# Using the SPACE intermediate model to help to implement agents

Dominique Fournier, Bruno Mermet, Gaële Simon, Marianne Flouret
Laboratoire d'informatique du Havre – BP 540
76058 Le Havre Cedex
FRANCE

**Abstract.** In this article, an intermediate model for specifying agents is described. This model makes easier the transition between a conceptual model (like the BDI one) and a multiagent environment. The SPACE model is specified. An example of application to BDI agents is also presented.

**Keywords:** Multiagent systems, model, software engineering, validation

## 1. Introduction

For the last decade, in order to answer to the growing interest for multi−agent systems (MAS) development, several models and tools have been proposed ([3, 6, 8]). Among them, there are some methods ([2, 8]), agents models ([1, 5]), MAS models and MAS development environments. Our research focuses on the design of a MAS development method allowing, at the same time, validation, verification or even system proof. Few already proposed methods provide enough elements necessary to achieve these tasks. In [7], we have described a first development of our method dedicated to optimization problems. The main goal of this paper is to propose an agent model allowing to completely and precisely specify agents produced by our method. This model must also provide an easy way to operationalize these agents.

This model, called SPACE (Structure, Perception, Action, Comportment and Environment) can be seen as an interface between agents models, like BDI, Aalaadin, Vowel, ... ([1, 5]) used to characterize agents behaviour, and their corresponding implementation in a MAS development environment.

A SPACE specification of an agent gives the most declarative operational view as possible of this agent. This property gives the basis to potentially future validations of the agent implementation.

The first part of this paper deals with a SPACE model description. The second part describes the operational semantics of the model. Finally, the third one focuses on a case study that deals with BDI model and gives its expression with SPACE.

## 2. The agent model : SPACE

In this part, we present the different elements of the SPACE agent model. As explained before, this model is designed in order to be a good compromise between a purely executable agent specification about which it is not possible to reason and a more formal specification not really implementable. We try to obtain an almost runnable agent specification while keeping the most declarative description as possible. As a consequence, the resulting specification can be considered as an operational one: we provide not only models but also an operational semantics describing the effective operation of the agent from these models.

In SPACE, an agent is represented by three complementary models and a structure: the perceptions model, the actions model, the behaviour model, the inner structure. These different elements are described in the following paragraphs. They are of course not independent from each other. That is why they must satisfy a set of consistency rules we yet defined [4] but that are not presented in this paper.

## 2.1 Perceptions model

This model describes the set of perceptions of the agent, that is to say, whatever is likely to modify the actions the agent may do. There are two kinds of perceptions: Messages perceptions and Variables Perceptions.

### 2.1.1 Variables Perceptions (EVP/IVP)

These perceptions are connected to inner and environment variables. The environment in which the agent is living is considered as a set of variables. The environment evolution implies changes on the values of the variables. *A priori*, an agent has only a partial and biased view of the environment. This view is also biased because it depends on the agent interpretation. By this way, it is easy to take into account the point of view of the agent in its behaviour. Thus, some elements of the environment may be considered differently according to the goal of the agent.

An Environment Variable Perception associates a particular value to a subset of values of significant environment variables for the agent. It allows to design the agent point of view on the environment. For example, let us consider weather variables like wind, hygrometry and temperature. The perception "weather" could have four values : nice, very nice, bad and very bad. Thus, when the wind speed is lower than 15 mph, the hygrometry fewer than 60% and the temperature between 20°c and 25°c, the weather perception shall be nice.

Inner variables perceptions rely on the same principles, but they are not based on the same kind of variables. It is assumed that an agent maintains some variables during its life. Generally, these variables are connected to the problems and goals the agent has to achieve. They are designed to manage some aspects of the agent behaviour.

Formally, variables perceptions are defined by a 5−tuple (N, C, V, T, F) with:

| ***name*** | ***type*** | ***role*** |
|---|---|---|
| N | String | Name of the perception |
| C | {EVP,IVP} | Kind of the perception |
| V | {variables} | Set of significant variables |
| T | {values} | Set of potential values |
| F | Function | Function from V to T describing the value of the perception according to the value of the concerned variables |

### 2.1.2 Messages Perceptions (IMP/RMP)

A first class of messages perceptions is *Information Messages Perceptions* (IMP). These perceptions are based on information messages. An information message is considered as a set of variables. Receiving these variables may modify a subset of receiver's inner variables. Yet, every values are not necessarily useful to the receiver. An IMP allows to define which elements of the message are relevant. The IMP also provides the way to read and use the information the message contains.

These perceptions are defined by a 5−tuple (N, C, S, V, F) with:

| *name* | *type* | *role* |
|---|---|---|
| N | String | Name of the perception |
| C | IMP | Kind of the perception |
| S | Class | Message structure (name and type of each variable) |
| V | {inner variables} | List of inner variables the perception may modify |
| F | Function | Casting function from S to V |

We defined also another class of messages perceptions : the *Request Messages Perceptions* (RMP), that are described in [4]**.**

### 2.2 The actions model

The actions model allows to describe the entire set of actions the agent is able to do. It also allows to conditions in which they can be performed (precondition), and also properties (on inner or environment variables) always having to be checked after their execution (postcondition). We can notice that sending messages is considered as specific actions. The associated precondition describes, in this case, when the agent can send the message.
An action is then defined by a 3−tuple (Pre, M, Post) with:

| *name* | *type* | *role* |
|---|---|---|
| PRE | Event | Precondition. Preconditions are specified by events (see below) |
| M | Method | Describes what the action does |
| POST | {properties} | Postcondition, expressed as a set of properties. A property is a constraint over one or more inner or environment variables |

An event is a couple (name, function). The function is defined on values of one or more perceptions or events, and its result is in B = {True, False}. This notion allows to specify a significant special context for the agent behaviour, i.e. in which the agent can implement specific actions.
For example, if an agent wants to play tennis, several conditions have to be checked. To express them, an event "tennis condition" will be defined. This event takes into account values of perceptions as "weather" (see paragraph 2.1.1) and "free partner". Thus, this event will be "true" if the "weather" perception value is "nice" or "very nice", and if the "free partner" perception value is "true".

### 2.3 The behaviour Model

A part of the agent behaviour is formalised by specific automata which are transducers [7]. Within the framework of our model, the agent automaton is in fact the representation of a decomposition of its goals. We give below the state and transition patterns with more details.
A state $p$ is defined by a 7−tuple (N, L, M, Ve, Vs, V, $\pi$) such that N is a string labelling the state, L={(E, A)} is an ordered list of couples (event, action) −an action A can be realized only if the associated event E is true −, M is a method, possibly neutral, which sorts actions of L if a specific order is needed, Ve is the set of input variables, i.e. variables the agent can read when it tries to reach the goal $p$, Vs is the set of output variables, i.e. the ones it can modify during its stay in state $p$, V is the variant of the goal $p$, and $\pi$ is the choice policy of transitions (priority order,...)

starting from state *p* when several events are realized.
A transition between two states Ed and Ea is defined by a 4−tuple (Ed, Ea, E, A). Such a transition shall be fired when the current state is Ed and when event E is true. The action A will then be run and the current state becomes Ea.

### 2.4 Inner structure of the agent

The inner structure of the agent contains the set of inner variables which represent, in real time, the evaluation the agent can do of its own situation. It contains also a list of waiting resquests.
The inner structure of the agent is also made up of its relationships, in fact the list of other agents it knows. This list shall be linked with a qualitative judgment about the kind of relationship (friends, enemies,...). Last but not least, the inner structure includes the behaviour automaton, and a reference to the current state in which the agent is.

## 3. Application to BDI Agents

BDI agents ([5]) rely on 3 essential characteristics : their beliefs, their desires and their intentions. Beliefs correspond to how the agents percieve their environment and their inner state. In our model, beliefs of BDI agents can be easily translated into perceptions.
Desires correspond to the set of goals an agent may have. In SPACE, the behaviour of an agent being described by an automaton whose states are associated to goals, the set of desires of a BDI agent will correspond to the set of states of the automaton.
Finally, intentions correspond to the current goal and eventually to an intention about next goals or actions. The current goal is represented, in SPACE, by the current state. The intentions about the next goals or actions can be modelised by inner variables. The latter must be taken into account by the events associated to transitions.
The typical algorithm describing how a BDI agent works is the following :

| | |
|---|---|
| *initial−state();*<br>*repeat*<br>  *options := option−generator(event−queue);*<br>  *selected−option := deliberate(options);*<br>  *update−intention(selected−options);* | *execute();*<br>  *get−new−external−events();*<br>  *drop−successful−attitudes();*<br>  *drop−impossible−attitudes();*<br>*end repeat* |

This algorithm can also be translated easily into SPACE. The option generation corresponds to the calculus of the perceptions associated to the transitions starting from the current state. The choice of the option and the updating of the intentions of the agent corresponds to the choice of the transition if several are fireable. The executed statement is similar to the standard behaviour of the SPACE agents when they are in a given state. Let us notice it is not necessary to translate the *get−new−external−events()* function because require events are automatically evaluated before executing actions. As well, the deletion of successful goals is a consequence of the automaton structure. Indeed, solving a goal implies the evaluation of the transitions coming from the state associated to this goal. By this way, the current state and as a consequence the current goal of the agent change. Finally, the deletion of impossible attitudes can be taken into account by adding an inner variable indicating when a goal is impossible. Then, transitions ending in the state associated to this goal must evaluate this variable.

## 4. Conclusion

The aim we had when we developed the SPACE model was to enlight as many properties as possible that could be necessary to validate the produced Multi–Agent System. Previous works were realized with the same goal. This is for instance the case of Metatem[2] and Gaïa[8]. A comparison of them with SPACE can be found in [4].
SPACE is designed to be a bridge between higher–level (in terms of abstraction) models like BDI, Eco–Agent, ... and their corresponding implementation in a MAS development platform. The main goal is to provide a runnable description of an agent, while keeping a description level which allows to reason about it, especially for verification and validation purpose. It is the main interest and contribution of the SPACE model.
Our work will focus on the design of a MAS development method integrating the SPACE model. More precisely, the goal is to extend the method we have already proposed in a previous paper [7], in order to produce an agent model expressed with SPACE. This method must also provide tools allowing to use the different components of the SPACE model in order to verify and validate the produced agents and then the entire MAS.

## References


1. Jacques Ferber and Olivier Gutknecht, *A meta–model for the analysis of Multi–agent Organisations in multi–agent systems*. In Third International Conference on Multi–Agent Systems (ICMAS 98) proceedings. IEEE Computer Society, p128–135, 1998.
2. M. Fisher. *A Survey of Concurrent MetateM – the Language and its Applications*. In D.M. Gabbay and H.J. Ohlbach, editors, Temporal Logic – Proceedings of the First International Conference, number 827 in LNAI, pages 480–505. Springer Verlag, 1994.
3. Carlos Iglesias, Mercedes Garrijo, and José Gonzalez. *A survey of agent–oriented methodologies*. In Jörg Müller, Munindar P. Singh, and Anand S. Rao, editors, Proceedings of the 5th International Workshop on Intelligent Agents: Agent Theories, Architectures, and Languages (ATAL–98), volume 1555, pages 317–330. Springer–Verlag: Heidelberg, Germany, 1999.
4. Bruno Mermet, Gaële Simon, Dominique Fournier, Marianne Flouret, *From the problem to its implementation, a methodology and the SPACE model to design Multi–Agents Systems*, ProMAS 2003, Melbourne (Australia), 2003 *to appear*.
5. A. Rao, M. Georgeff, *BDI Agents from Theory to Practice*, Technical Note 56, AAII, April 1995.
6. Arsène Sabas, Sylvain Delisle, and Mourad Badri. *A comparative analysis of multiagent system development methodologies : Towards a unified approach*. In Robert Trappl, editor, Cybernetics and Systems, pages 599–604. Austrian Society for Cybernetics Studies, 2002.
7. Gaële Simon, Marianne Flouret, Bruno Mermet, A methodology to solve optimisation problems with MAS ; application to the graph colouring problem, AIMSA 2002, Varna 2002.
8. Michael Wooldridge, Nicholas R. Jennings, and D. Kinny, *The Gaia Methodology for Agent Oriented Analysis and Design*, Journal of Autonomous Agents and Multi–Agent Systems, 3(3), p 285–312, 2000.