

The seminar 08: Modelling architecture of the entity type(s)

Objectives of the seminar:

- to develop the perceptor-effector diagram
- to develop the service diagram

Update: 18. 9. 2018

Theoretical background

Perceptor-effector diagram

Behavioural entity types perceive their environment with the perceptrs (sensors). They react on the stimuli coming from the environment with the effectors (actuators). Perceptor-effector diagram is able to represent the components that the entity type (most often the agent type) can use for sensing the environment and reacting on stimuli. These elements are available in the (AML) Behavior toolbox. The main elements of this diagram are the following, see table 1:

- **Perceptor Type** is the UML class used to specify (by means of owned perceiving acts) the observations an owner of a perceptor of that type can make. Perceptor type has perceiving acts. Perceiving acts are specialized UML operations which can be owned by perceptor types and thus used to specify what perceptions their owners, or perceptrs of given type, can perform. The result is the evaluation of the receiving stimuli. This evaluation is also a method – a signal leading to the next behaviour of the entity type, e. g. turning on the left – 45 degrees, see figure 1.
- **Effector Type** is the UML class - a specialized BehavioredSemiEntityType used to model type of Effectors, in terms of owned EffectingActs. Effecting act is a specialised operation (from the UML) that can be used to specify what effecting acts the owning EffectorType, or an Effector of that EffectorType, can perform (e. g. sending warning message), see figure 1.

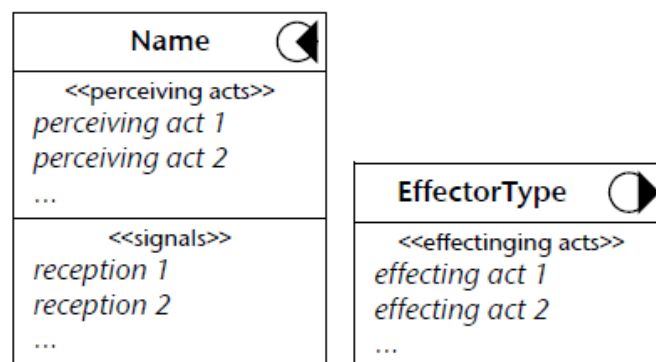





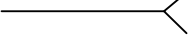


Figure 1: Perceptor type and Effector type

- **Perceptor** (UML port) is a mean to enable its owner, a behaved semi-entity, to observe, i.e. perceive a state of and/or to receive a signal from its environment (surrounding entities). It is no operations or attributes. These ones are located in its type.
- **Effector** (UML port) is a mean to enable its owner, a behaved semi-entity, to bring about an effect on others, i.e. to directly manipulate with (or modify a state of) some other entities. It is no operations or attributes. These ones are located in its type.
- **Perceives**: It is a specific type of the UML dependency for modelling the fact that the entity is able to sense (perceive) its environment with the preceptors.
- **Effects**: It is a specific type of the UML dependency for modelling the fact that the entity is able to influence the environment with the effectors.

Table 1: Main elements of the perceptor-effector diagram

Name	StarUML toolbox	UML element	Visual representation	Example
Perceptor type	(AML) Behavior	UML class		Sensorical scanner
Effector type				Movement actuator
Perceptor		UML port		Eye
Effektor				Leg
Perceives		UML dependency		-
Effects				-

Perceptor-effector diagram – practical tasks

1. Open your file from the previous seminar. If you did not create the own one, open the file from the last seminar Sem07-EntityDiagram-solution.uml.
2. Develop the new model with the name Perceptor-effector model in the Behavior package.
3. Create the new diagram in the Perceptor-effector model – Agent diagram.
4. Choose the (AML) Behavior toolbox from the palletes that are on the left side of the StarUML environment, see fig. 2.

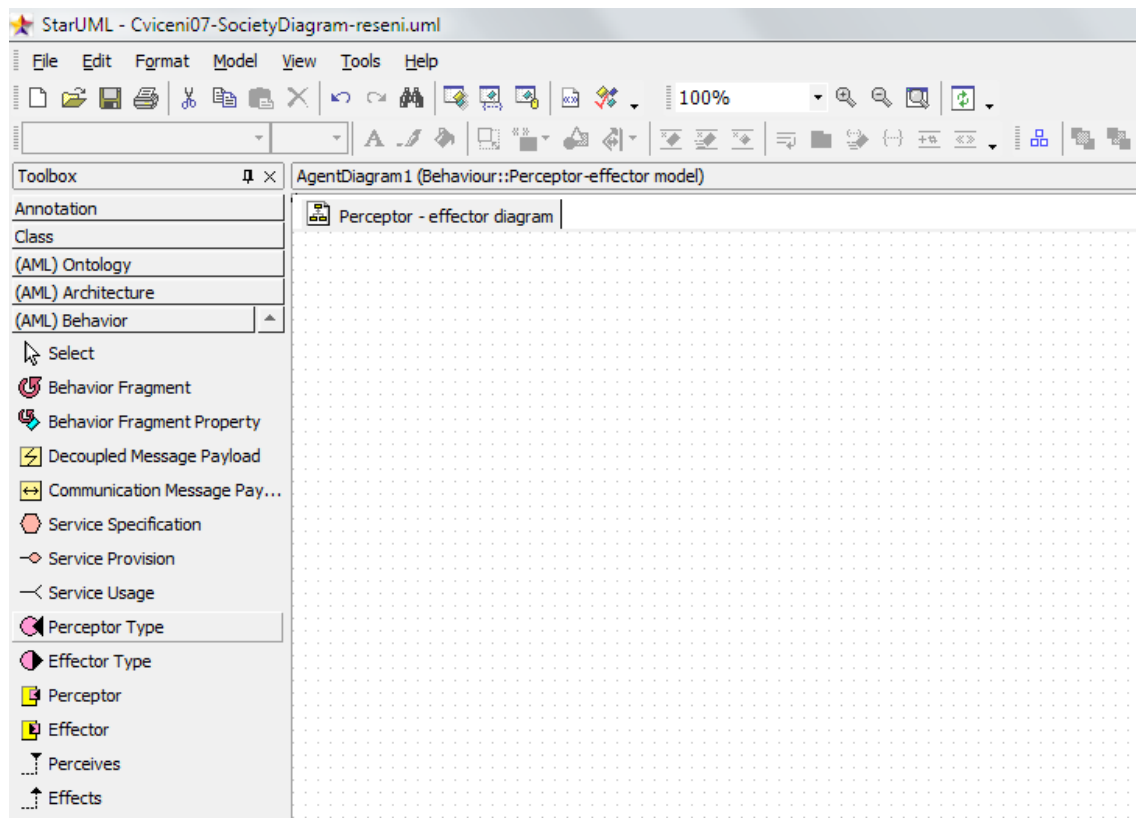


Figure 2: (AML)Behavior – elements for the Perceptor-effector diagram development

5. **Develop the perceptor type with the name Eye. This preceptor type will be used by the agent type Person that will play the role named Player. This perceptor type will be used for sensing the environment.**

Choose the Perceptor type element from the (AML)Behavior toolbox. Put this element into the work place and give it the name Eye. This perceptor type will offer the information about the state of the environment to agent type Person. Perceptor type contains two perceiving acts: look and localise. Perceptor type will contain the signal that will be activated when the object will be near the agent type.

6. **Add the perceiving acts to the agent type Person: look and localise.**

Perceiving act >look< has the argument direction with the return value Object. Perceiving act >localise< has the argument object with return value Position. Both acts have the private visibility type.

Specification or editing of acts for preceptor type is the same as the specification of properties or methods in classical UML (AML) class. You will set up properties/methods by doubleclicking on the UML(AML) class. Squared icon with red rectangle will appear on the right side of the element. This icon indicates the possibility to add method (or act). Click on this icon and write the name for operation localize(object):Position, see fig. 3. The icon for the editing of a visibility appears on the left side of the selected element (e. g. UML class). You can also use the Properties tab (on the right side of the StarUML) for editing of various parameters of selected elements – properties or methods, see fig. 4. It is necessary to specify corresponding stereotypes for corresponding <<stereotypes>> for distinguishing what is the perceiving act and what is the signal (as the reaction on the perceiving

input). Perceiving acts have the <<pa>> stereotype (perceiving act) and signals have the <<signal>> stereotype. Stereotypes are placed in front of the operations names, see fig. 4.

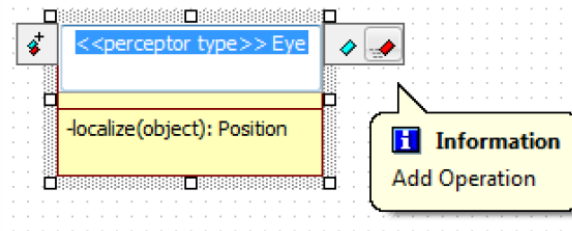


Figure 3: New acts specification – double-click on the AML element

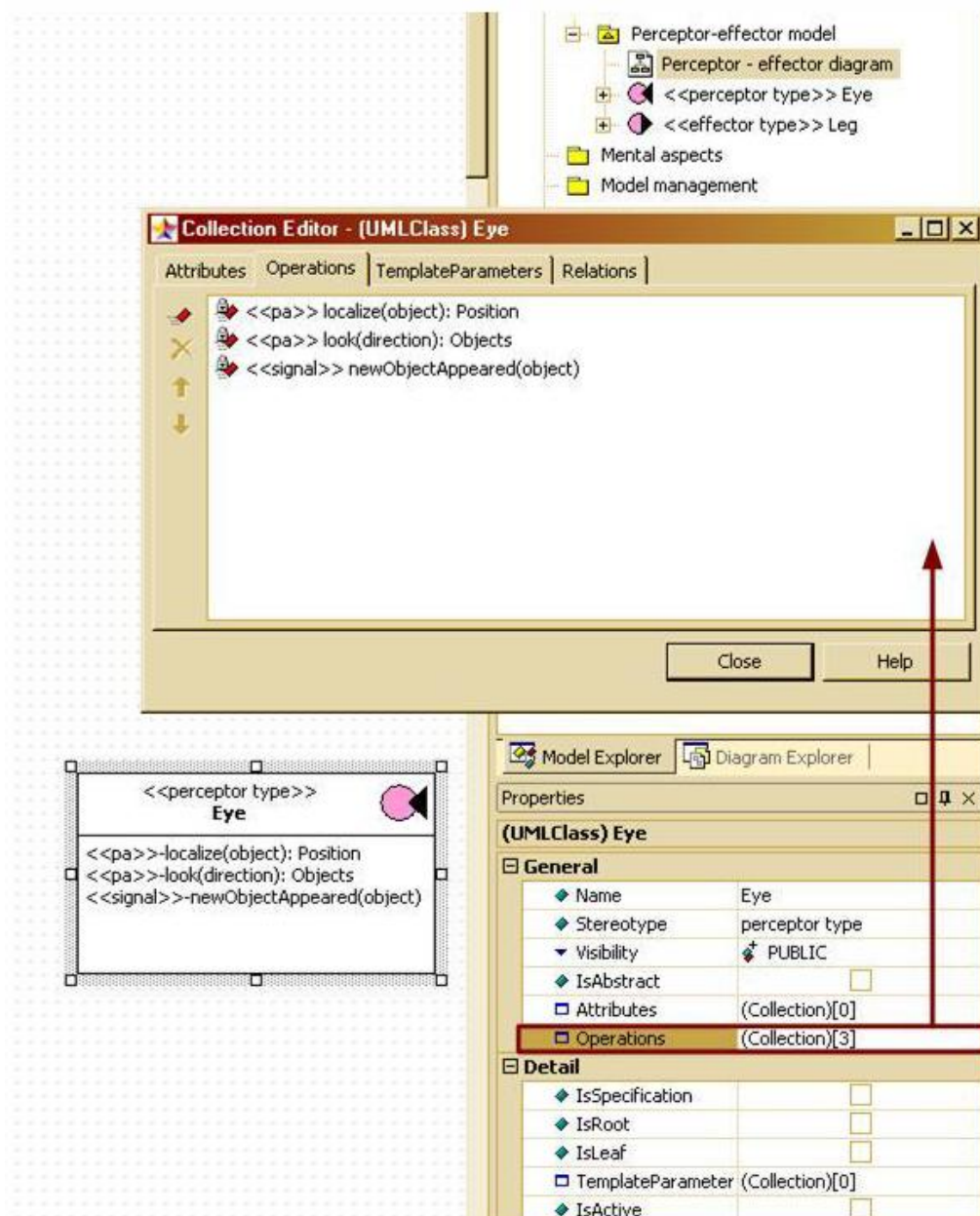


Figure 4: New acts specification – Properties tab

7. Specify effector type with the name Leg that will be used for movement of agent type Person playing the role Player.

Specification of the effector type is very similar to the perceptor type. We only use different element type – Effector type from the (AML) Behaviour toolbox. This effector type has the <<ea>> stereotype.

8. Add the five effecting acts for the effector type Leg with the following arguments (methods will be without return values).

- step(direction): krok vpřed v požadovaném směru
- walk(to): chůze určitým směrem
- run(to): běh určitým směrem
- kick(what, direction): kop do předmětu a do určitého směru
- leadBall(ball, direction, distance): vedení míče do určité vzdálenosti a v určitém směru

9. Model the fact that the agent type Player use two perceptor types Eye and two effector types Leg.

Add the entity role type named Player on the working space from the Model Explorer. We use the UML ports for modelling particular perceptrs and effectors from the (AML)Behavior toolbox. We add these ports on the edges of the UML (AML) classes, in our case Player, see fig. 5. We add the name eye for the perceptor. We choose the corresponding perceptor type Eye in the Properties/Type panel. We add the multiplicity 2 representing two perceptor types Perceptor in the same panel (Multiplicity), see fig. 5. We apply the same steps for modelling effector type Leg.

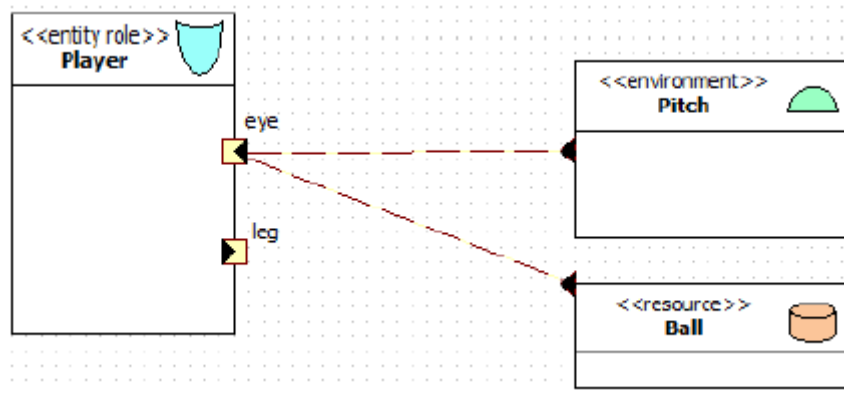


Figure 5: Perceptrs and effectors, Perceives dependency

10. Represent that the entity role player Player perceives Pitch and Ball.

Add the environment type Pitch and resource type Ball into the working space. We use the Perceives dependency from the toolbox (AML) Behavior. This dependency is used for modelling relations between sensors and entity types, see fig. 5.

11. Represent that the entity role player Player interact with the Ball with the effector type Leg and this leg influences the position of the Player.

Use the different dependency called Effects from the (AML) Behavior toolbox. Effector can influence the entity type itself, see fig. 6.

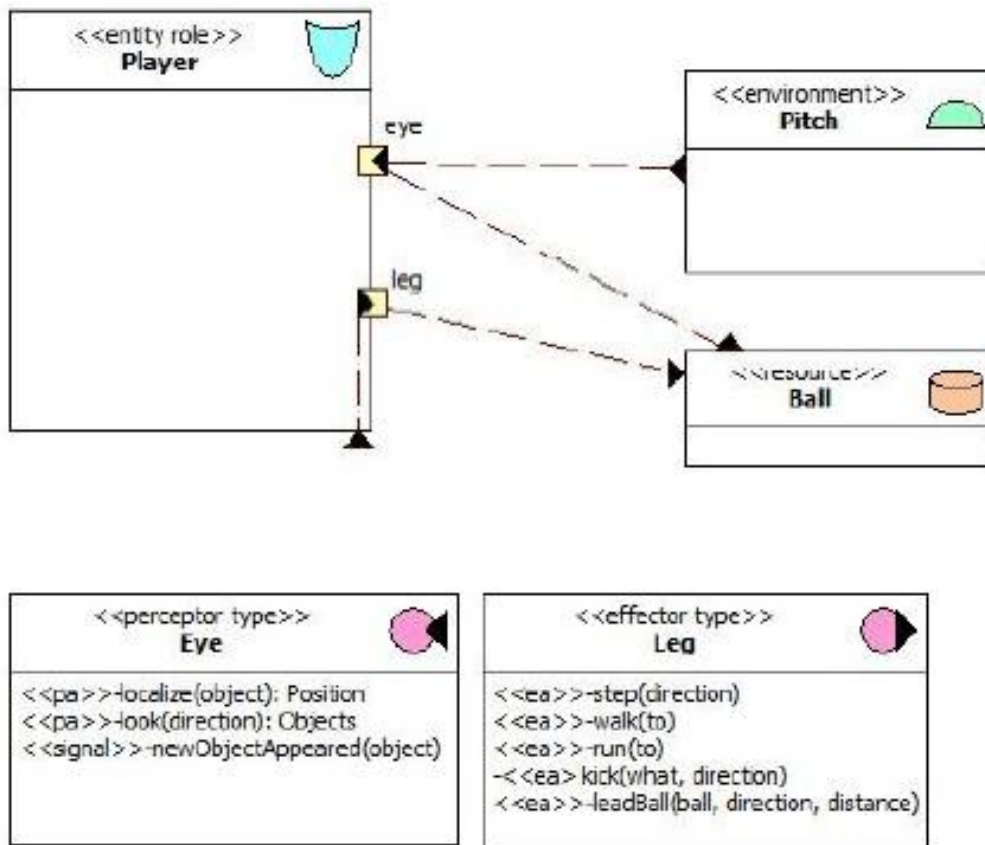


Figure 6: Perceptor-effector diagram

12. Check the own perceptor-effector diagram with the solution, see fig. 6.

Theoretical background

Service diagram

Service diagram is a special case of the UML composite structure diagram that is used for representation services, their users (clients) and providers of services. Service is a coherent block of functionality. We can model the situations where the client uses the service and provider provides the service. Services can be perceived in the external and internal point of view. External services are located in the outer environment of the entity type, e. g. the agent type can use the service that is offered by the particular environment type. Internal services are related to the internal structure of the entity type. Entity type has more or less general structure consisting of various components, e. g. cpu, knowledge base, memory system, etc. These components can use or provide services to other components. This is the internal view on the services. Figure 7 depicts the difference between internal and external services. Table 2 depicts the most substantial elements that can be used for development of service diagram.

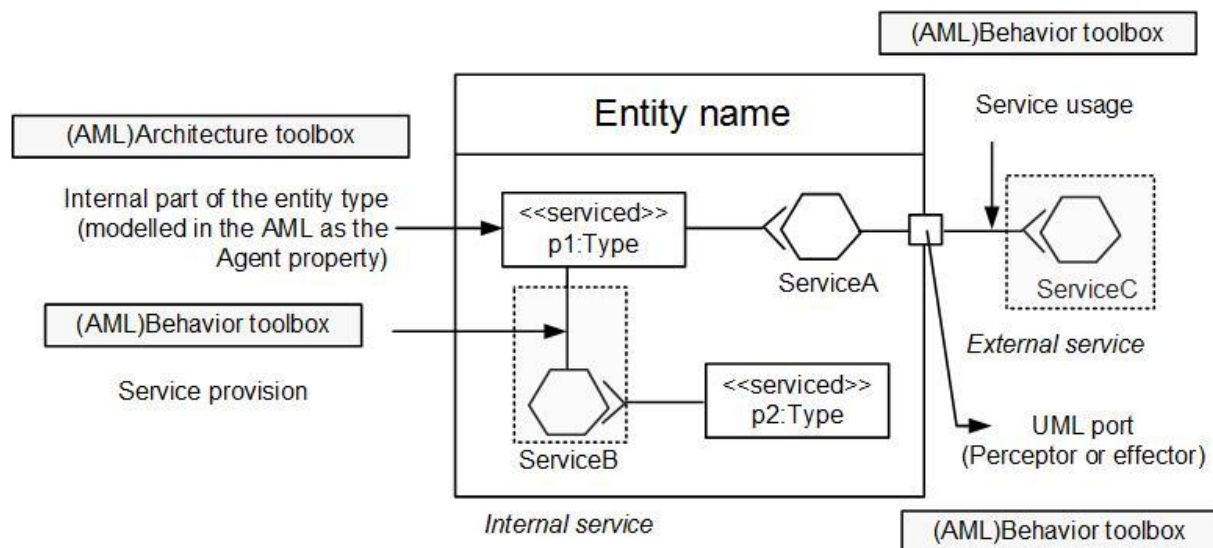

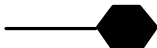
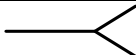


Figure 7: Internal and external services

Table 2: Main elements sof the service diagram

Name	StarUML toolbox	UML element	Visual representation	Example
Service specification	(AML) Behavior	UML collaboration		Collision detection Movement checking
Provide a service		UML dependency		-
Use a service		UML dependency		-

Service diagram - practical tasks

We will develop the model of the agent type – mobile robot that will move in the room with the usage of two effectors – one front wheel and two rear wheels. This mobile robot can be simplified representation of football player. Both effector types will offer the Movement service that be used for changing the position of the robot in the room. Front wheel will offer the Wheel Control service to the CPU that is the integral part of the agent type mobile robot. The CPU will be a client of this service. It is necessary to be in touch with the environment, e. g. in case of collisions detection. Front wheel will use the service Collision Detection that will be offered by the Room environment. Room will be also in touch with the environment, e. g. it will perceive the agent type(s) on the basis of its movement. The environment type will use the Movement service from the effectors of the mobile

robot. Knowledge base is the secondary integral part of the mobile robot. This component will be used by the mobile robot for manipulation with information and knowledge. Pieces of information and knowledge will be used for communication with effectors – wheels of the mobile robot.

13. Create the new directory in the Model Explorer (package Behavior), i. e. the Services model.
14. Add the new Agent diagram into this Service model, i. e. Service diagram.
15. Choose the (AML) Architecture toolbox. Choose the agent type and put it into the work space. Its name will be – Robot.
16. Specify the two integral parts of this Robot, i. e. one cpu of the CPU type (resource type) and one knowledge base (knowledgeBase) of the Knowledge base type (resource type). Represent these integral parts as the properties of the agent type (use the Agent property from the (AML) Architecture toolbox). Do not forget on the multiplicity and corresponding stereotypes for new resource types. Do not forget to add the types for new sources).
17. Add the frontWheel effector of the type Wheel. Add the two rearWheel effectors of the type Wheel. Do not forget to implement the new effector type Wheel.
18. Specify the new environment type Room where the mobile robot will move, see fig. 8.

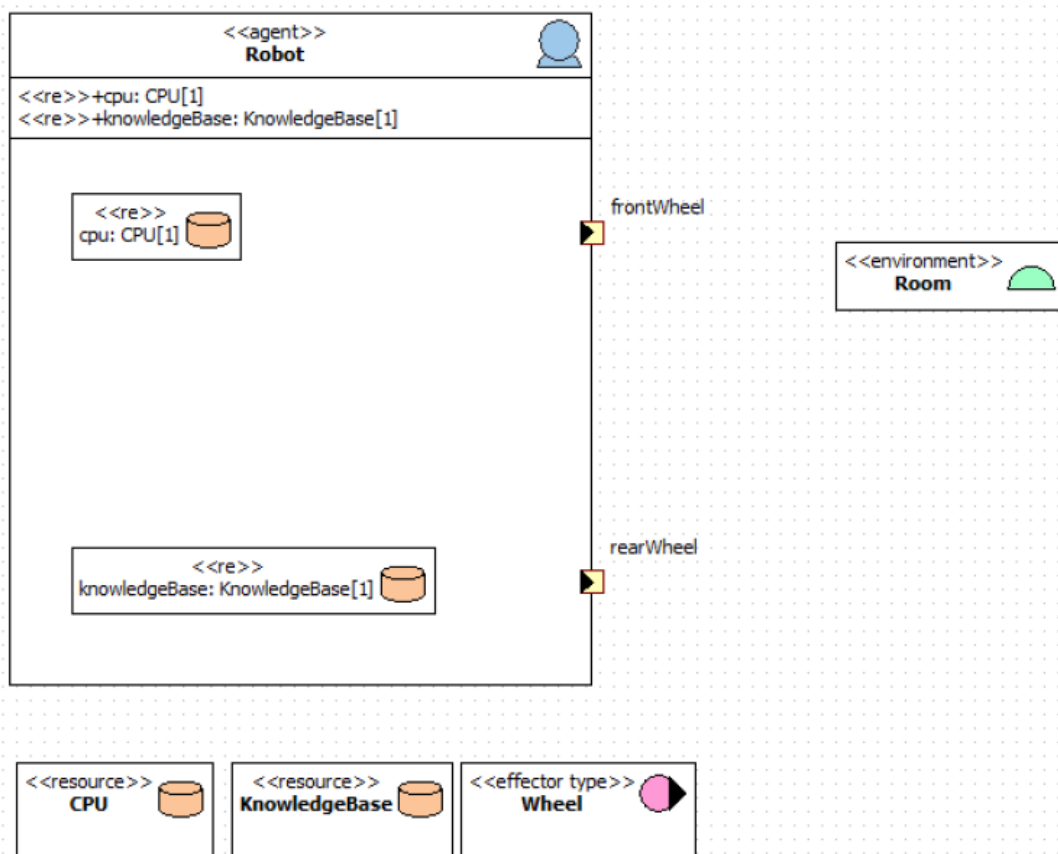


Figure 8: Entity types for service diagram

19. Specify the Effects dependency (relation) between the effectors and the environment type Room.
20. Choose the (AML)Behavior toolbox and right click on the Service specification element. Create the service named WheelControl. The visual style will be Iconic.
21. Create the service named KnowledgeManipulation in the same way as the WheelControl.
22. Represent the situation that the cpu uses the service KnowledgeManipulation and WheelControl. Use the Service Usage from the (AML) Behavior toolbox for specification the fact that the CPU uses the service WheelControl and KnowledgeManipulation.

23. Save the AML project.

24. Check your results with the result that is on the fig. 9 below and with the files with both diagrams (Perceptor-Effector and Service), see the file Sem08-Perceptor-Effector-solution.uml and Sem08-Services-solution.uml.

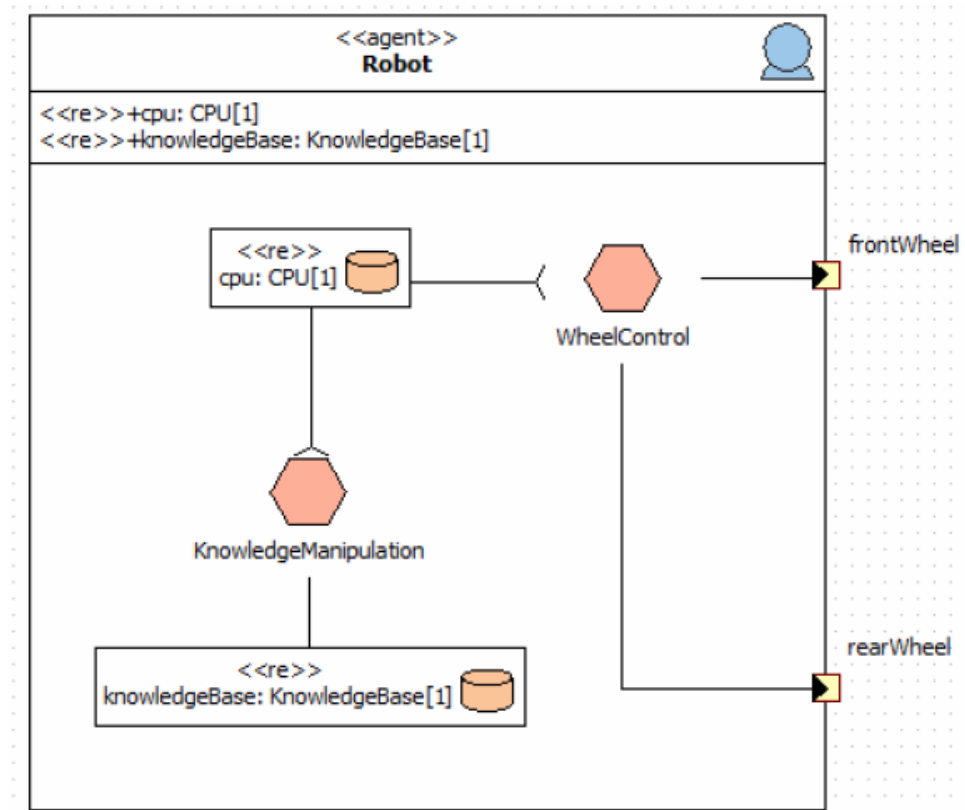


Figure 9: Representation of the internal services for mobile Robot

End of the seminar

Exercises

- Homework: prepare the above mentioned diagrams for your project.

The most important keywords

- Preceptor type
- Perceptor
- Effector type
- Effector
- UML dependency Perceives
- UML dependency Effects
- Service
- Use the service
- Provide the service

Resources:

[1] Červenka, R. a Trenčanský, I., 2007. The Agent Modeling Language – AML: A Comprehensive Approach to Modeling Multi-Agent Systems Birkhäuser Verlag AG, Basel - Boston - Berlin, 355 p., ISBN 978-3-7643-8395-4.