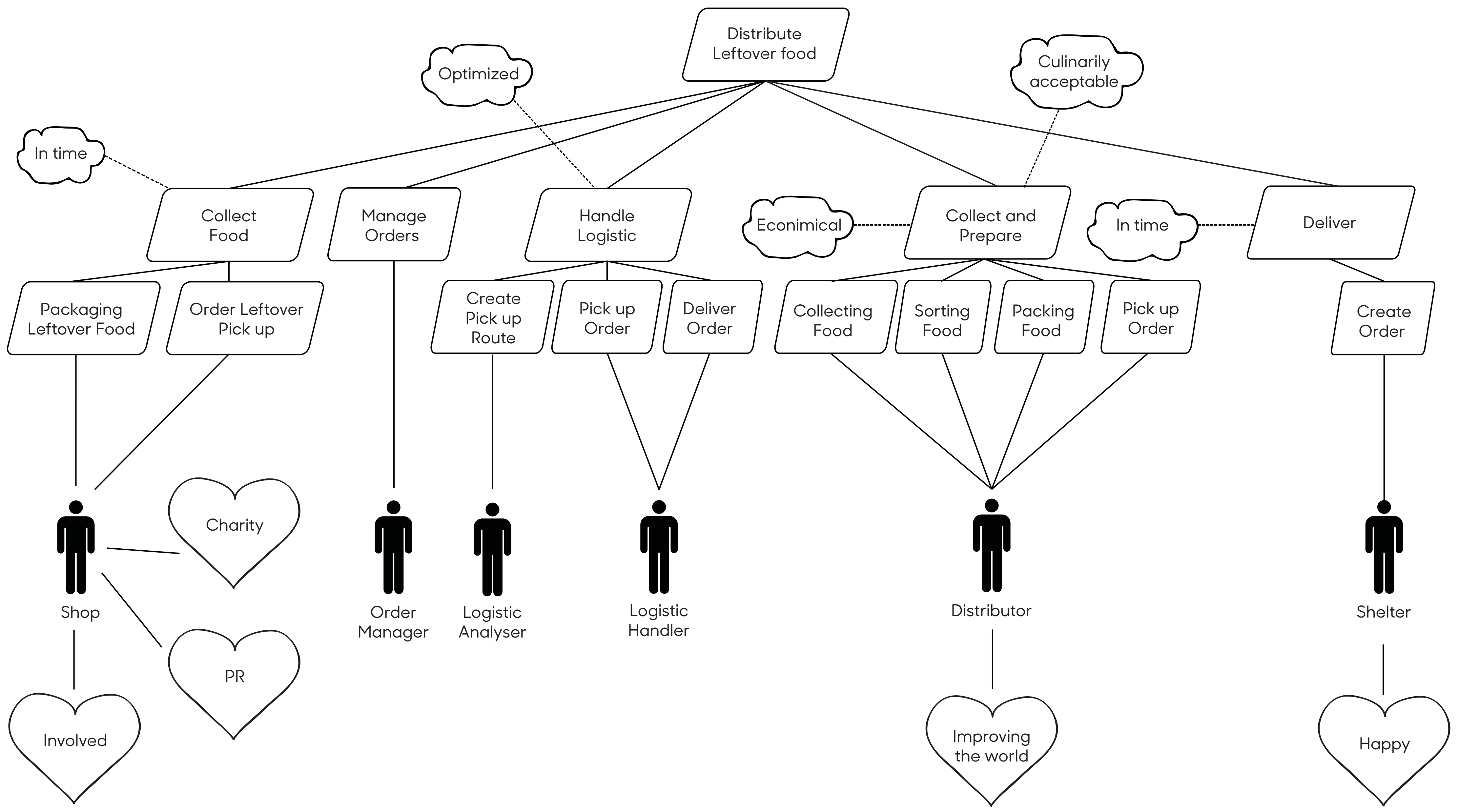
The process of our system, mainly supply and demand is reliant upon software to organize, notify and also promote human interaction for the common good.

1. Motivation layer

This section contains the following models from the motivation layer of conceptual space:

1.1. Goal model

The goal model describes the goals and connects them to specific roles. The model also connects the goals with specific quality requirements that the system is trying to achieve. Goal model is a part of agent organization behavior analysis. It contains functional goals and also quality goals for agent roles. In current project the main goal is to make people in need happy by receiving food from the supermarkets. There are also sub-goals marked with heart symbols.



1.2. Role model

There are three different roles in current project: “Shop”, “Shelter”, “Distributor”, “Order Manager”, “Logistic Handler” and “Logistic Analyser”.

The roles are described in detail below.

Role Name	**Shop**
Description	The role of providing food for the service
Responsibilities	• Packs leftover food • Creates pick up order
Constraints	• Food appropriable for consuming • Pick up order is created when suitable amount of food is collected

Role Name	**Shelter**
Description	The role of people in need represented by Shelter
Responsibilities	• Creates food order • Receives delivery • Stores food
Constraints	• Distributes the food to people in need

Role Name	**Distributor**
Description	The role of Distributing the food packages
Responsibilities	• Receiving the delivery • Sorts food • Packs food • Orders pick up of ready packs
Constraints	• Follow quality standards • Sorts Food by type • Sorts Food by date of appropriability • Sorts Food by Order created by Shelter

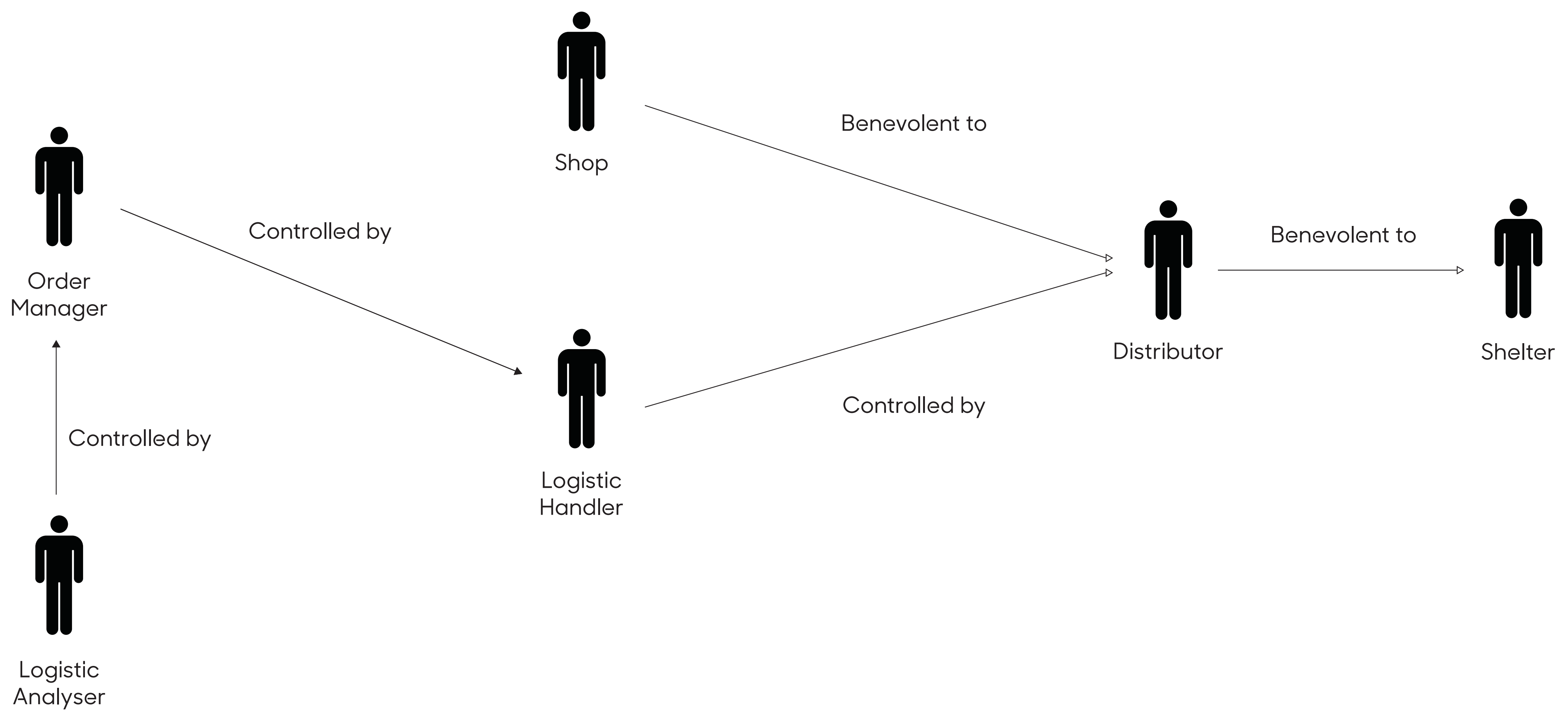
Role Name	**Order Manager**
Description	The Role of receiving and managing orders
Responsibilities	• Receives food orders from Shelters • Receives pick up orders from Shops • Receives pick up orders from Ditributors • Sends notification to Logistic Handler • Sends notification to Shop • Sends notification to Shelter
Constraints	• Communicate with Logistic Analyser • Keep history of orders and data of Shops

Role Name	**Logistic Handler**
Description	The role of picking up and delivering food
Responsibilities	• Receives notification from Order Manager • Picks up leftover food from Shop • Uses Pick up route • Delivers leftover food to Ditributor • Picks up packs from Distributor • Delivers packs to Shelters
Constraints	• Quality communication • Fast response • Abide by business standards

Role Name	**Logistic Analyser**
Description	The role of selecting the most effective delivery method by stored datadelivery plans
Responsibilities	• Analyses pick up points • Creates pick up route • Analyses delivery points • Creates delivery route
Constraints	• Calculate the shortest route • Decides about the pick up way

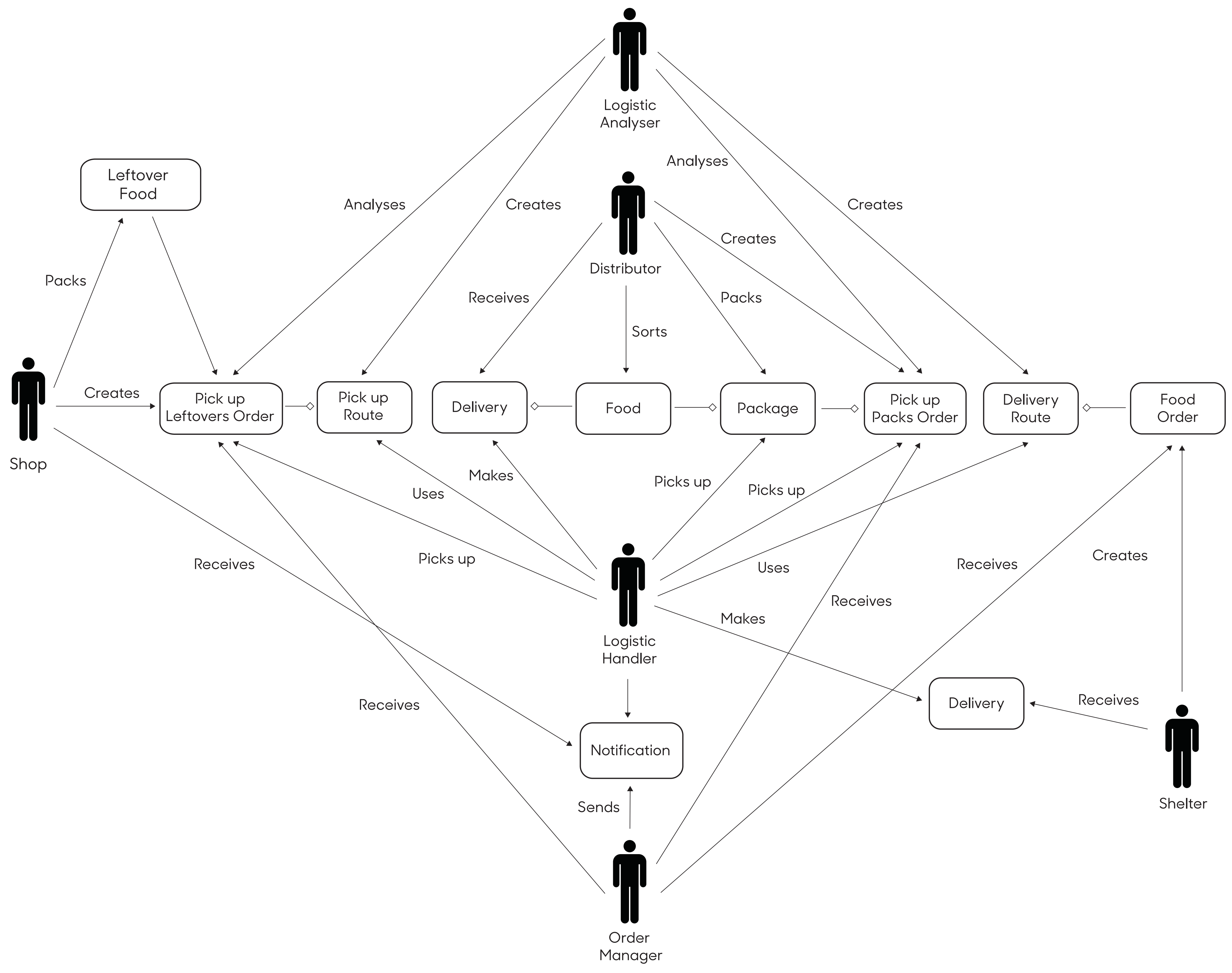
1.3. Organization model

The Organization model describes the relationships between the roles in the agent organization. There are two standard types of relationships used: “control” and “benevolence”. These are the roles that are represented within our system:



1.4. Domain model

Domain model represents the knowledge within the system that our sociotechnical system has to be equipped to handle. The model is mostly self-explanatory: the leftover food will be packed in the shop, picked up from there, sorted and then delivered to the shelters. The shelters will create food orders based on their actual needs. Shops after collecting some amount of leftover food will create pick up order.

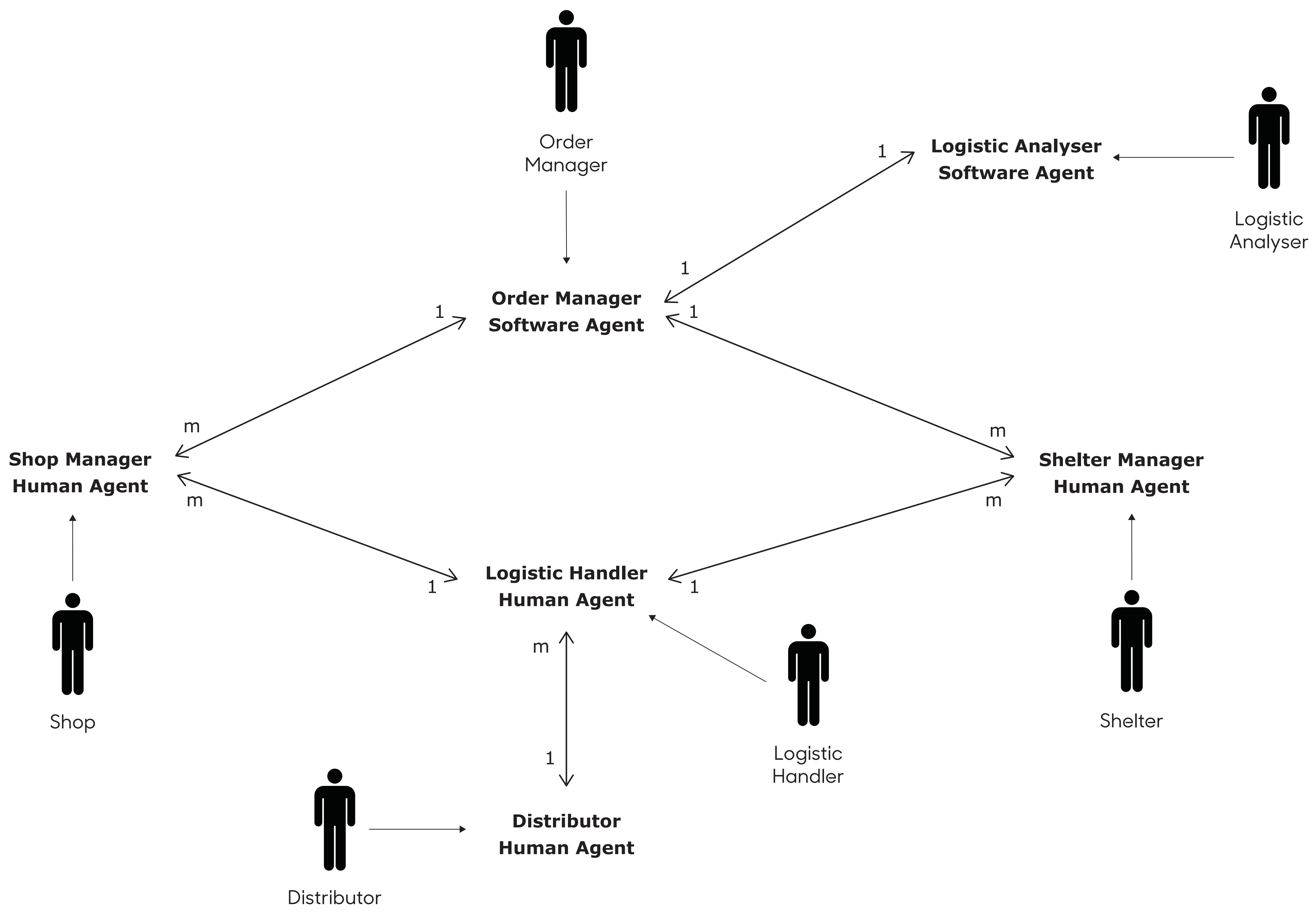


2. System design layer

This section contains the following models of the system design layer:

2.1. Acquaintance model

Acquaintance Model Describes Roles and Agent Types and who interacts with who.



2.2. Agent model

Agent Models show who the agents are (4 human agents and 2 software agents in our case).

Agent Name	**Shop Manager Human Agent**
Description	Representative person from the shop
Roles	Shop
Responsibilities	• Packs leftover food • Creates pick up order

Agent Name	**Shelter Food Manager Human Agent**
Description	Representative person from a shelter who takes care of food resources
Roles	Shop
Responsibilities	• Creates food order • Receives delivery • Stores food • Distributes the food to people in need

Agent Name	**Distributor Human Agent**
Description	Distributor who can be a volunteer or a worker of our organization
Roles	Distributor
Responsibilities	• Receiving the delivery • Sorts food • Packs food • Orders pick up of ready packs

Agent Name	**Order Manager Software Agent**
Description	The Intelligent Digital System which handles orders
Roles	Order Manager
Responsibilities	• Receives pick up orders • Receives food orders • Sends notifications

Agent Name	**Logistic Handler Human Agent**
Description	Driver who can be our worker or volunteer
Roles	Logistic Handler
Responsibilities	• Receives routes and requests • Picks up orders • Delivers food

Agent Name	**Logistic Analyser Software Agent**
Description	Analyses Data and creates pick up and delivery plans
Roles	Logistic Analyser
Responsibilities	• Analyses pick up points • Creates pick up route • Analyses delivery points • Creates delivery route

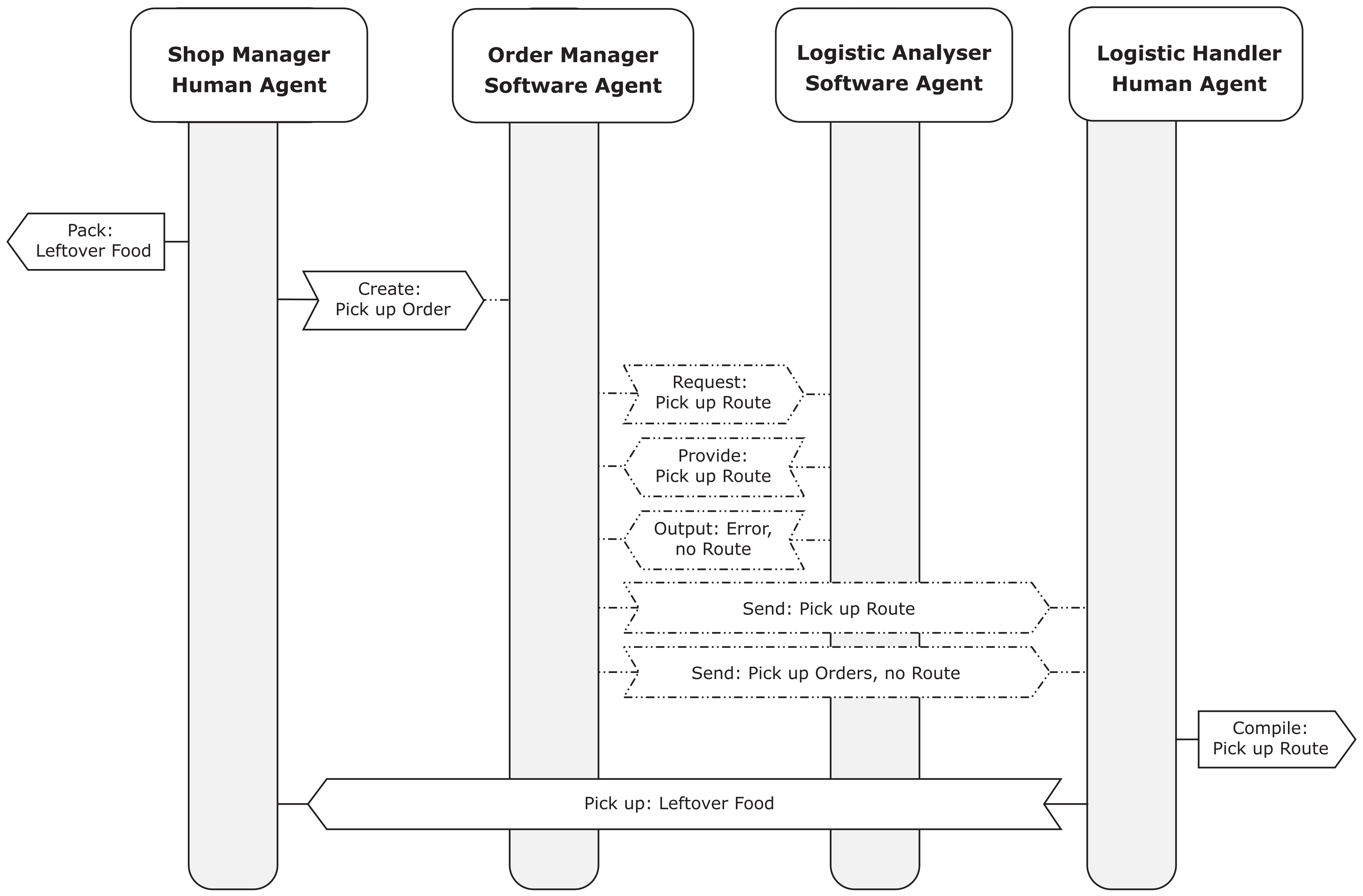
2.3. Interaction models

The basic interactions required by the Human and Software agents are shown in the following 3 models grouped by functionality and sub goals. These diagrams illustrate common examples of communication between different agents. Main interaction pathways are pickup, distributing and delivery and the requests that are associated with them.

2.3.1. Interaction Model of Handling Shop Pick up Order

Shop Manager (Human Agent) packs the leftover food and creates pick up order. Order Manager (Software Agent) requests a Pick up Route from Logistic Analyser (Software Agent). When the Route is provided Order Manager will send it to Logistic Handler (Human Agent) that will pick up the Shop Package from Shop Manager (Human Agent).

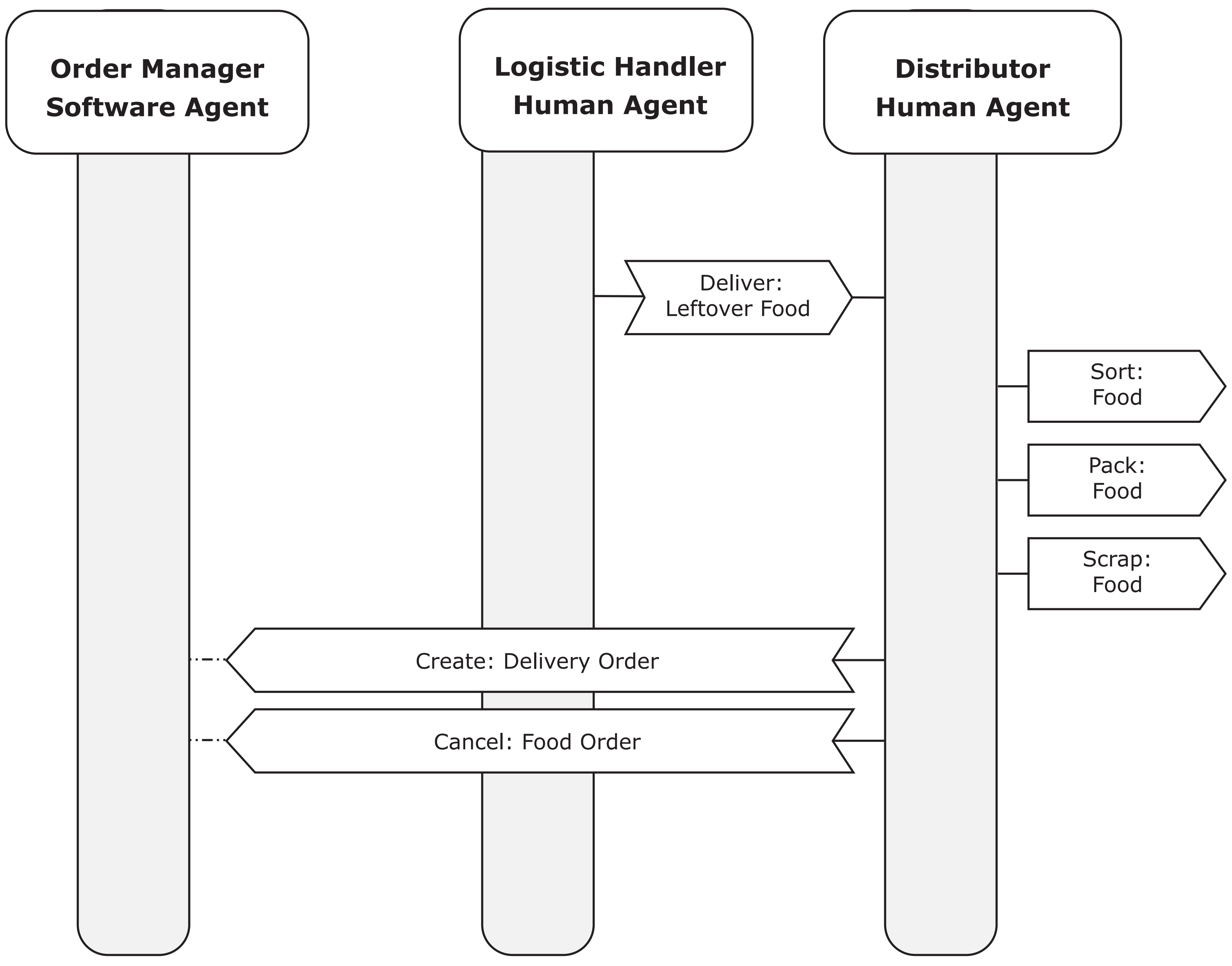
In case of a failure or error, there will be no route calculated. SO the Logistic Handler (Human Agent) has to compile it by himself.



2.3.2. Interaction Model of Handling Distribution

Distributor (Human Agent) gets the Leftover Food from Logistic Handler (Human Agent) who got it from a Shop, then the food will be sorted and packed for the Shelter and when they are ready the Delivery Order will be sent to Order Manager (Software Agent).

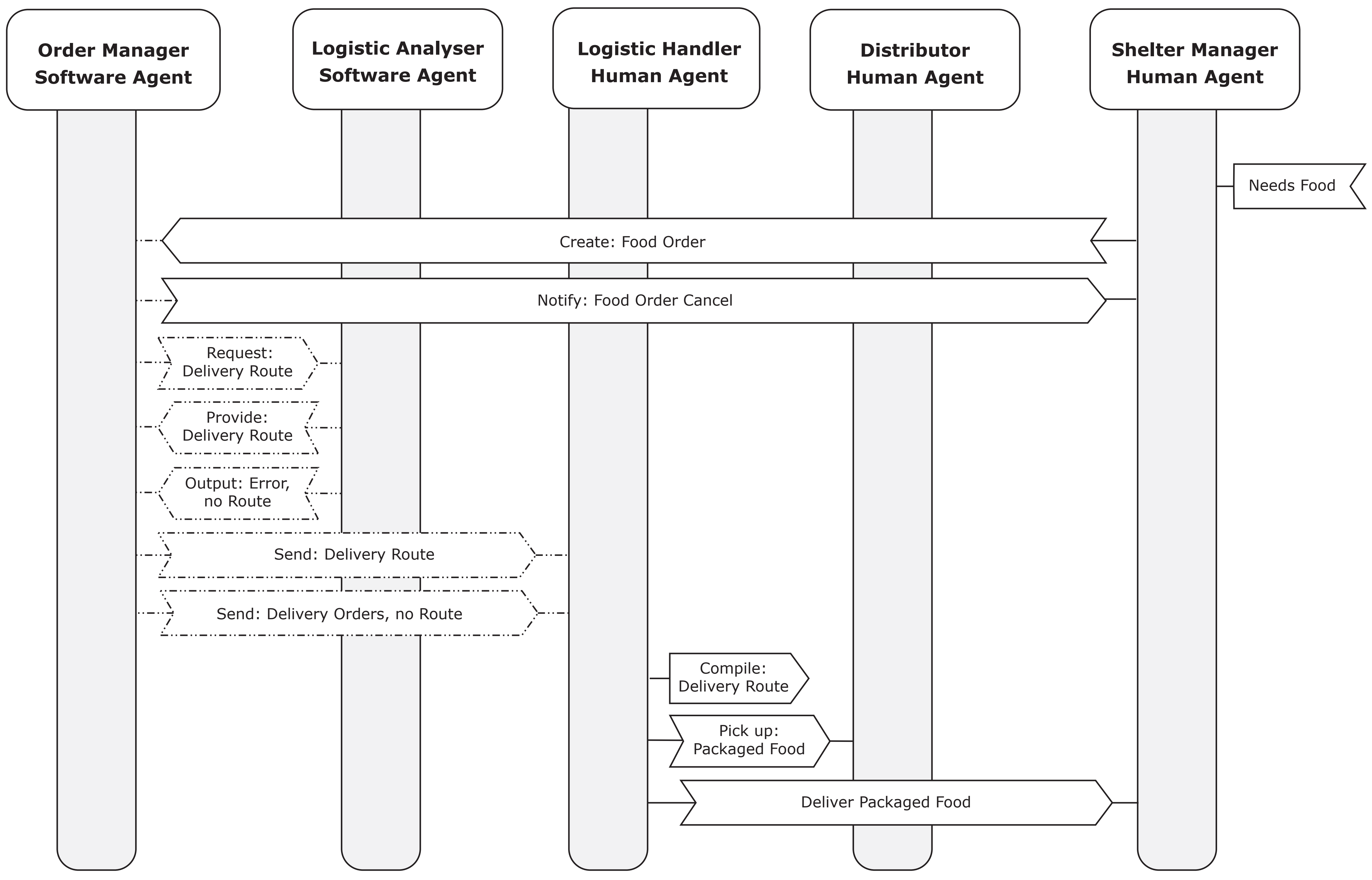
In case of a Scrap Food the Distributor (Human Agent) will cancel the Food Order.



2.3.3. Interaction Model of Handling the Food Delivery

Shelter Manager (Human Agent) needs food and will create Food Order for Order Manager (Software Agent). When the Food will be ready and packed for Delivery to Shelters Order Manager (Software Agent) will request the Logistic Analyser (Software Agent) for Delivery Route and after getting the Delivery Route, Order Manager will sent the Delivery Route to the Logistic Handler (Human Agent) who will pick up Packaged Food from Distributor and deliver it to the Shelter Manager (Human Agent).

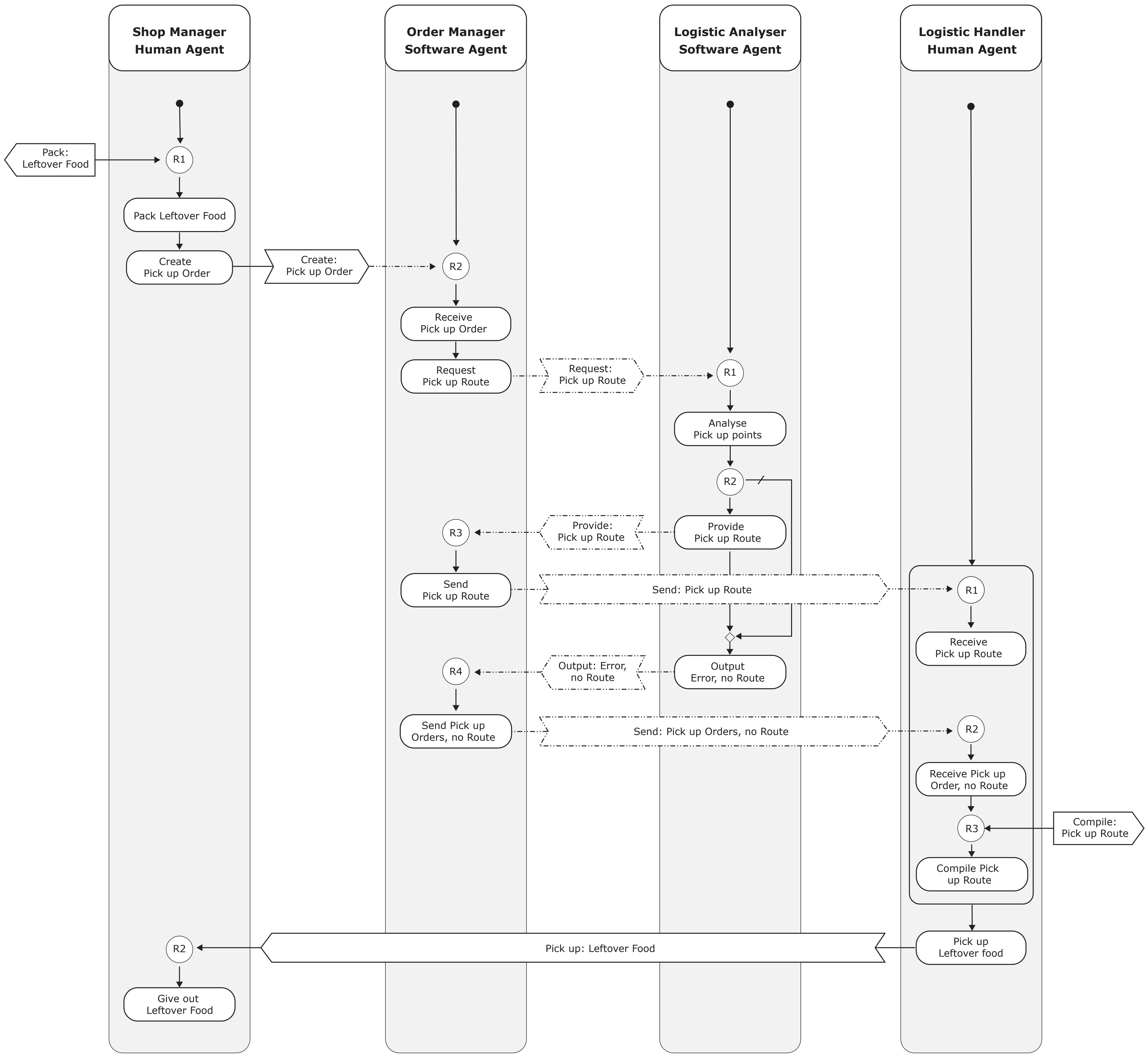
The shelter cannot receive the food and the food order should be cancelled by Distributor.



2.4. Behaviour models

Behaviour model represents more specifically what is happening at each step of the process and what duties of the agents are.

2.4.1. Behaviour Model of Handling Shop Pick up Order



Shop Manager

- R1- makes sure that the leftover food is packed and Pick up Order created
- R2- gives out leftover food

Order Manager

- R2- receives Pick up Order and requests Pick up Route
- R3- sends Pick up Route out
- R4- sends Pick up Route but will not receive any Route

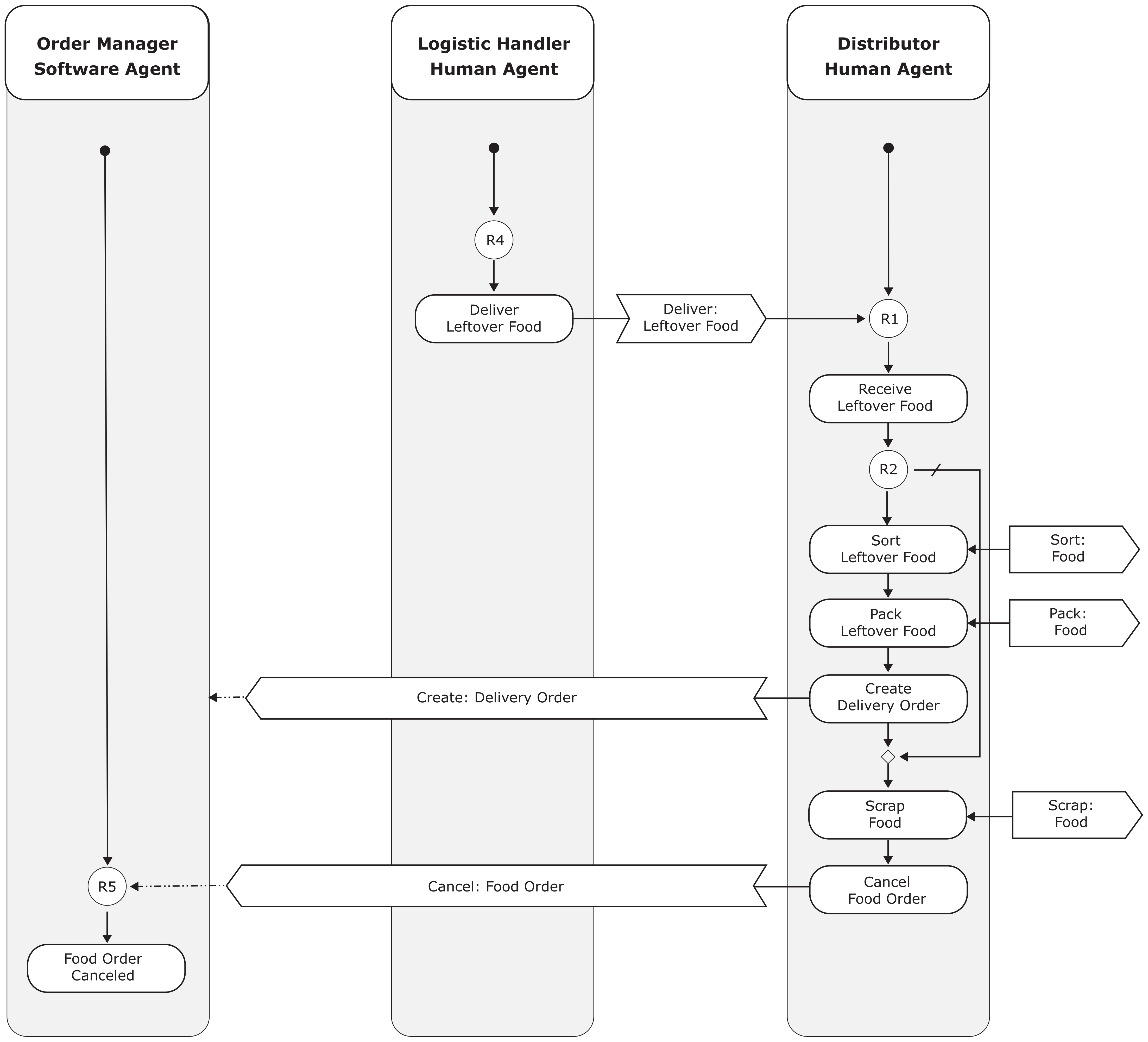
Logistic Analyser

- R1- analyses Pick up points and provides Pick up Route
- R2- provides Pick up Route or in case of error will not

Logistic Handler

- R1- receives and accepts the Pick up Route and picks up packages from the Shop
- R2- receives Pick up Order, but no Route
- R3- compiles new Pick up Route

2.4.2. Behaviour Model of Handling Distribution



Distributor

- R1- receives Leftover Food
- R2- makes sure that the food will be sorted and packed for the shelter and the delivery will be ordered, but in case of Scrap Food, will cancel the Food Order

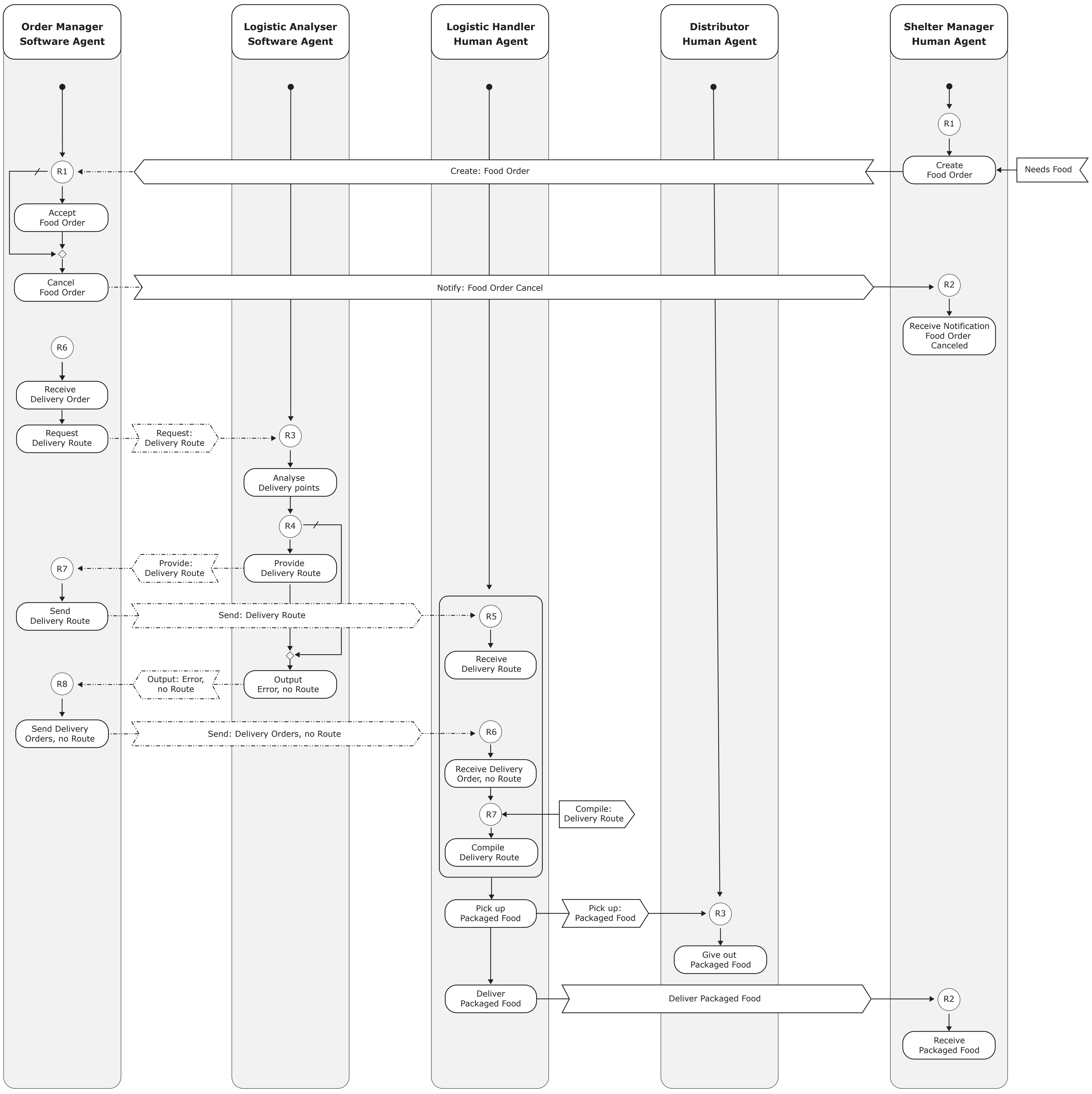
Logistic Handler

- R4- delivers Leftover Food

Order Manager

- R5- in case of Scrap Food, will cancel Food Order

2.4.3. Behaviour Model of Handling the Food Delivery



Shelter Manager

- R1- creates Food order
- R2- receives notification that the Food Order is cancelled or if there are no failures, receive Packaged Food

Distributor

- R3- gives out packaged Food

Logistic Handler

- R5- receives Delivery Route
- R6- receives Delivery Order but no Route
- R7- Compiles own Delivery Route

Logistic Analyser

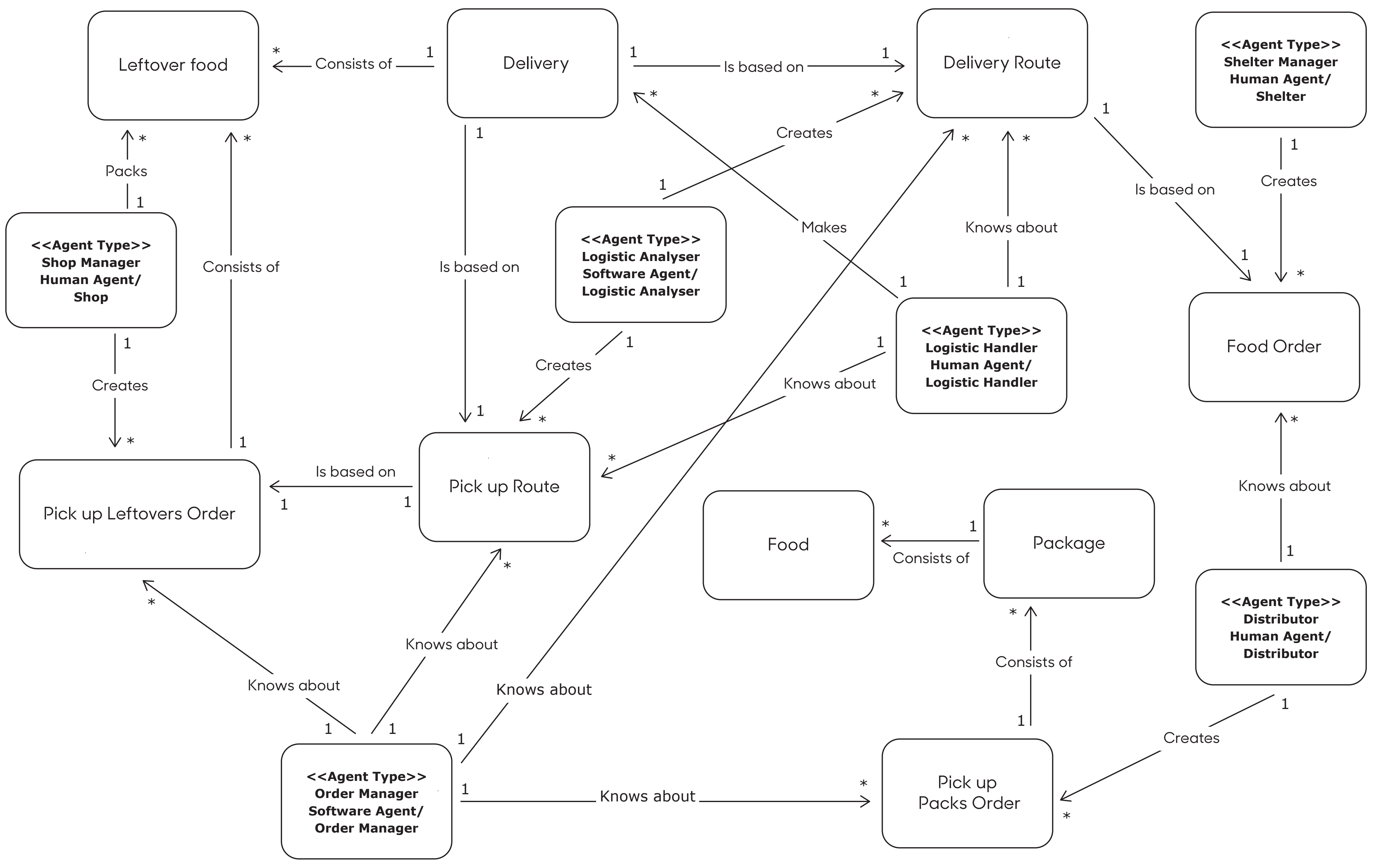
- R3- analyses Delivery points
- R4- provides Delivery Route or in case of an error doesn’t

Order Manager

- R1- accepts Food Order (in case of an error, the Food Order is cancelled)
- R6- receives Delivery Order and requests Delivery Route

2.5. Knowledge model

This model shows what the knowledge of every agent is. Information about themselves and their environment. Only the most relevant attributes are included with the data objects.



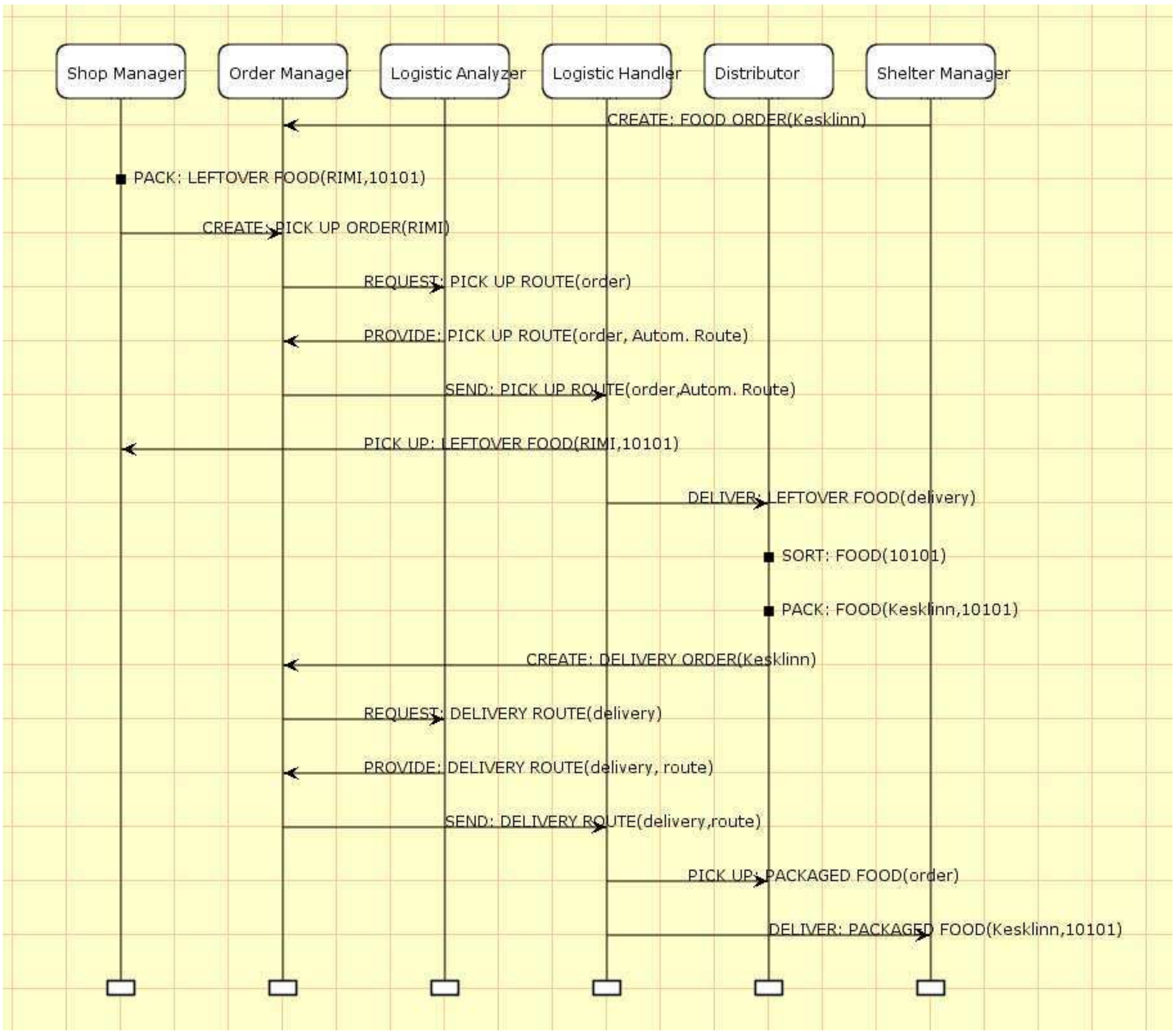
Leftover food
Delivery
Delivery Route
<<Agent Type>> Shelter Manager Human Agent/ Shelter
Consists of
Is based on
Creates
Makes
Knows about
Packs
<<Agent Type>> Shop Manager Human Agent/ Shop
<<Agent Type>> Logistic Analyser Software Agent/ Logistic Analyser
<<Agent Type>> Logistic Handler Human Agent/ Logistic Handler
Food Order
Pick up Leftovers Order
Pick up Route
Food
Package
<<Agent Type>> Distributor Human Agent/ Distributor
<<Agent Type>> Order Manager Software Agent/ Order Manager
Pick up Packs Order

3. Implementation

3.1 CPN Model Validation

3.1.1. Scenario 1

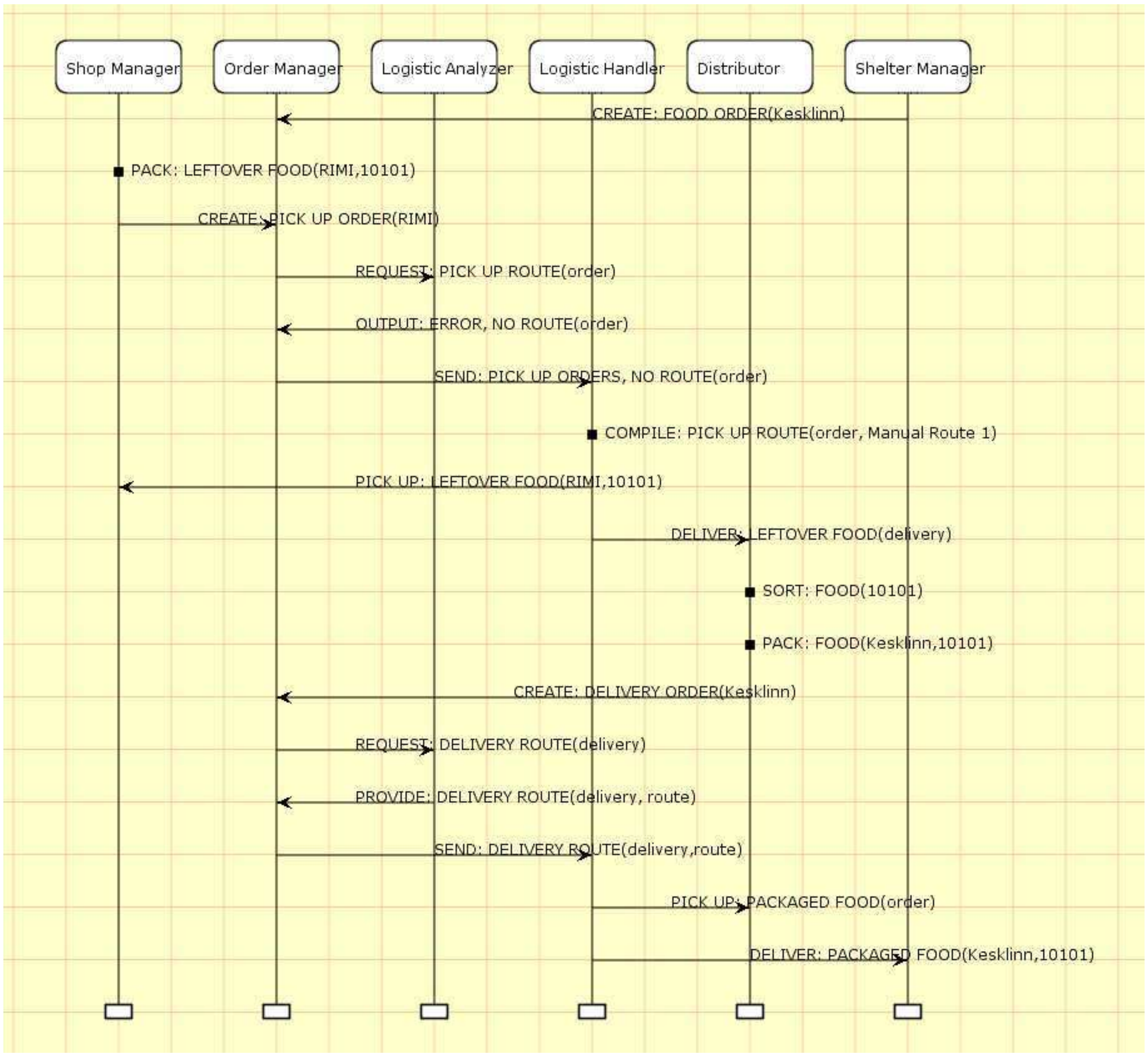
The perfect scenario when all the services are up and running and the food is OK to package and deal. Figure 1 shows perfect interaction flow in the Sequence Diagram.



Shop Manager
Order Manager
Logistic Analyzer
Logistic Handler
Distributor
Shelter Manager
CREATE: FOOD ORDER(Kesklinn)
PACK: LEFTOVER FOOD(RIMI,10101)
CREATE: PICK UP ORDER(RIMI)
REQUEST: PICK UP ROUTE(order)
PROVIDE: PICK UP ROUTE(order, Autom. Route)
SEND: PICK UP ROUTE(order,Autom. Route)
PICK UP: LEFTOVER FOOD(RIMI,10101)
DELIVER: LEFTOVER FOOD(delivery)
SORT: FOOD(10101)
PACK: FOOD(Kesklinn,10101)
CREATE: DELIVERY ORDER(Kesklinn)
REQUEST: DELIVERY ROUTE(delivery)
PROVIDE: DELIVERY ROUTE(delivery, route)
SEND: DELIVERY ROUTE(delivery,route)
PICK UP: PACKAGED FOOD(order)
DELIVER: PACKAGED FOOD(Kesklinn,10101)

3.1.2. Scenario 2

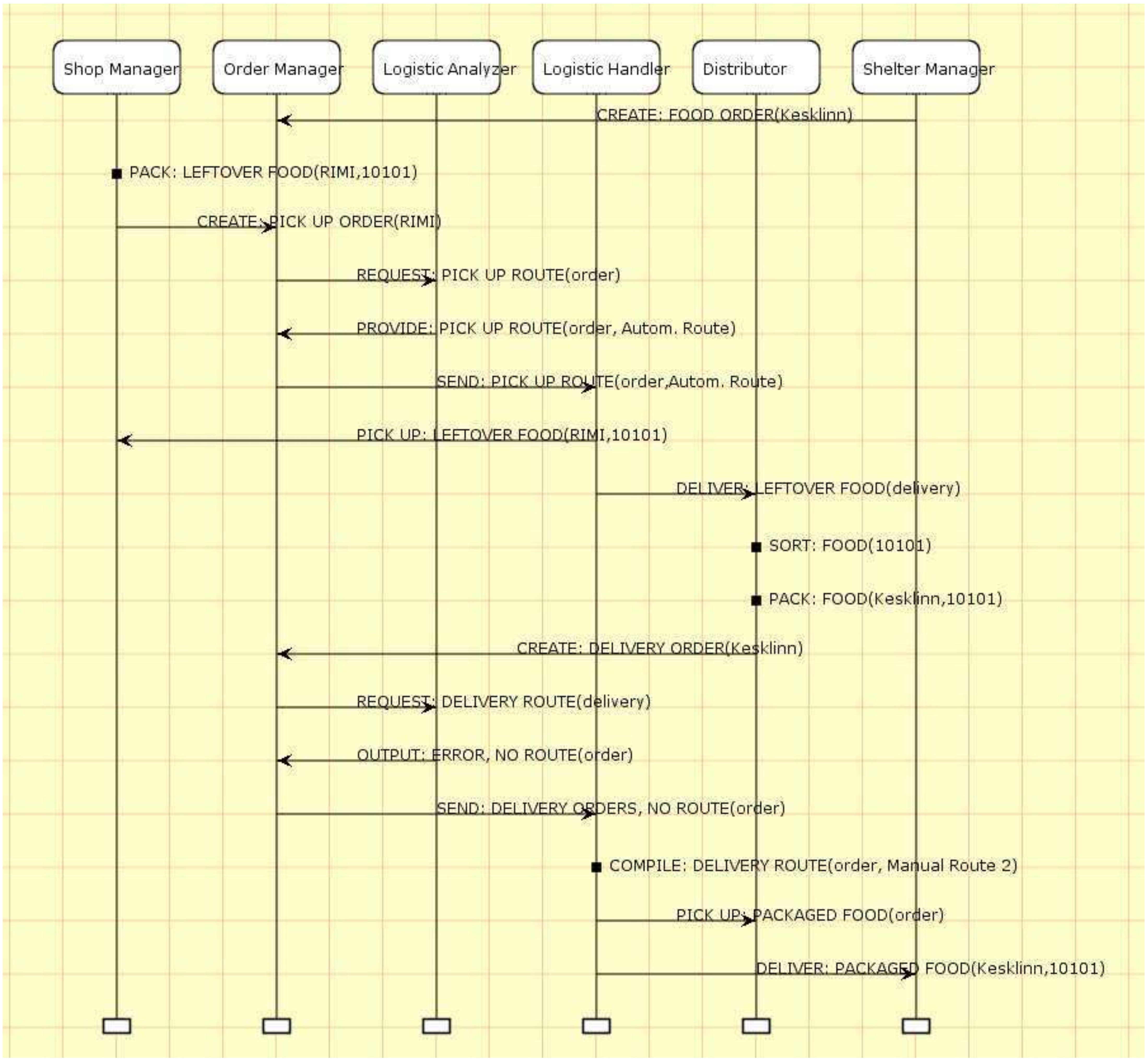
This scenario has a failure in Logistic Analyzer request when asking Pick up Route. Alternative scenario is to output error from Logistic Analyzer and send orders from Order Manager to Logistic Handler without route. If route is not provided, Logistic Handler will compile suitable route itself. Figure 2 shows interaction flow with alternate pick up route calculation in the Sequence Diagram.



Shop Manager
Order Manager
Logistic Analyzer
Logistic Handler
Distributor
Shelter Manager
CREATE: FOOD ORDER(Kesklinn)
PACK: LEFTOVER FOOD(RIMI,10101)
CREATE: PICK UP ORDER(RIMI)
REQUEST: PICK UP ROUTE(order)
OUTPUT: ERROR, NO ROUTE(order)
SEND: PICK UP ORDERS, NO ROUTE(order)
COMPILE: PICK UP ROUTE(order, Manual Route 1)
PICK UP: LEFTOVER FOOD(RIMI,10101)
DELIVER: LEFTOVER FOOD(delivery)
SORT: FOOD(10101)
PACK: FOOD(Kesklinn,10101)
CREATE: DELIVERY ORDER(Kesklinn)
REQUEST: DELIVERY ROUTE(delivery)
PROVIDE: DELIVERY ROUTE(delivery, route)
SEND: DELIVERY ROUTE(delivery,route)
PICK UP: PACKAGED FOOD(order)
DELIVER: PACKAGED FOOD(Kesklinn,10101)

3.1.3. Scenario 3

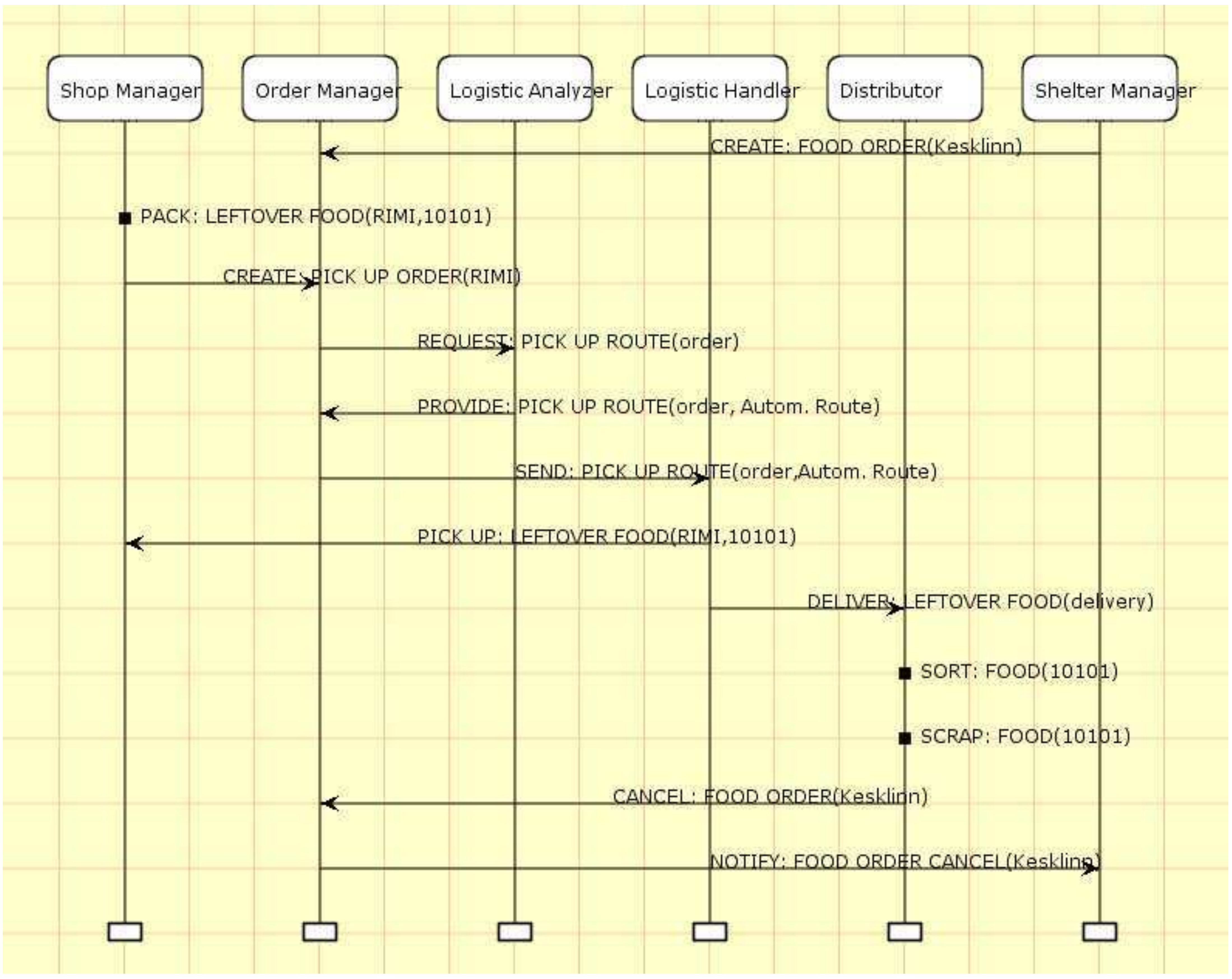
This scenario has a failure in Logistic Analyzer request when asking Delivery Route. Alternative scenario is to output an error from Logistic Analyzer and send orders from Order Manager to Logistic Handler without route. If route is not provided, Logistic Handler will compile suitable route itself. Figure 3 shows interaction flow with alternate delivery route calculation in the Sequence Diagram.



Shop Manager
Order Manager
Logistic Analyzer
Logistic Handler
Distributor
Shelter Manager
CREATE: FOOD ORDER(Kesklinn)
PACK: LEFTOVER FOOD(RIMI,10101)
CREATE: PICK UP ORDER(RIMI)
REQUEST: PICK UP ROUTE(order)
PROVIDE: PICK UP ROUTE(order, Autom. Route)
SEND: PICK UP ROUTE(order,Autom. Route)
PICK UP: LEFTOVER FOOD(RIMI,10101)
DELIVER: LEFTOVER FOOD(delivery)
SORT: FOOD(10101)
PACK: FOOD(Kesklinn,10101)
CREATE: DELIVERY ORDER(Kesklinn)
REQUEST: DELIVERY ROUTE(delivery)
OUTPUT: ERROR, NO ROUTE(order)
SEND: DELIVERY ORDERS, NO ROUTE(order)
COMPILE: DELIVERY ROUTE(order, Manual Route 2)
PICK UP: PACKAGED FOOD(order)
DELIVER: PACKAGED FOOD(Kesklinn,10101)

3.1.4. Scenario 4

This scenario has an alternate flow when food is scrapped, not packed in Distributor. This means that shelter cannot receive the food and the food order should be cancelled by Distributor. Order Manager then notifies Shelter Manager about order cancellation. Figure 3 shows interaction flow with alternate order cancellation flow in the Sequence Diagram.


Shop Manager
Order Manager
Logistic Analyzer
Logistic Handler
Distributor
Shelter Manager
CREATE: FOOD ORDER(Kesklinn)
PACK: LEFTOVER FOOD(RIMI,10101)
CREATE: PICK UP ORDER(RIMI)
REQUEST: PICK UP ROUTE(order)
PROVIDE: PICK UP ROUTE(order, Autom. Route)
SEND: PICK UP ROUTE(order,Autom. Route)
PICK UP: LEFTOVER FOOD(RIMI,10101)
DELIVER: LEFTOVER FOOD(delivery)
SORT: FOOD(10101)
SCRAP: FOOD(10101)
CANCEL: FOOD ORDER(Kesklinn)
NOTIFY: FOOD ORDER CANCEL(Kesklinn)

3.2. Verification

It was possible to calculate State Space when CPN model had only three tokens (1 shop, 1 food, 1 shelter). State Space status was Full.

```
Home Properties

  Home Markings
     Initial Marking is not a home marking

* There is no terminal strongly connected component.

Liveness Properties

  Dead Markings
     [33,42]

* There are dead markings, probably because empty lists implementations.

  Dead Transition Instances
     None

  Live Transition Instances
     None

Fairness Properties

     No infinite occurrence sequences.
```

4. Conclusion

CPN is very good tool to validate AOM methodology, more exactly interaction and behavior models. It enables validating different scenarios quite easily. In our CPN we used lists and timed tokens to simulate more real behavior. At the end it seemed overkill, because State Space was not able to fully calculate with so many tokens.

All different scenarios played out nicely, but still there are Dead Markings. Sadly State Space To Sim did not reveal anything.

If State Space verification is so important, it should be looked more closely in lectures.

State Space report result

```
CPN Tools state space report for:
/cygdrive/C/Users/Silver/Desktop/#Leftover Food Distributing.cpn
Report generated: Mon May 16 23:36:24 2016


 Statistics
------------------------------------------------------------------------

  State Space
     Nodes:  42
     Arcs:   57
     Secs:   27
     Status: Full

  Scc Graph
     Nodes:  42
     Arcs:   57
     Secs:   0


 Boundedness Properties
------------------------------------------------------------------------

  Best Integer Bounds
                             Upper      Lower
     CPN_Diagram'DISTRIBUTOR_food_received 1
                             1          0
     CPN_Diagram'LOGISTIC_ANALYSER 1
                             1          1
     CPN_Diagram'LOGISTIC_HANDLER_1 1
                             1          1
     CPN_Diagram'LOGISTIC_HANDLER_2 1
                             1          1
     CPN_Diagram'LOGISTIC_HANDLER_delivery_order_received 1
                             1          0
     CPN_Diagram'LOGISTIC_HANDLER_pick_up_order_received 1
                             1          0
     CPN_Diagram'ORDER_MANAGER_delivery_order_created 1
                             1          1
     CPN_Diagram'ORDER_MANAGER_food_order_created 1
                             1          0
     CPN_Diagram'ORDER_MANAGER_pick_up_order_created 1
                             1          1
     CPN_Diagram'SHELTER 1   1          0
     CPN_Diagram'SHELTER_package_received 1
                             1          0
     CPN_Diagram'SHOP 1      1          0
     CPN_Diagram'arrived_to_distributor 1
                             1          0
     CPN_Diagram'delivery_orders_sent 1
                             1          0
     CPN_Diagram'delivery_route_ready 1
                             1          0
     CPN_Diagram'delivery_route_requested 1
                             1          0
     CPN_Diagram'dumster 1   1          0
     CPN_Diagram'error_sent_no_delivery_route 1
                             1          0
     CPN_Diagram'error_sent_no_pick_up_route 1
                             1          0
     CPN_Diagram'food_has_been_scrapped 1
                             1          0
     CPN_Diagram'food_order_is_canceled 1
                             1          0
     CPN_Diagram'food_ready_for_deliver 1
                             1          0
     CPN_Diagram'food_ready_to_pick_up 1
                             1          0
     CPN_Diagram'food_sorted 1
                             1          0
     CPN_Diagram'leftover_food 1
                             1          0
     CPN_Diagram'leftover_food_is_collected 1
                             1          1
     CPN_Diagram'leftover_food_packed 1
                             1          0
     CPN_Diagram'next_delivery_order_selected 1
                             1          0
     CPN_Diagram'next_pick_up_order_selected 1
                             1          0
     CPN_Diagram'package_ready_for_shelter 1
                             1          0
     CPN_Diagram'pick_up_orders_sent 1
                             1          0
     CPN_Diagram'pick_up_route_ready 1
                             1          0
     CPN_Diagram'pick_up_route_requested 1
                             1          0
     CPN_Diagram'shelter_has_been_notified 1
                             1          0
     CPN_Diagram'shelter_packages_collected 1
```

```
                             1          1

  Best Upper Multi-set Bounds
     CPN_Diagram'DISTRIBUTOR_food_received 1
                         1`[]++
1`[("RIMI","10101")]
     CPN_Diagram'LOGISTIC_ANALYSER 1
                         1`"Autom. Route"
     CPN_Diagram'LOGISTIC_HANDLER_1 1
                         1`"Manual Route 1"
     CPN_Diagram'LOGISTIC_HANDLER_2 1
                         1`"Manual Route 2"
     CPN_Diagram'LOGISTIC_HANDLER_delivery_order_received 1
                         1`([("Kesklinn","10101")],"Autom. Route")++
1`([("Kesklinn","10101")],"Manual Route 2")
     CPN_Diagram'LOGISTIC_HANDLER_pick_up_order_received 1
                         1`([("RIMI","10101")],"Autom. Route")++
1`([("RIMI","10101")],"Manual Route 1")
     CPN_Diagram'ORDER_MANAGER_delivery_order_created 1
                         1`[]++
1`[("Kesklinn","10101")]
     CPN_Diagram'ORDER_MANAGER_food_order_created 1
                         1`"Kesklinn"
     CPN_Diagram'ORDER_MANAGER_pick_up_order_created 1
                         1`[]++
1`[("RIMI","10101")]
     CPN_Diagram'SHELTER 1
                         1`"Kesklinn"
     CPN_Diagram'SHELTER_package_received 1
                         1`("Kesklinn","10101")
     CPN_Diagram'SHOP 1  1`"RIMI"
     CPN_Diagram'arrived_to_distributor 1
                         1`[]++
1`[("Kesklinn","10101")]
     CPN_Diagram'delivery_orders_sent 1
                         1`[("Kesklinn","10101")]
     CPN_Diagram'delivery_route_ready 1
                         1`([("Kesklinn","10101")],"Autom. Route")
     CPN_Diagram'delivery_route_requested 1
                         1`[("Kesklinn","10101")]
     CPN_Diagram'dumster 1
                         1`"10101"
     CPN_Diagram'error_sent_no_delivery_route 1
                         1`[("Kesklinn","10101")]
     CPN_Diagram'error_sent_no_pick_up_route 1
                         1`[("RIMI","10101")]
     CPN_Diagram'food_has_been_scrapped 1
                         1`"10101"
     CPN_Diagram'food_order_is_canceled 1
                         1`"Kesklinn"
     CPN_Diagram'food_ready_for_deliver 1
                         1`("Kesklinn","10101")
     CPN_Diagram'food_ready_to_pick_up 1
                         1`("RIMI","10101")
     CPN_Diagram'food_sorted 1
                         1`"10101"
     CPN_Diagram'leftover_food 1
                         1`"10101"
     CPN_Diagram'leftover_food_is_collected 1
                         1`[]++
1`[("RIMI","10101")]
     CPN_Diagram'leftover_food_packed 1
                         1`("RIMI","10101")
     CPN_Diagram'next_delivery_order_selected 1
                         1`[]++
1`[("Kesklinn","10101")]
     CPN_Diagram'next_pick_up_order_selected 1
                         1`[]++
1`[("RIMI","10101")]
     CPN_Diagram'package_ready_for_shelter 1
                         1`("Kesklinn","10101")
     CPN_Diagram'pick_up_orders_sent 1
                         1`[("RIMI","10101")]
     CPN_Diagram'pick_up_route_ready 1
                         1`([("RIMI","10101")],"Autom. Route")
     CPN_Diagram'pick_up_route_requested 1
                         1`[("RIMI","10101")]
     CPN_Diagram'shelter_has_been_notified 1
                         1`"Kesklinn"
     CPN_Diagram'shelter_packages_collected 1
                         1`[]++
1`[("Kesklinn","10101")]

  Best Lower Multi-set Bounds
     CPN_Diagram'DISTRIBUTOR_food_received 1
                         empty
     CPN_Diagram'LOGISTIC_ANALYSER 1
                         1`"Autom. Route"
     CPN_Diagram'LOGISTIC_HANDLER_1 1
                         1`"Manual Route 1"
     CPN_Diagram'LOGISTIC_HANDLER_2 1
                         1`"Manual Route 2"
     CPN_Diagram'LOGISTIC_HANDLER_delivery_order_received 1
```

```
                    empty
    CPN_Diagram'LOGISTIC_HANDLER_pick_up_order_received 1
                    empty
    CPN_Diagram'ORDER_MANAGER_delivery_order_created 1
                    empty
    CPN_Diagram'ORDER_MANAGER_food_order_created 1
                    empty
    CPN_Diagram'ORDER_MANAGER_pick_up_order_created 1
                    empty
    CPN_Diagram'SHELTER 1
                    empty
    CPN_Diagram'SHELTER_package_received 1
                    empty
    CPN_Diagram'SHOP 1  empty
    CPN_Diagram'arrived_to_distributor 1
                    empty
    CPN_Diagram'delivery_orders_sent 1
                    empty
    CPN_Diagram'delivery_route_ready 1
                    empty
    CPN_Diagram'delivery_route_requested 1
                    empty
    CPN_Diagram'dumster 1
                    empty
    CPN_Diagram'error_sent_no_delivery_route 1
                    empty
    CPN_Diagram'error_sent_no_pick_up_route 1
                    empty
    CPN_Diagram'food_has_been_scrapped 1
                    empty
    CPN_Diagram'food_order_is_canceled 1
                    empty
    CPN_Diagram'food_ready_for_deliver 1
                    empty
    CPN_Diagram'food_ready_to_pick_up 1
                    empty
    CPN_Diagram'food_sorted 1
                    empty
    CPN_Diagram'leftover_food 1
                    empty
    CPN_Diagram'leftover_food_is_collected 1
                    empty
    CPN_Diagram'leftover_food_packed 1
                    empty
    CPN_Diagram'next_delivery_order_selected 1
                    empty
    CPN_Diagram'next_pick_up_order_selected 1
                    empty
    CPN_Diagram'package_ready_for_shelter 1
                    empty
    CPN_Diagram'pick_up_orders_sent 1
                    empty
    CPN_Diagram'pick_up_route_ready 1
                    empty
    CPN_Diagram'pick_up_route_requested 1
                    empty
    CPN_Diagram'shelter_has_been_notified 1
                    empty
    CPN_Diagram'shelter_packages_collected 1
                    empty


 Home Properties
------------------------------------------------------------------------

  Home Markings
    Initial Marking is not a home marking


 Liveness Properties
------------------------------------------------------------------------

  Dead Markings
    [33,42]

  Dead Transition Instances
    None

  Live Transition Instances
    None


 Fairness Properties
------------------------------------------------------------------------
    No infinite occurrence sequences.
```

5. References

Sterling, Leon S., and Kuldar Taveter. The Art of Agent-Oriented Modeling (2009). The MIT Press

TALLINN UNIVERSITY OF TECHNOLOGY

Erasmus Learning Agreement

Agent-Oriented Modelling and Multiagent Systems
Final Project Report

in the
Faculty of Information Technology
Department of Informatics

May 2016

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The idea behind our mini-project has come from the experience of all Erasmus students. Erasmus students are student who decided to go abroad, hosted by other universities, to take exams. This experience is ruled by some documents that must be filled in and validated by three entities that are: student, a employer of host university (usually named host university coordinator) and the coordinator for the home university.

The main document of the Erasmus Process is the *"Learning Agreement"*, that is a document where the student must write down the exams that he intends to do abroad and the exams (chosen from his study plan) that he wants that home university validates after his come back.

Now this process is not so easy and the student often has to change this document during the first weeks of his mobility. In this way in the first weeks the Erasmus student is forced to spend more time to handle bureaucratic stuff than study for his exams.

Our idea is to get this process easier, to help student and other involved entities to get the agreement faster, adding on some features that can be handled by software agents. In this way the whole process becomes faster and the Erasmus Experience can be lived better by the student.

Model multi-agent system is the opportunity that let students to have the possibility to require a list of exams given by the host university that should be fit with his study plan. In addition the system can improve the communication between students, host and home Erasmus coordinator as well as teachers.

We have also decided to handle with an agent-oriented model the end of Erasmus period, getting automatic the submit of final grade to the home university, included the conversion in a different system grade.

In this report we are going to explain in the following two chapters our Agent-Oriented System for Learning Agreement Process on two different layers:

- Motivation Layer
- System Design Layer

The first one highlights the goals of the different roles involved in the system and shows the domain of the system. We use for this purpose the following models:

- Goal Model
- Role Models
- Organization Model
- Domain Model

The second one refers to the design of the system, showing the interaction and the behaviour of the different agents that operate in the system. The models that land in this layer are:

- Agent models
- Acquaintance model
- knowledge model
- Interaction Models
- Behaviour Models

The fourth chapter shows our CPN models we have used to simulate and validate our Multi-agent system.

Chapter 2

Motivation Layer

As written in the previous one, this chapter contains the following models:

1. Goal Model
2. Role Models
3. Organization Model
4. Domain Model

2.1 Goal Model

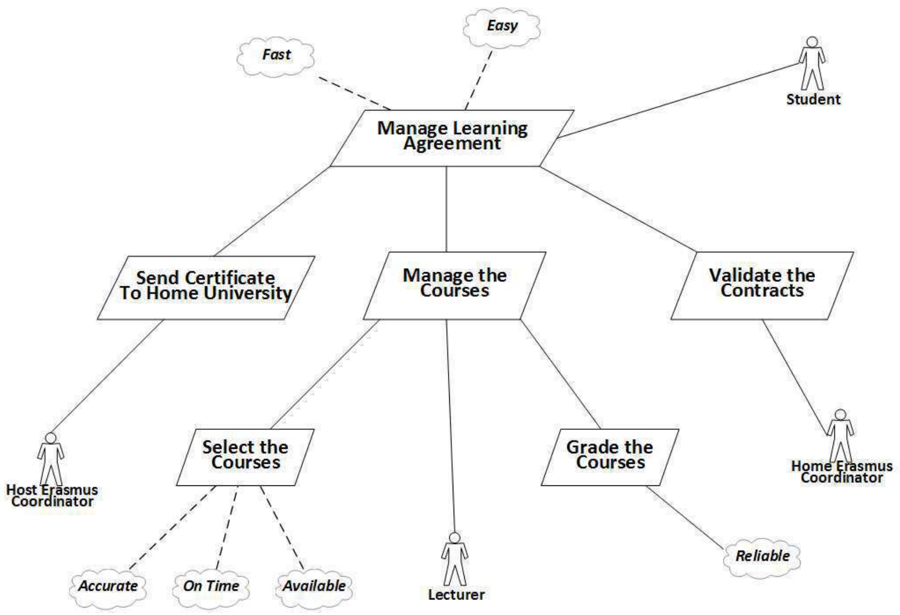


Figure 2.1: Goal Model

The goal model describes the goals and it connects them to specific roles. The model also connects the goals with specific quality requirements that the system is trying to achieve. The goals are described hierarchically and the goal model for current project is described on the figure 2.1

The main goal of the our system stated as *"Management of Learning Agreements"*, with the emphasis on providing easy and fast service for all students. In order to achieve this goal, we have defined some sub-goals. The Sub goals are: *Manage the courses*, *Validate the contracts*, *Send Final Certificate.*

2.2 Role Models

The following topic includes a short overview of different roles. The purpose of the role model is to describe different roles in the system. The role models include: *role names*, *descriptions*, *responsibilities* and *constraints.*

There are four different roles in our project:

- Student;
- Lecturer;
- Home Erasmus Coordinator;
- Host Department Coordinator.

The role models show the different roles implemented in the Erasmus Learning Agreement system. These roles are played by humans as well as software agents.

The roles are described in detail below:

TABLE 2.1: Role Model - Student

Role Name	Student
Description	The Role of the students is to propose the Learning Agreement following the instruction by the home university
Responsibilities	- Propose a list of host university courses that he wants to do - Propose a list of home university courses that he wants that Home University validate - Following Contract Rules - Sending questions - Signing documents - Listening Advice - Following Coordinators' instructions - Agree with the Erasmus contract rules - Following the courses - Having appointment with the coordinator - Apply registration in the host university - Enter the start of the Erasmus Experience - Enter the end of the Erasmus Experience - Submit Erasmus Period
Constraints	- Unexpected changes in the Learning Agreement must be submitted quickly - Information submitted must be correct

TABLE 2.2: Role Model - Lecturer

Role Name	Lecturer
Description	The role of the lecturer is to suggest available courses and grade the students
Responsibilities	- Enter the grades - Submit the grades - Evaluating students - Receiving questions - Entering answers - Sending the answer - Give course advice - Submit course advice - Submit syllabus
Constraints	- The list of available course must be available on time without error - Grades should be evaluated correctly - Information to student must be reliable

Table 2.3: Role Model - Host Department Coordinator

Role Name	Host Department Coordinator
Description	The role of the host department coordinator is to validate the Learning Agreement submitted by the student and accepted by the sending university
Responsibilities	- Check the Learning Agreement - Set the courses - Connecting the lecturers about current course list - Evaluating student suitability - Giving information about courses to Home University - Receive the grades - Enter the grades in the certificate - Sending the certificate
Constraints	- Information to student must be available and easy to get - Changes in timetables must be communicated quickly - Student Registration must be sent to home university immediately

Table 2.4: Home Erasmus Coordinator

Role Name	Home Erasmus Coordinator
Description	The role of the home Erasmus coordinator is to manage and evaluate the Learning Agreement proposed by the student
Responsibilities	- Informing the students about Erasmus tasks - Validate the Learning Agreement - Print the contract - Check the compatibility between the host and home courses -Inform students about the Erasmus Bureaucracy - Receiving certificate from host university
Constraints	- The Learning Agreement must be validate quickly - After the submit of the Learning Agreement the contract must be printed immediately - The information about Erasmus project and host university must be up to date

Organization Model

The organization model is a part of the interaction analysis of the agent organization. It shows the relationships between the roles in the agent organization system.

As written previously, our system is acted by 4 different roles. In this system we can find the relationship shown in figure 2.2.

2.3 Domain Model

The domain model explains the knowledge within the system and its relationship to the roles. The domain model for our project is shown on Figure 2.3. There are seven

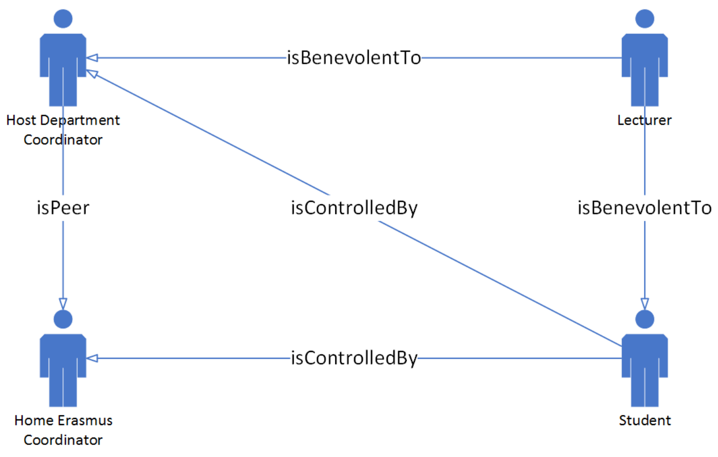


FIGURE 2.2: Organization Model

different domain entities: Answers, Certificate, Learning Agreement, Courses available, Grades, Project/Exams, Question.

In the domain model Students ask questions to Home Eramus Coordinator, receive answers from Home Erasmus Coordinator and grades from Lecturers. Lecturer evaluates submitted Project/Exams by Students and they give grades to the students. Also Lecturer submits courses which is available to the Host Department Coordinator.

Learning Agreement is submitted by student, accepted by Home Erasmus Coordinator and validated by Host Department Coordinator.

Host Department Coordinator submits final certificate and Home Erasmus Coordinator receives it.

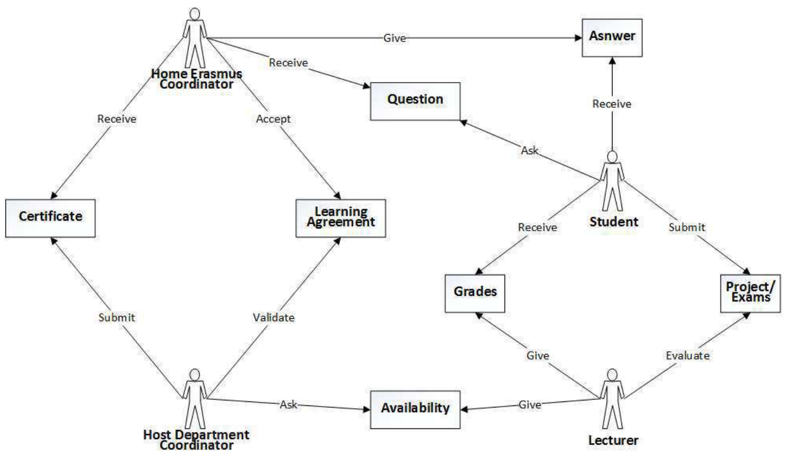


FIGURE 2.3: Domain Model

Chapter 3

System Design Layer

The *System Design Layer* includes the following models:
- Agent models
- Acquaintance model
- Knowledge model
- Interaction Models
- Behaviour Models

3.1 Agent Models

Table 3.1: Student Agent

Agent Name	Student Agent
Description	The Role of the Student is to create L.A without problem
Role	Student
Responsibilities	- Propose a list of host university course that he wants to attend - Propose a list of home university courses that he wants to validate - Following contract rules - Formulate Questions - Signing documents - Listening advice - Following Coordinators' instruction - Following the courses - Having appointment with the coordinator - Enter the start date of the Erasmus - Enter the finish date of the Erasmus Experience for students - Apply registration in host university

Table 3.2: Teacher Agent

Agent Name	Teacher Agent
Description	The Role of the Teacher is to send availability for the courses and put the grade to the exams
Role	Lecturer
Responsibilities	- Enter the grades - Evaluating the Students - Read Questions - Entering Answers

Table 3.3: Home coordinator Agent

Agent Name	Home Coordinator Agent
Description	The Role of the Home Coordinator Agent is to help student to develop The Learning Agreement Successfully
Role	Home Erasmus Coordinator
Responsibilities	- Informing the students about Erasmus tasks - Inform the students about the Erasmus bureaucracy - Check the compatibility between the host and home courses

Table 3.4: Host Erasmus Coordinator

Agent Name	Host Coordinator Agent
Description	The Role of the Host Coordinator Agent is to help the students to choose the right exams and send certificate to Home University
Role	Host Department Coordinator
Responsibilities	- Check the Learning Agreement - Set the courses - Giving information about courses to Home University - Evaluating student about suitability

Table 3.5: My caption

Agent Name	Student Software Agent
Description	The role of the Student Software Agent is to create an interface in the peer-to-peer system for the Student
Roles	Student
Responsibilities	- Submit the Learning Agreement - Sending Questions - Agree with the Erasmus contract rules - Submit Erasmus Period

Table 3.6: Teacher Software Agent

Agent Name	Teacher Software Agent
Description	The role of the Teacher Software Agent is to create an interface in the peer-to-peer system for the Lecturer
Roles	Lecturer
Responsibilities	- Submit syllabus - Submit course advice - Sending the Answer - Receiving Questions - Submit the grades

Table 3.7: Host Coordinator Software Agent

Agent Name	Host Coordinator Software Agent
Description	The role of the Host Coordinator Software Agent is to create an interface in the peer-to-peer system for the Host Department Coordinator
Roles	Host department coordinator
Responsibilities	- Connecting the lecturers about current courses list - Enter the grade in the certificate - Receive the grades

Table 3.8: Home Coordinator Software Agent

Agent Name	Home Coordinator Software Agent
Description	The role of the Home Coordinator Software Agent is to create an interface in the peer-to-peer system for the Home Erasmus coordinator
Roles	Home Erasmus Coordinator
Responsibilities	- Validate the Learning Agreement - Print the contract - Receiving the certificate from the host university

3.2 Acquaintance Model

The aim of the acquaintance model is to describe the interactions between agents of our system. All agents have their software interface agents and they are interacted each other. The Acquaintance model for our project is displayed in the figure 3.1

As we can see in the model, the interactions between human agents are limited and most of the communications involve software agents.

3.3 Interaction Models

3.3.1 Require Availability for the Exams

This interaction involves the Host University and the Teachers. The purpose of this interaction is to create the list of exams that the Erasmus student can choose to take

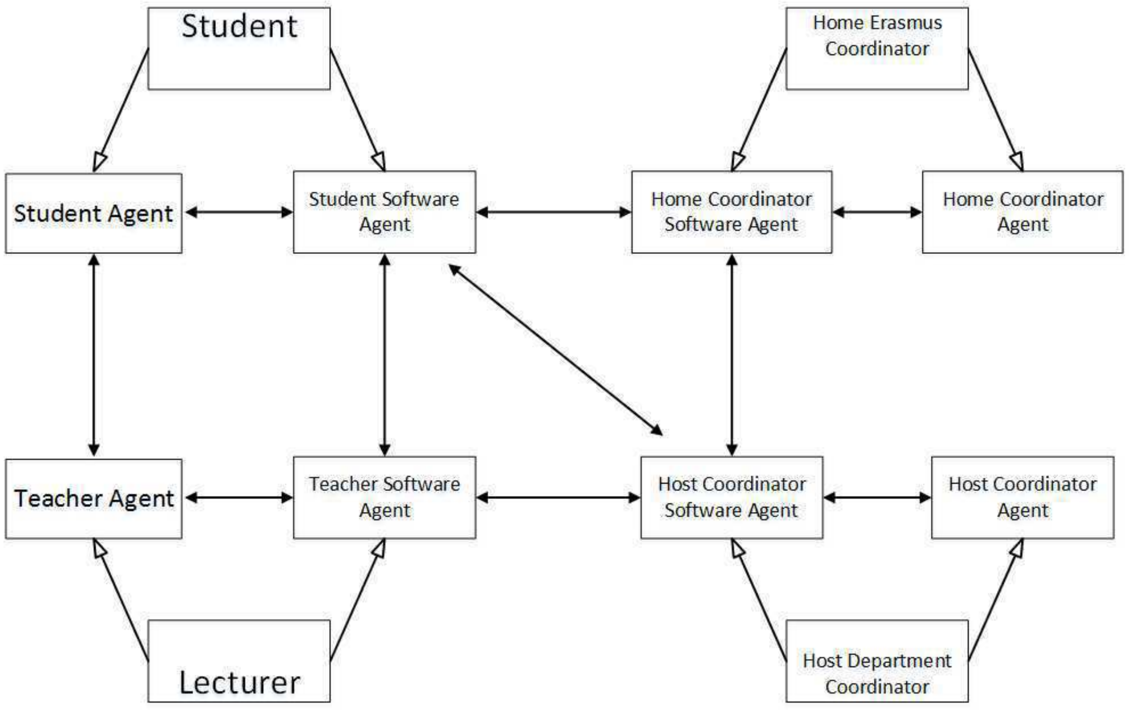


FIGURE 3.1: Acquaintance Model

once arrived in the host University.

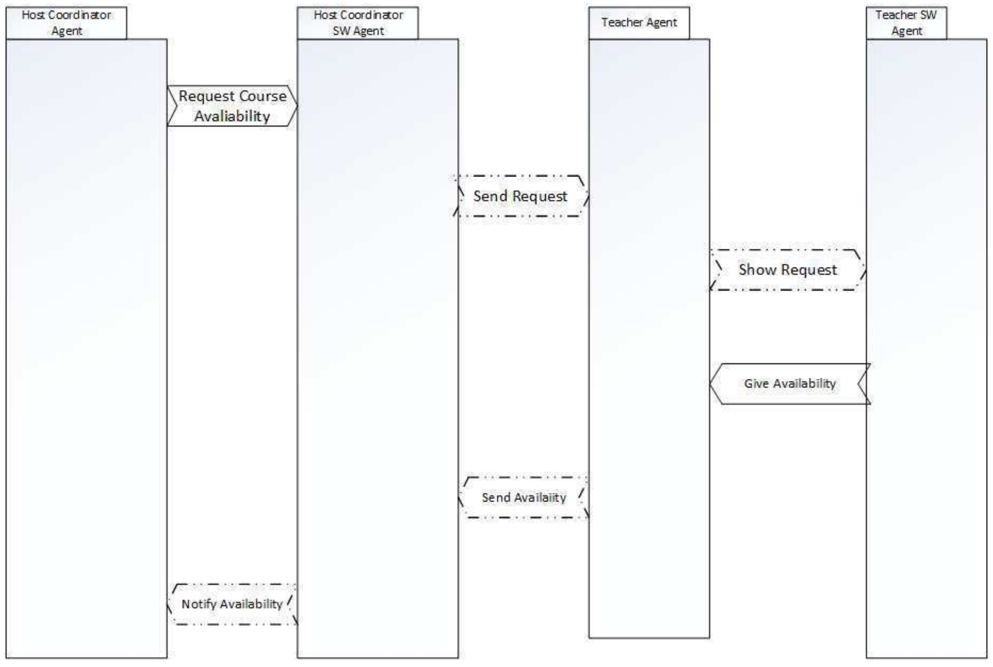


FIGURE 3.2: Interaction Model - Available Exam List

As shown in the figure 3.2, the interaction between the human agents *"Host Coordinator Agent"* and *"Student Agent"* is linked by other intermediary interaction between the human agents and their software agents as well as the two software agents.

3.3.2 Answer-Question Student-Teacher

This interaction involves the students and the teachers agents. The purpose of this interaction is to give to the student the opportunity to have a fast and reliable way to communicate to teacher to ask the main questions regarding the exams that he wants to take.

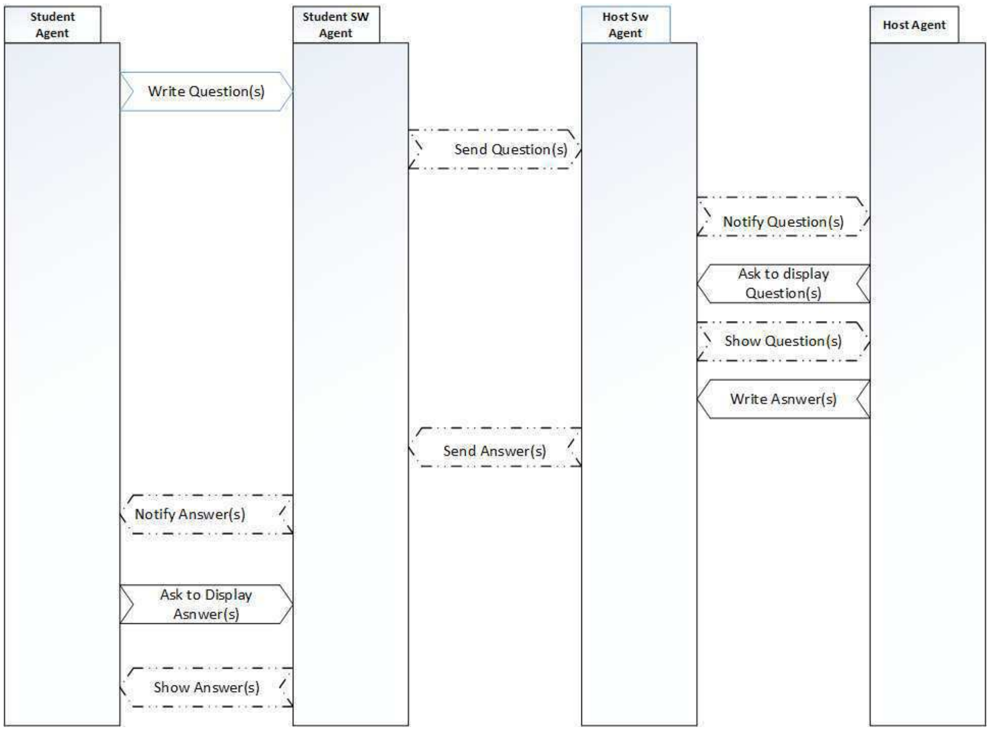


FIGURE 3.3: Interaction Model - Answer/Question Teacher-Student

Also in this case, the communication between the two human entities is mediated by software agents inside.(figure 3.3)

3.3.3 Learning Agreement Process

This interaction is the main important for the systems because it regards the main purpose of the project. Since this interaction involves many agents, we decide to split this interaction model in two models to let it to be more readable.

In the first one we show all the interactions that occur when a student has to require the Learning Agreement. The Process starts by the Student Software Agent when he received the confirm that the student is definitively an Erasmus Student. So when the period arrives, it asks to student to complete the Learning Agreement and he asks, through the software agent, to see the available courses that he can choose from the host university. The request is forwarded to the Host Coordinator Software Agent that show the request to the host coordinator agent that launch the process. At the end the list of the courses is shown to the Student. (figure 3.4)

In the second one there is the submitting of learning agreement. The student fill in the Learning Agreement adding on the exams that he wants to take. The exams are evaluated by the home university according to his policy and study plan and the outcome is sent back to student.

If the Learning Agreement is valid, the student forward the Learning Agreement to the Host University that sign and validate the contract.

Also in these interactions the communications are mediated by the software agents. (figure 3.5)

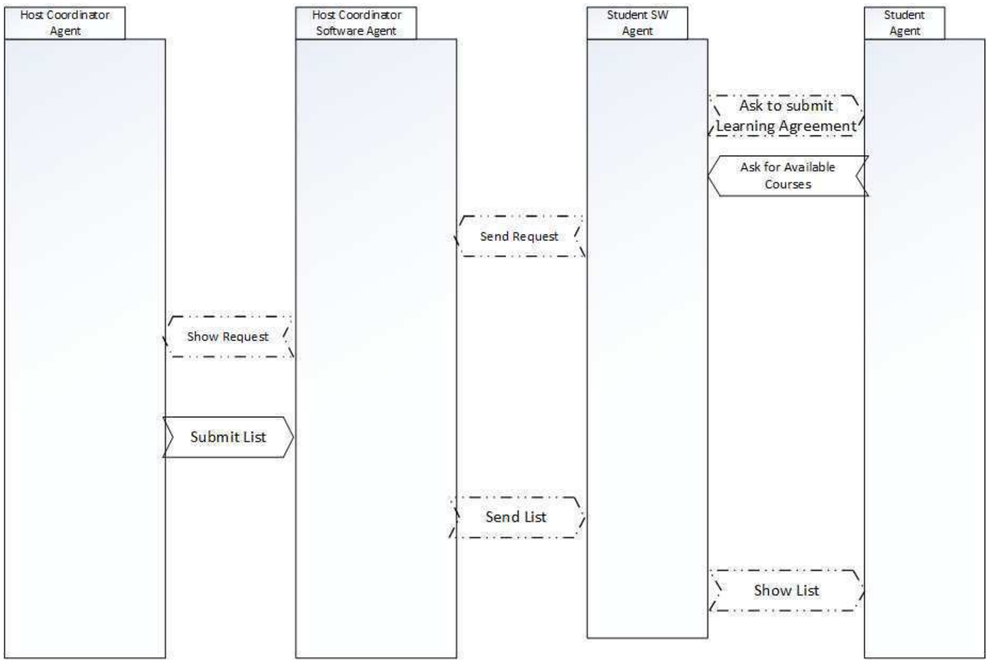


FIGURE 3.4: Interaction Model - Learning Agreement Request

3.3.4 Sending Certificate according the exams taken

The final interaction Model involves all agents at the end of mobility period for the student. Its purpose is to send to home universities the final certificate that show to home university the result taken by the student. In this interaction we have added the possibility to have the conversation between different *Grading Systems* automatically though the software agents. Obviously each university can kept its grading system and conversion rules. It is only necessary that Home university send to host university this rules to let it the possibility to send final grades already converted.

The Scenario described above is shown in the figure 3.6

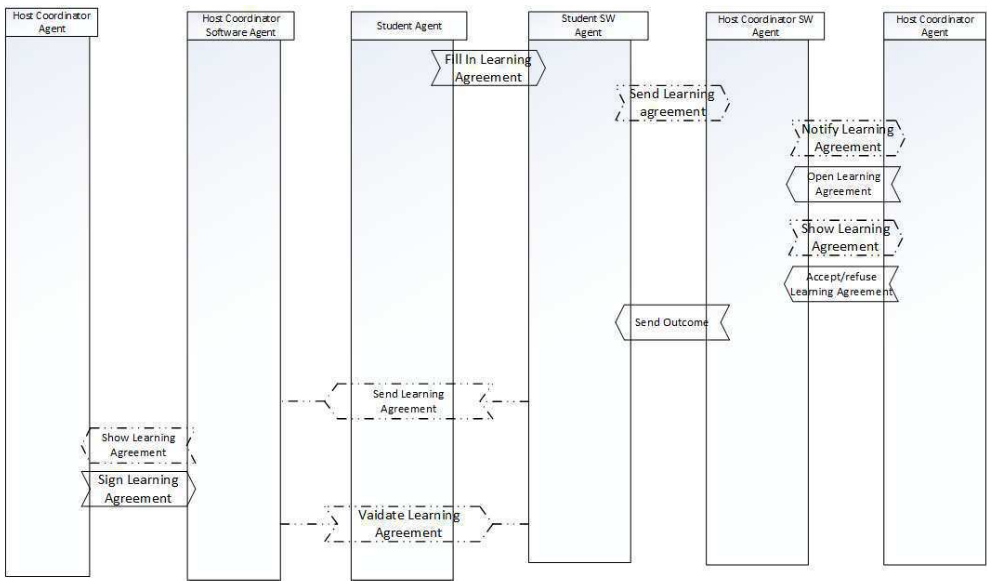


FIGURE 3.5: Interaction Model - Learning Agreement Submit

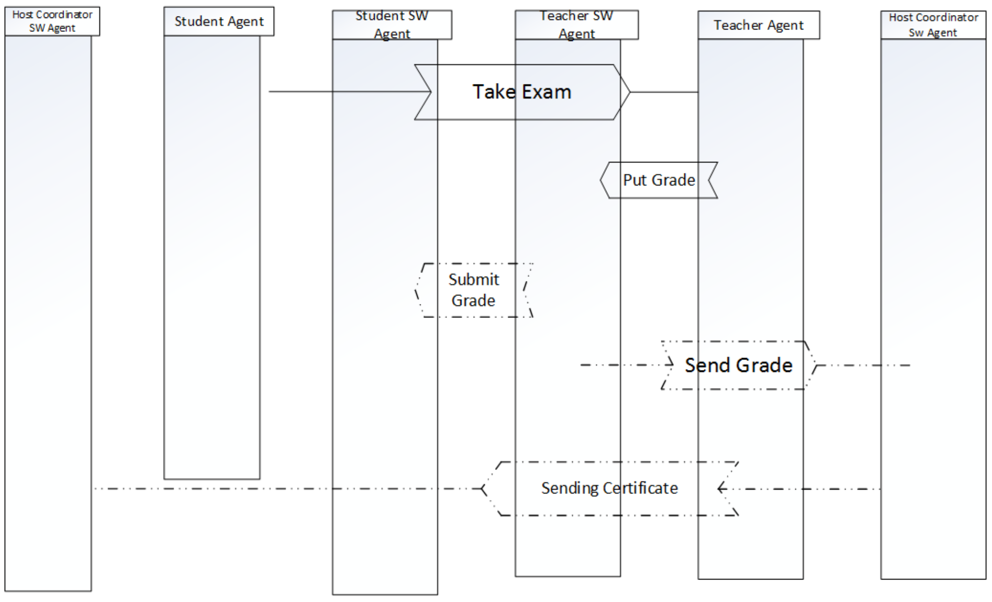


FIGURE 3.6: Interaction Model - Final Certificate

3.4 Knowledge Model

This kind of model represents the knowledge requirements for the agents. The knowledge model is detailed version of our domain model. We can see more extended view of domain entities here.

In the Figure 3.7 we can see agents and goals interact with each other and the kind of "messages“ that they exchange themselves.

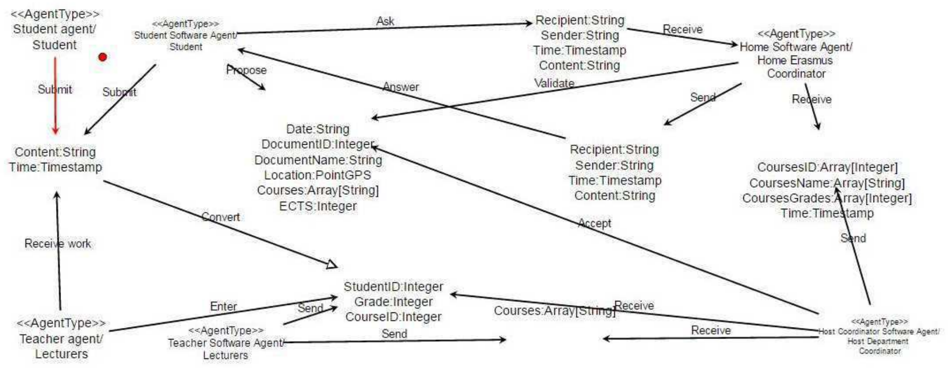


FIGURE 3.7: knowledge Model

3.5 Behaviour Models

The Behaviour model illustrates loop inside the agent types and the activities and actions between the agent types in our system design. We can create rules and events in the agent type. There are several behaviour models in the system but we are just going to describe the primary ones following the scenarios described by the interaction models. So there are several behaviour models that describe our system and show the behaviour and the triggered communication between the different agents.

3.5.1 Behaviour Model - Available Course List

As written in the previous sections, the first requirement to have a fast and safe Learning Agreement Process is to give to the student the possibility to refers to a list of available courses. The first behaviour model (figure 3.8) shows this process.

3.5.2 Behaviour Model - Learning Agreement Process

As done for the interaction models, also for the behaviour one we decide to split the behaviour model regarding the main goal of the project, that is the Learning Agreement Management.

In the first one (figure 3.9) we show how the student software Agent behave in this phase and his interaction with the student (human) agent and the host coordinator Software Agent. In this phase there is the request to have an available courses list.

In the second one (figure 3.10) we focus on the behaviour of the Student Software Agent and Home Coordinator Software Agent. The first one submit the Learning Agreement filled in by the student and send it to home coordinator software Agent. This Agent check the exams following home university rules and decide if the exams are valid or

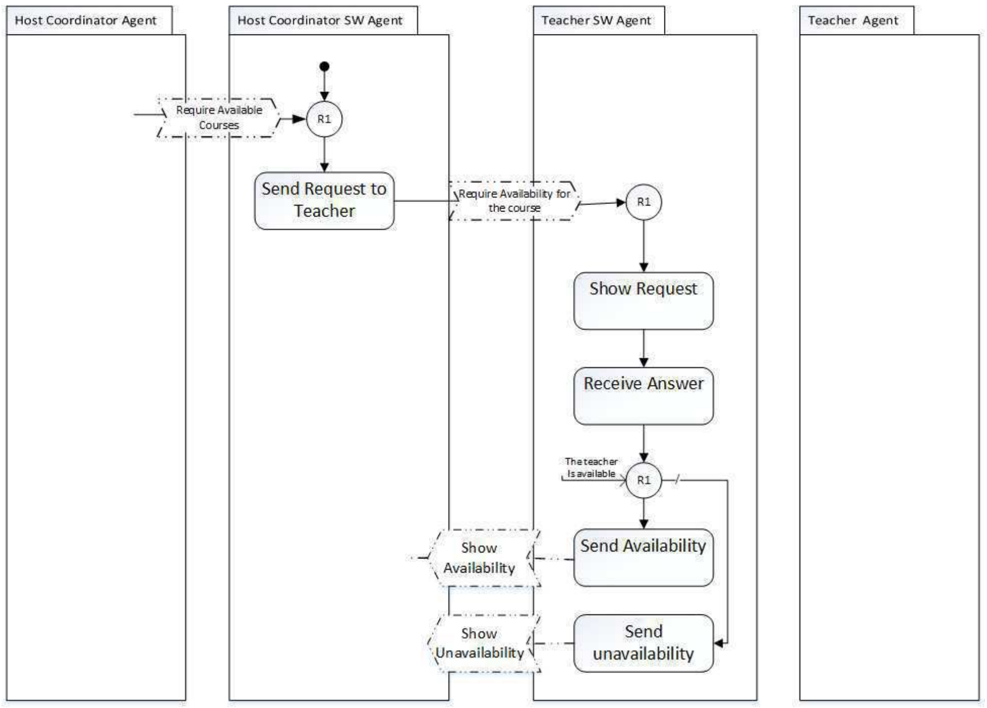


FIGURE 3.8: Behaviour Model - Available Courses

not. At the end he evaluated the whole learning agreement and send the outcome to the student software agent. If the learning agreement received is good, the student software agent send it to the host university. If the learning agreement is not valid, the student must start the initial process again and fill in another Learning Agreement.

3.5.3 Behaviour Model - Sending final Certificate

This behaviour Model is particularly interesting because we add on here the possibility to compute the grade according to home university grading systems. In addition we have shown the possibility for the student to take the exam again if he has failed the first one, according the rules that there are available dates to repeat it.

Since the complexity of the model, we decide again here to split the model in two ones.

The first one (3.11) shows the behaviour of the student software agent that register the exam sessions, receive the grades by the teacher software agent, and notify all events to the student (human) agent (included the possibility or not to do again the exam in the failure case).

The second one (3.12) focuses on the behaviour of the Host University Software Agent that receives grades from the teachers and convert these grades according the home university grading system. In this model we show above all the communication between the host and home university regarding the request of the grading system by the home university.

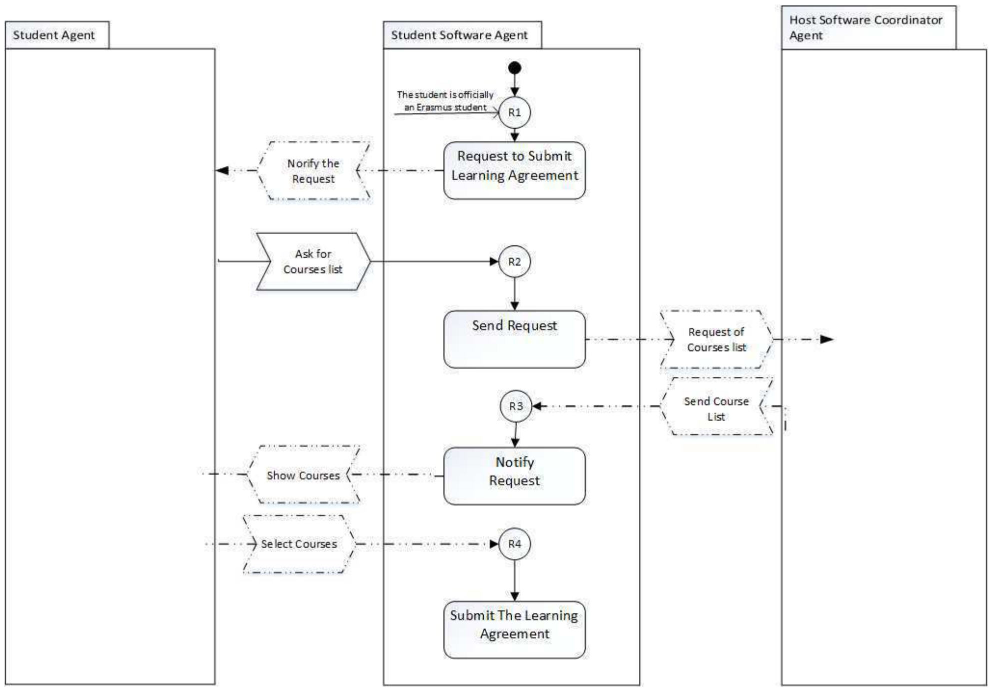


FIGURE 3.9: Behaviour Model - Learning Agreement Process - Request

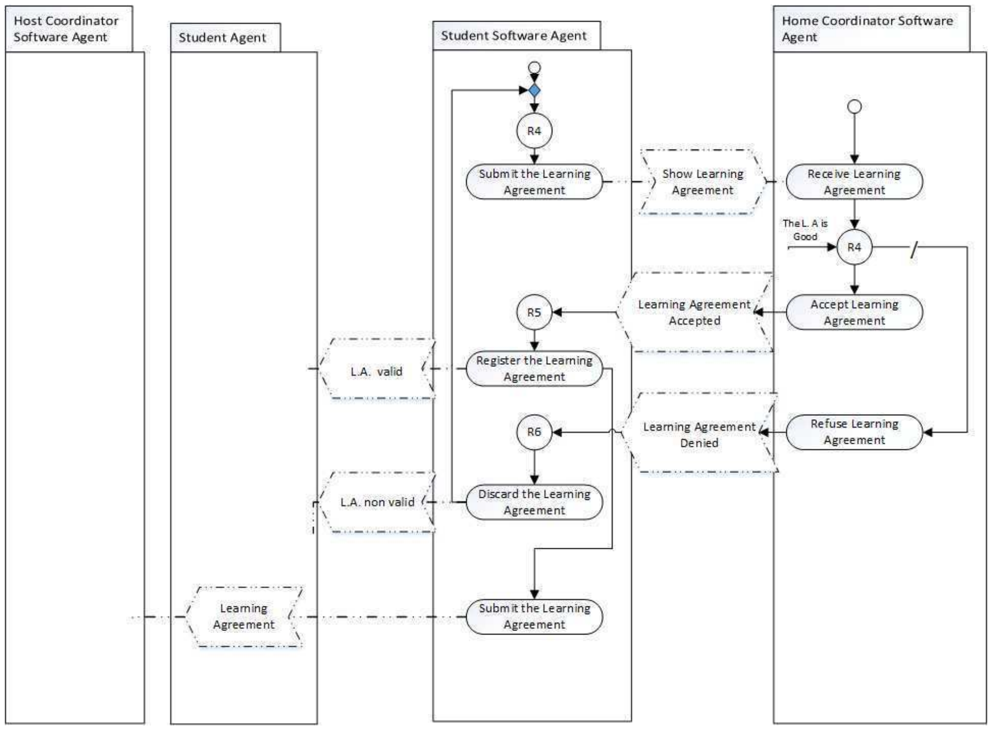

Host Coordinator Software Agent
Student Agent
Student Software Agent
Home Coordinator Software Agent
R4
Submit the Learning Agreement
Show Learning Agreement
Receive Learning Agreement
The L. A is Good
R4
Accept Learning Agreement
Learning Agreement Accepted
R5
Register the Learning Agreement
L.A. valid
Learning Agreement Denied
R6
Refuse Learning Agreement
Discard the Learning Agreement
L.A. non valid
Learning Agreement
Submit the Learning Agreement

FIGURE 3.10: Behaviour Model - Learning Agreement Process - Submit

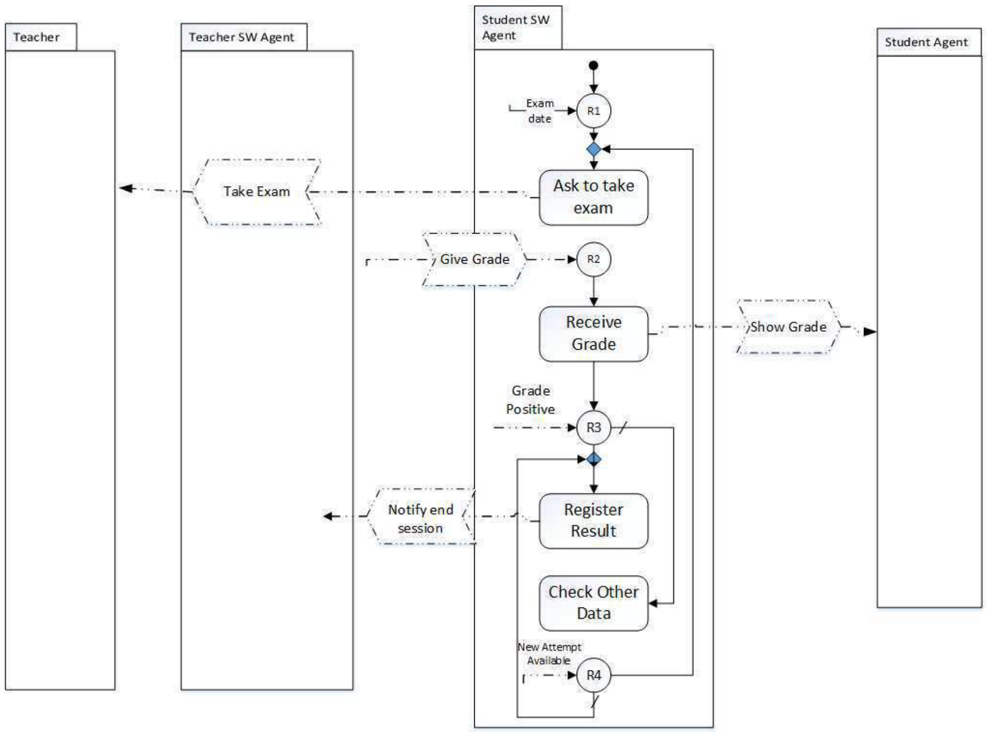


FIGURE 3.11: Behaviour Model - Sending Final Certificate - Exam Sessions

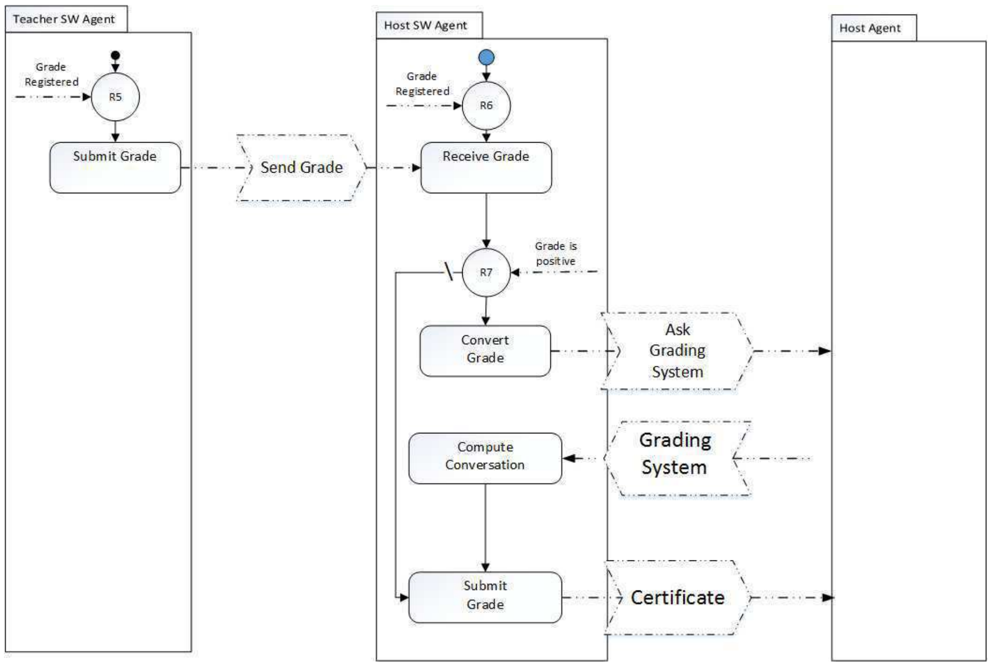


FIGURE 3.12: Behaviour Model - Sending Final Certificate - Submit Phase

Chapter 4

CPN tools - Simulation and Verification

The Erasmus Experience involves two different phases: Before mobility and after mobility. For this reason we have decided to use two different CPN model to validate and take the verification of our AOM models.

The first one refers to the Learning Agreement Management, included the requirements that are the gathering of available courses by the host university and the request by the student to receive the courses that he can attended.

4.1 CPN tools - Before Mobility

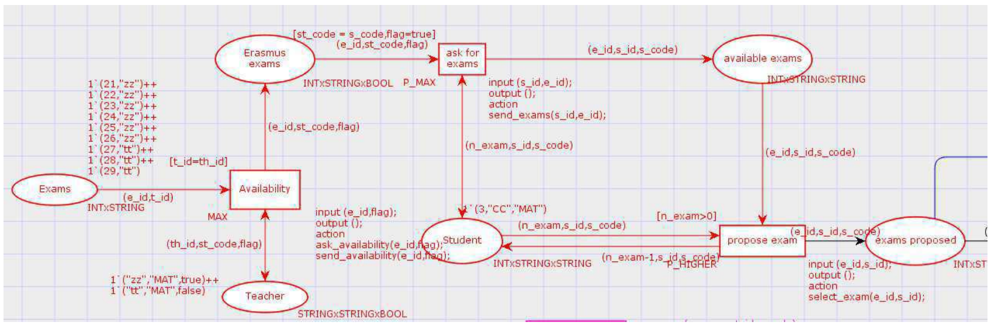
4.1.1 CPN model - Learning Agreement Process

Since the model referring to the Learning Agreement Process is too big to show completely in one figure, we split this in three figures.

The first one (figure 4.1) shows the interactions between the host university, the teachers and the student. The Host University gathers the availability of the teachers to hold the courses that host university wants to offers for Erasmus students.

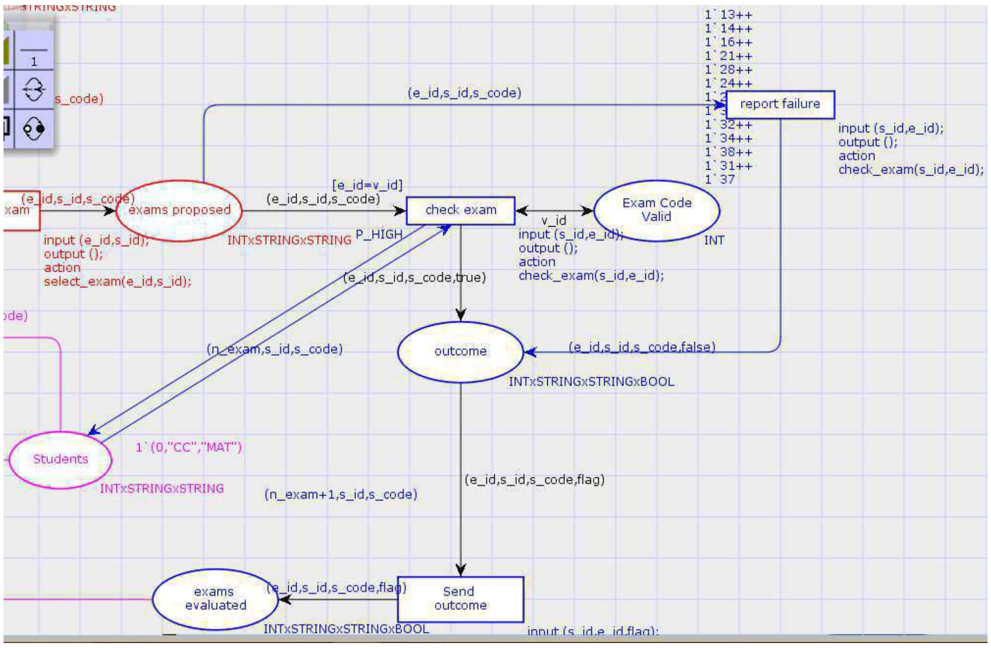
After that the student, in according his study plan, receive the courses that he can choose and select some of that.

The second one (figure 4.2) displays the process of the Home University regarding the acceptance of the courses proposed by the student. In the figure we can see a list of courses that University can accept (exams are identified here by a id code to simulate one possible policy). According to the code of the exams proposed, the Home University can accept or deny the exam. This information is used also to validate the Learning



```
input (s_id,e_id);
output ();
action
send_exams(s_id,e_id);
```
```
input (e_id,flag);
output ();
action
ask_availability(e_id,flag);
send_availability(e_id,flag);
```
```
input (e_id,s_id);
output ();
action
select_exam(e_id,s_id);
```

FIGURE 4.1: CPN Model - Learning Agreement Management (exams availability and selecting process)



```
input (s_id,e_id);
output ();
action
check_exam(s_id,e_id);
```
```
input (e_id,s_id);
output ();
action
select_exam(e_id,s_id);
```
```
input (s_id,e_id);
output ();
action
check_exam(s_id,e_id);
```

FIGURE 4.2: CPN Model - Learning Agreement Management (exams evaluating)

Agreement, because we have considered (just to have a simple example) that the Learning Agreement proposed is valid only if the student can take at least two exams in the Host University.

The third one (figure 4.3) shows the classification of the student proposals and the sending process by the student to host university of the the exams that he can do during Erasmus period. Only student that received a positive feedback for the Learning Agreement can send this information to the Host University, and he can only communicate the exams that he can effectively take, and not the ones that he proposed but are refused by the Home University.

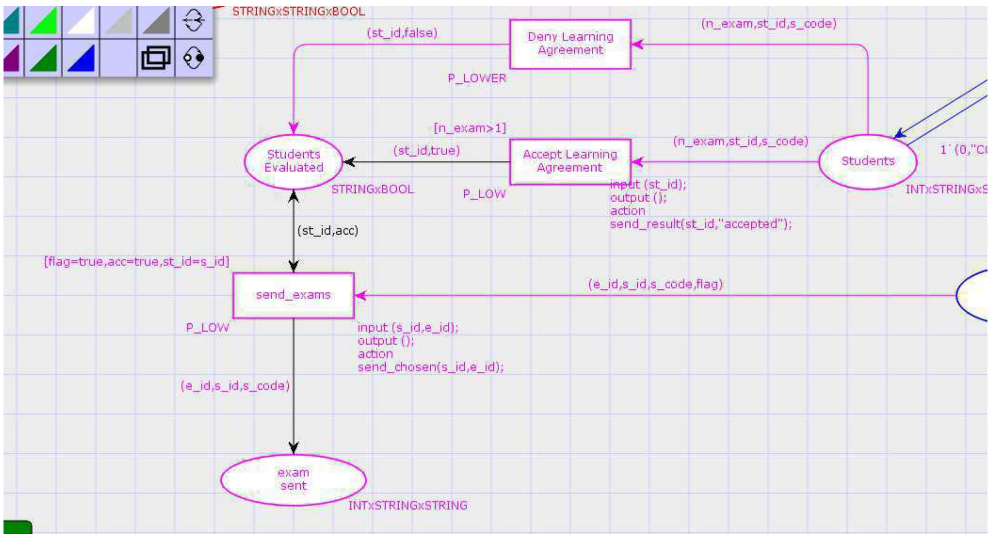


FIGURE 4.3: CPN Model - Learning Agreement Management (communication to Host University)

4.1.2 Scenarios

About the Learning Agreement Management, we have decided to show here two different possible scenario that can occurs in this phase.

In the first one we will show a successful case where the student chooses at least two exams that are good for the home university. In this case the Learning Agreement is accepted and the the communication of the exams that the student can take during his Erasmus are sent to the Host University.

In the second one we will show a failure case where less than two exams are accepted by the Home university. In this case no Learning Agreement should be sent to the host University.

Notice that the first part of the two scenarios about the request of available course is important-less for the purpose of this project, so we will show only the result of one run of simulation for this phase.

4.1.3 Learning Agreement Management - Simulation and MSC results

The figure 4.4 displays the communication between the host university and the teachers to have the availability of them to take the course for Erasmus Project. As it is possible to see by the figure, not all the exams receive availability by the teachers.

The figure 4.5 displays the success case of a student that choose at least two exams that are accepted by the home university getting a valid Learning Agreement.

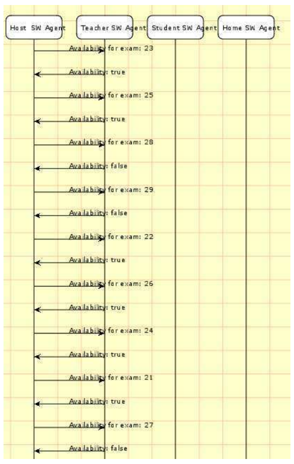

Host SW Agent
Teacher SW Agent
Student SW Agent
Home SW Agent
Availability for exam: 23
Availability: true
Availability for exam: 25
Availability: true
Availability for exam: 28
Availability: false
Availability for exam: 29
Availability: false
Availability for exam: 22
Availability: true
Availability for exam: 26
Availability: true
Availability for exam: 24
Availability: true
Availability for exam: 21
Availability: true
Availability for exam: 27
Availability: false

FIGURE 4.4: CPN Model Learning Agreement - Available courses list

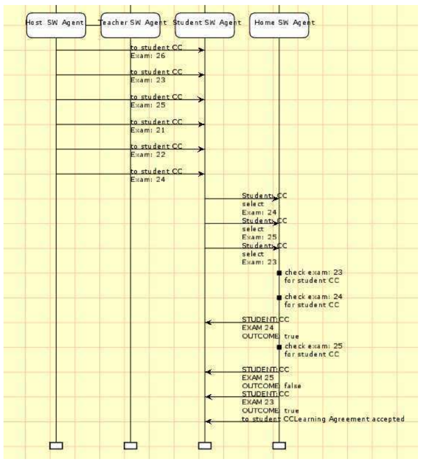

Host SW Agent
Teacher SW Agent
Student SW Agent
Home SW Agent
to student CC Exam: 26
to student CC Exam: 23
to student CC Exam: 25
to student CC Exam: 21
to student CC Exam: 22
to student CC Exam: 24
Student: CC select Exam: 24
Student: CC select Exam: 25
Student: CC select Exam: 23
check exam: 23 for student CC
check exam: 24 for student CC
STUDENT: CC EXAM 24 OUTCOME: true
check exam: 25 for student CC
STUDENT: CC EXAM 25 OUTCOME: false
STUDENT: CC EXAM 23 OUTCOME: true
to student CCLearning Agreement accepted

FIGURE 4.5: CPN Model Learning Agreement - Learning Agreement Accepted

In particular this simulation shows a student that has chosen three exams. One of them is refused but the other two are accepted, so the Learning Agreement is Valid and the student apply for the Host University.

The figure 4.6 displays the failure case of a student that choose less than two exams that are accepted by the home university getting a invalid Learning Agreement. The simulation refers to a student that has chosen three exams and only one of them is accepted, so the Learning Agreement is Valid and it has not been sent to host university as expected.

4.2 CPN tools - During Mobility

4.2.1 CPN model - Sending Final Certificate

The model shown in figure 4.7 displays the interaction between students, teachers and Host University during the Erasmus mobility. In particular it refers to the exams session

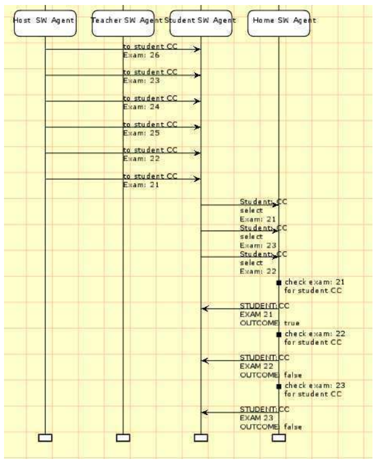
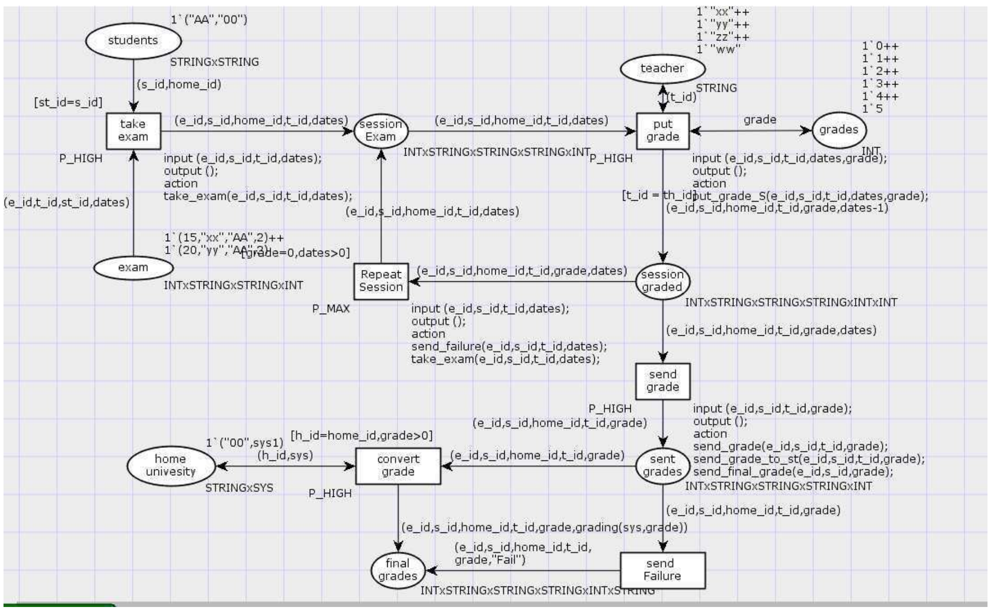

Host SW Agent
Teacher SW Agent
Student SW Agent
Home SW Agent
to student CC Exam: 26
to student CC Exam: 23
to student CC Exam: 24
to student CC Exam: 25
to student CC Exam: 22
to student CC Exam: 21
Student: CC select Exam: 21
Student: CC select Exam: 23
Student: CC select Exam: 22
check exam: 21 for student CC
STUDENT: CC EXAM 21 OUTCOME: true
check exam: 22 for student CC
STUDENT: CC EXAM 22 OUTCOME: false
check exam: 23 for student CC
STUDENT: CC EXAM 23 OUTCOME: false

FIGURE 4.6: CPN Model Learning Agreement - Learning Agreement Denied

process, included grading phase, and the final certificate sending for the exams that the students passed.


students
1`("AA","00")
STRINGxSTRING
(s_id,home_id)
[st_id=s_id]
take exam
P_HIGH
(e_id,t_id,st_id,dates)
exam
1`(15,"xx","AA",2)++
1`(20,"yy","AA",2)
INTxSTRINGxSTRINGxINT
(e_id,s_id,home_id,t_id,dates)
session Exam
INTxSTRINGxSTRINGxSTRINGxINT
(e_id,s_id,home_id,t_id,dates)
[grade=0,dates>0]
Repeat Session
P_MAX
(e_id,s_id,home_id,t_id,grade,dates)
teacher
1`"xx"++
1`"yy"++
1`"zz"++
1`"ww"
STRING
(t_id)
(e_id,s_id,home_id,t_id,dates)
put grade
P_HIGH
[t_id = th_id]
grade
grades
1`0++
1`1++
1`2++
1`3++
1`4++
1`5
INT
(e_id,s_id,home_id,t_id,grade,dates-1)
session graded
INTxSTRINGxSTRINGxSTRINGxINTxINT
(e_id,s_id,home_id,t_id,grade,dates)
send grade
P_HIGH
(e_id,s_id,home_id,t_id,grade)
sent grades
INTxSTRINGxSTRINGxSTRINGxINT
(e_id,s_id,home_id,t_id,grade)
(e_id,s_id,home_id,t_id,grade)
convert grade
P_HIGH
[h_id=home_id,grade>0]
(h_id,sys)
1`("00",sys1)
home univesity
STRINGxSYS
(e_id,s_id,home_id,t_id,grade,grading(sys,grade))
final grades
INTxSTRINGxSTRINGxSTRINGxINTxSTRING
(e_id,s_id,home_id,t_id, grade,"Fail")
send Failure

```
input (e_id,s_id,t_id,dates);
output ();
action
take_exam(e_id,s_id,t_id,dates);
```

```
input (e_id,s_id,t_id,dates,grade);
output ();
action
put_grade_S(e_id,s_id,t_id,dates,grade);
```

```
input (e_id,s_id,t_id,dates);
output ();
action
send_failure(e_id,s_id,t_id,dates);
take_exam(e_id,s_id,t_id,dates);
```

```
input (e_id,s_id,t_id,grade);
output ();
action
send_grade(e_id,s_id,t_id,grade);
send_grade_to_st(e_id,s_id,t_id,grade);
send_final_grade(e_id,s_id,grade);
```

FIGURE 4.7: CPN Model- Sending Final Certificate

In this model we can notice the loop that let the student to take again one exam if he has failed in the first attempt. The loop ends if the student passes the exam or he does not have more attempt for that exam (in this case we have considered two attempts for each exam).

About the sending of final certificate, we have implemented the automatic conversion of grades. Before doing this, the Host University asks the Home University to receive its grading system.

Notice that only for passed exams the grade is converted, while the failed ones are certificated with grade "fail“.

4.2.2 Scenarios

About the period spent by the student in the host university, ended with the exams sessions, we run three simulation to show these different scenarios. (In all three scenarios we consider a student that has taken two exams and for each of them he has only two attempts)

Scenario 1: The student takes two exams and the he passes both of them in the first attempt. So the exams are immediately handled by software agents and the certificate of them to home university.

Scenario 2: The student takes two exams and he passes one of them in the first attempt, and the other one in the second attempt. The simulations will show the interaction between the agents and the loop the let the student to repeat the exams.

Scenario 3: The student takes two exams and he fails one of them twice. The simulations will show that the loop that let the student to repeat the exams ends if the student doesn't have more attempts.

All scenario show the feature we have thought about an automated conversation of the grade. For simplification reason we have considered the conversation between 5-point grading systems to 10-point grading systems. The rules used (just to simplify the model since it is not important per the purpose of the project) is to multiply the grade obtained for two.

4.2.3 Sending Final Certificate - Simulation and MSC results

Figure 4.8 displays the interaction between the agents. In particular we can notice that the student take the exams only one time because he passes both of them in the first attempt.

In the figure 4.9 we can notice that the student take one of exams twice because he fails one of them in the first attempt.

Figure 4.10 shows the case where the student fails one exam twice (the student has only two attempts in our simplified policy). As expected for this exam the student send the grade failure (conversation is useless only for positive results).

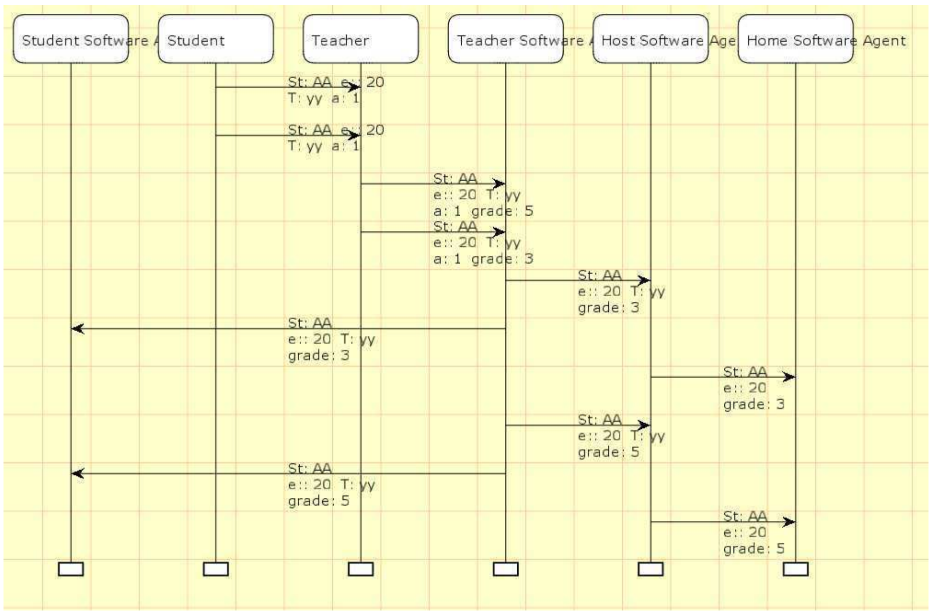


FIGURE 4.8: CPN Model Sending Final Certificate - Two Exams passed in the firs attempt

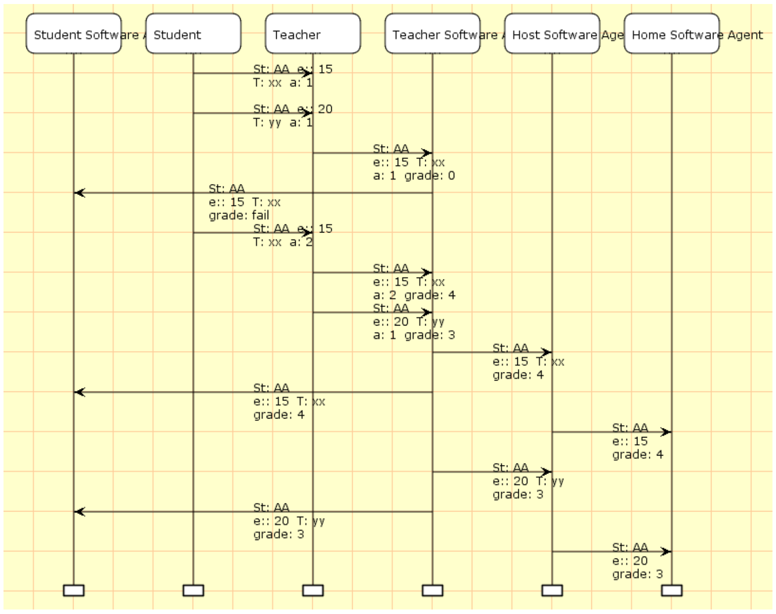


FIGURE 4.9: CPN Model Sending Final Certificate - Two Exams passed in the firs attempt

4.3 Verification

In order to verify our models, we run the state-space analysis for both of them.

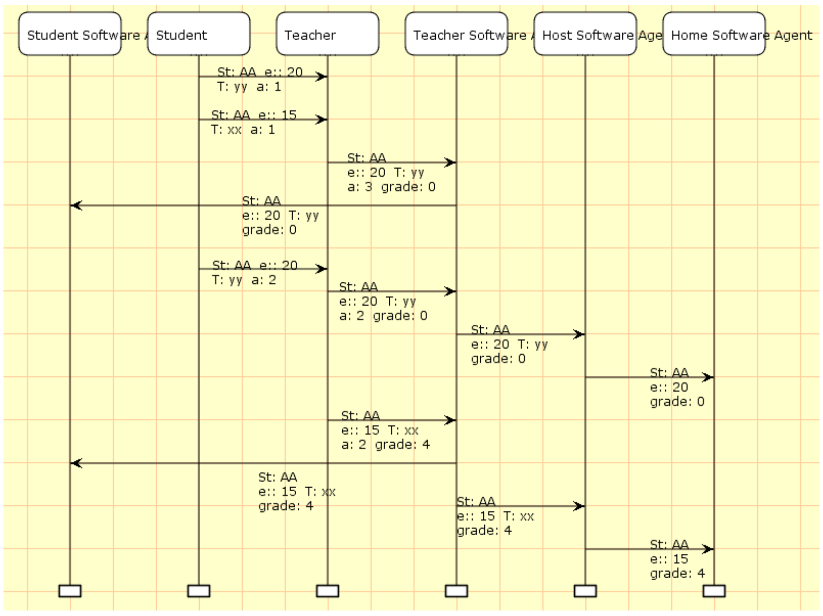
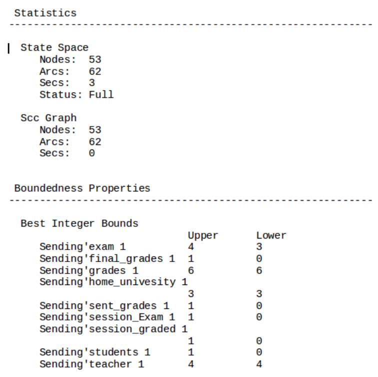


FIGURE 4.10: CPN Model Sending Final Certificate - Two Exams passed in the firs attempt

4.3.0.1 Verification - Learning Agreement Management model

Figure 4.11 shows the statistics of one run for the CPN model referring the Learning Agreement Management process. Since the status is full, the results shown in the state-space analysis are meaningful.



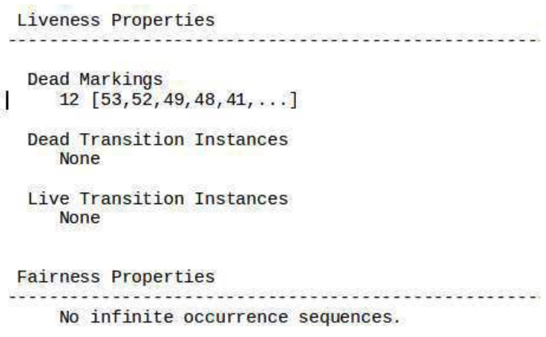
```
 Statistics
------------------------------------------------------------

| State Space
     Nodes:  53
     Arcs:   62
     Secs:   3
     Status: Full

  Scc Graph
     Nodes:  53
     Arcs:   62
     Secs:   0


 Boundedness Properties
------------------------------------------------------------

  Best Integer Bounds
                             Upper      Lower
     Sending'exam 1          4          3
     Sending'final_grades 1  1          0
     Sending'grades 1        6          6
     Sending'home_univesity 1
                             3          3
     Sending'sent_grades 1   1          0
     Sending'session_Exam 1  1          0
     Sending'session_graded 1
                             1          0
     Sending'students 1      1          0
     Sending'teacher 1       4          4
```

FIGURE 4.11: CPN model 1 - Statistics

Figure 4.12 displays the results of the simulation run.

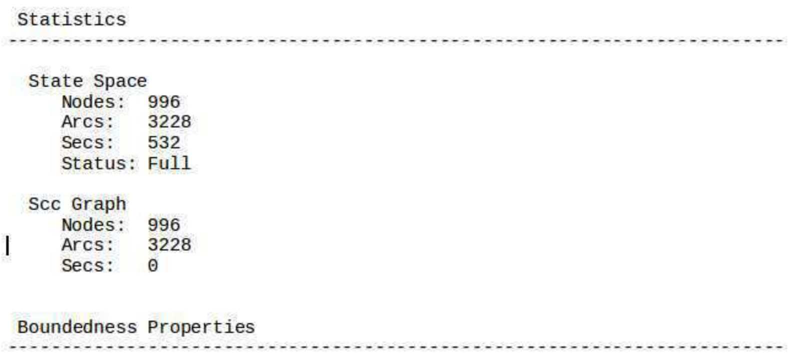
4.3.0.2 Verification - Sending Final Certificate model

Figure 4.13 shows the statistics of one run for the CPN model referring the Sending Final Certificate process. Also in this case we get the full status, so the results are significant.



```
 Liveness Properties
------------------------------------------------------------

  Dead Markings
     12 [53,52,49,48,41,...]

  Dead Transition Instances
     None

  Live Transition Instances
     None


 Fairness Properties
------------------------------------------------------------
     No infinite occurrence sequences.
```

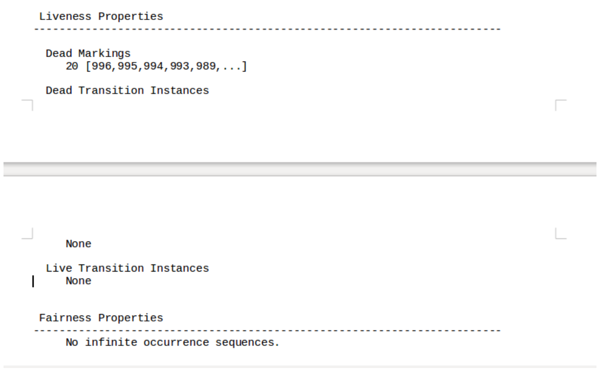
FIGURE 4.12: CPN model 1 - Verification results



```
 Statistics
------------------------------------------------------------------------

  State Space
     Nodes:  996
     Arcs:   3228
     Secs:   532
     Status: Full

  Scc Graph
     Nodes:  996
     Arcs:   3228
     Secs:   0


 Boundedness Properties
------------------------------------------------------------------------
```

FIGURE 4.13: CPN model 2 - Statistics

Figure 4.14 shows the result for the simulation run.



```
 Liveness Properties
------------------------------------------------------------------------

  Dead Markings
     20 [996,995,994,993,989,...]

  Dead Transition Instances
     None

  Live Transition Instances
     None

 Fairness Properties
------------------------------------------------------------------------
     No infinite occurrence sequences.
```

FIGURE 4.14: CPN model 3 - Verification results

Chapter 5

Conclusion

This project let us to learn how could be useful and efficient use AOM methodology to model systems that are meaningful in the society.

In particular the use of different agents that share the tasks of a single role could be a real improvement in the system since it is possible add a software agent that can do automated operation faster than human agent.

In addition the AOM methodology, such as the use of CPN tools to validate and verify the AOM models, has been very useful since it let us to know before a possible implementation problems that can occur in the test phase. In this way the CPN tools could get the AOM system development faster and safer, since he let people to save time to fix problems that without CPN tools can see only in the test phase.

The use of CPN tools, after a brief starter phase, has let us to validate and verify our models fast and easily.

For all these reasons we suggest to use CPN tools as support for the development of multi-agent systems.

Bibliography

[1] Leon S. Sterling, Kuldar Taveter *The Art of Agent-Oriented Modeling*

[2] CPN tools - `cpntools.org`

TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology

User to User Borrowing and Renting Service

Agent-Oriented Modelling and Multiagent Systems (IDY0303)

Tallinn
2016

Table of Contents

1. Introduction

For this project we are going to create a mobile application that enables people to borrow and use products and services directly from user to user.

Based on their location, the users would be able to search for products and services of their interest, that are nearest to them, and then contact their providers.

The application shows a list of all the products and allows the users to view detailed information and also the status of the product, for example if it is currently available or being borrowed by someone and for how long.

It will be possible to borrow products for free, to rent for money and to trade one product for another in return, according to the user's preferences.

2. Motivation layer

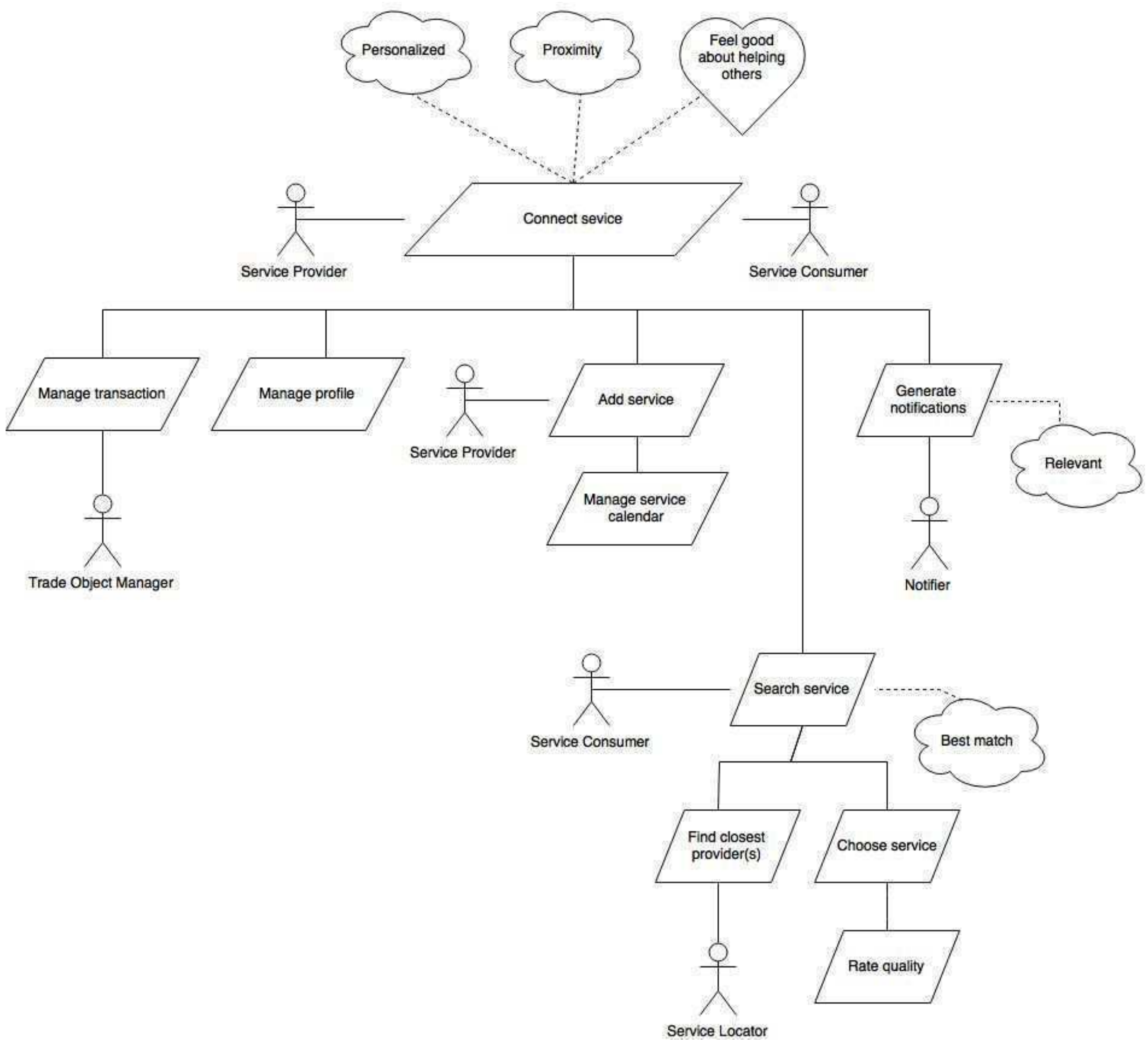
This section contains the following models: Goal model, Role model, Organizational model and Domain model.

2.1 Goal model

The goal model describes the hierarchy of functional goals, roles associated with functional goals, quality and emotional goals attached to functional goals.

The main goal of the system is to help connect lenders and borrowers (or service providers and consumers) to help out each other and to help decrease unnecessary spendings or to get affordable services more conveniently. The main goal is dependent on 5 subgoals. Most important of these are adding new services and searching for services. Adding a new service also includes managing service calendar to keep track of when the service is available. When searching for a service, the system tries to find best matching services and prioritizes those closest to the searcher.

Other subgoals are managing transactions between providers and consumers, managing user profiles, generating notifications about relevant services and rating the quality of products and services.



2.2 Role model

There are five roles in the system: Service Consumer, Service Provider, Service Locator, Notifier and Trade Object Manager. Below are more detailed descriptions of all roles.

Role name	Service Consumer
Description	The role of the user
Responsibilities	Register an account Allow location data

	Edit profile View products and services Search products and services Use products and services Rate products and services
Constraints	The application must be installed The user must be registered

Role name	Service Provider
Description	The role of the provider
Responsibilities	Register an account Allow location data Edit profile Offer new products and services View users Rate users
Constraints	The application must be installed The user must be registered

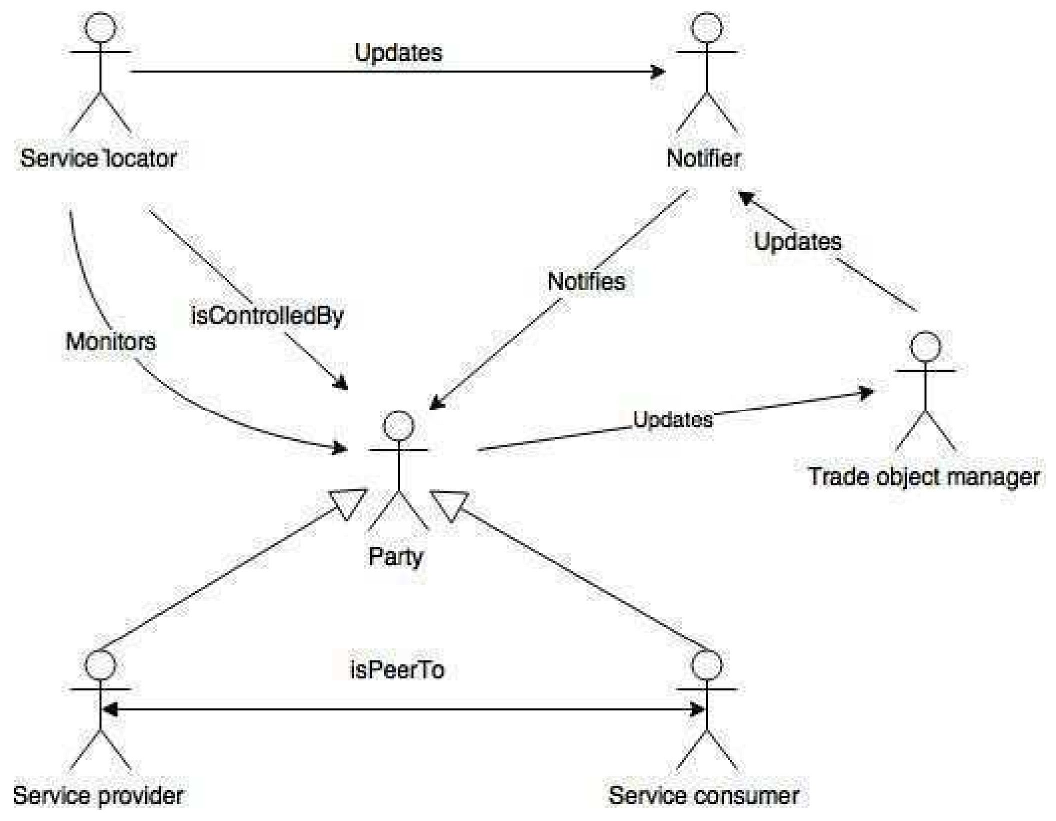
Role name	Service Locator
Description	The role of the GPS service
Responsibilities	Return coordinates of current location
Constraints	GPS services should be enabled on the device The application must be installed, where to return the coordinates

Role name	Notifier
Description	The role of the notification service
Responsibilities	Notify of incoming messages Notify if products or services become available Notify changes in transaction
Constraints	Connection to the database of the service

Role name	Trade Object Manager
Description	The role of the trade object manager
Responsibilities	Manage transactions Manage trade objects and states
Constraints	Connection to the database of the service

2.3 Organizational model

This figure shows the organizational model which describes different types of relationships between roles. We have five main roles: Service Locator, Notifier, Trade Object Manager, Service Consumer and Service Provider. Service Provider and Service Consumer are both Party-type roles. We have five different types of relationships: *Monitors*, *Updates*, *isControlledBy*, *Notifies*, *isPeerTo*.



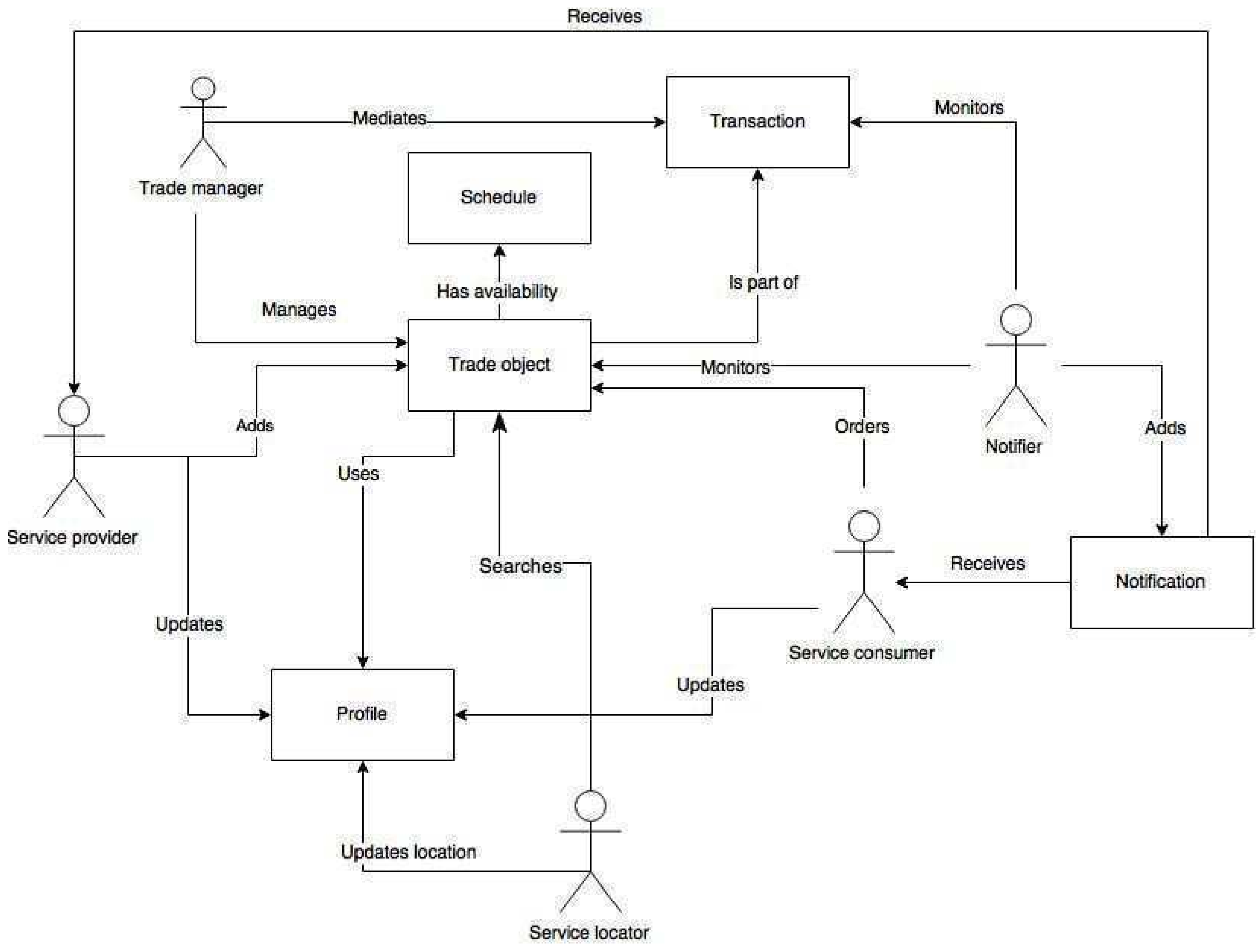
At the center of our service are service providers and service consumers. A provider can also be a consumer and vice versa. Service providers and consumers are monitored by the service locator. That information is then passed on to the Notifier, who decides whether to notify a particular party or not. If the needs of a Service Consumer is met by some Service Provider, the Service Consumer will be notified. If the Service Consumer decides to use that particular service, the Trade Object Manager will be updated and those updates will then be sent forward to the Notifier service.

2.4 Domain model

The purpose of the domain model is to describe the relationships between the roles and the resources. There are 5 different domain entities:

- Transaction
- Trade object
- Notification
- Profile
- Schedule

At the center of this domain model is the Trade object. Service Providers can update their profile and add trade objects. Service Consumers can order trade objects and also update their profile. Both Service Providers and Consumers will receive relevant notifications about trade objects and transactions. The Trade object has a schedule and can be a part of a particular Transaction. Notifier monitors Trade objects and Transactions and notifies Service Providers and Service Consumers when necessary. The Service Locator updates the location in the user profiles and searches for Trade object's location.

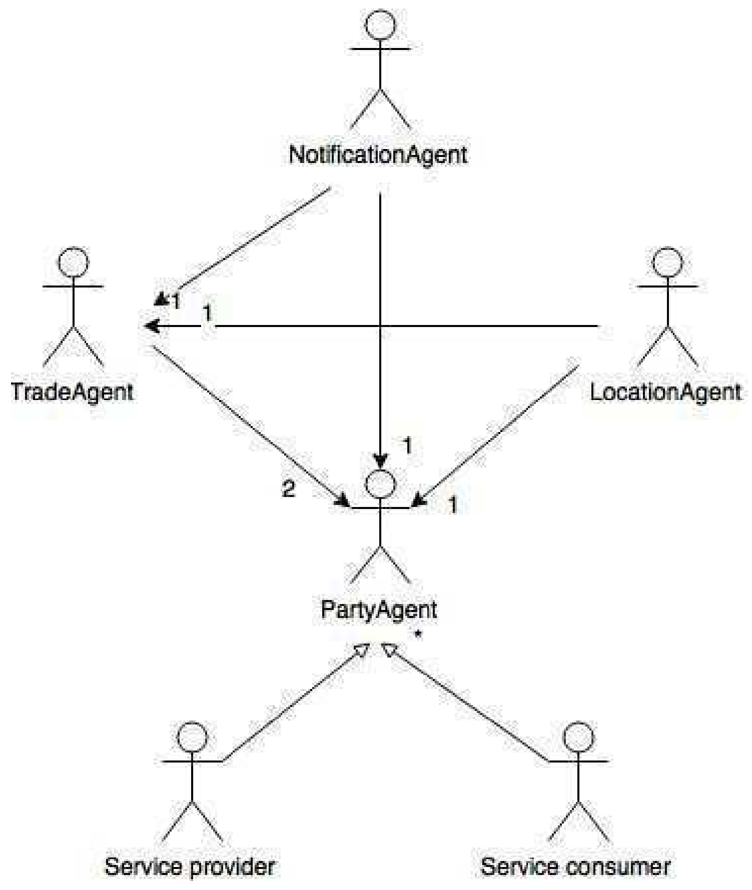


3. System design layer

This section contains the following models of system design layer: Agent and acquaintance model, interaction models, knowledge model and behaviour models.

3.1 Agent and acquaintance model

The agent model is created by designing agent types to fulfil the roles. Each role may be mapped to one or more agent types. The service provider and the service consumer agents are both mapped as Party Agents. The notification agent can send notifications both to the trade agent and the party agent . The precise roles and responsibilities are described in the table below.



Agent name	Party agent
Description	Human type agent
Roles	Service provider, Service consumer
Responsibilities	Register an account Allow location data Edit profile View products/services Offer products/services Order products/services Search products/services Rate products/services

Agent name	Notification agent
Description	Notification software component
Roles	Notification service
Responsibilities	Notify of incoming messages Notify if products or services become available Notify changes in transaction

Agent name	Location agent
Description	Location data provider
Roles	Location service
Responsibilities	Update current location Search nearby matching services

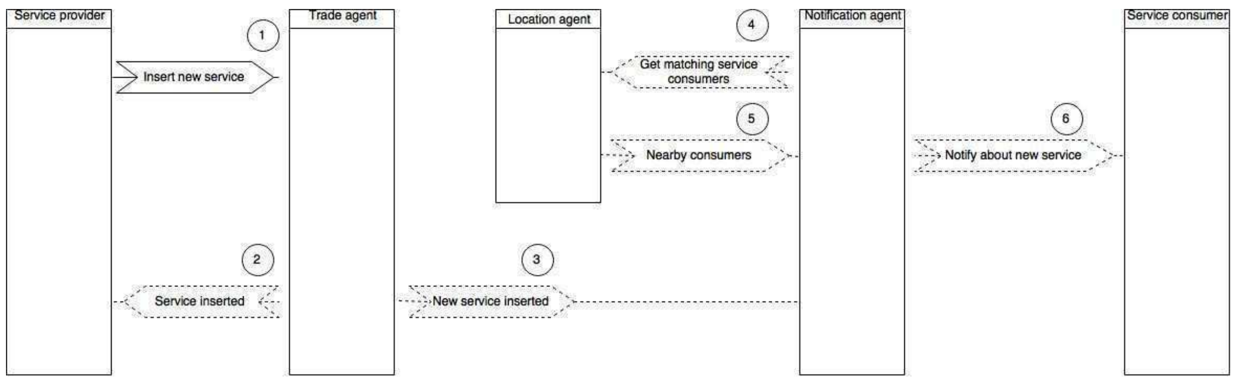
Agent name	Trade agent
Description	Trade object management software component
Roles	Trade object manager
Responsibilities	Manage transactions Manage trade objects and states

3.2 Interaction models

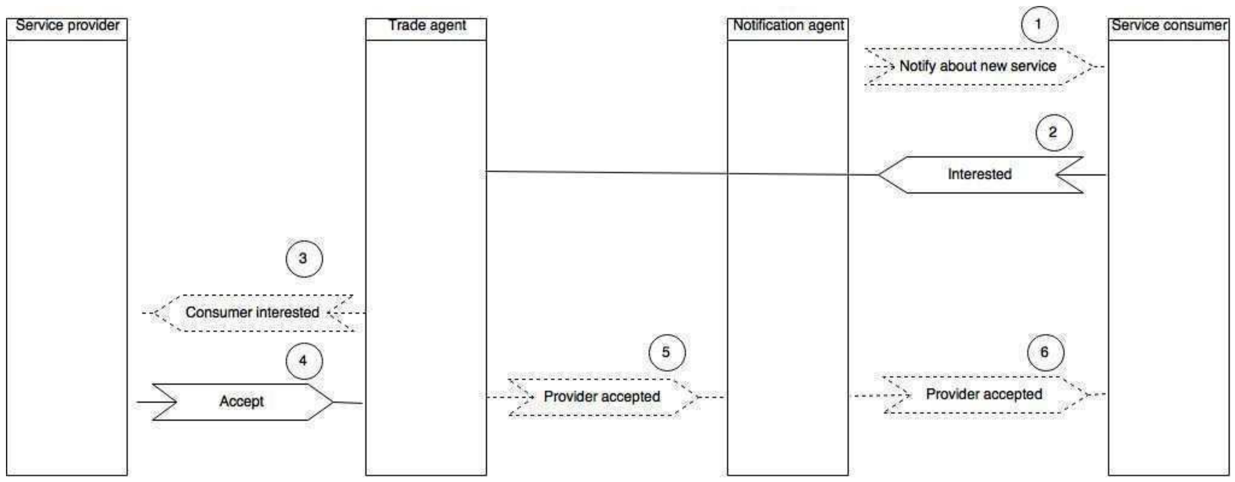
The interaction links described in section 3.1 show which agents interact with other agents and which party can initiate an interaction. Interaction model represent interaction patterns between agents in more detail. Interaction-sequence diagram models prototypical interactions as action events. Action event is an event that is caused by the action of an agent, like sending a message or starting a machine.

Direct actions performed by human agents are shown as continuous lines, messages sent between and by software agents are shown as dotted lines.

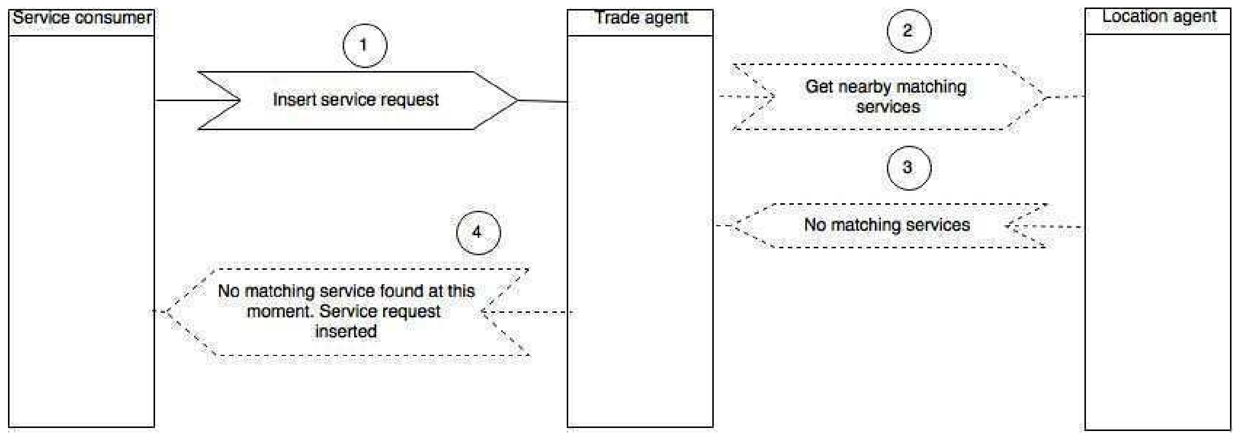
3.2.1 Service providing



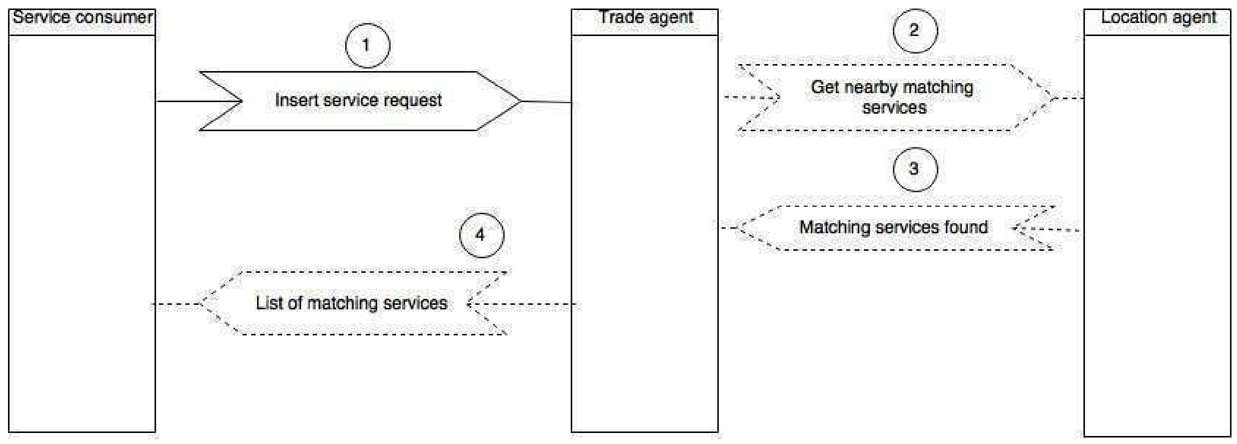
3.2.2 New service availability notification



3.2.3 Service request with no immediate match


Service consumer
Trade agent
Location agent
1
Insert service request
2
Get nearby matching services
3
No matching services
4
No matching service found at this moment. Service request inserted

3.2.4 Service request with immediate match

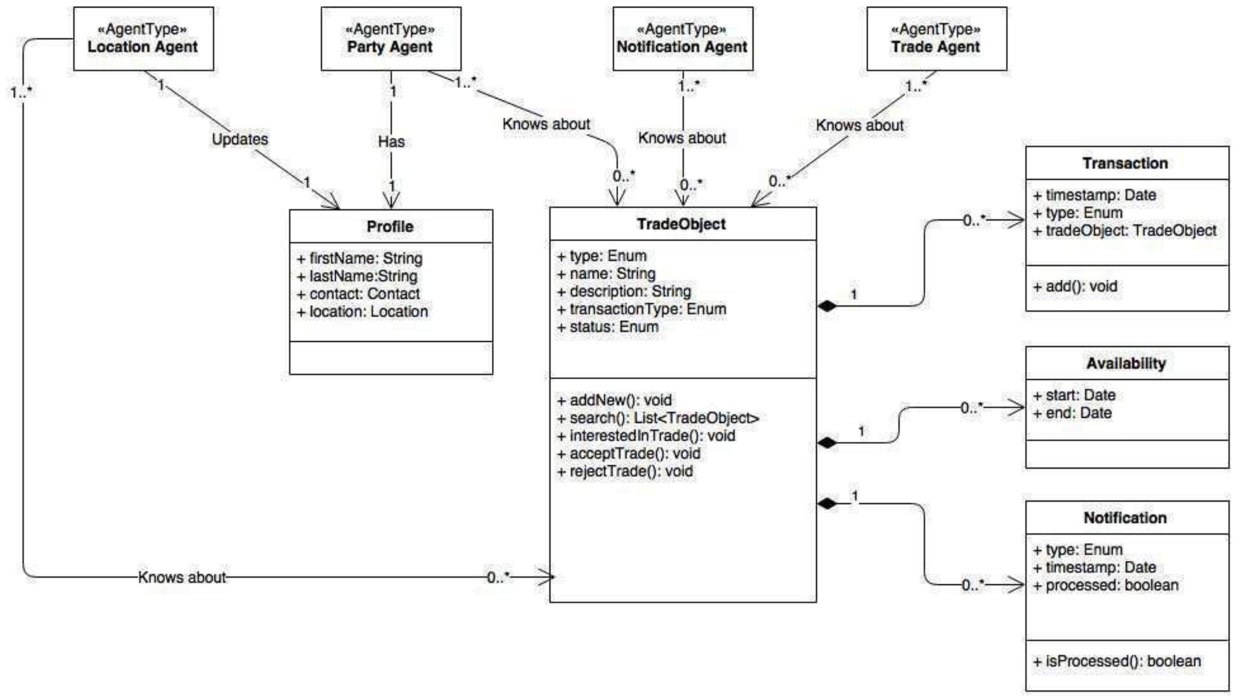


3.3 Knowledge model

Knowledge model represents private and shared knowledge that the agents need for functioning in the sociotechnical system. There are two most important data entities in our knowledge model: profile, which every party agent has, and trade object, which every party agent knows about.

Location agent updates the Party agents location and has information about all the trade objects. The profile contains basic knowledge like first name, last name and contact information.

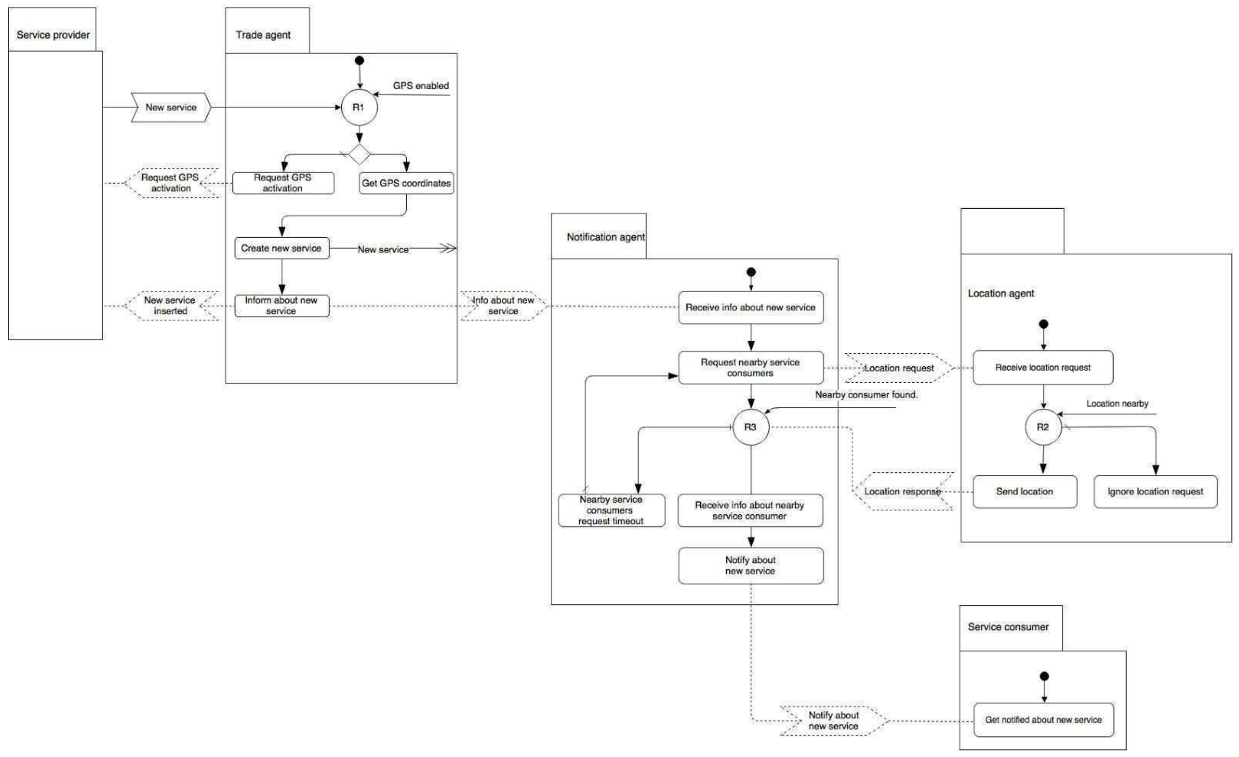
The trade object contains the following information: a type enum, name, description, transaction type, status. Enum is a data type consisting of a set of named values called elements, members, enumeral, or enumerators of the type of trade object it is (service, physical object etc.). The trade object has several methods: *addNew*, *search*, *interestedInTrade*, *acceptTrade, rejectTrade*. A Trade object can have multiple date ranges when it is available (Availability). It can also have multiple notifications, which notification agent has generated about this object. The trade object can have multiple transactions.



3.4 Behaviour models

Behaviour models are based on interaction models. While interaction models describe what happens between the agents, behaviour models go into more detail about what happens inside agents as a result of interacting with another agent.

3.4.1 Service providing behavior model



Service providing behaviour model describes what happens inside agents when a new service is added to the system. Trade agent asks the provider to enable GPS if necessary, acquires providers coordinates and inserts the service. Then it signals Notification agent, who in turn asks Location agent for nearby consumers. If any suitable consumers are found, Notification agent sends them a notification about new service.

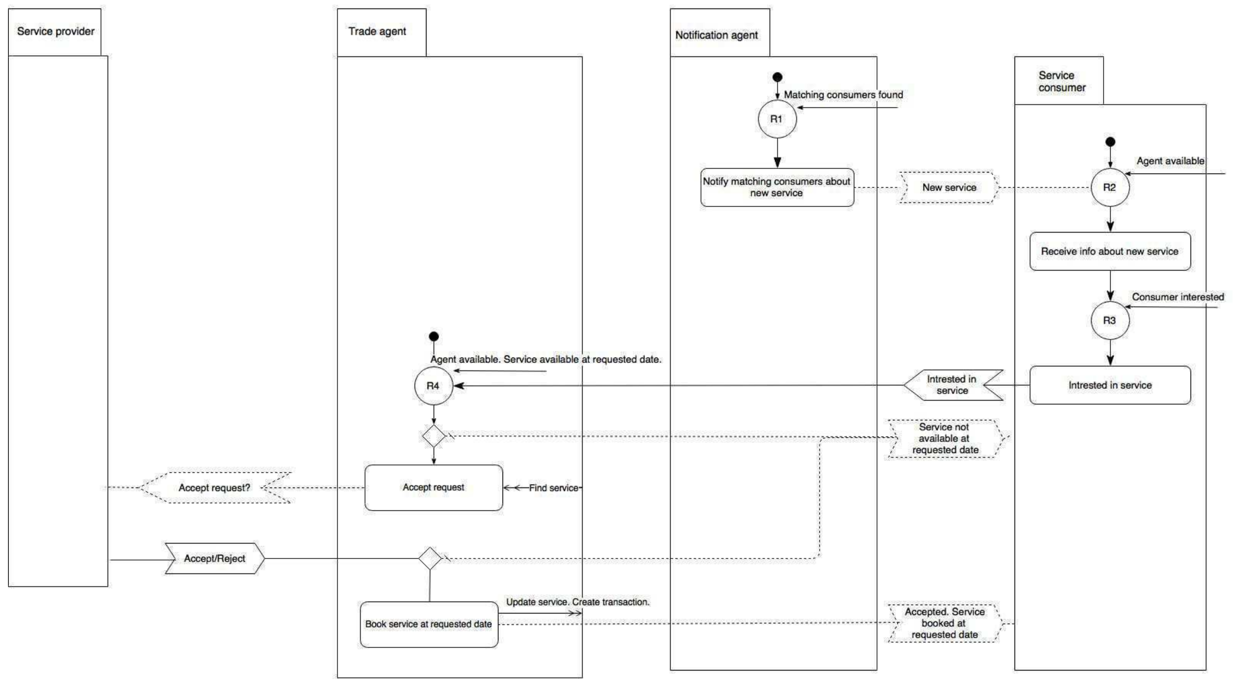
We are only interested in nearby consumers because it makes much more sense to suggest a new service to them, than to someone who is halfway around the world, too far to actually use the service. The only time distance doesn't matter, is when the service in question is virtual and doesn't require consumer/provider to be physically present. However, we in our model, all services do require physical presence.

R1 - A new service is inserted. Check if GPS is enabled. If yes, get coordinates and add new service to the system, else request provider to activate GPS.

R2 - If matching consumer is found and they happen to be nearby, location agent returns their GPS coordinates to notification service. If consumer isn't near, location agent ignores them.

R3 - If suitable consumer(s) are found, notify them about new available service.

3.4.2 New service availability behavior model


Service provider
Trade agent
Notification agent
Service consumer
Matching consumers found
R1
Notify matching consumers about new service
New service
Agent available
R2
Receive info about new service
Consumer interested
R3
Intrested in service
Intrested in service
Agent available. Service available at requested date.
R4
Service not available at requested date
Accept request?
Accept request
Find service
Accept/Reject
Book service at requested date
Update service. Create transaction.
Accepted. Service booked at requested date

New service availability behaviour model describes what happens when consumer gets notification about new service. Effectively, this model is a continuation of Service providing behaviour model (section 3.4.1).

Notification agent sends consumer a notice about new service. If consumer is available, they receive the notice and if they are interested in the service, they alert the Trade agent. Consumer also gives Trade agent a timeframe when they would be interested in the service. If the service is not available at requested date, Trade agent let's consumer know about it. On the other hand, if the service is available, Trade agent asks service provider to either accept or reject the request. In case service provider accepts, Trade agent creates a new transaction. In either case, Trade agent also informs consumer about provider's decision.

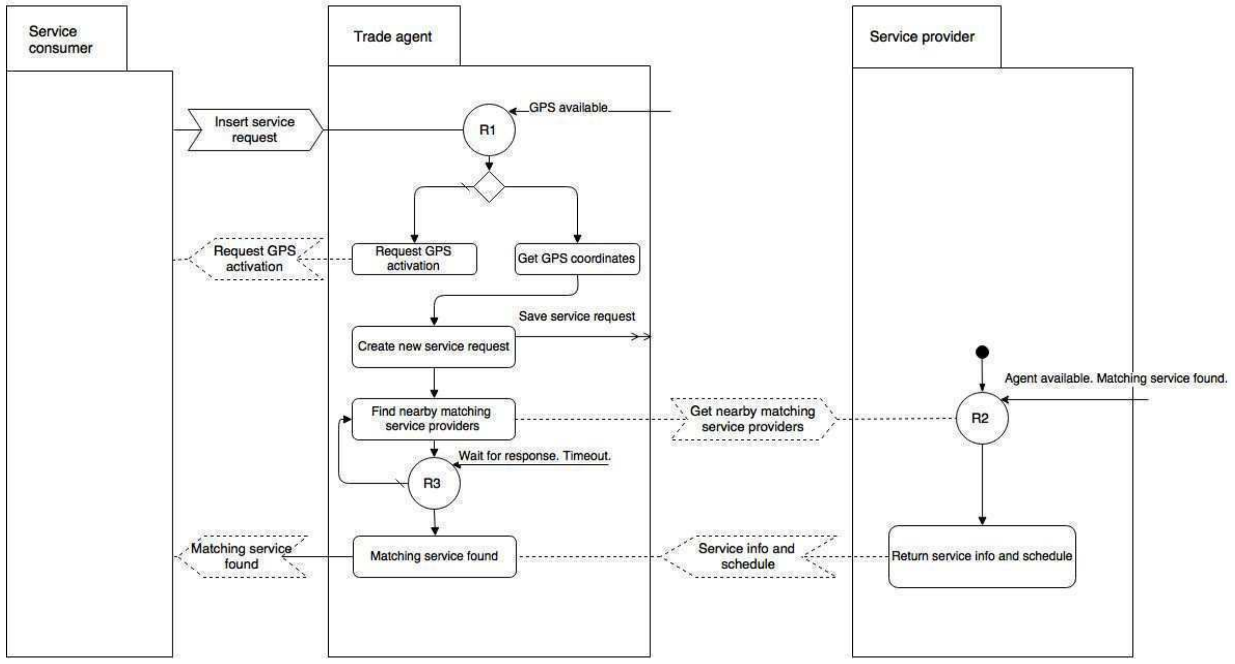
R1 - If matching consumers are found, notify them about new service.

R2 - If consumer is near, they get notification about new service.

R3 - If consumer is interested in the service, they alert Trade agent about it.

R4 - If consumer is available and service is available at the time consumer wishes, trade agent finds service provider and asks them to accept or reject the consumer. If service is not available et requested date, trade agent informs consumer about it.

3.4.3 Service request behavior model


Service consumer
Trade agent
Service provider
Insert service request
GPS available
R1
Request GPS activation
Request GPS activation
Get GPS coordinates
Save service request
Create new service request
Find nearby matching service providers
Get nearby matching service providers
Agent available. Matching service found.
R2
Wait for response. Timeout.
R3
Matching service found
Matching service found
Service info and schedule
Return service info and schedule

This model describes the behaviour of agents when a new service request is made. Consumer finds a service they want to consume (this is not described in the model) and requests it. Trade agent receives the request, asks consumer to activate their GPS if it's not enabled already, and creates a new request in the system. After that Trade agent searches for nearby providers who could respond to the request. If suitable providers are found, Trade agent returns found provider's service to the consumer.

R1 - Consumer requests a service. If they have GPS enabled, the coordinates are used to create a new request in the system. If consumer's GPS is not enabled, they are asked to enable it.

R2 - If matching provider is found, they are nearby and available, return info about provided service and the times when it is available.

R3 - Search for nearby matching providers is in progress. If the search times out, repeat it. If providers are found in reasonable time, return info about found matching service(s) to consumer.

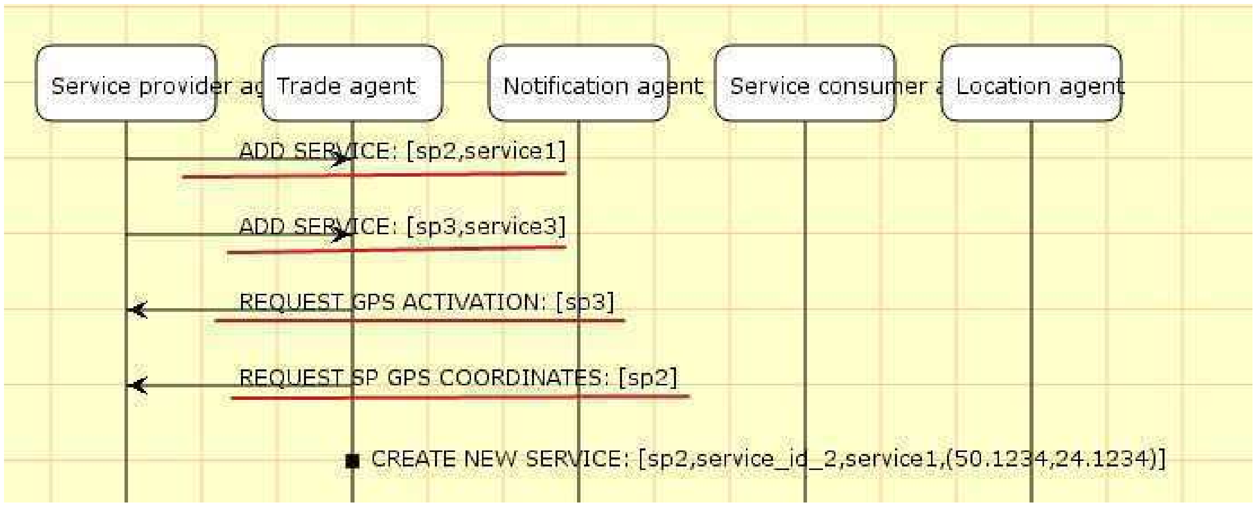
4. Analysis in CPN Tools

4.1 Validation

For validation we chose a subsection of our system that deals with providing a new service. The behaviour is also described in section 3.4.1 (Service providing behaviour model). All scenarios selected for validation are based on rules from behaviour model. Below are all scenarios in detail along with screenshot of MSC and analysis whether simulation behaved as expected. Due to the asynchronous nature of agents, there are some unrelated function calls between the calls specific to one scenario. Calls that pertain to a specific scenario are underlined.

4.1.1 Scenario 1

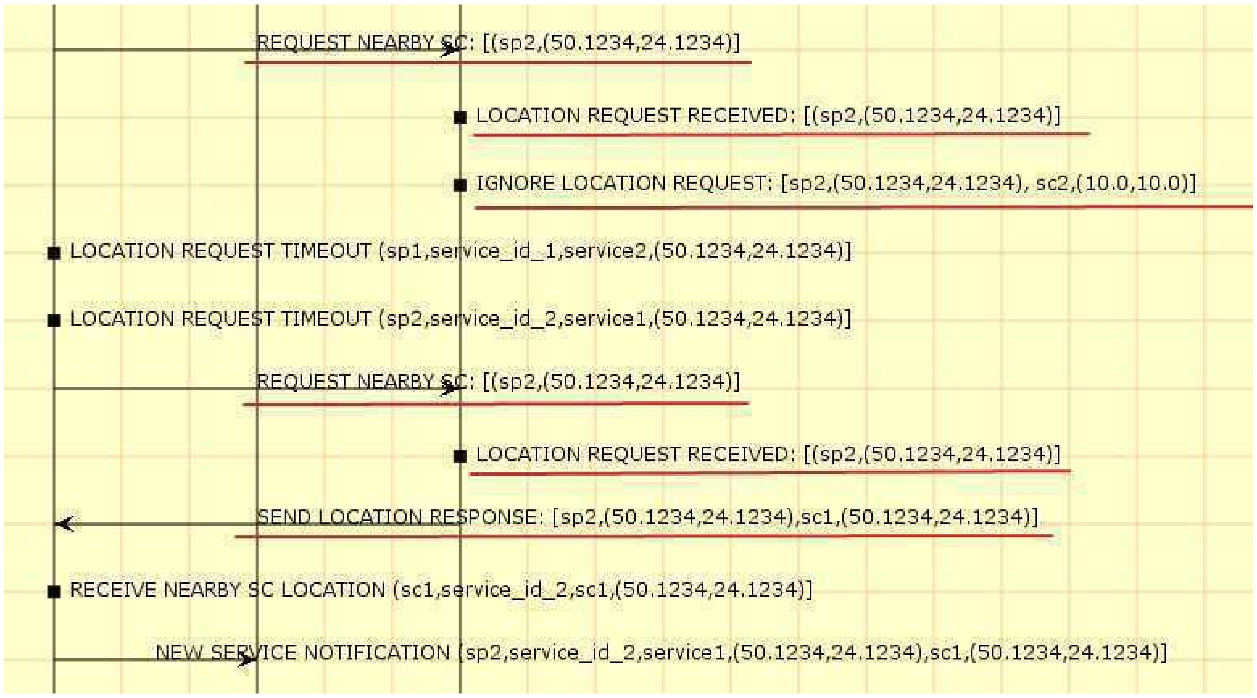
Description: Provider submits a new service to Trade agent. Trade agent checks if provider has their GPS enabled. If the GPS is enabled, Trade agent gets provider's coordinates and inserts new service along with these coordinates into the system. If GPS is not enabled, Trade agent asks provider to enable it.


Service provider ag
Trade agent
Notification agent
Service consumer a
Location agent
ADD SERVICE: [sp2,service1]
ADD SERVICE: [sp3,service3]
REQUEST GPS ACTIVATION: [sp3]
REQUEST SP GPS COORDINATES: [sp2]
CREATE NEW SERVICE: [sp2,service_id_2,service1,(50.1234,24.1234)]

Analysis: The part of the simulation for first scenario worked exactly as expected. Service provider sp2 submits new service (first ADD SERVICE function call), Trade agent requests providers GPS coordinates. Provider gives the coordinates (this is unfortunately not displayed on the screenshot), and trade agent creates new service in the system along with these coordinates. The screenshot also illustrates another option where providers GPS was not enabled (second function call) and trade agent correctly requested that provider enable it.

4.1.2 Scenario 2

Description: Location agent gets request for consumers that are near the provider. Location agent searches for consumers based on their current GPS coordinates. If consumer is near provider, location agent return the consumer. If consumer is somewhere far away, location agent ignores it.



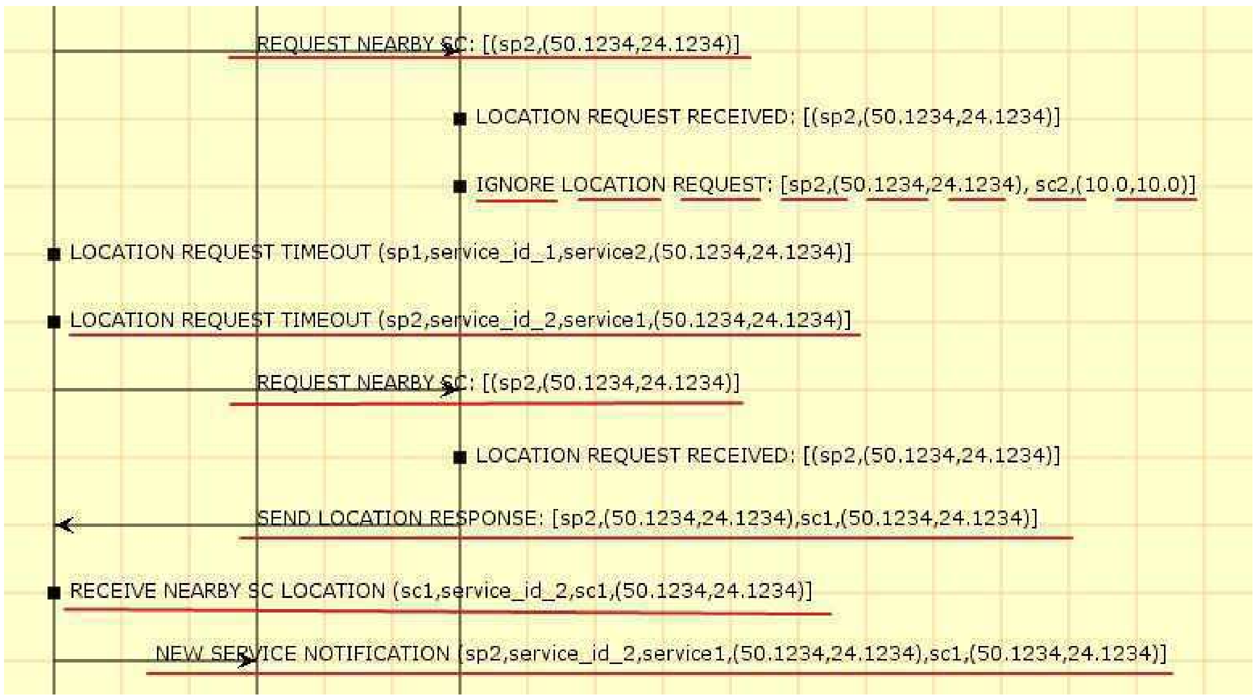
```
REQUEST NEARBY SC: [(sp2,(50.1234,24.1234)]
LOCATION REQUEST RECEIVED: [(sp2,(50.1234,24.1234)]
IGNORE LOCATION REQUEST: [sp2,(50.1234,24.1234), sc2,(10.0,10.0)]
LOCATION REQUEST TIMEOUT (sp1,service_id_1,service2,(50.1234,24.1234)]
LOCATION REQUEST TIMEOUT (sp2,service_id_2,service1,(50.1234,24.1234)]
REQUEST NEARBY SC: [(sp2,(50.1234,24.1234)]
LOCATION REQUEST RECEIVED: [(sp2,(50.1234,24.1234)]
SEND LOCATION RESPONSE: [sp2,(50.1234,24.1234),sc1,(50.1234,24.1234)]
RECEIVE NEARBY SC LOCATION (sc1,service_id_2,sc1,(50.1234,24.1234)]
NEW SERVICE NOTIFICATION [sp2,service_id_2,service1,(50.1234,24.1234),sc1,(50.1234,24.1234)]
```

Analysis: Simulation of second scenario also runs as expected. Notification agent requests consumer, who is near coordinates 50.1234, 24.1234. Location agent finds consumer sc2, whose coordinates are 10.0, 10.0. Based on these coordinates location agent decides that consumer is not anywhere near provider, and ignores the consumer.

On the lower part of the screenshot is also positive outcome for this scenario, where a consumer with matching GPS coordinates was found and returned to notification service. It should be noted that in a real system, consumer coordinates would need to fall into some +/- range of the provider, but to simplify our model, exact coordinates are good enough.

4.1.3 Scenario 3

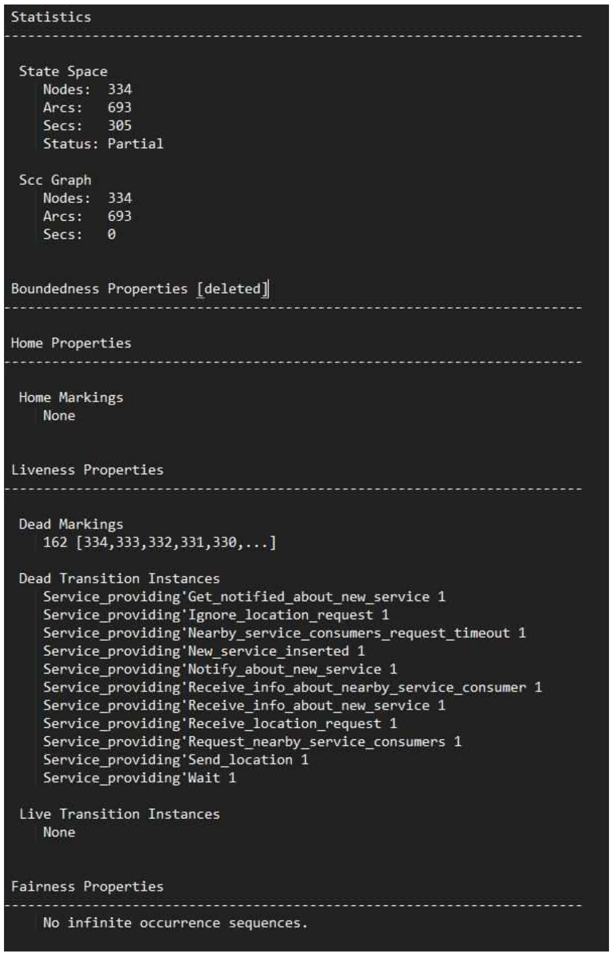
Description: Notification agent waits for location service to return nearby consumers. If there are any found, notification agent receives information about them and sends them notifications about the service. If location agent takes too long to respond (effectively times out), notification agent requests consumers again.


```
REQUEST NEARBY SC: [(sp2,(50.1234,24.1234)]
LOCATION REQUEST RECEIVED: [(sp2,(50.1234,24.1234)]
IGNORE LOCATION REQUEST: [sp2,(50.1234,24.1234), sc2,(10.0,10.0)]
LOCATION REQUEST TIMEOUT (sp1,service_id_1,service2,(50.1234,24.1234)]
LOCATION REQUEST TIMEOUT (sp2,service_id_2,service1,(50.1234,24.1234)]
REQUEST NEARBY SC: [(sp2,(50.1234,24.1234)]
LOCATION REQUEST RECEIVED: [(sp2,(50.1234,24.1234)]
SEND LOCATION RESPONSE: [sp2,(50.1234,24.1234),sc1,(50.1234,24.1234)]
RECEIVE NEARBY SC LOCATION (sc1,service_id_2,sc1,(50.1234,24.1234)]
NEW SERVICE NOTIFICATION [sp2,service_id_2,service1,(50.1234,24.1234),sc1,(50.1234,24.1234)]
```

Analysis: Simulation of the third scenario also behaves as expected. When notification agent requests nearby consumer, and none is found (in the simulation, the one that is found is not near), the request times out as it was supposed to do. After timeout notification agent tries again to find a suitable consumer. This time location agent returns a nearby consumer. On getting information about the consumer, notification agent sends a notification to them, alerting them of a new service.

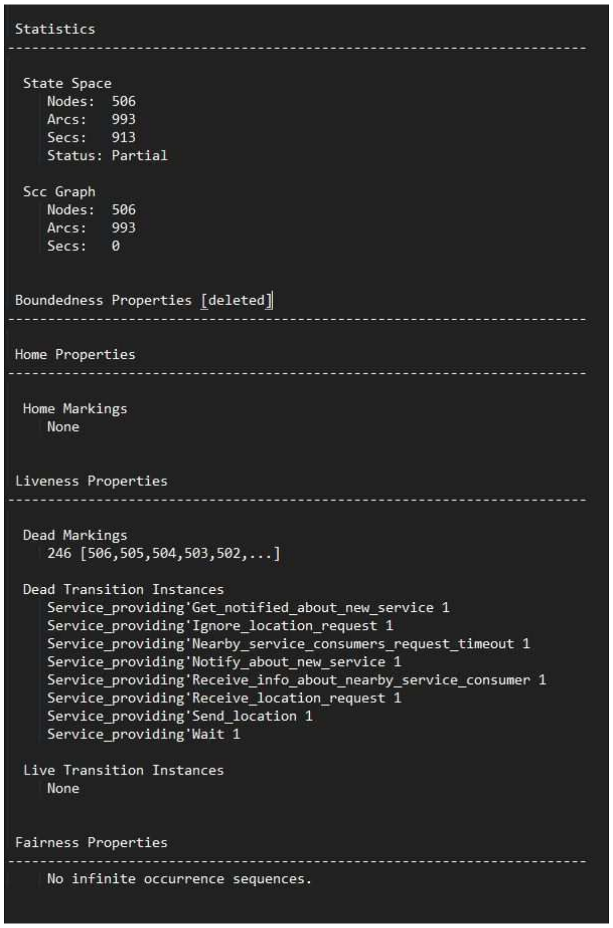
4.2 Verification

Despite our best efforts we didn’t manage to get a full verification status. Below are 2 screenshots of first and third run of State Space analysis. From the screens it is visible that number of travelled nodes grew by ~200 and the number of arcs grew by ~300. It can also be seen that the number of dead markings grew, but the number of dead transition instances decreased by 3.



```
Statistics
------------------------------------------------------------------------

 State Space
    Nodes:  334
    Arcs:   693
    Secs:   305
    Status: Partial

 Scc Graph
    Nodes:  334
    Arcs:   693
    Secs:   0


Boundedness Properties [deleted]
------------------------------------------------------------------------

Home Properties
------------------------------------------------------------------------

 Home Markings
    None


Liveness Properties
------------------------------------------------------------------------

 Dead Markings
    162 [334,333,332,331,330,...]

 Dead Transition Instances
    Service_providing'Get_notified_about_new_service 1
    Service_providing'Ignore_location_request 1
    Service_providing'Nearby_service_consumers_request_timeout 1
    Service_providing'New_service_inserted 1
    Service_providing'Notify_about_new_service 1
    Service_providing'Receive_info_about_nearby_service_consumer 1
    Service_providing'Receive_info_about_new_service 1
    Service_providing'Receive_location_request 1
    Service_providing'Request_nearby_service_consumers 1
    Service_providing'Send_location 1
    Service_providing'Wait 1

 Live Transition Instances
    None


Fairness Properties
------------------------------------------------------------------------
    No infinite occurrence sequences.
```

State-space analysis of first run



```
Statistics
------------------------------------------------------------------------

 State Space
    Nodes:  506
    Arcs:   993
    Secs:   913
    Status: Partial

 Scc Graph
    Nodes:  506
    Arcs:   993
    Secs:   0


Boundedness Properties [deleted]
------------------------------------------------------------------------

Home Properties
------------------------------------------------------------------------

 Home Markings
    None


Liveness Properties
------------------------------------------------------------------------

 Dead Markings
    246 [506,505,504,503,502,...]

 Dead Transition Instances
    Service_providing'Get_notified_about_new_service 1
    Service_providing'Ignore_location_request 1
    Service_providing'Nearby_service_consumers_request_timeout 1
    Service_providing'Notify_about_new_service 1
    Service_providing'Receive_info_about_nearby_service_consumer 1
    Service_providing'Receive_location_request 1
    Service_providing'Send_location 1
    Service_providing'Wait 1

 Live Transition Instances
    None


Fairness Properties
------------------------------------------------------------------------
    No infinite occurrence sequences.
```

State-space analysis of third run

There were quite many dead markings, 162 on the first run, 246 on the third. Dead markings denote nodes without outgoing arcs[1], in our model, there are 5 such nodes.

Dead transition instances correspond to parts of the model that can never be activated. Therefore they could be removed from the model without changing its behaviour[1]. However, in our model all the transitions that are marked as dead were actually traversed when we ran the simulation manually. Therefore it is difficult to see why state space analysis would mark these transitions as dead.

Since we couldn’t get a full verification status, it’s impossible to say if our model was correct.

Conclusion

Our goal was to analyse, model, validate and verify a system for borrowing and renting services. The main idea was that providers could add a new service, consumers would be notified about it, and could also search for existing services.

Creating all the analysis and design models was a great way to get a bigger picture of our system and to really think through all the interactions and necessary relationships between our agents and to model the functional and nonfunctional goals.

For validation and verification we used CPN Tools. Validation part seems to be correct, messages corresponded to scenario descriptions. However, we were unable to completely verify our model because we couldn’t get a complete state space analysis.

CPN Tools is an interesting tool to model and verify agent oriented systems. It’s probably very useful to domain experts and people who have much experience using it.

However, it has quite steep learning curve. That is both because none of us had any prior experience with agent-oriented modelling, and because CPN Tools has a very unique user interface that works differently from any other GUI we had ever seen.

It also has same weird bugs, for example the program tends to crash if you leave it unattended long enough for screensaver to appear.

References

1. Kurt Jensen, Lars M. Kristensen, Coloured Petri Nets: Modelling and Validation of Concurrent Systems, Springer, 2009

Tallinn University of Technology
Department of Informatics
Chair of Software Engineering

Adaptive Personal Training Advisor

A mini-project in the course of
“Agent-oriented modelling and multi-agent systems”

Tallinn 2016

Table of Contents

Introduction

The goal of this Project is to learn the basics of agent-oriented modelling by building a multi-agent system, that helps people in improving their health by composing a gym training plan. The person willing to train defines a goal and the system creates a personalized training plan. The key feature of the system is adaptivity in terms of training plan difficulty and gym equipment availability.

The document consists of three main parts: Requirements analysis, Design and CPN simulation. The first two parts consist mainly of the corresponding diagrams together with descriptions. The third part contains verification and validation of a simplified system simulation using Coloured Petri nets (CPN). Lastly a Conclusion is presented with the overview of the work done.

Requirements analysis

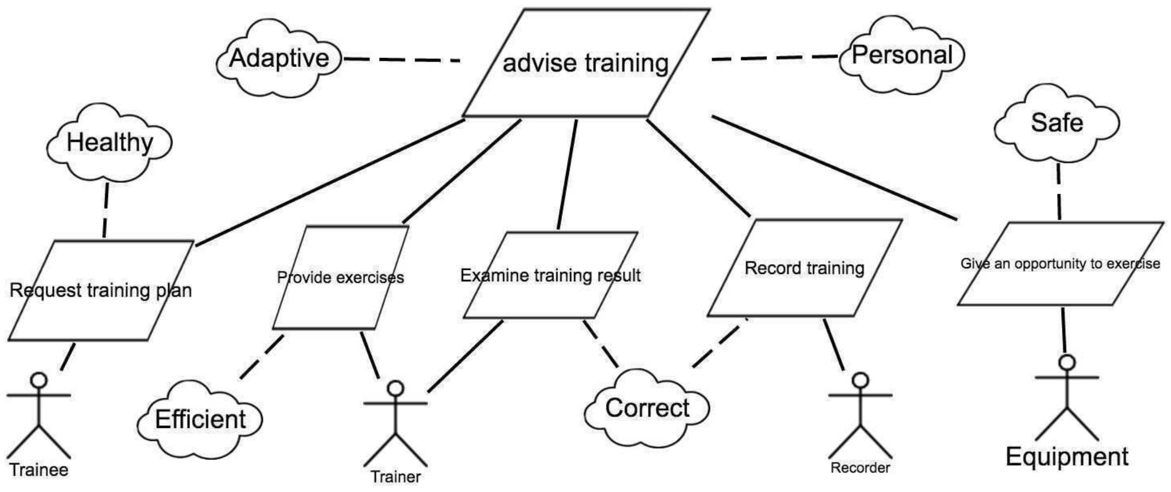
Goal model

The main goal of the system is to advise an adaptive and personalized training for the person (Trainee) willing to improve his/her health. To reach this goal a number of subgoals is created.

Trainee requests a training plan, that must be healthy (sufficiently difficult to present a challenge and be beneficial, but not too difficult to adversely affect well-being).

Training plan consists of exercises, that the Trainer must provide. Trainee has to do the exercises using the Equipment which gives safe opportunity to exercise. The Recorder must correctly record the training session, and the Trainer must analyze the recorded data to make training plan improvements if needed.

The advised training plans are adaptive (an alternative exercise is provided if the Equipment is used by someone else or unavailable at specific gym, the difficulty is adjusted if it is too easy or too hard for the Trainee) and personal (based on the age, weight, height, gender and other personal data the Trainee provides).



Role model

There are 4 roles in the system: Trainee, Trainer, Recorder and Equipment. Below are the detailed characteristics of the roles.

Role name	Trainee
Description	The person who wants to improve his/her physical health

Responsibilities	• Defines goal • Goes to training • Does exercises provided by Trainer
Constraints	Has to do exercises provided by Trainer

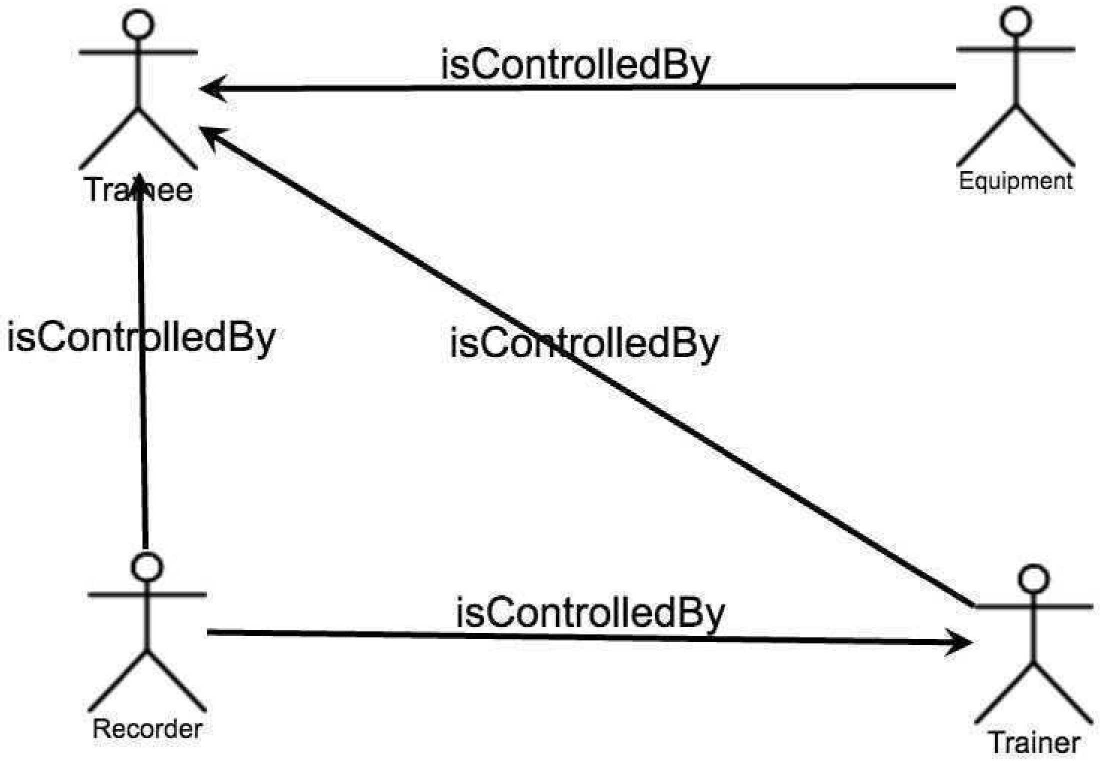
Role name	Trainer
Description	Helps Trainee to achieve his physical goal, provides training plan and exercise information
Responsibilities	• Provides Trainee a training plan • Provides information about exercise (how to do, etc.) • Provides Trainee an alternative exercise • Examine training results
Constraints	• Has to provide adaptive and personal training plan • Has to provide efficient exercises (training plan) • Has to examine training results correctly

Role name	Recorder
Description	Records result of the training (done exercises, order of exercises, etc.)
Responsibilities	• Records done exercises (and their order) • Records Trainee's health/body data
Constraints	• Has to record data without loss • Has to record data correctly

Role name	Equipment
Description	Gym equipment
Responsibilities	• Gives an opportunity to do exercise
Constraints	Has to provide safe opportunity to do exercise

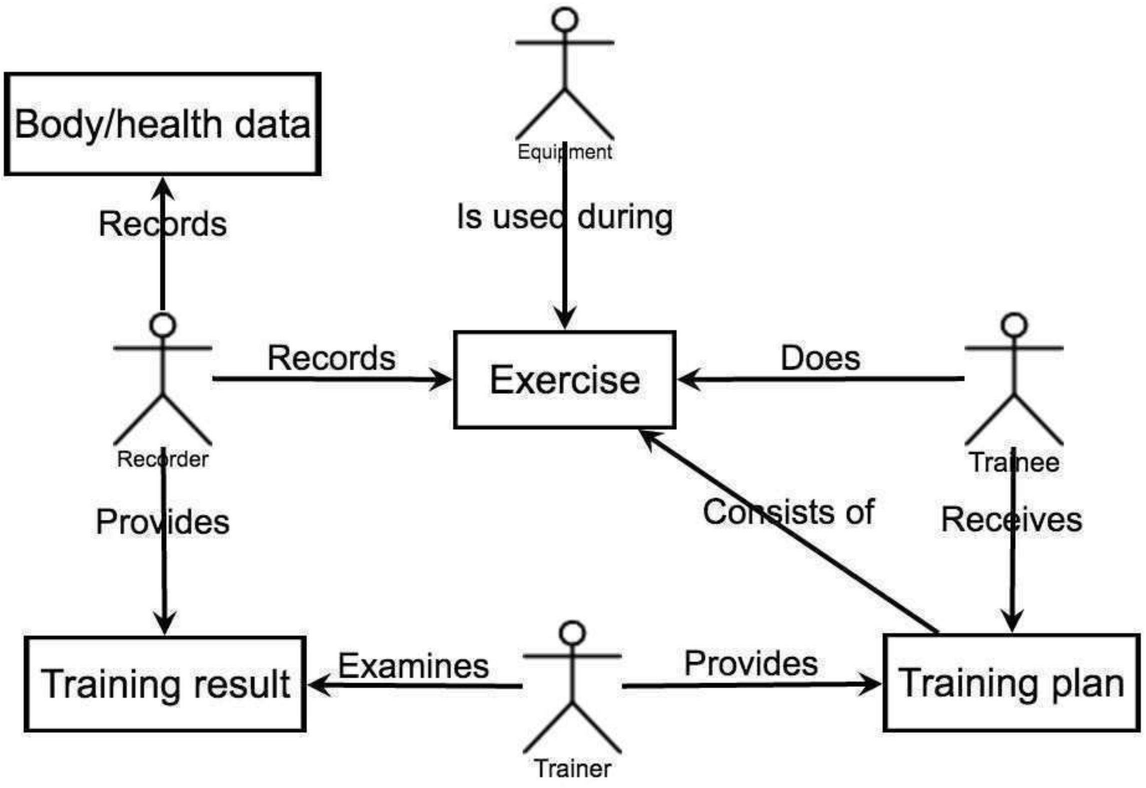
Organization model

Trainee being the single human role has the control over other non-human roles. Meaning its requests are always being complied with. Recorder is also controlled by the Trainer, as Trainer needs recorded data to improve Trainee's training plan.



Domain model

The Trainee receives a Training plan provided by the Trainer. Training plans consist of Exercises. The Trainee does the Exercises using some Equipment. During training the Recorder logs the completed Exercises, records Body/Health data and provides the Training result to the Trainer. The Trainer examines the Training result and makes a new Training plan.


Body/health data
Records
Equipment
Is used during
Recorder
Records
Exercise
Does
Trainee
Provides
Consists of
Receives
Training result
Examines
Provides
Training plan
Trainer

Design

Agent model

There are four agents in the system - Trainee human agent, Trainee software agent, Trainer agent and Equipment agent. Below are the detailed characteristics of the agents.

Agent name	Trainee human agent
Description	Human Agent. The one who wants to improve his/her physical health
Roles	Trainee
Responsibilities	• Defines goal • Goes to training • Does exercises provided by Trainer

Agent name	Trainee software agent
Description	Software Assistant of Trainee Human Agent
Roles	Trainee
Responsibilities	-

Agent name	Trainer agent
Description	Agent that is responsible for providing exercises, recording training and examining training result
Roles	• Trainer • Recorder
Responsibilities	• Provides Trainee a training plan • Provides information about exercise (how to do, etc.) • Provides Trainee an alternative exercise • Examine training results • Records done exercises (and their order) • Records Trainee's health/body data

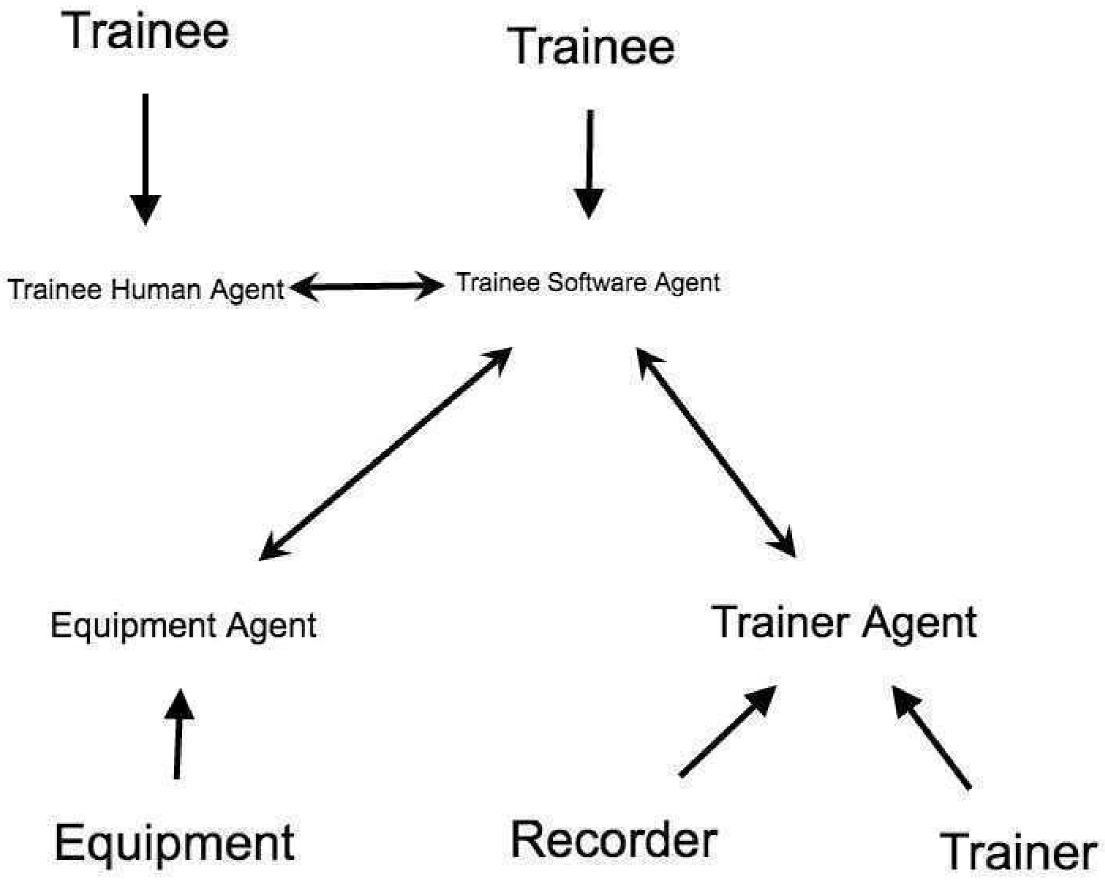
Agent name	Equipment agent
Description	Gym equipment agent
Roles	Equipment
Responsibilities	• Gives an opportunity to do exercise

Acquaintance model

The Trainee role is represented by two agents - a human agent representing a real person and a software agent, which forwards the actions of the human agent to other agents and displays information from them.

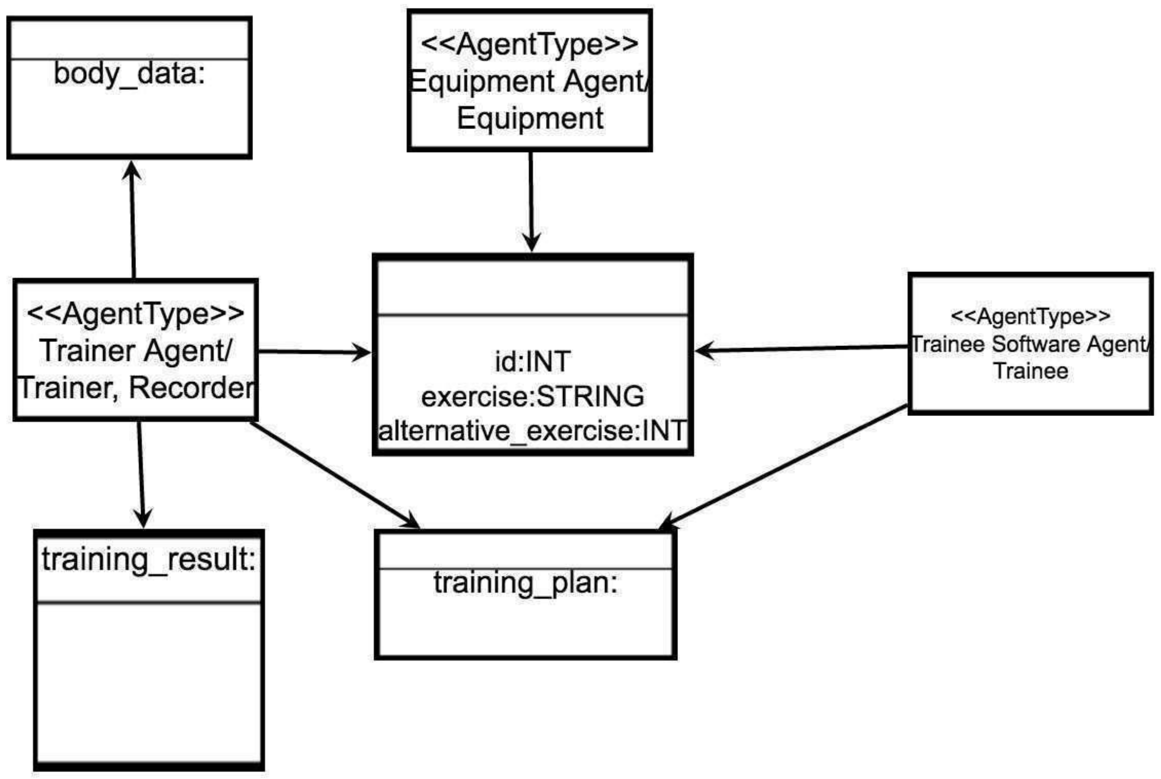
The Trainer agent has two roles - the Recorder and the Trainer and is typically comprised of a software application on a smartwatch or a smartphone.

The Equipment agent has a single role with the same name. Equipment is a single piece of gym equipment.



Knowledge model

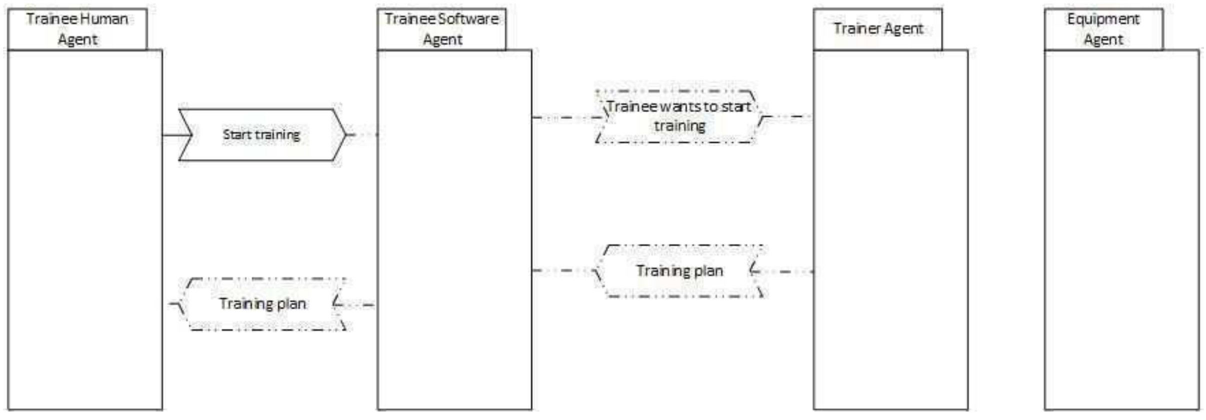
The following model contains the knowledge requirements of the agents. Only the basic attributes are shown.


body_data:
<<AgentType>> Equipment Agent/ Equipment
<<AgentType>> Trainer Agent/ Trainer, Recorder
id:INT
exercise:STRING
alternative_exercise:INT
<<AgentType>> Trainee Software Agent/ Trainee
training_result:
training_plan:

Interaction models

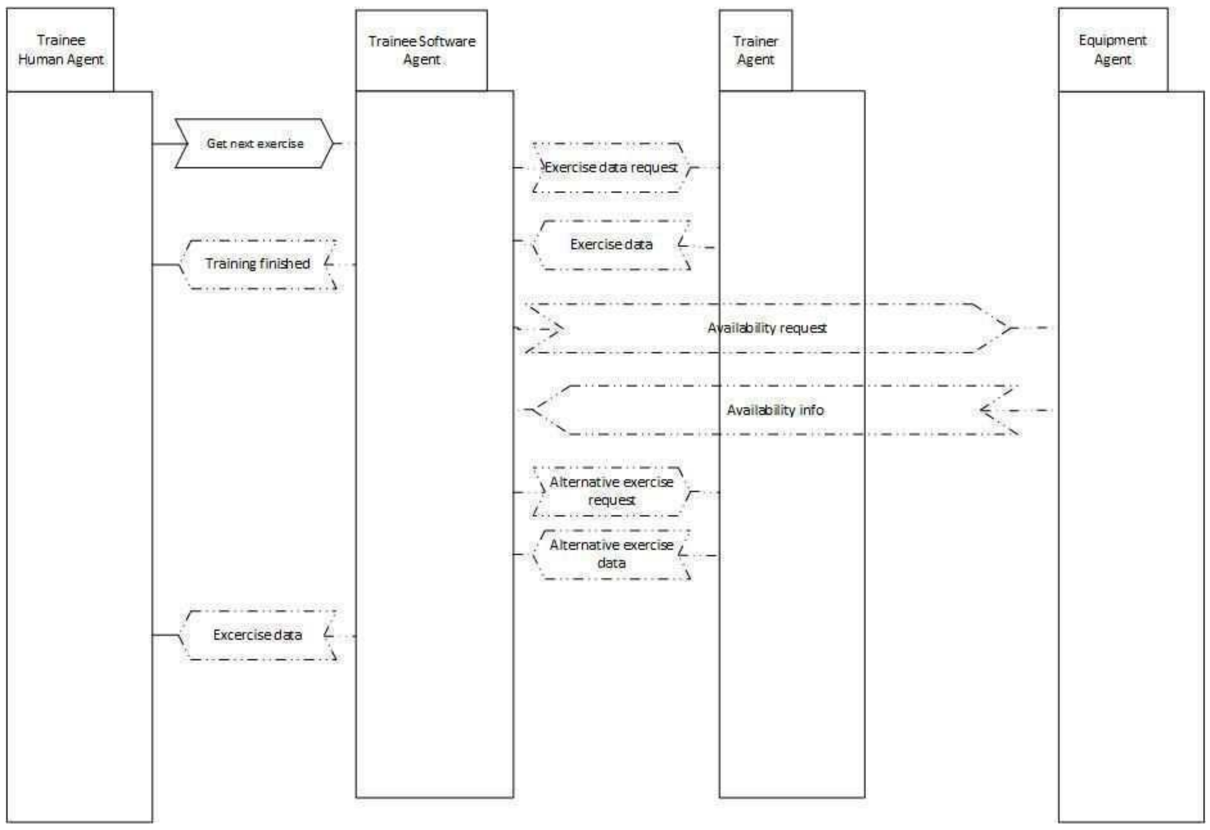
The basic interactions between human and software agents are shown below. Three interaction models were chosen, that capture the key functionality of the project - adaptive training.

Beginning of training



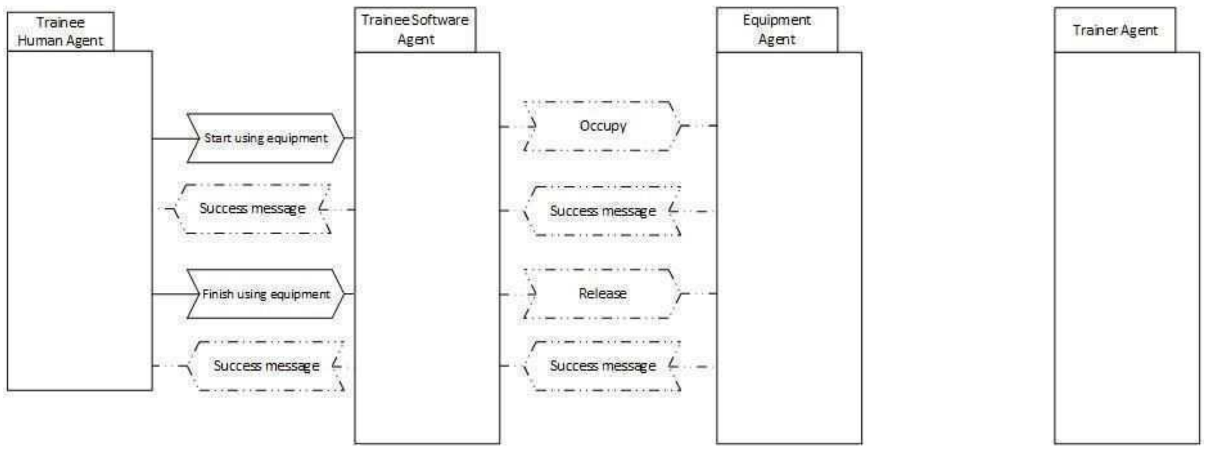
This interaction model shows beginning of the training. After Trainee reached the gym, he tells the Trainer (some wearable device or smartphone) that he wants to start training. All this is done through some UI (Trainee Software Agent). In response Trainer sends an overview of today’s training to the Trainee, what is also displayed through some UI.

Exercise selection



Given model is a sequel of previous one (Beginning of training). After Trainee receives today’s plan, he requests information about first exercise from Trainer. This information contains different types of media that describe how and using what gym equipment the exercise has to be done. Before Trainee can start exercising, he must to make sure that necessary gym equipment is available. If it is not available, then Trainee (Software Agent) asks for alternative exercise from the Trainer. Then the process is repeated until there is no more exercises left to do.

Exercise

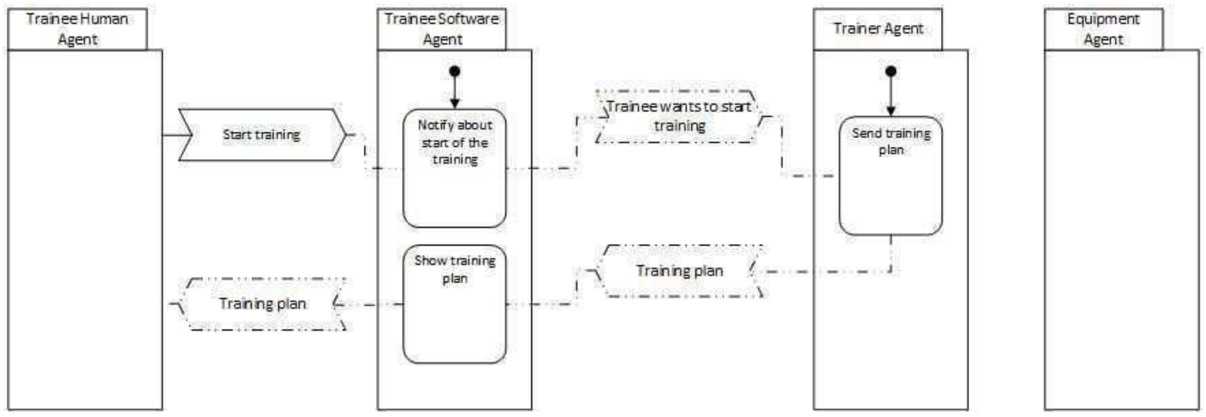


After Trainee receives all needed information, he can finally start exercising. Trainee occupies the gym equipment, does the exercise and, finally, releases the equipment, so others could also use it.

Behaviour models

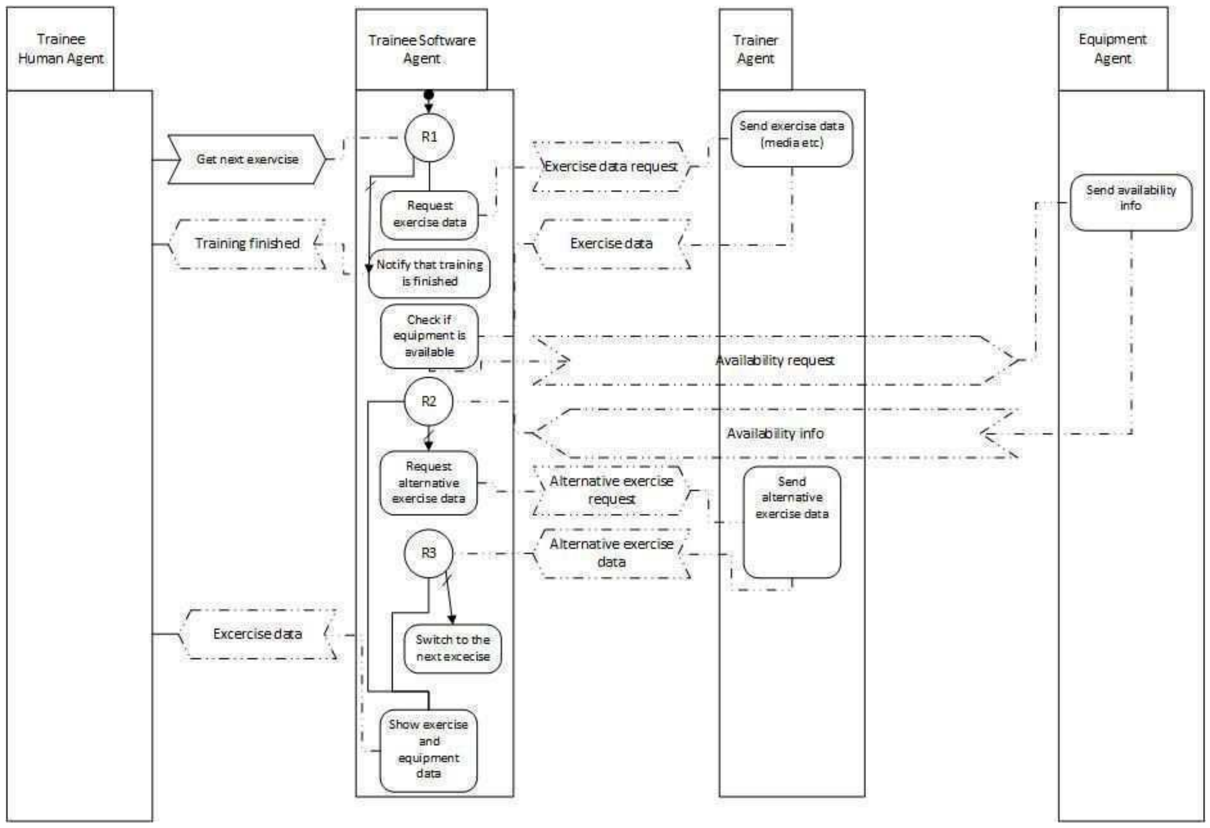
Following are the behavior models for the corresponding interaction models together with rule descriptions. Description of the models can be found in previous section (Interaction models).

Beginning of training



Given model shows beginning of the training. Equipment Agent does not participate in this phase.

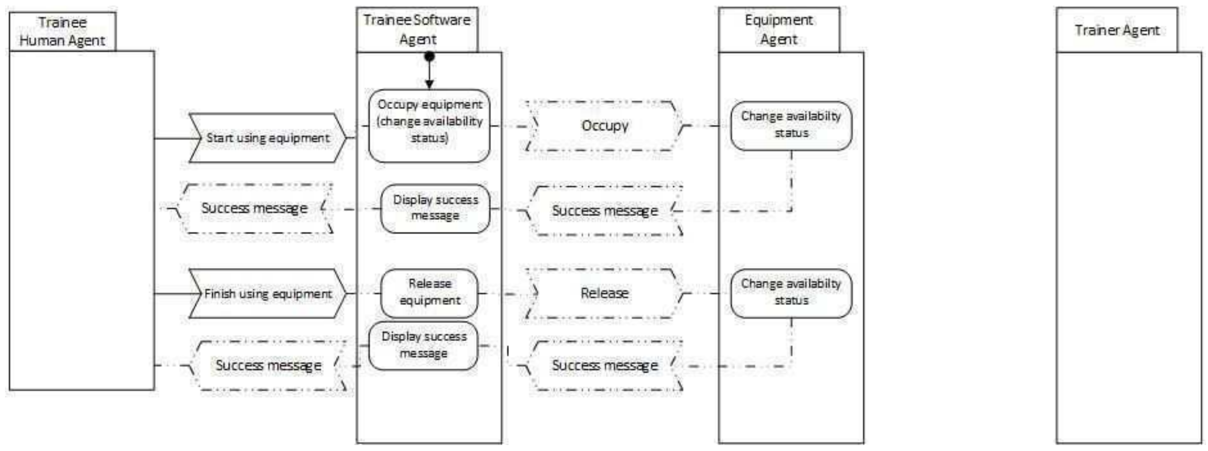
Exercise selection


Trainee Human Agent
Trainee Software Agent
Trainer Agent
Equipment Agent
R1
Get next exercise
Request exercise data
Exercise data request
Send exercise data (media etc)
Exercise data
Training finished
Notify that training is finished
Send availability info
Check if equipment is available
Availability request
R2
Availability info
Request alternative exercise data
Alternative exercise request
Send alternative exercise data
R3
Alternative exercise data
Excercise data
Switch to the next excecise
Show exercise and equipment data

Given model show exercise selection logic.

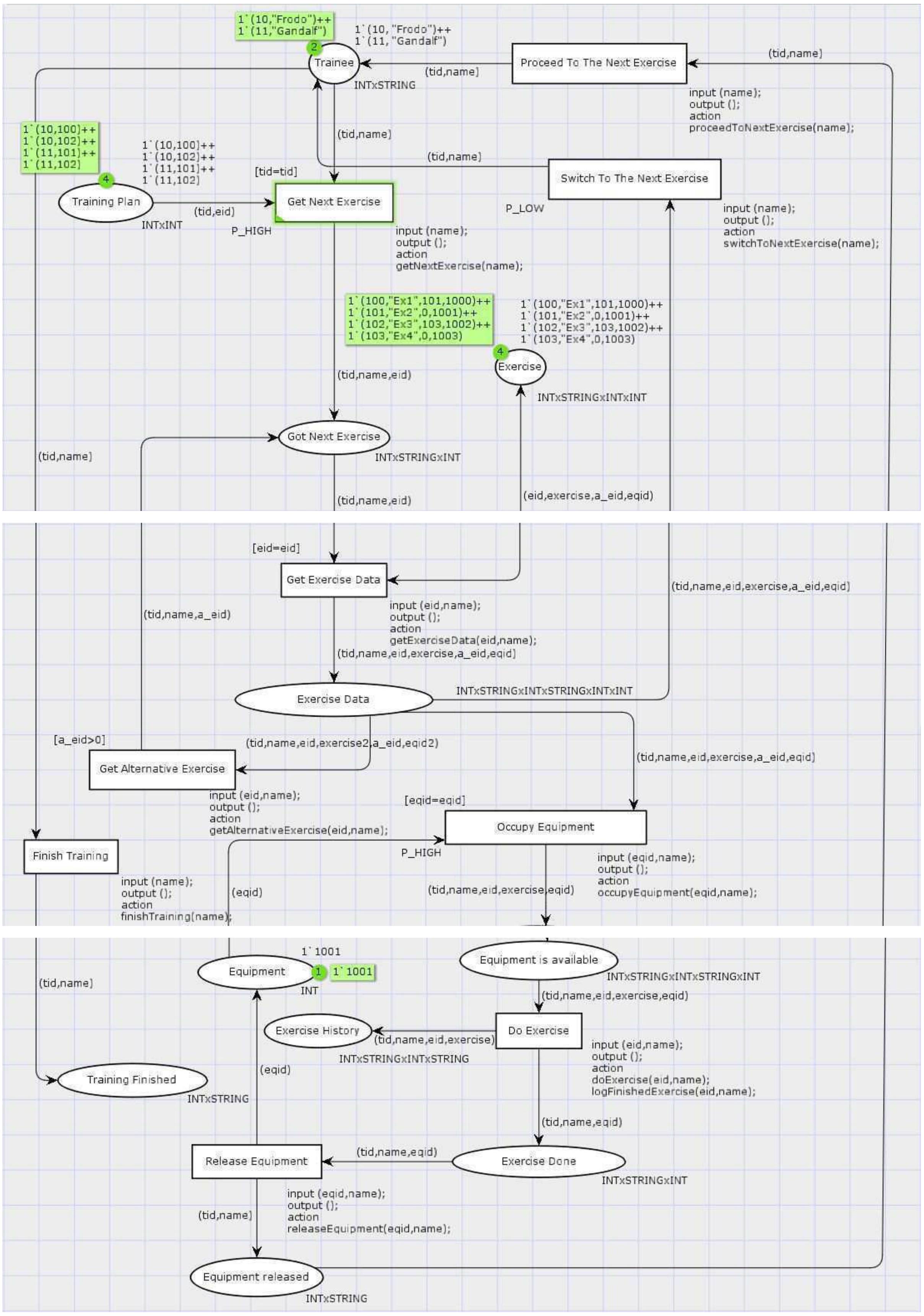
Rules:
R1 - Check if there are exercises left in today’s training plan.
R2 - If gym equipment is taken, ask for alternative exercise from Trainer.
R3 - If gym equipment for alternative exercise is also taken, switch to the next exercise.

Exercise



Given model shows training process. Trainer Agent does not participate in this phase.

CPN simulation



```
input (name);
output ();
action
proceedToNextExercise(name);
```

```
input (name);
output ();
action
switchToNextExercise(name);
```

```
input (name);
output ();
action
getNextExercise(name);
```

```
input (eid,name);
output ();
action
getExerciseData(eid,name);
```

```
input (eid,name);
output ();
action
getAlternativeExercise(eid,name);
```

```
input (eqid,name);
output ();
action
occupyEquipment(eqid,name);
```

```
input (name);
output ();
action
finishTraining(name);
```

```
input (eid,name);
output ();
action
doExercise(eid,name);
logFinishedExercise(eid,name);
```

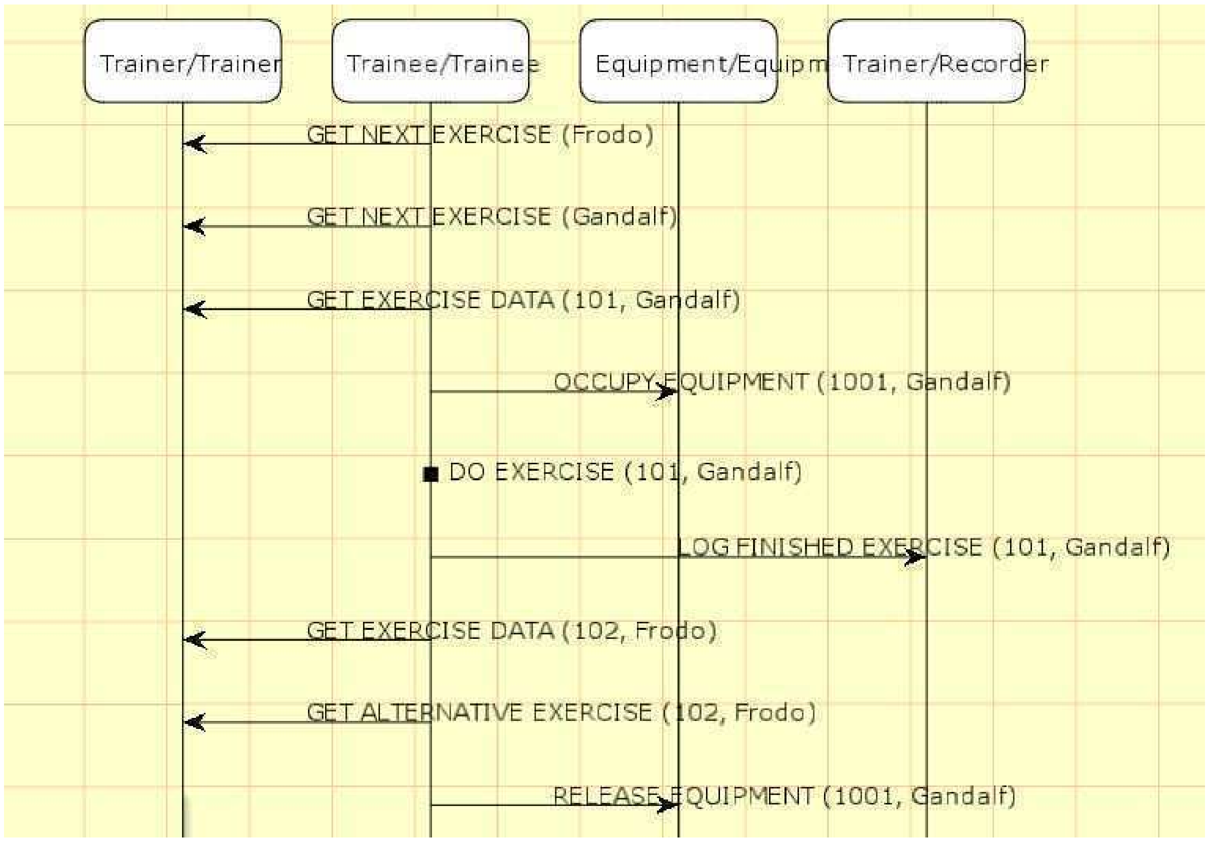
```
input (eqid,name);
output ();
action
releaseEquipment(eqid,name);
```

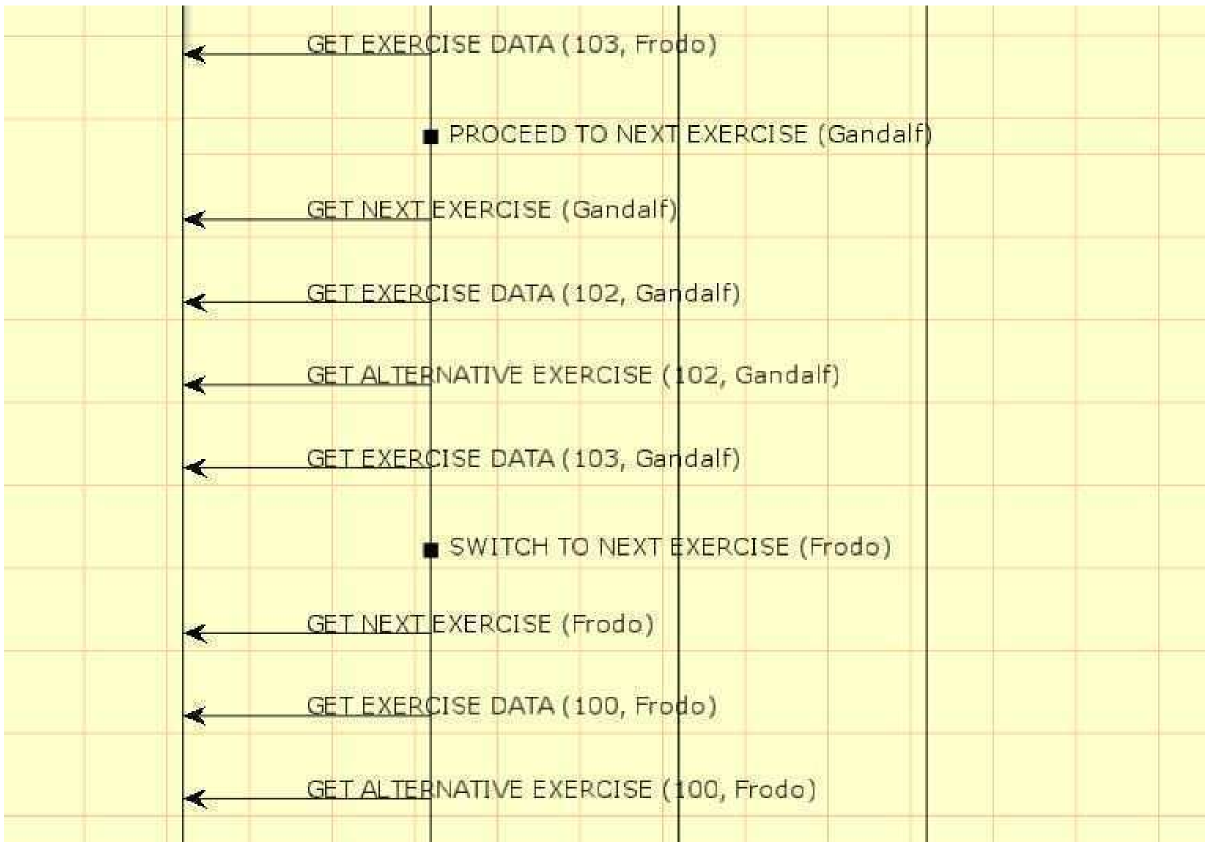
Validation

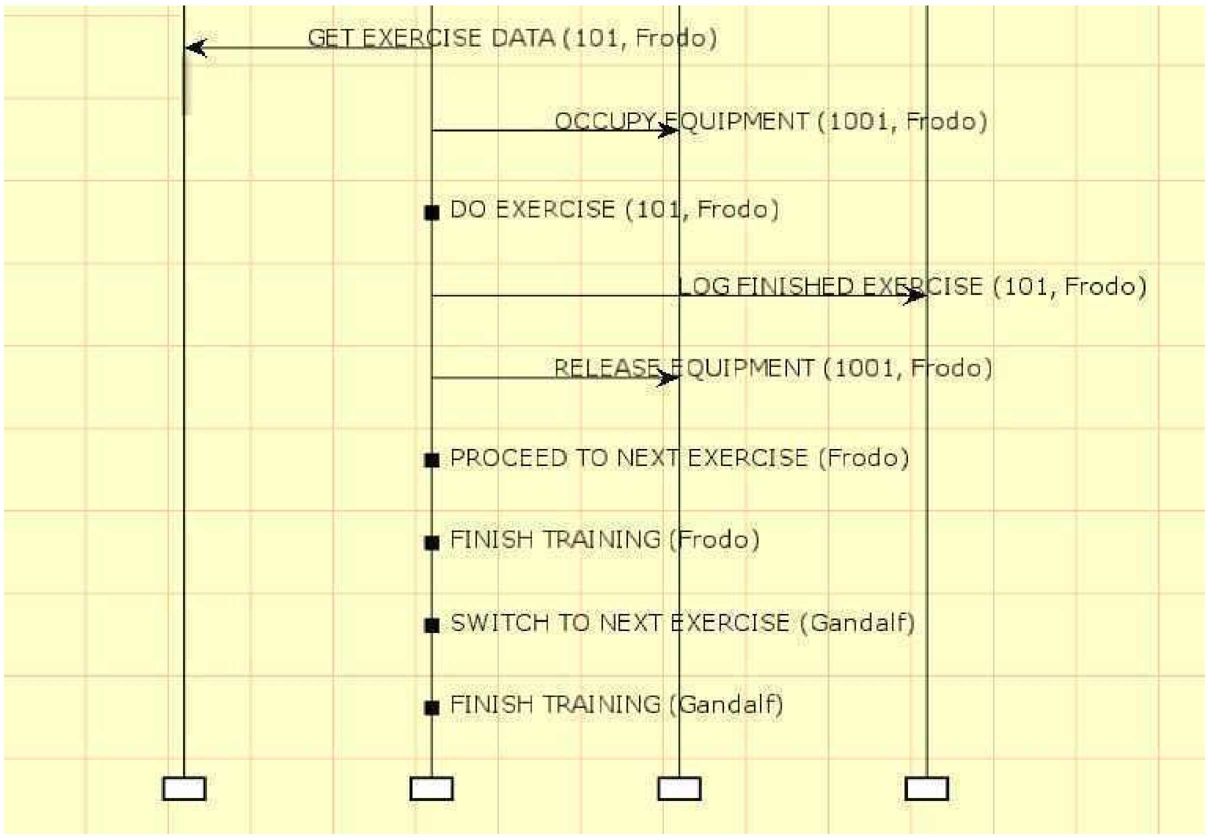
For the mini project 3 different scenarios were chosen that describe situations that can happen during the training. Scenarios are following:

1. Trainee wants to start exercise that is provided by Trainer using specified equipment. This Equipment is available at the gym and is not held by other trainees at the moment. Trainee successfully performs the exercise.
2. Trainee wants to start exercise that is provided by Trainer using specified equipment, but he can't because there is no available equipment. Trainee asks for alternative exercise. The Equipment is available and Trainee successfully performs alternative exercise.
3. Trainee wants to perform exercise, but Equipment used in this and alternative exercise are taken and Trainee has to switch to the next exercise.

Sequence diagram


Trainer/Trainer
Trainee/Trainee
Equipment/Equipm
Trainer/Recorder
GET NEXT EXERCISE (Frodo)
GET NEXT EXERCISE (Gandalf)
GET EXERCISE DATA (101, Gandalf)
OCCUPY EQUIPMENT (1001, Gandalf)
DO EXERCISE (101, Gandalf)
LOG FINISHED EXERCISE (101, Gandalf)
GET EXERCISE DATA (102, Frodo)
GET ALTERNATIVE EXERCISE (102, Frodo)
RELEASE EQUIPMENT (1001, Gandalf)


GET EXERCISE DATA (103, Frodo)
PROCEED TO NEXT EXERCISE (Gandalf)
GET NEXT EXERCISE (Gandalf)
GET EXERCISE DATA (102, Gandalf)
GET ALTERNATIVE EXERCISE (102, Gandalf)
GET EXERCISE DATA (103, Gandalf)
SWITCH TO NEXT EXERCISE (Frodo)
GET NEXT EXERCISE (Frodo)
GET EXERCISE DATA (100, Frodo)
GET ALTERNATIVE EXERCISE (100, Frodo)


GET EXERCISE DATA (101, Frodo)
OCCUPY EQUIPMENT (1001, Frodo)
DO EXERCISE (101, Frodo)
LOG FINISHED EXERCISE (101, Frodo)
RELEASE EQUIPMENT (1001, Frodo)
PROCEED TO NEXT EXERCISE (Frodo)
FINISH TRAINING (Frodo)
SWITCH TO NEXT EXERCISE (Gandalf)
FINISH TRAINING (Gandalf)

Verification

Applying CPN Tools on our mini project we obtain the following result from Report:

Home Markings: [274],
Dead Marking: [274],
Dead Transition Instances: None
Live Transition Instances: None
Fairness Properties: No infinite occurrence sequences

The part of the state space report shown above says that there is one home and one dead marking. These markings have the same node number 274, which means that this node is both a home and a dead marking. One dead marking means that the CPN model is partially correct and if execution terminates then we have the correct result. Furthermore, because node 274 is also a home marking, it is always possible to terminate the protocol with the correct result. Also we can see that there are no infinite occurrence sequences. [4]

Report

```
CPN Tools state space report for:
/cygdrive/C/Users/Sergei/Desktop/cpn_07052016.cpn
Report generated: Sat May  7 22:25:53 2016


 Statistics
------------------------------------------------------------------------

  State Space
     Nodes:  274
     Arcs:   400
     Secs:   13
     Status: Full

  Scc Graph
     Nodes:  274
     Arcs:   400
     Secs:   0


 Boundedness Properties
------------------------------------------------------------------------

  Best Integer Bounds
                           Upper      Lower
     New_Page'Equipment 1    1          0
     New_Page'Equipment_is_available 1
                           1          0
     New_Page'Equipment_released 1
                           2          0
     New_Page'Exercise 1     4          4
     New_Page'Exercise_Data 1
                           2          0
     New_Page'Exercise_Done 1
                           1          0
     New_Page'Exercise_History 1
                         2          0
     New_Page'Got_Next_Exercise 1
                         2          0
     New_Page'Trainee 1      2          0
     New_Page'Training_Finished 1
                         2          0
     New_Page'Training_Plan 1
                         4          0

  Best Upper Multi-set Bounds
     New_Page'Equipment 1
                        1`1001
     New_Page'Equipment_is_available 1
                        1`(10,"Frodo",101,"Ex2",1001)++
1`(11,"Gandalf",101,"Ex2",1001)
     New_Page'Equipment_released 1
                        1`(10,"Frodo")++
1`(11,"Gandalf")
     New_Page'Exercise 1 1`(100,"Ex1",101,1000)++
1`(101,"Ex2",0,1001)++
1`(102,"Ex3",103,1002)++
1`(103,"Ex4",0,1003)
     New_Page'Exercise_Data 1
                        1`(10,"Frodo",100,"Ex1",101,1000)++
1`(10,"Frodo",101,"Ex2",0,1001)++
1`(10,"Frodo",102,"Ex3",103,1002)++
1`(10,"Frodo",103,"Ex4",0,1003)++
1`(11,"Gandalf",101,"Ex2",0,1001)++
1`(11,"Gandalf",102,"Ex3",103,1002)++
1`(11,"Gandalf",103,"Ex4",0,1003)
     New_Page'Exercise_Done 1
                        1`(10,"Frodo",1001)++
1`(11,"Gandalf",1001)
     New_Page'Exercise_History 1
                        1`(10,"Frodo",101,"Ex2")++
1`(11,"Gandalf",101,"Ex2")
     New_Page'Got_Next_Exercise 1
                        1`(10,"Frodo",100)++
1`(10,"Frodo",101)++
1`(10,"Frodo",102)++
1`(10,"Frodo",103)++
1`(11,"Gandalf",101)++
1`(11,"Gandalf",102)++
1`(11,"Gandalf",103)
     New_Page'Trainee 1  1`(10,"Frodo")++
1`(11,"Gandalf")
     New_Page'Training_Finished 1
                        1`(10,"Frodo")++
1`(11,"Gandalf")
     New_Page'Training_Plan 1
                        1`(10,100)++
1`(10,102)++
1`(11,101)++
1`(11,102)

  Best Lower Multi-set Bounds
     New_Page'Equipment 1
                       empty
     New_Page'Equipment_is_available 1
                       empty
     New_Page'Equipment_released 1
```

```
                        empty
     New_Page'Exercise 1 1`(100,"Ex1",101,1000)++
1`(101,"Ex2",0,1001)++
1`(102,"Ex3",103,1002)++
1`(103,"Ex4",0,1003)
     New_Page'Exercise_Data 1
                        empty
     New_Page'Exercise_Done 1
                        empty
     New_Page'Exercise_History 1
                        empty
     New_Page'Got_Next_Exercise 1
                        empty
     New_Page'Trainee 1  empty
     New_Page'Training_Finished 1
                        empty
     New_Page'Training_Plan 1
                        empty


 Home Properties
------------------------------------------------------------------------

  Home Markings
     [274]


 Liveness Properties
------------------------------------------------------------------------

  Dead Markings
     [274]

  Dead Transition Instances
     None

  Live Transition Instances
     None


 Fairness Properties
------------------------------------------------------------------------
     No infinite occurrence sequences.
```

Conclusion

During the project, all of the project requirements were achieved. Different level analysis and design models were constructed, core functionality CPN simulation was created. During the implementation of the project team members were introduced to web-based agent-oriented modelling tool and CPN Tools software package.

CPN Tools has a nice idea of model validation and verification, but is severely lacking a modern user interface and a decent documentation. The software package is not widely used, thus it is impossible to compensate the lack of proper documentation with online research on resources such as stackoverflow.com.

Most of the time using CPN Tools was spent on trying to find out how to implement the simplest programming constructs (e.g. loops and branches) for the most basic business logic, and refactoring (simplifying) the AOM model when the CPN created was becoming too complex. CPN Tools software package aims to be cross-platform but is not working properly on anything other than Microsoft Windows. Out of three team members only one uses Windows as the main OS.

Overall the team members find the usage of CPN Tools package counter-productive and suggest finding at least an alternative editor for CPN files. Also more complex CPN examples should be provided on course home page.

Nevertheless the members are pleased with the project and the experience gained from making it.

References

1. Laboratory of Socio-Technical Systems. "Agent-Oriented Modelling and Multiagent Systems (2016)". [WWW] http://maurus.ttu.ee/sts/?page_id=2222 (08.05.2016)
2. CPN Tools. “Documentation”. [WWW] http://cpntools.org/documentation/start (08.05.2016)
3. Sterling, Leon S., and Kuldar Taveter. The Art of Agent-Oriented Modeling (2009). The MIT Press.
4. Kurt Jensen, Lars M. Kristensen. Coloured Petri Nets: Modelling and Validation of Concurrent Systems (2009). Springer.
5. Bogdan Aman, Gabriel Ciobanu. Mobility in Process Calculi and Natural Computing (2011). Springer.