Reconfigurable Intelligent Agents

Michal Radecký
Petr Gajdoš
Faculty of Electrical Engineering and Computer Science
VŠB-TU Ostrava
17.listopadu 15, Ostrava, Czech Republic
michal.radecky@vsb.cz, petr.gajdos@vsb.cz

Abstract

*The development process of Multi-Agent Systems has similar features to standard information systems. However, there are special requests that have to be taken into account, e.g. general MAS architecture, internal architecture of particular agents, their autonomy and communication. This paper describes a new development tool (**Agent**Studio) that covers main phases of agent development and simulation. Our approaches support the creation of intelligent agents and reconfiguration of their behaviours.*

1 Motivation

A Multi-Agent System (MAS) attracts attention as an approach to complexity systems in recent years. Fundamental elements of MAS are represented by agents, individual software units in distributed systems. Efficiency of such systems depends on the quality of agents' internal architecture, features and their cooperation ability with other agents. Recent research deals with so called *intelligent agents* which dispose of additional skills that appear from logic, processes, information retrieval, etc. This paper describes a specification of agent behaviour based on extended UML technique. Agents, which can modify their behaviours upon defined processes are main goal of our research.

Nowadays, there are standard approaches to develop whole information systems from the requirement specification to the deployment (UML, RUP, etc.). They can be used also for MAS, however, this is not proper way to build complex multi agent systems. Specific features of these systems (e.g. autonomy of element, communication among elements, logic) require some specific tools and/or approaches. The Agent UML (AUML) [12] and new features of UML 2.1 and SysML [10, 9] can cover some phases of software process with respect to MAS. Nevertheless, the AUML is a quite old approach, that is no longer supported. On the other hand, new UML standards are too complicated for common users and they bring a lot of extensions which are not necessary for MAS development. Selection of suitable implementation framework plays also an important role. A Java Agent Development framework (JADE) has been chosen for our purpose [1].

2 Agent behaviour modelling

Each agent is determined by its own objectives and the way to meet these objectives is founded on the internal behaviour of a given agent. In the case of processional intelligent agents [7, 5], internal behaviour is specified by an algorithm. Agent lives, behaves and reacts with respect to environment stimuli and given algorithm. Outer MAS behaviour results from communication of particular agents and from interconnection of several internal agent behaviours. The behaviours can be changed dynamically with the aid of the *Reconfiguration approach* and communication is supported by *message passing*. These points are included in activity diagrams extension.

2.1 UML Activity Diagrams Extension

The UML Activity Diagrams represent a standard diagrammatic technique which describes the series of activities, processes and other control elements that together express an algorithm from the dynamical point of view [14, 4]. Diagrams capture internal behaviour of each agent, but they do not describe the interactions and communication between agents. Therefore, additional modifications and extensions are required to provide reconfiguration of behaviours and interaction among processes.

From now on, a *process* is a flow of atomic activities and sub-processes, that finally represents particular non-trivial action of the agents. An agent behaviour consists of a set of such processes.

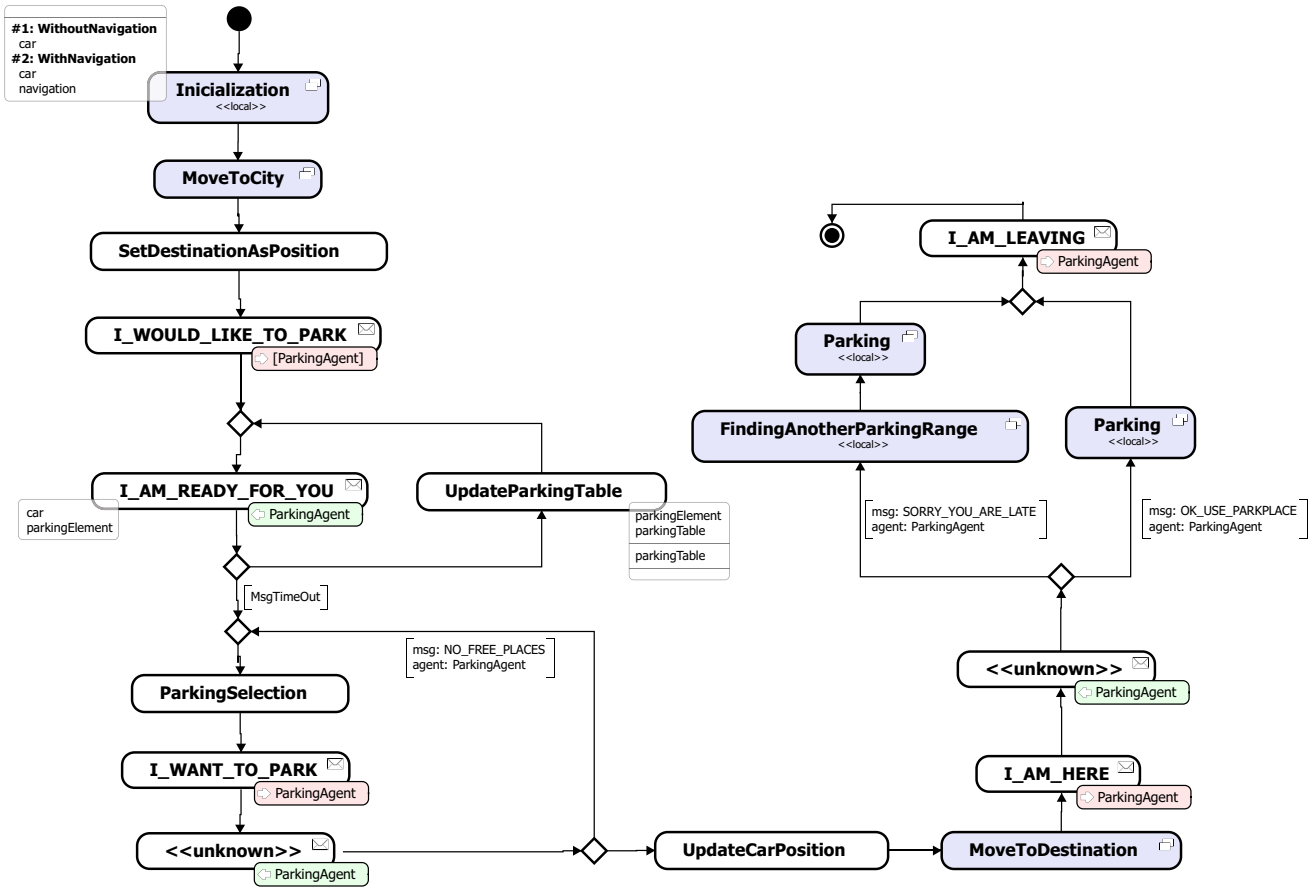

Figure 1. Example of ABD (Agent Behaviour Diagrams) with additional elements. It illustrates Car agent behaviour during parking process.

Our *Agent Behaviour Diagrams* contain all the elements of the standard UML Activity Diagrams and new elements concerned with message passing, process hierarchy, input/output objects and resulting process scenarios (see the figure 1).

The simple, clear and formal specification of internal agent behaviours is a precondition for the following phases of multi-agent software process. Also other types of diagrams (e.g. sequential diagrams, maps of the agents communication) can be generated thanks to the new stereotypes and information (on objects, scenarios, activity scores, etc.), which were established in the *Agent Behaviour Diagram*. The same holds for semi-automatic creation of source code templates of agents (e.g. agent interfaces, classes and methods).

2.2 ABD rules

Several rules must be kept to build complex MAS model.

- Each process, as well as diagram, has just one *initial node* and just one *final node*. It is necessary for further joining of processes to obtain overall agent behaviour.

- In the case of more final nodes within a process, the nodes are merged with appropriate process modification. Then, the previous final nodes are represented by *scenarios*, which are stored as an extra information in the merged node.

- ABD are *well-formed* [13]. Van der Aalst defined a set of general structural rules for workflow nets (e.g. the level of split/join nodes preservation, no crossing of the levels of control flow). These rules were acquired and due this fact, the diagrams can be verified and formalized.

3 Reconfiguration of Behaviours

Every process, expect the *primary process*, can be specified by more than one diagram (see the figure 2). Then, each diagram describes one *realization* of a given process. Realizations depend upon knowledge, experiences, environment and states of agents. The realizations can be stored within agent internal knowledge base or global MAS repository. Moreover, the agents can extend own sets of realizations thanks to communication and cooperation with other agents and/or platform facilities. This feature enables agents to learn.

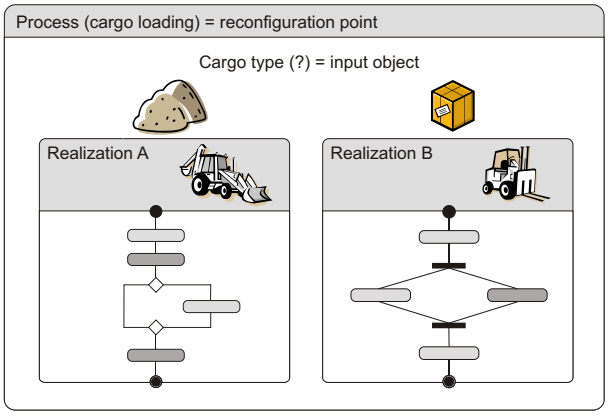

Figure 2. Example of process composition.

The *behaviour reconfiguration approach* represents the way how to implement intelligent agents with respect to processes [6]. The idea behind the reconfiguration comes from the hypothesis, that each process (*reconfiguration point*) can be realized by different ways - realizations. Reconfiguration algorithm is applied in time of process firing. Each process requires a set of input objects and can produce the outputs. The same holds for the realizations.

The figure 3 depicts the basic scheme of mentioned reconfiguration method. At the beginning, the set of all processes and their realizations is defined. Next, the selection phase is initiated. Depicted selection consists of two steps. The first one represents a simple selection of applicable realizations, based on input objects occurrences. The second one chooses the most suitable realization according to input objects properties, scores, etc. Methods of multicriterial analysis or logical tools can be used during the selection phase.

4 Traffic Simulations

The relationship between our MAS model and subsequent simulation is described in this part (see fig. 4). Mentioned ideas and approaches are projected in an implementation of MAS. Developed **Agent**Studio applications are suitable for this purpose.

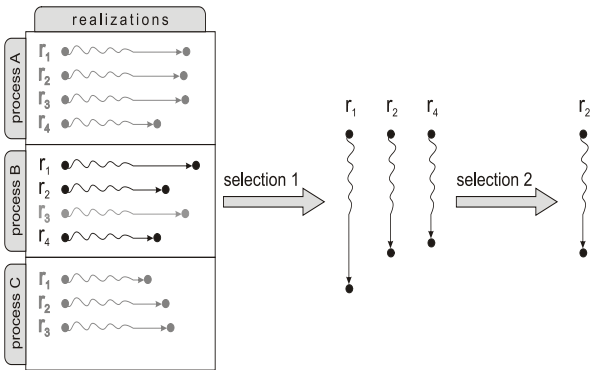

Figure 3. The reconfiguration algorithm scheme

Proposed approaches and methods have been proven in the area of traffic management. These are the reasons, that led us to choose this application area. First, the computational simulations are becoming increasingly important because, in some cases, it is the only way to study and interpret designed processes. These simulations may require very large computational power and the calculations have to be distributed on several computers. The MAS technology supports such kind of computation because of its independence from platforms, operation systems, etc. However, just selected traffic situations were taken into account to demonstrate the power of MAS technology, logic and **Agent**Studio. Next, several commercial systems pick up actual traffic data, create digital models of traffic infrastructure (roads, crossroads, traffic signs, etc.) and provide such data sets to use them within other projects. Then it is quite easy to use provided data in simplified form for **Agent**Studio Simulator and test agents' behaviours on real data. Logic and intelligent decision-making process play an important part in our project. Also this point is inherent with traffic simulations, e.g. if a traffic light is red, cars should stop before a crossroad. In other words, the car has to change its behaviour. Most of such rules are well described in Highway Code and can be rewritten into Prolog and/or TIL (Transparent Intensional Logic) formulas. The fourth reason consists in agent behaviour description based on UML modelling which is covered by ABD. Last but not least, an eye appealing way of presentation of simulation results is important. Visualization tool makes one part of the **Agent**Studio Simulator and helps us to see what the agents really do and how they behave depending on the environment.

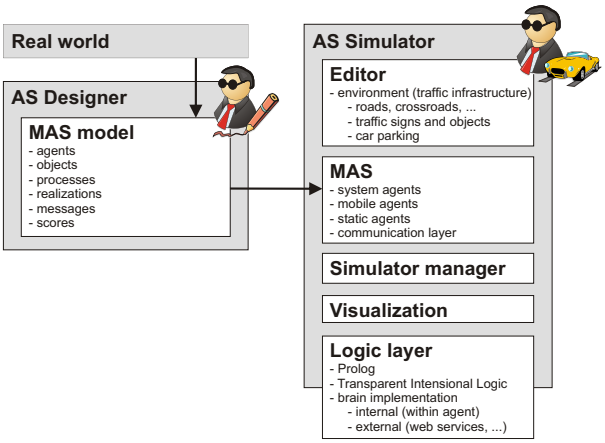

Figure 4. The scheme of the AgentStudio applications

4.1 Target area description

Traffic simulations try to reflect real situations taking place on roads. Nowadays, **Agent**Studio Simulator allows us to design and edit simplified infrastructure to test agents' behaviours. In the future, it will also enable to import real GIS data. These are important situations which we focused on:

- Cars overtake each other and they will recognize traffic obstacles.
- They safety pass through crossroads.
- They keep safety distance from other agents (cars).
- They keep basic rules defined in Highway Code.

Particular situation is solved during the agent's life with respect to agent's ability to make decisions. Finally, MAS development and simulation can be divided into steps mentioned below.

4.2 Process of Modelling and Simulating

A complete scenario of **Agent**Studio utilization can be divided into several parts:

1. Real world requirement analysis → identification of the agents and their goals, objects, processes, etc.
2. Agents' behaviours modelling in **Agent**Studio Designer → MAS model based on ABDs.
3. Source code generation and its completion (Complex behaviours have to be modified by programmers.) → source code in specific programming language and selected MAS framework.
4. MAS environment specification → traffic infrastructure description based on retrieved real data.
5. Initial phase of simulation process → starting MAS platforms, encapsulation of traffic information into platform data structures.
6. Simulation, visualization and management.

4.3 Traffic MAS architecture

The figure 5 illustrates mentioned MAS architecture. Platforms **1** and **3** represent two parts of the real world, e.g. a town district with a separated parking lot. The data of such platform consists of a traffic infrastructure map. Next, each platform has a description of traffic elements located on the traffic infrastructure. Finally, the data holds the information on mobile agents obtained from *proxy agents*. Generally, particular platform data reflects the state of the real world. Second parts of these platforms make environments for system agents which are responsible for communication with other agents (Proxy agents), map services (MapDispatcher agent) and for agent registration (WorldRegister agent). The platform **2** consists of mobile agents that represent cars moving in the real world. The platform **2** can be distributed on many hardware nodes according to FIPA standard.

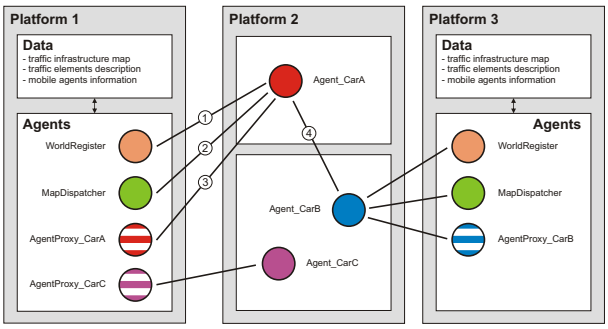

Figure 5. MAS architecture for traffic simulation

The main relationships between platforms/agents are described in the following points (see the figure 5).

- A single car (Agent_CarA) registers itself into a given part of the world (Platform 1). It is done through the communication with WorldRegister agent that also creates a proxy agent for Agent_CarA (st. like proxy in the Object Oriented Programming).
- This connection provides an access to map services (road finding, infrastructure description, etc.) ensured by MapDispatcher agent (see the Yellow Pages in [3]).

- This communication realizes a synchronization of mobile agent data. It runs during whole car agent's life.
- Mobile agents can communicate between each other to negotiate emergency situations and/or to get some new knowledge, addition information on surrounding world, etc.

5 Car as an Intelligent Agent

Previously mentioned approach of reconfiguration is applied within every car agent. Moreover, a car agent needs certain form of inner structure for intelligent determination. The basic features of an intelligent mobile agent are perception, decision making and acting.

5.1 Perception

It is a natural feature of each live organism and the same holds for the intelligent agent. The perceptions are based on technical facilities. There are several information sources, e.g:

- sensors
- GIS data
- communication with other agents

An Implementation of a car agent in the real operation needs some hardware sensors, e.g. digital cameras, ultrasonic detectors, GPS, etc. Nevertheless, this research deals with software simulated perceptions. According to the mentioned architecture, the proxy agents ensure such kind of sensors. These agents have full access to platform information and can simulate sight and location of substituted agents. Provided data is sent by proxy agent to its car agent via ACL message.

Spatial data represents the most important data for mobile agent. For the simulation, it is better to appear from real data on traffic infrastructure, which can be obtained from some Geographical Information Systems (GIS). The mission of GIS is not to provide detailed description of infrastructure. It consists in map services ensuring road finding, traffic signs positioning, traffic element description, etc.

The last information source appears from communication among agents. Agents can interchange some knowledge (traffic jam, accident location, etc.) or they use services (parking payment, call for help, etc.).

5.2 Decision making

A rational agent in a multi-agent world is able to reason about the world (what holds true and what does not), about its own cognitive state, and about that of other agents [8]. A theory formalizing reasoning of autonomous intelligent agents has thus to be able to "talk about" and quantify over the objects of agents' attitudes, iterate attitudes of distinct agents, express self-referential statements and respect different inferential abilities of agents. Since agents have to communicate, react to particular events in the outer world, learn by experience and be less or more intelligent, a powerful logical tool is of a critical importance.

To this end we make use of Prolog and Transparent Intensional Logic [11, 2]. The logic tool is represented by an agent brain; currently, an external software component used as a remote service. The communication with the brain is based on request/response approach, where the request has to contain relevant information on agent state, intention, it's seeing, nearest surround and other knowledge. The response of this service should determine next step of agent life process with respect to request content. Next, the intelligent decision making is used within reconfiguration process, which was already mentioned.

5.3 Acting

Acting is a natural consequence of two previous features. It means the concrete process firing which leads to pass agent's objectives. At the end of this, agent's state, as well as the state of whole MAS, has to be updated. Then, the process compound of perception, decision making, and acting is repeated.

5.4 Illustrative example

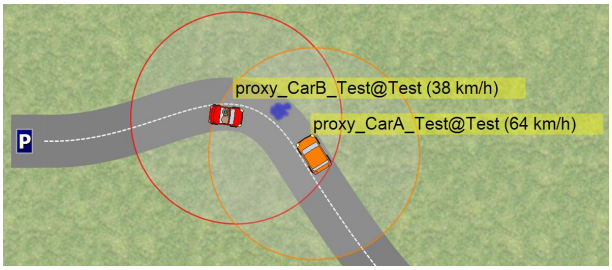

Figure 6. Illustration of situation within the simulated world

The figure 6 illustrates a common traffic situation. There are two cars on a simple road. They are on different lanes, however they have to solve conflict situation because of water puddle on the road. The CarB can continue to drive but the CarA has to change its behaviour. The solution of the behaviour change is described below.

5.5 Perception of the CarA

The agent sees everything in the circle bounded by the orange colour. Concretely:

Infrastructure - road element that is composed of two lanes.
Objects - CarB driving on the contra-flow lane in distance 150 meters with the speed 38 kilometres/hour; Water puddle placed on the current lane in distance 105 metres.

Such data is described by XML in **Agent**Studio Simulator and it is used within decision making process.

5.6 Decision making of the CarA

Agent CarA holds some information on its state, e.g. current speed, orientation, intention, physical properties (car weight, maximal speed and acceleration). The state data together with data obtained from perceptions and/or traffic rules form the input data for logic resulting. In our example (see the figure 6), the agent CarA implements a behaviour, that consists of *NEXT_STEP* process. This process has following set of realizations:

- WITHOUT_CHANGE
- CHANGE_LANE
- SLOW_DOWN
- SPEED_UP

Then, the *NEXT_STEP* process is reconfigured according to existing data and the best realization is selected, included into behaviour and initialized. The *SLOW_DOWN* realization is selected because there is no chance to drive safety around the obstacle. There is the puddle on the road and the oncoming car is too close, therefore the CarA can not change the lane.

5.7 Acting of the CarA

The CarA decreases its speed and tries to solve the situation again with new values and conditions.

6 Conclusion & Future Work

The result of our present research on Multi-Agent Systems was described in this paper. The extended UML diagrams (ABD) were used to model agents' behaviours. The reconfiguration principles were developed and implemented in the **Agent**Studio application. The future work attention has to be paid to imperfect and vague information, its sharing and knowledge storing. Automatic code generation must be improved to reduce human work during MAS development.

This research has been supported by the program Information Society of the Czech Academy of Sciences, project No. 1ET101940420 "Logic and Artificial Intelligence for multi-agent systems"

References

[1] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. JADE - A White Paper, 1999.
[2] M. Duží. Concepts, Language and Ontologies (from the logical point of view). In *Information Modelling and Knowledge Bases XV. Amsterdam*. IOS Press, 2004.
[3] J. Odell. The Foundation for Intelligent Physical Agents. http://www.fipa.org/, 2006.
[4] D. Pilone and N. Pitman. *UML 2.0 in a Nutshell*. O'Reilly Media, Sebastopol, 2004.
[5] M. Radecký and P. Gajdoš. Process and Logic Approaches in the Intelligent Agents Behaviours. In *EJC 2006*, pages 264–270, 2006.
[6] M. Radecký and P. Gajdoš. Process and Logic Approaches in the Intelligent Agents Behaviours. In *Information Modelling and Knowledge Bases XVIII.*, Amsterdam, 2007. IOS Press.
[7] M. Radecký and I. Vondrák. Agents and Their Behavior's Reconfiguration. In *ECEC 2006*, pages 113–120, 2006.
[8] K. Schelfthout and T. Holvoet. An environment for coordination of situated multiagent systems. In *1st Int. Workshop on Environments for MAS (E4MAS)*, 2004.
[9] S. M. L. Specification. http://www.sysml.org, 2007.
[10] U. . O. Specification. http://www.uml.org, 2007.
[11] P. Tichý. *The Foundations of Frege's Logic*. De Gruyter, 1988.
[12] A. UML. http://www.auml.org, 2005.
[13] W. van der Aalst. Verification of Workflow Nets. In *Application and Theory of Petri Nets 1997*, 1997.
[14] I. Vondrák. *Methods of Business Modeling*. VŠB-TUO, Czech Republic, Ostrava, 2004.