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

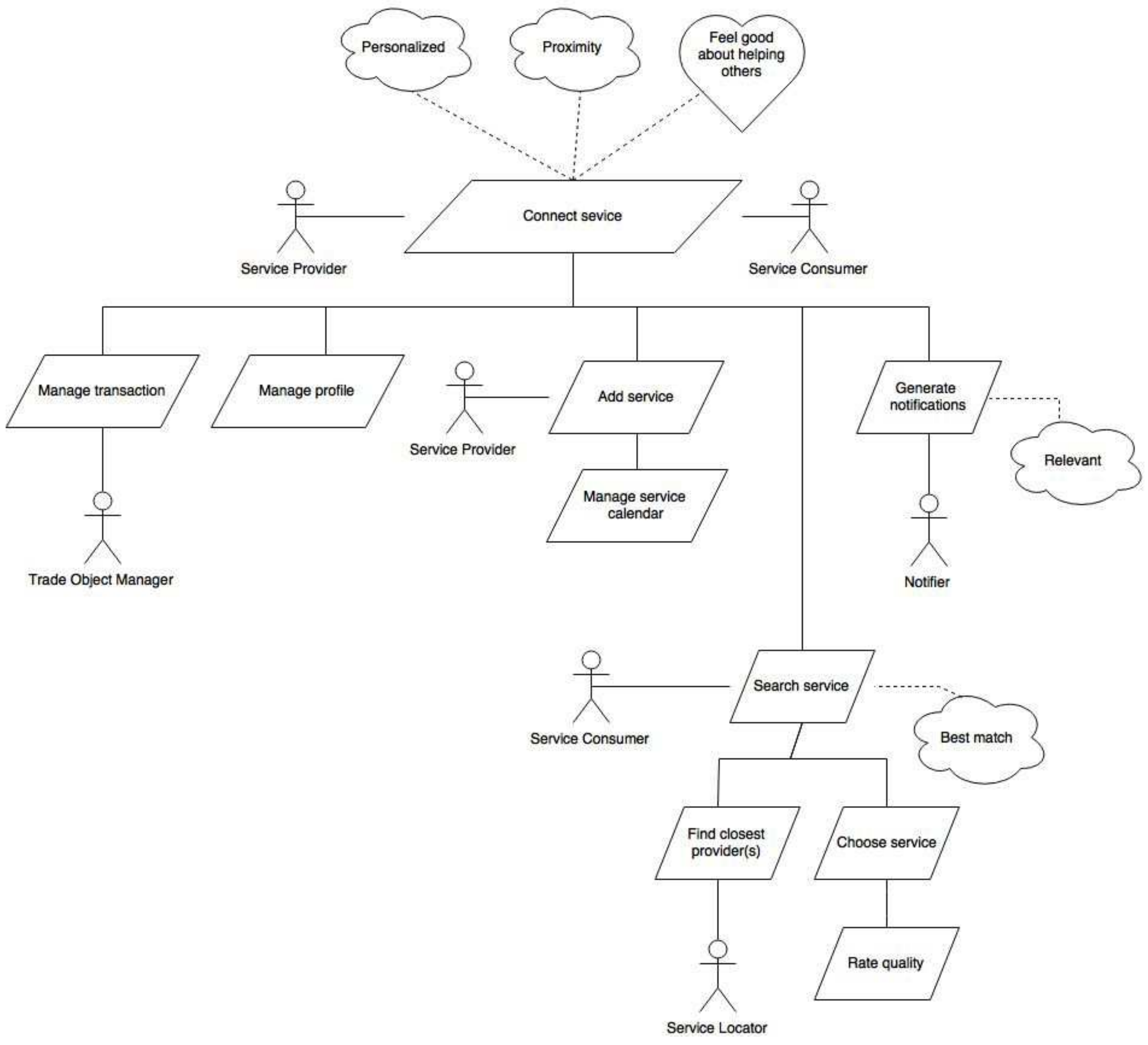
This section contains the following models: Goal model, Role model, Organizational model and Domain model.

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

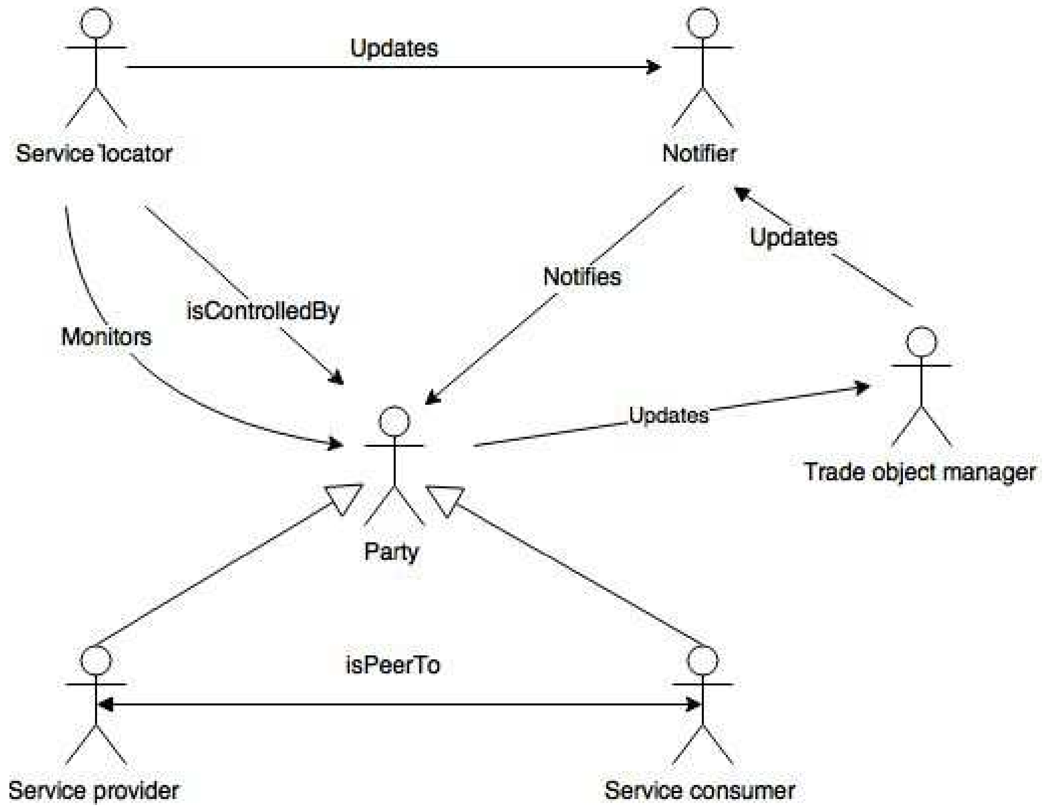
Role name	Service Locator
Description	The role of the GPS service
Responsibilities	Return coordinates of current location
Constraints	GPS services should be enabled on the device The application must be installed, where to return the coordinates

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*.



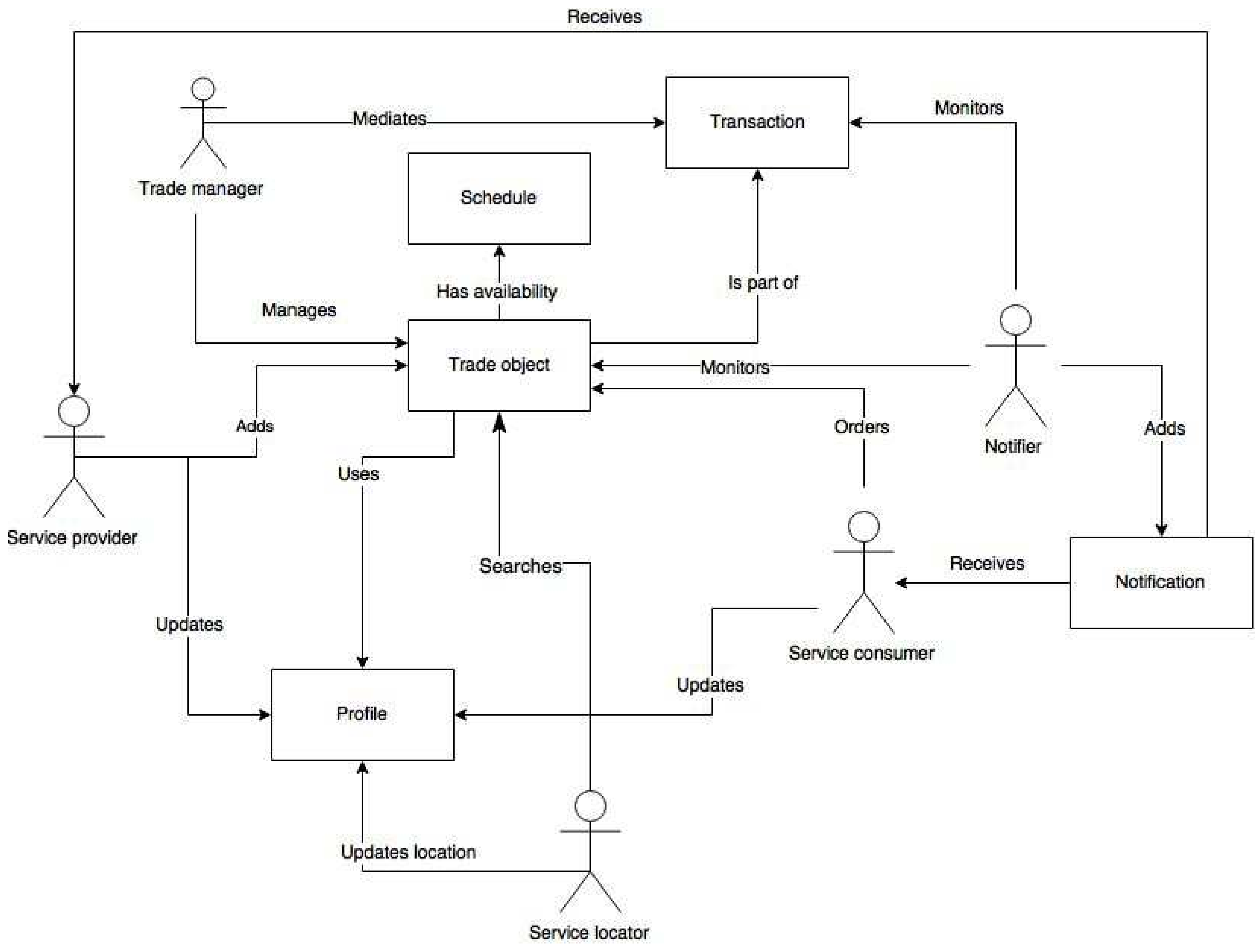
At the center of our service are service providers and service consumers. A provider can also be a consumer and vice versa. Service providers and consumers are monitored by the service locator. That information is then passed on to the Notifier, who decides whether to notify a particular party or not. 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.

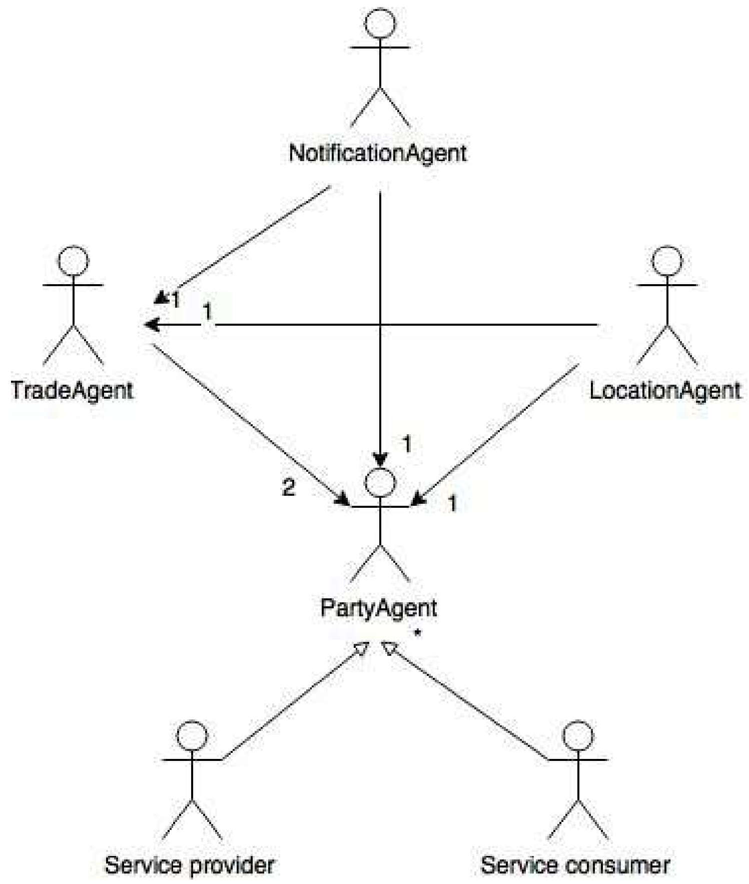


3. System design layer

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

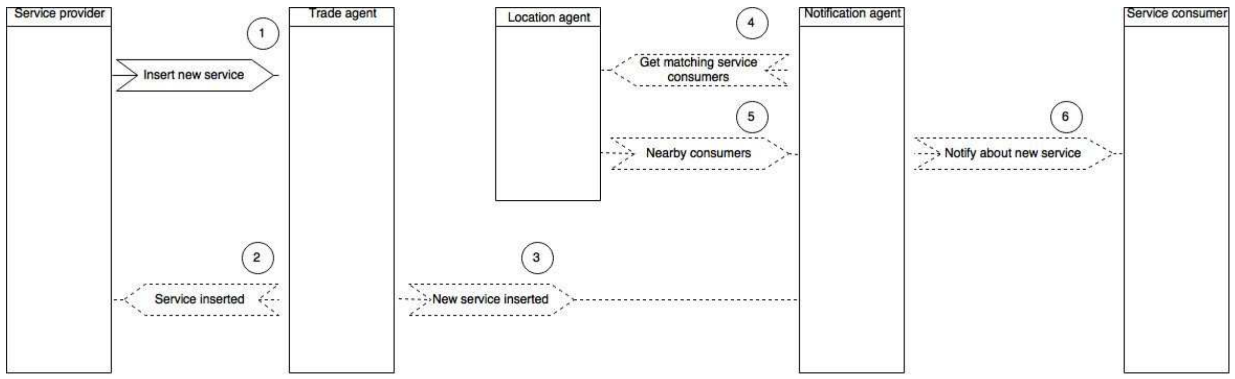
Agent name	Trade agent
Description	Trade object management software component
Roles	Trade object manager
Responsibilities	Manage transactions Manage trade objects and states

3.2 Interaction models

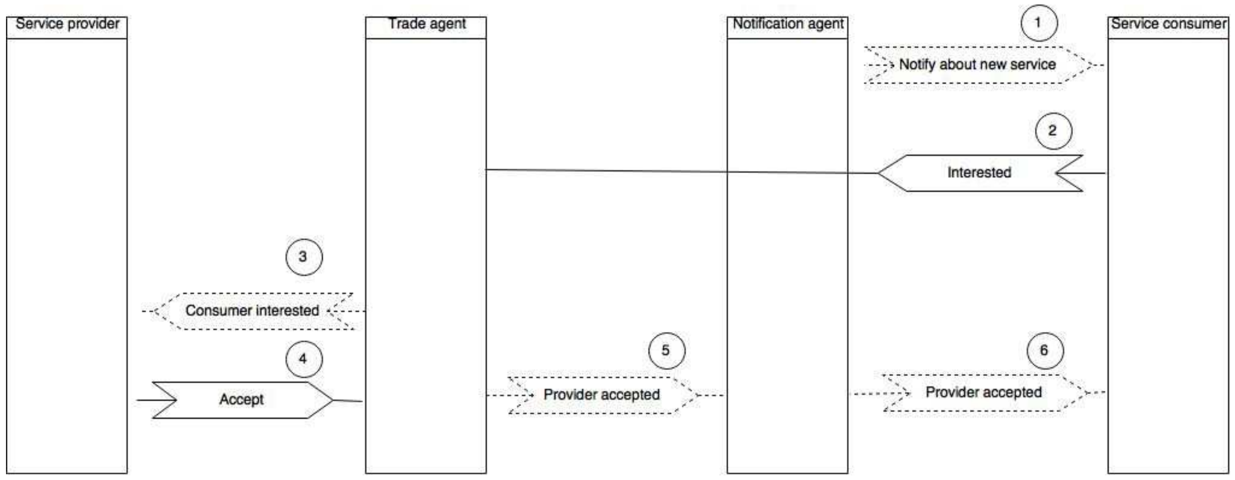
The interaction links described in section 3.1 show which agents interact with other agents and which party can initiate an interaction. Interaction model represent interaction patterns between agents in more detail. Interaction-sequence diagram models prototypical interactions as action events. Action 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.

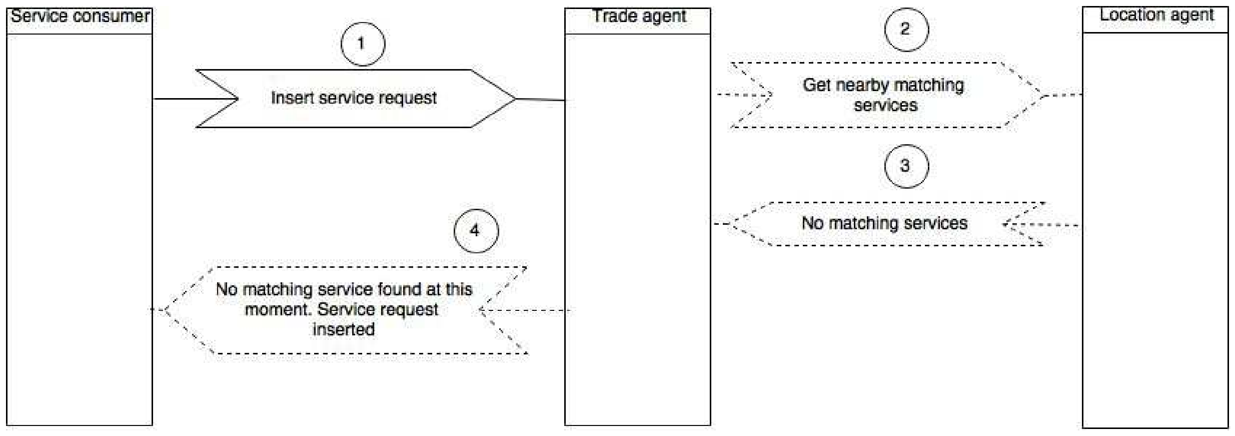
3.2.1 Service providing



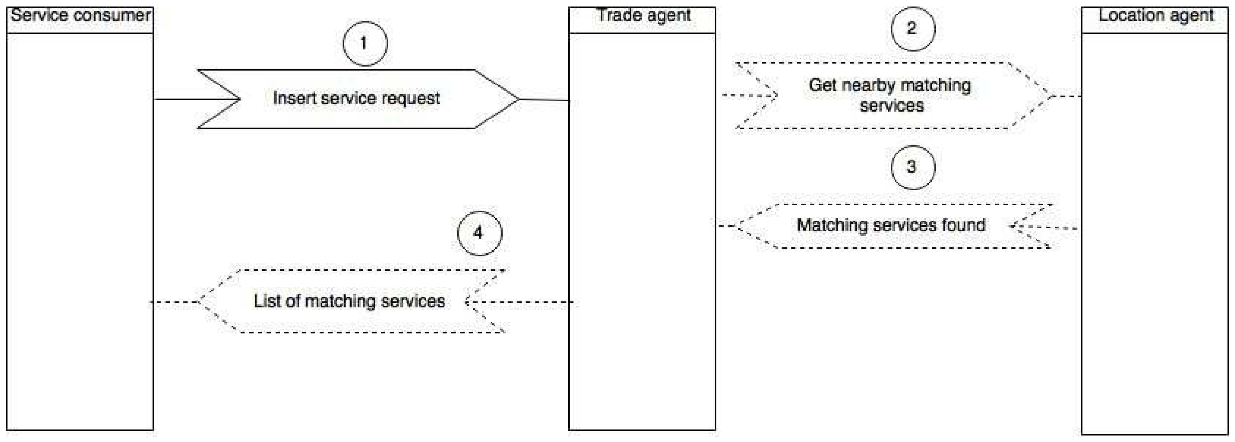
3.2.2 New service availability notification



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

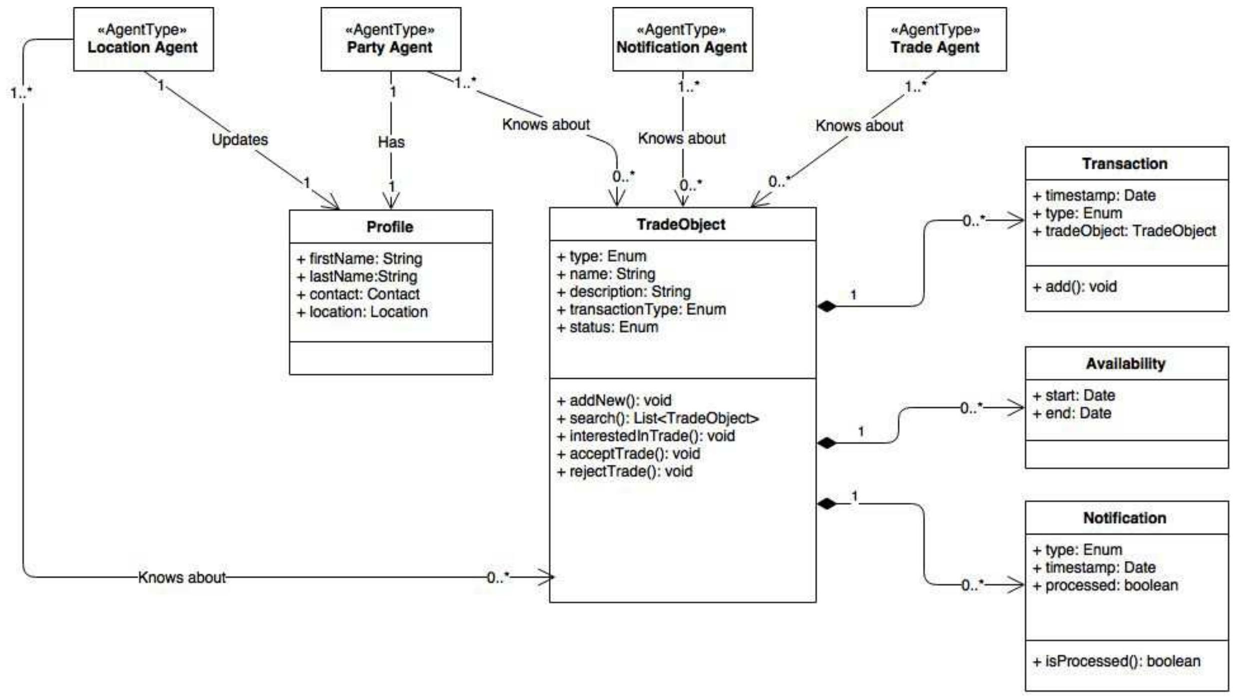


3.3 Knowledge model

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.

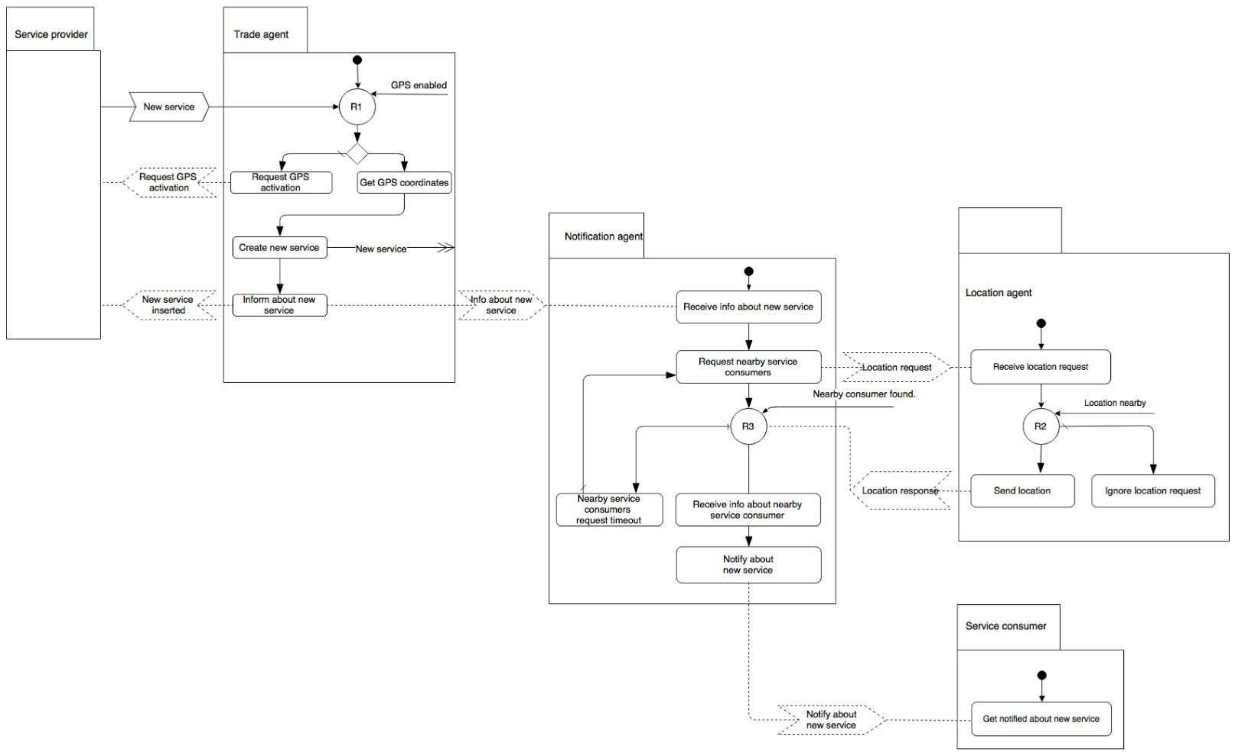
The trade object contains the following information: a type enum, name, description, transaction type, status. Enum is a data type consisting of a set of named values called elements, members, enumeral, or enumerators of the type of trade object it is (service, physical object etc.). The trade object 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.

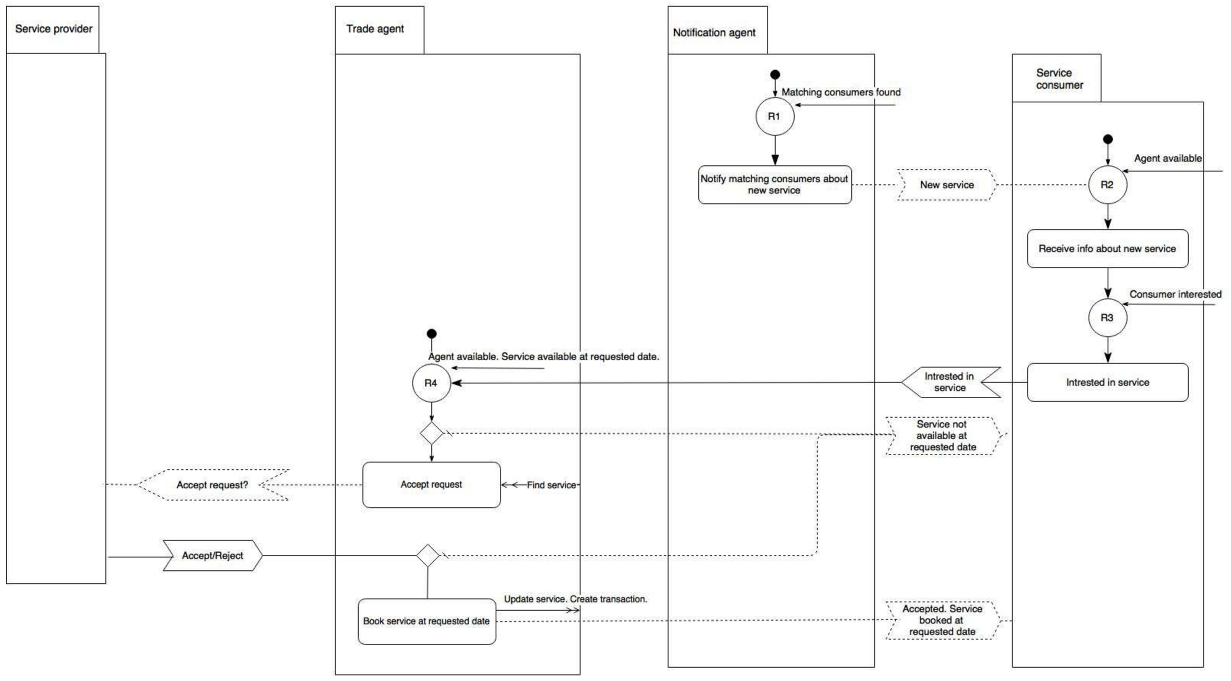
We are only interested in nearby consumers because it makes much more sense to suggest a new service to them, than to someone who is halfway around the world, too far to actually use the service. The only time distance doesn't matter, is when the service in question is virtual and doesn't require consumer/provider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

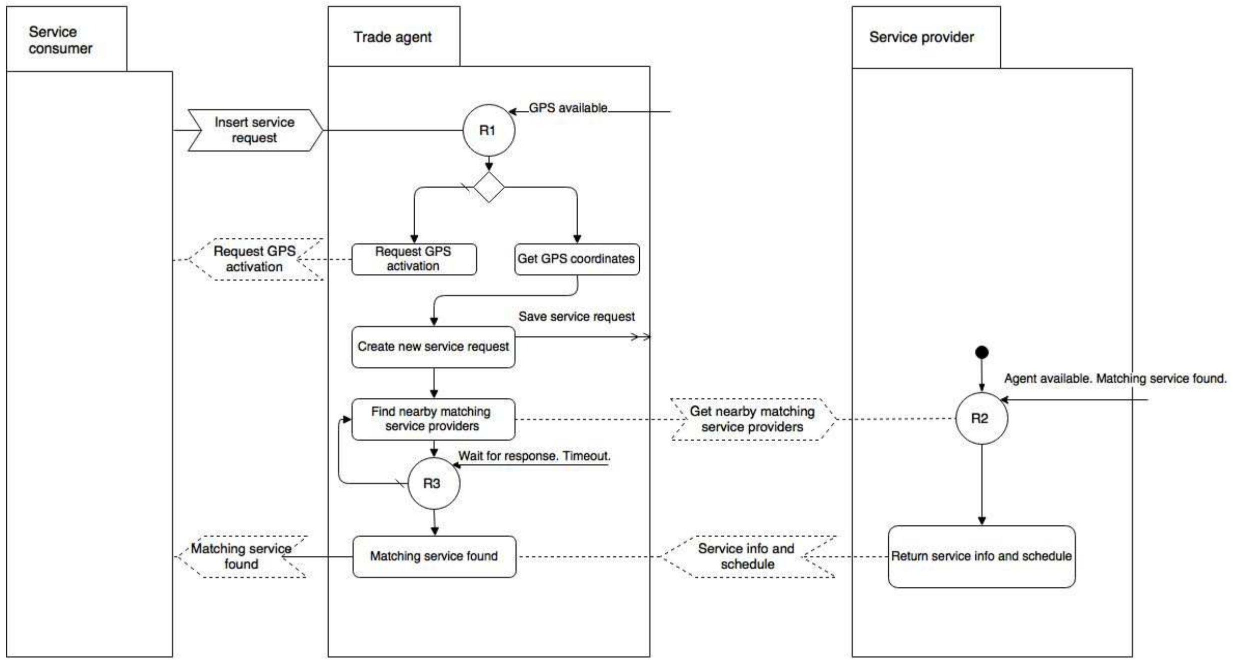
R1 - If matching consumers are found, notify them about new service.

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
Agent available. Matching service found.
Find nearby matching service providers
Get nearby matching service providers
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.

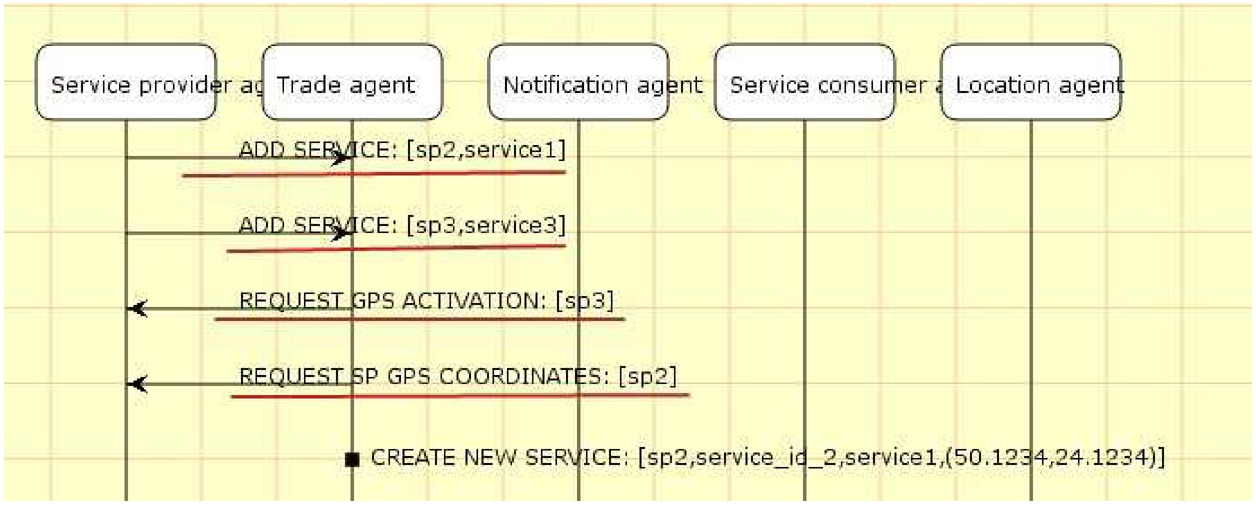
4. Analysis in CPN Tools

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

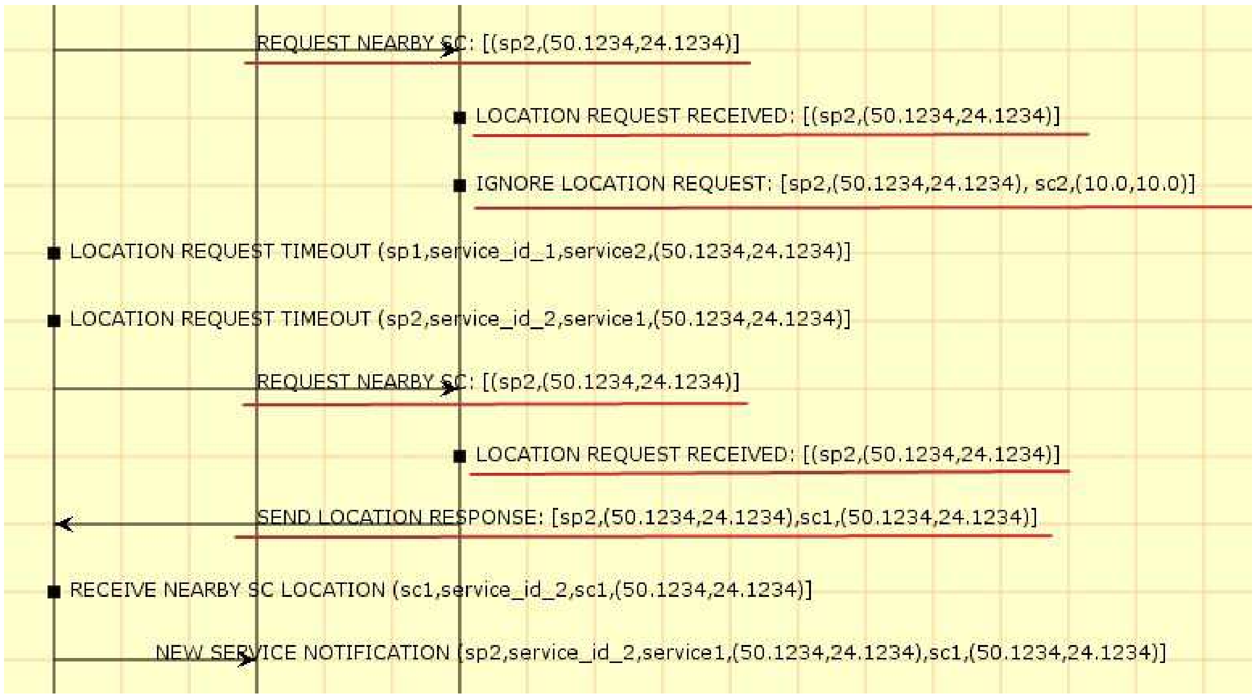
Description: Provider submits a new service to Trade agent. Trade agent checks if provider has their GPS enabled. If the GPS is enabled, Trade agent gets provider's coordinates and inserts new service along with these coordinates into the system. If GPS is not enabled, Trade agent asks provider to enable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



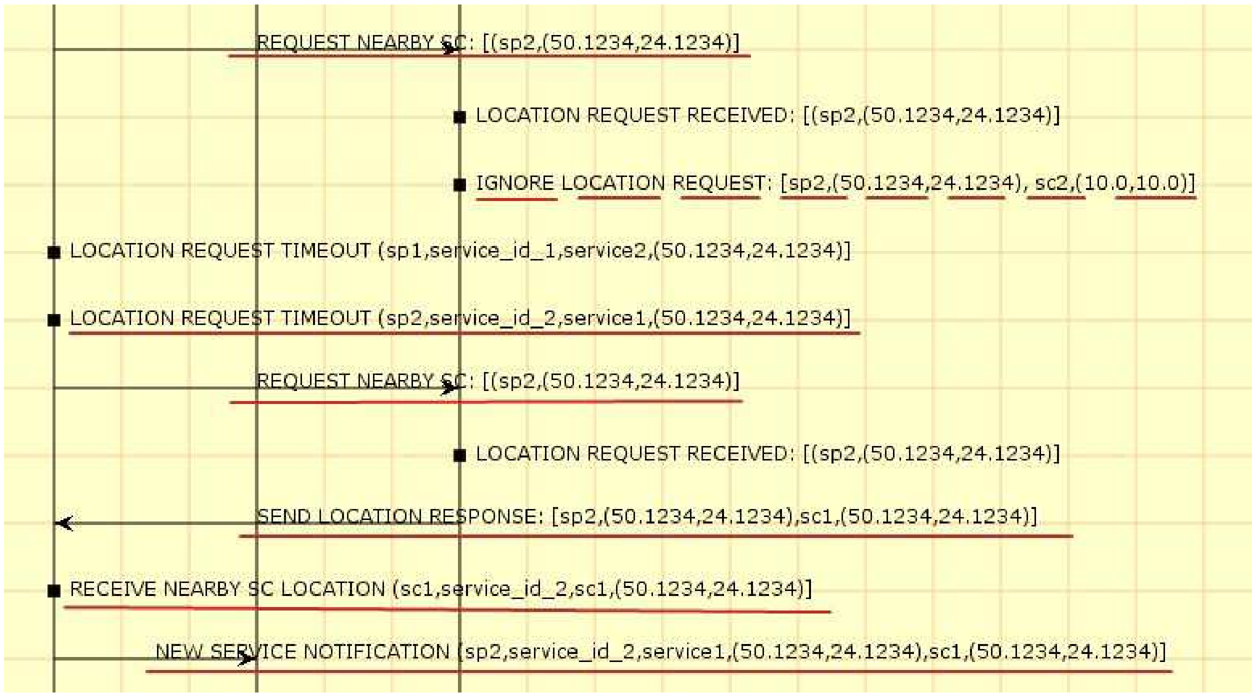
```
REQUEST NEARBY SC: [(sp2,(50.1234,24.1234)]
LOCATION REQUEST RECEIVED: [(sp2,(50.1234,24.1234)]
IGNORE LOCATION REQUEST: [sp2,(50.1234,24.1234), sc2,(10.0,10.0)]
LOCATION REQUEST TIMEOUT (sp1,service_id_1,service2,(50.1234,24.1234)]
LOCATION REQUEST TIMEOUT (sp2,service_id_2,service1,(50.1234,24.1234)]
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.

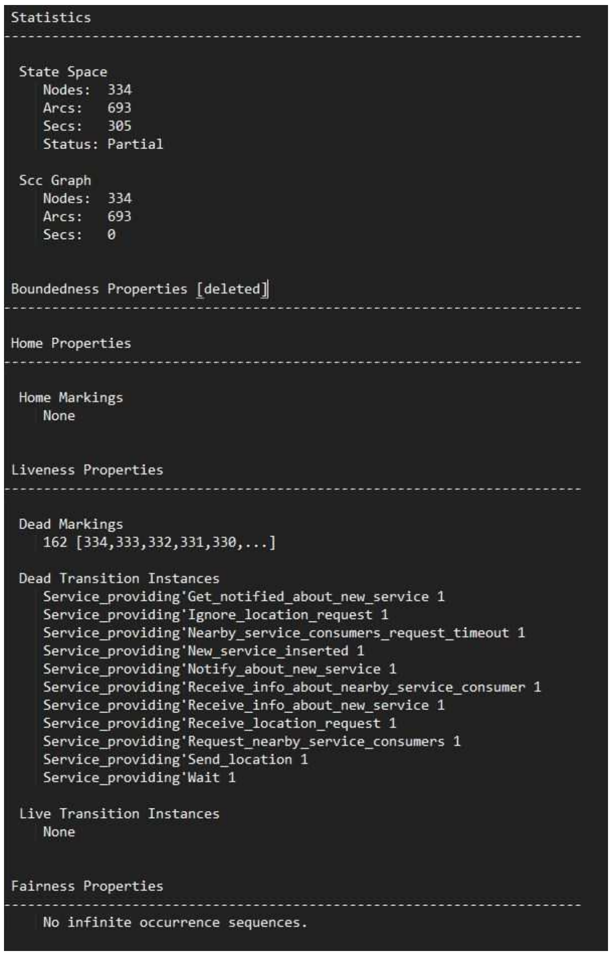


```
REQUEST NEARBY SC: [(sp2,(50.1234,24.1234)]
LOCATION REQUEST RECEIVED: [(sp2,(50.1234,24.1234)]
IGNORE LOCATION REQUEST: [sp2,(50.1234,24.1234), sc2,(10.0,10.0)]
LOCATION REQUEST TIMEOUT (sp1,service_id_1,service2,(50.1234,24.1234)]
LOCATION REQUEST TIMEOUT (sp2,service_id_2,service1,(50.1234,24.1234)]
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.

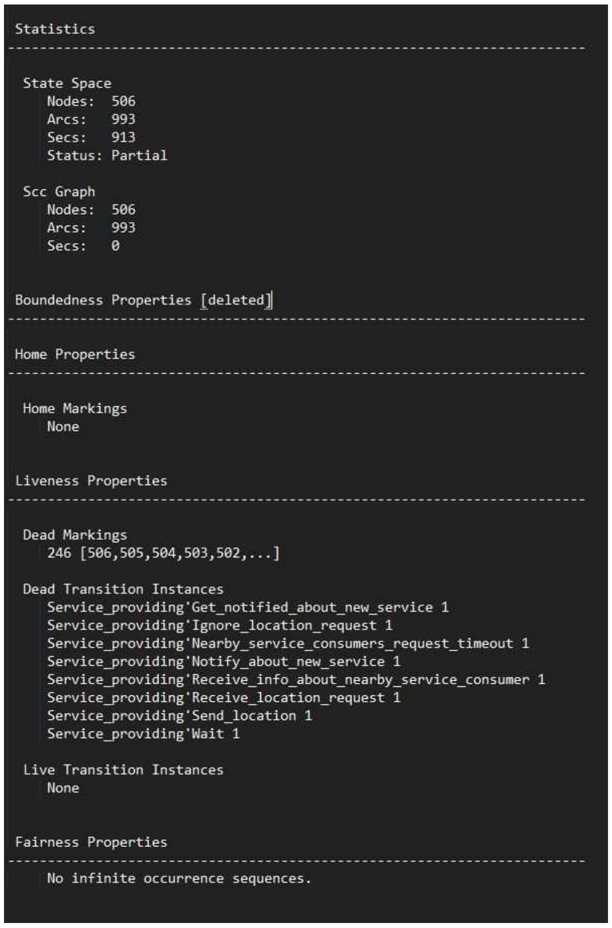
4.2 Verification

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

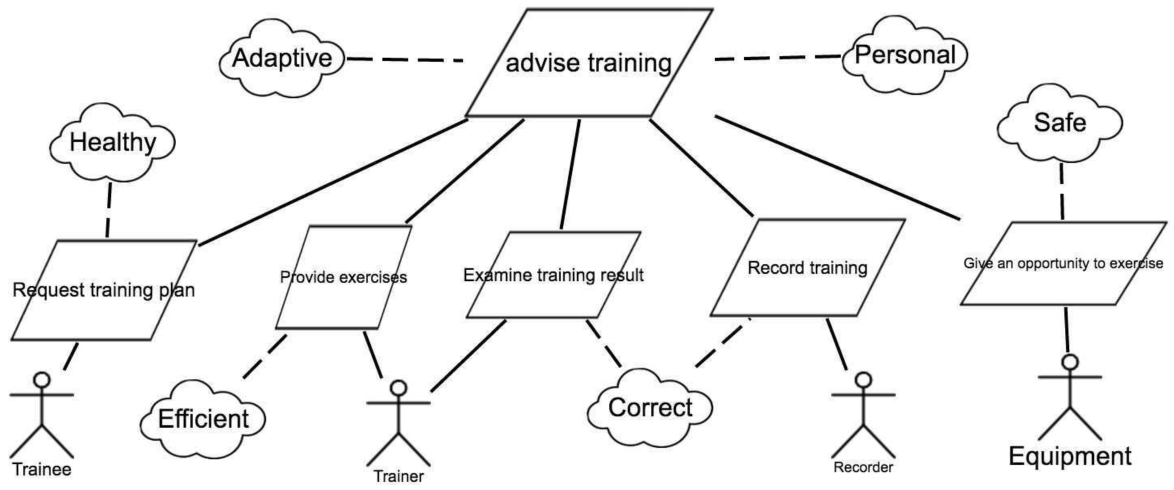
Goal model

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

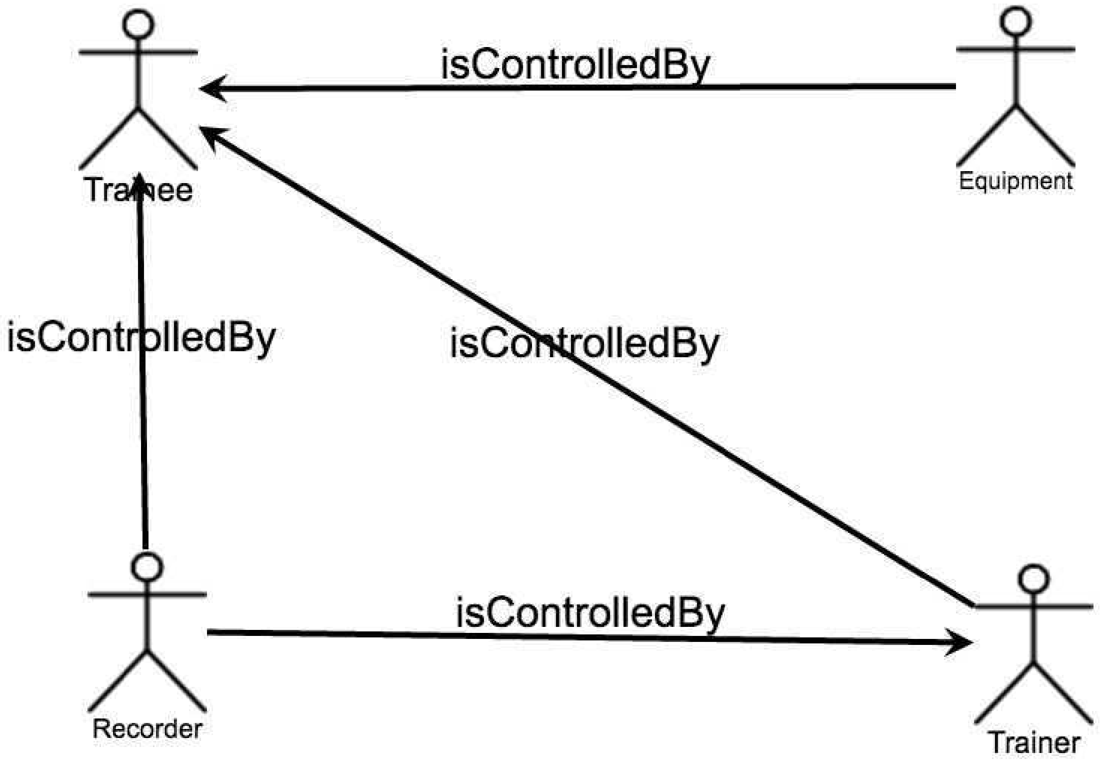
Role name	Trainer
Description	Helps Trainee to achieve his physical goal, provides training plan and exercise information
Responsibilities	• Provides Trainee a training plan • Provides information about exercise (how to do, etc.) • Provides Trainee an alternative exercise • Examine training results
Constraints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

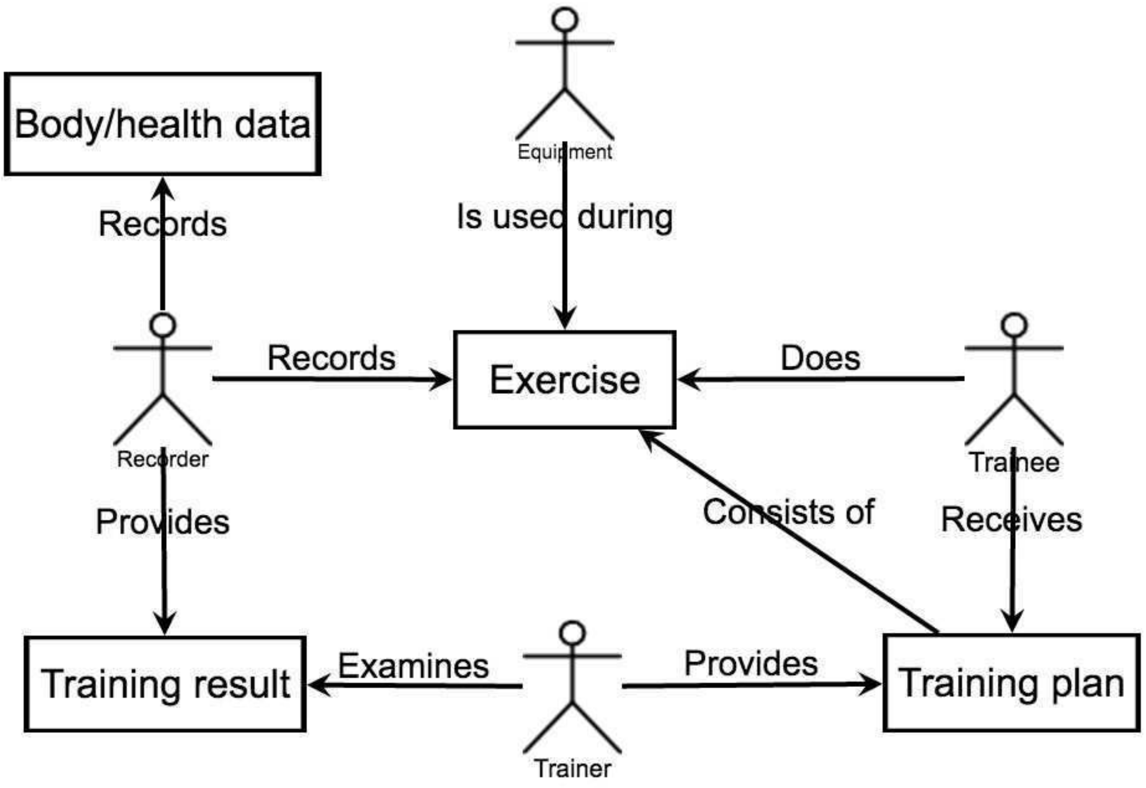
Organization model

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.


Equipment
Body/health data
Records
Is used during
Records
Exercise
Does
Recorder
Trainee
Consists of
Provides
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

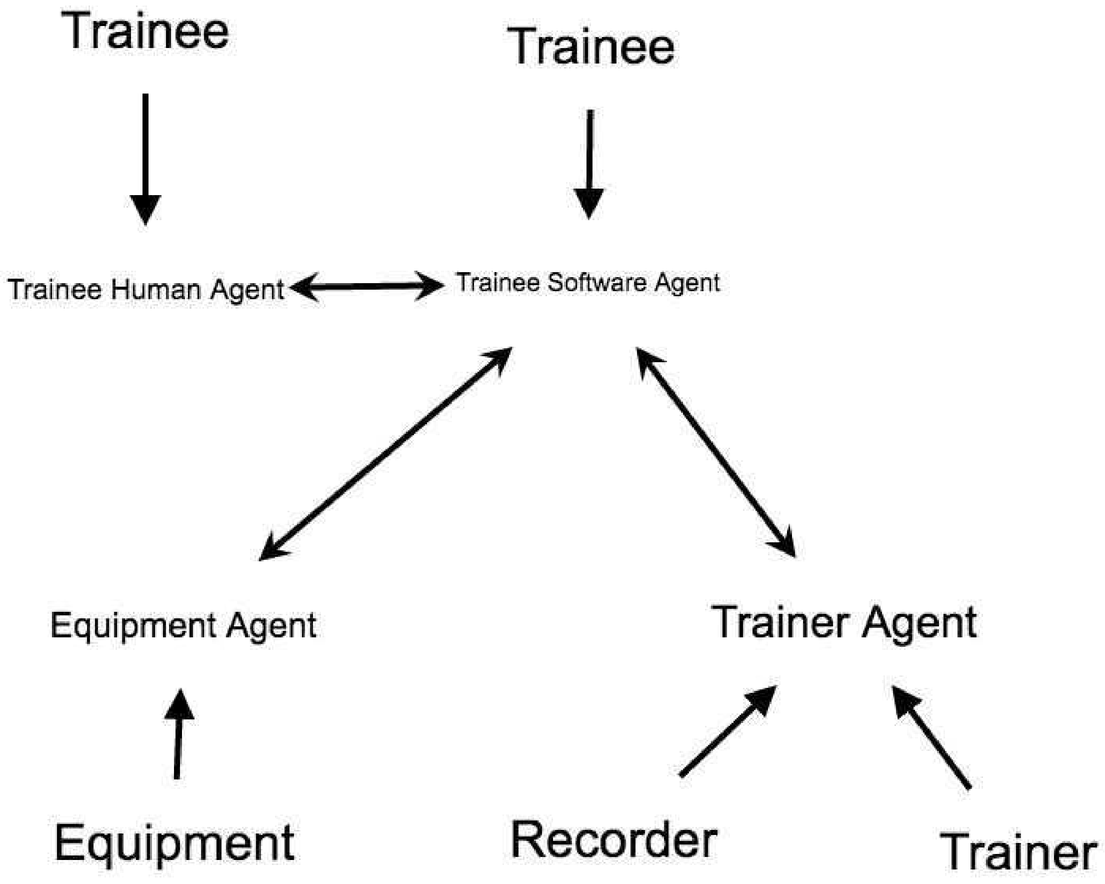
Agent name	Equipment agent
Description	Gym equipment agent
Roles	Equipment
Responsibilities	• Gives an opportunity to do exercise

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.

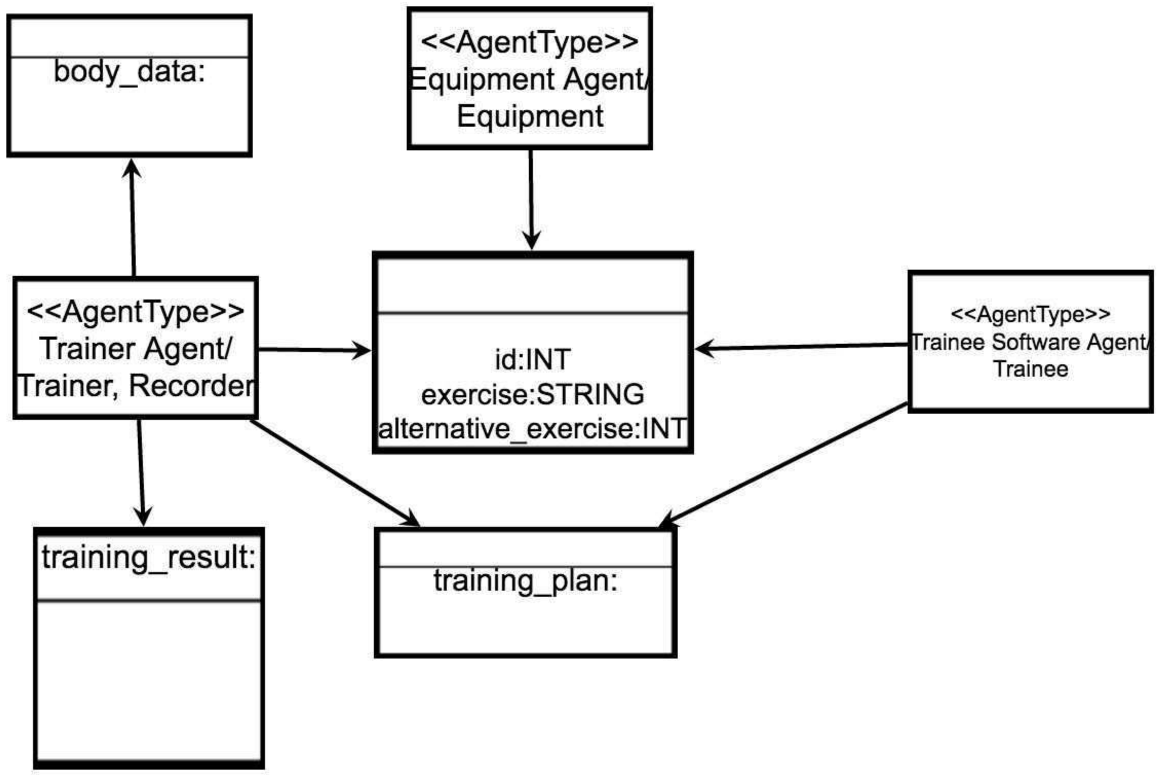
The Trainer agent has two roles - the Recorder and the Trainer and is typically comprised of a software application on a smartwatch or a smartphone.

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



Knowledge model

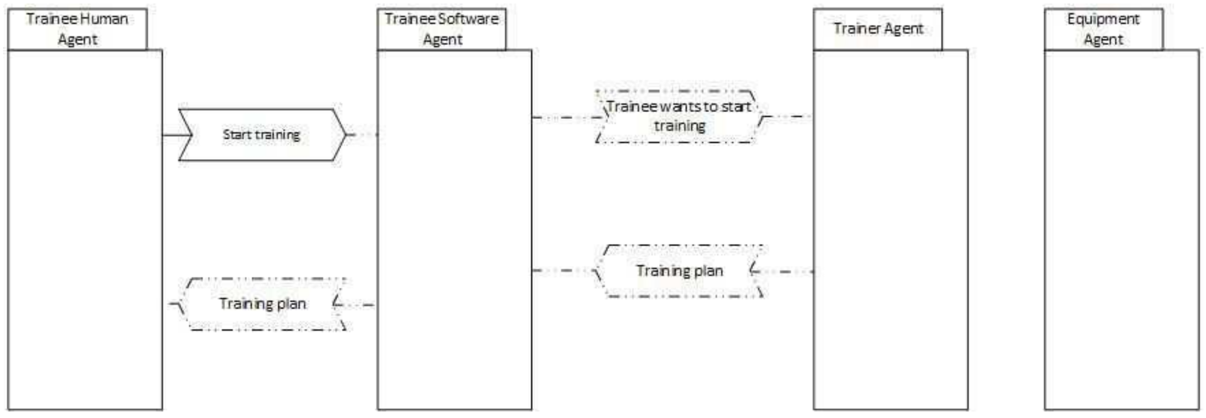
The following model contains the knowledge requirements of the agents. Only the basic attributes are shown.



Interaction models

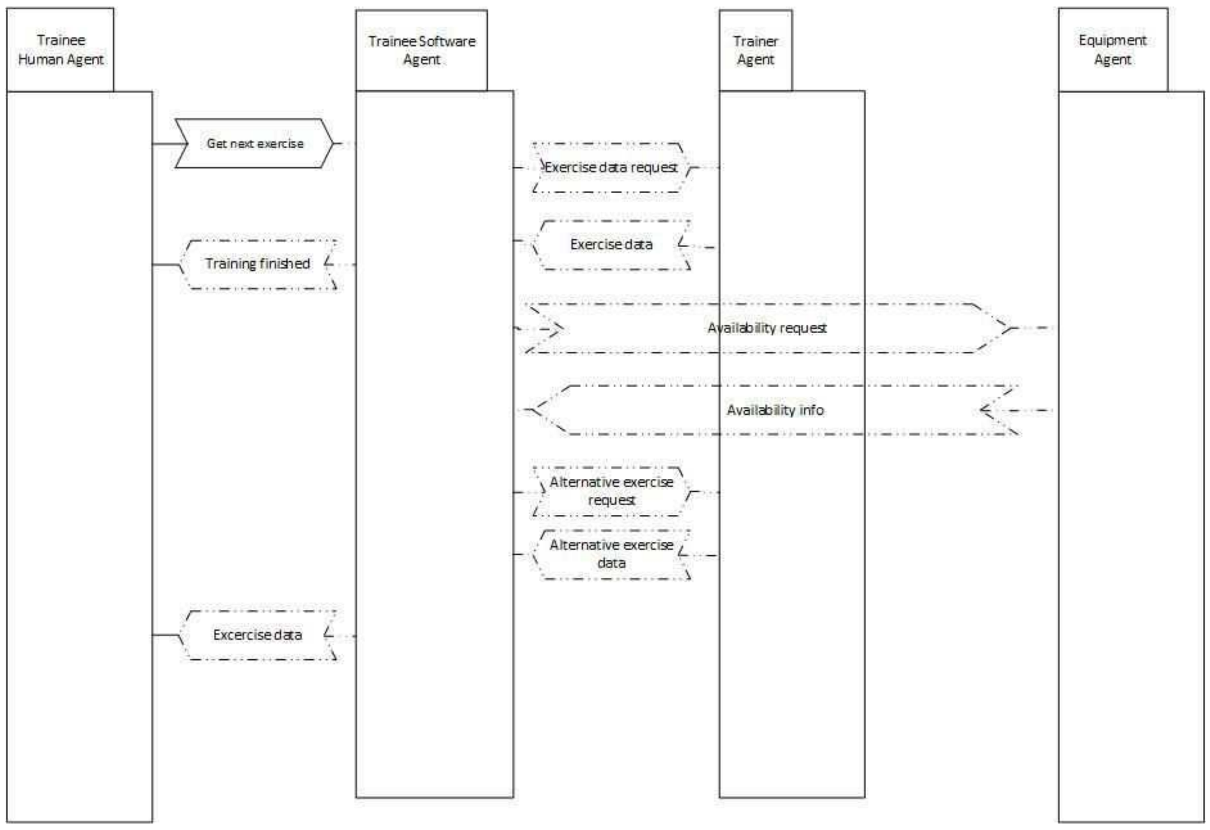
The basic interactions between human and software agents are shown below. Three interaction models were chosen, that capture the key functionality of the project - adaptive training.

Beginning of training



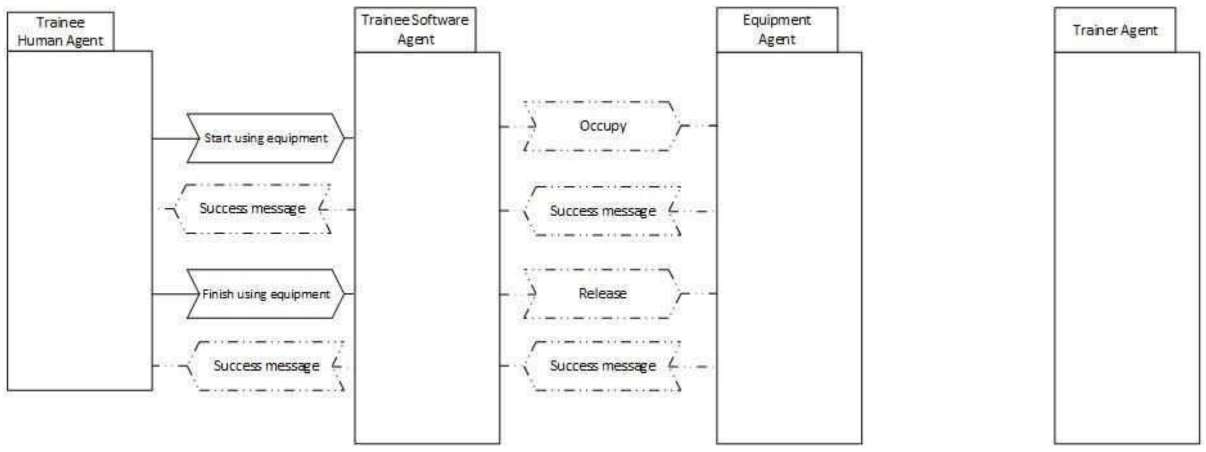
This interaction model shows beginning of the training. After Trainee reached the gym, he tells the Trainer (some wearable device or smartphone) that he wants to start training. All this is done through some UI (Trainee Software Agent). In response Trainer sends an overview of today's training to the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

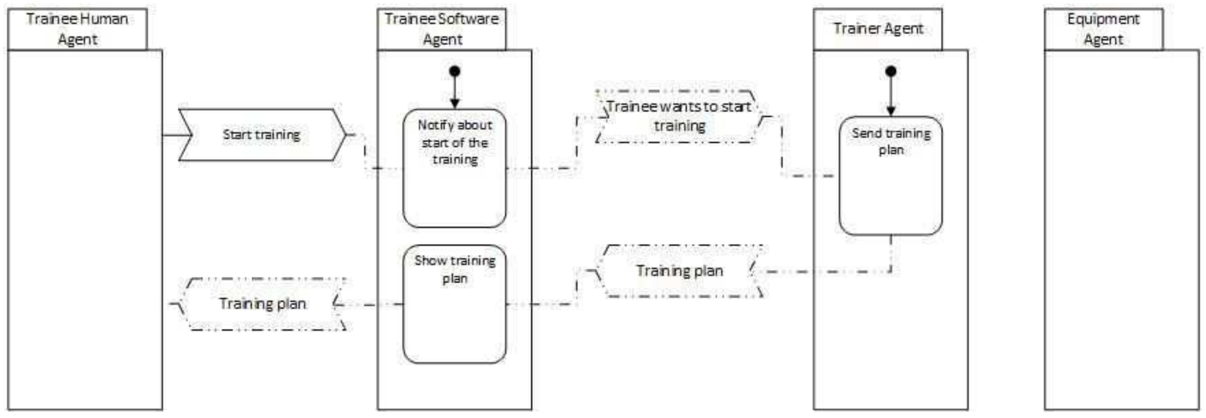


After Trainee receives all needed information, he can finally start exercising. Trainee occupies the gym equipment, does the exercise and, finally, releases the equipment, so others could also use it.

Behaviour models

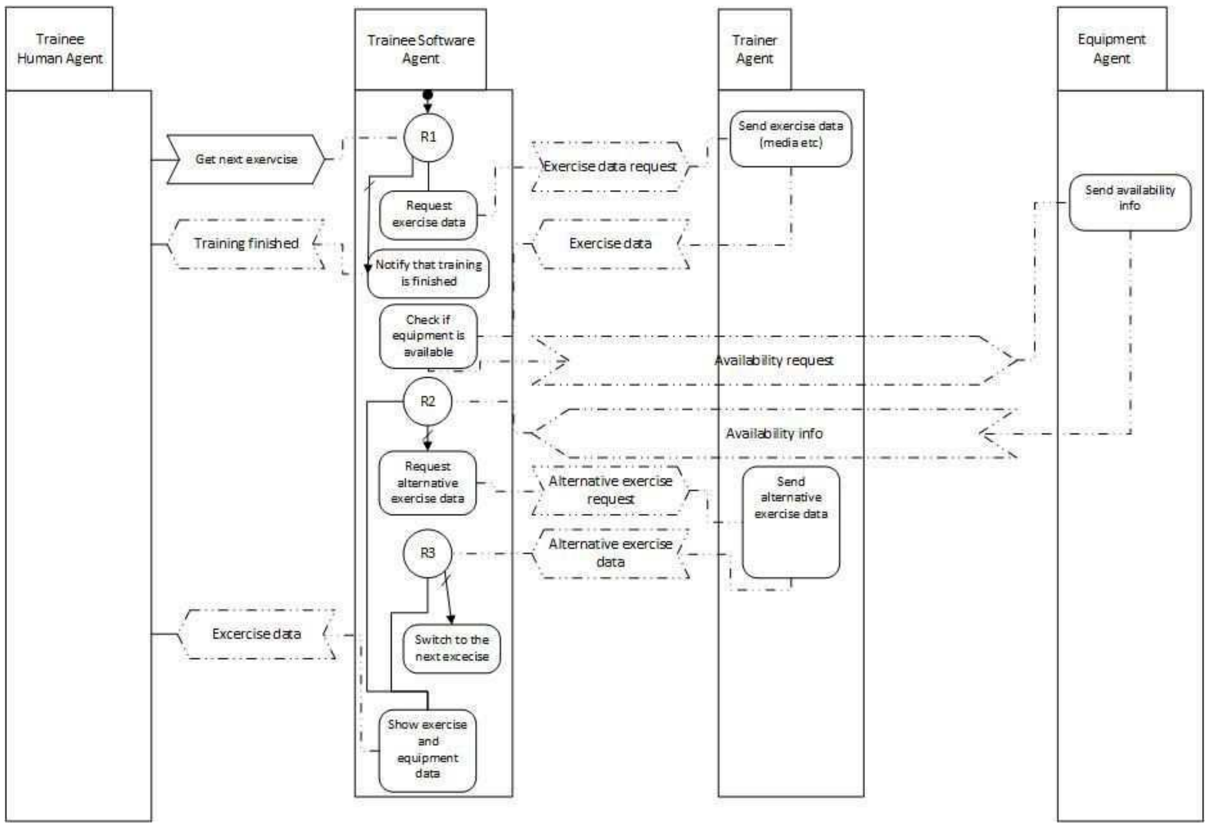
Following are the behavior models for the corresponding interaction models together with rule descriptions. Description of the models can be found in previous section (Interaction models).

Beginning of training



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

Exercise selection

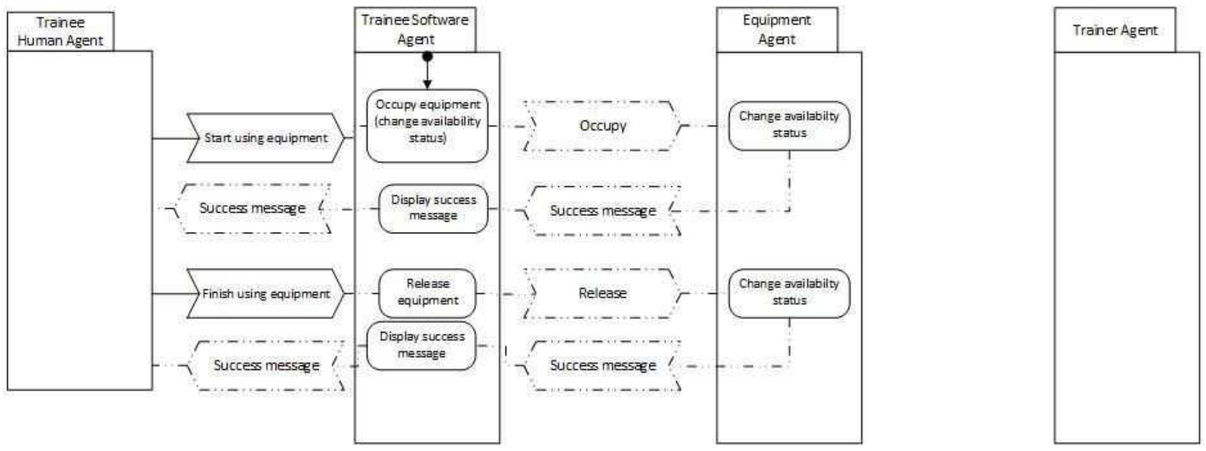


Trainee Human Agent
Trainee Software Agent
Trainer Agent
Equipment Agent
R1
Get next exervcise
Exercise data request
Send exercise data (media etc)
Request exercise data
Training finished
Exercise data
Notify that training is finished
Send availability info
Check if equipment is available
Availability 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.

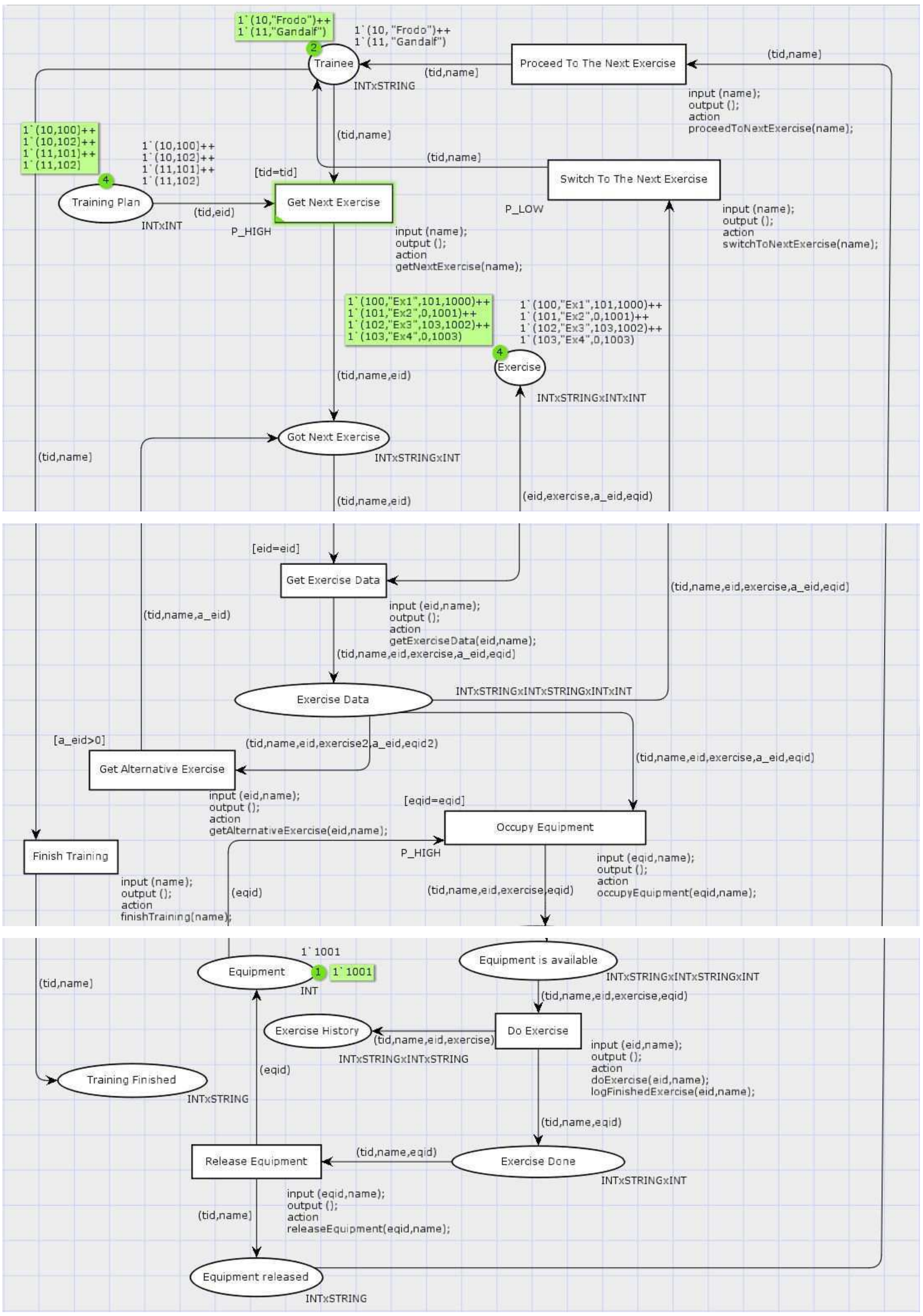
Rules:
R1 - Check if there are exercises left in today’s training plan.
R2 - If gym equipment is taken, ask for alternative exercise from Trainer.
R3 - If gym equipment for alternative exercise is also taken, switch to the next exercise.

Exercise



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

CPN simulation

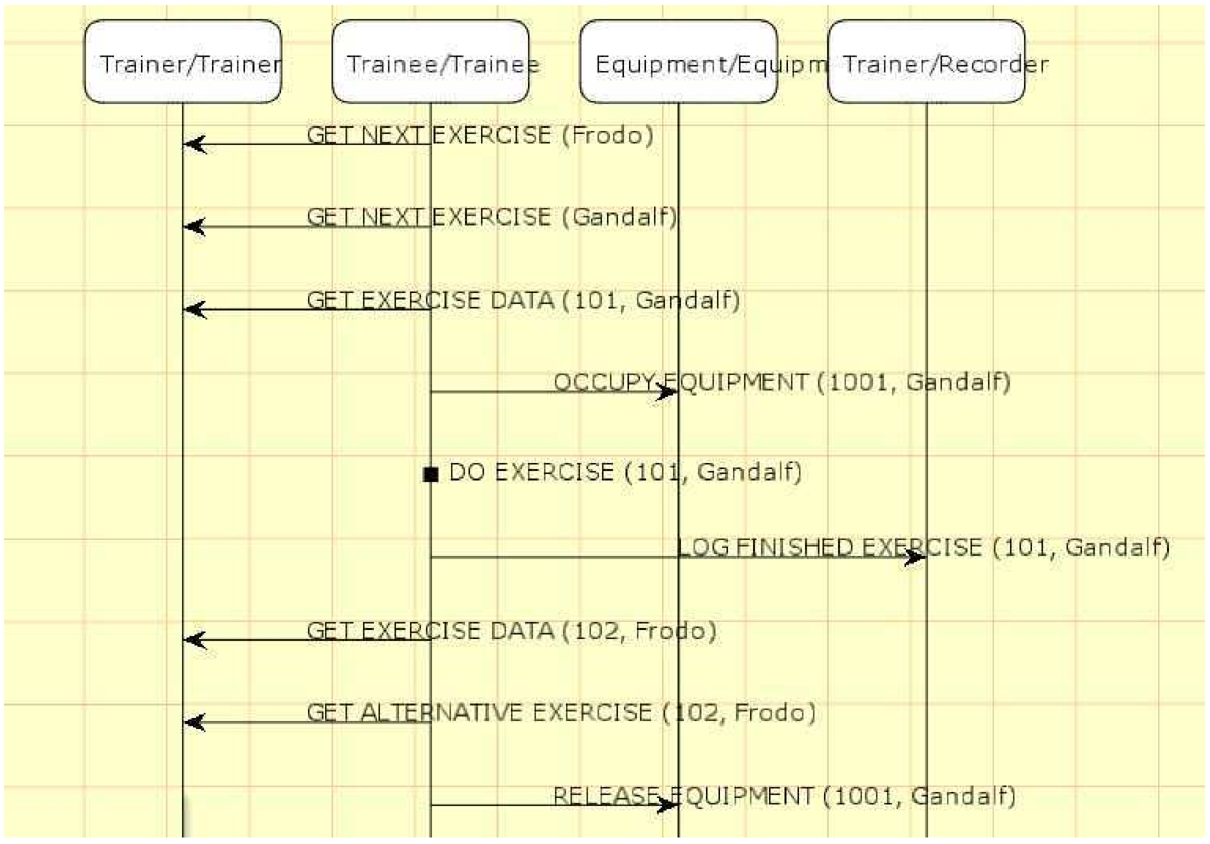


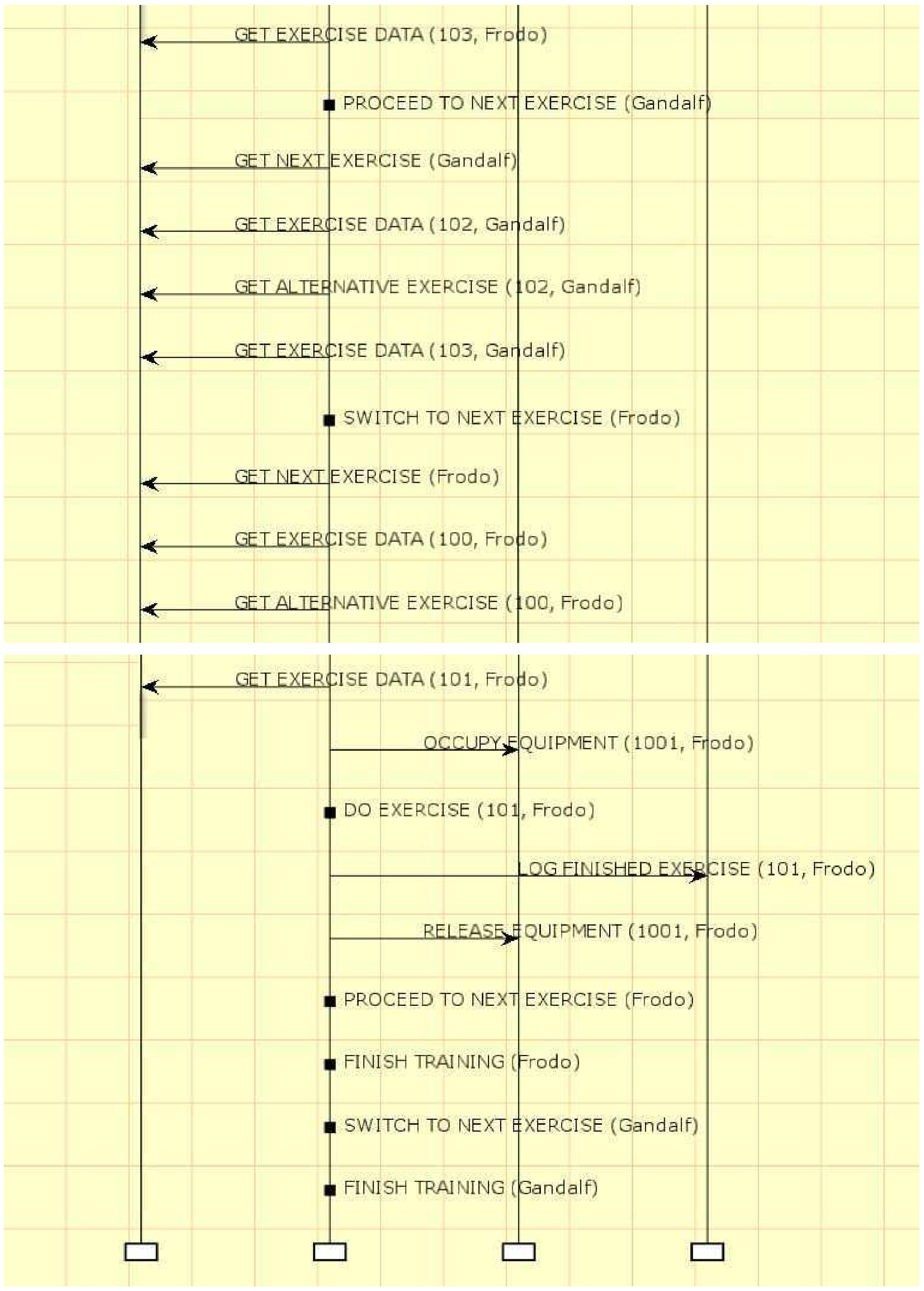
Validation

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.