MInD: don't use agents as objects

Renato Lenz Costalima[1], Amauri Holanda Souza Junior[1], Cidcley Teixeira de Souza[1], and Gustavo Augusto Lima de Campos[2]

[1] Instituto Federal do Ceará, Av. Treze de Maio, 2081, Benfica, Fortaleza - Ceará, Brasil
[2] Universidade Estadual do Ceará, Av. Dr. Silas Munguba, 1700, Campus do Itaperi, Fortaleza - Ceará, Brasil

Abstract. What is intelligence? Since it is not possible to see the internal details of intelligence, it is described by its behaviours, that include: problem solving, learning and language [1]. These behaviours are expected outputs of intelligence, but they are not the intelligence itself. Intelligence is rather what makes them possible. That could be: "The capacity to acquire and apply knowledge". With that goal, the **MInD**, a **M**odel for **In**telligence **D**evelopment, is an in development framework for multi-agent systems.

1 MInD - Model for Intelligence Development

Russell *et al.*[2] defines an agent as "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors" (Figure 1). A quick Abbott's textual analysis of this definition identifies the interfaces: `Agent`, Environment, Sensor and Effector or Actuator; and the methods: `perceive()` and `act()`. A representation using the UML sequence diagram (Figure 2) helps to enlighten a sensor gathering information from the environment and representing it into a perception sent to the agent. Some agent's internal decision must choose an actuator and make it act, somehow modifying the environment.

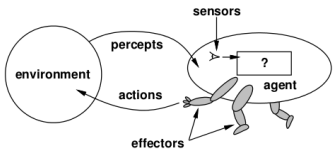

Fig. 1. An agent

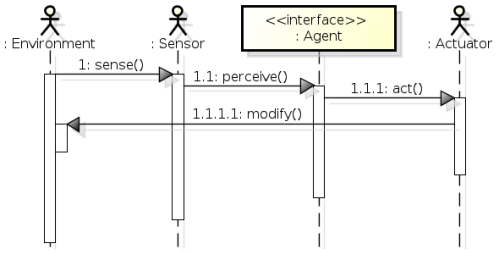

Fig. 2. The UML sequence diagram for an agent

The only method in `Agent` interface is `perceive()`, for short `see()`, and every interaction with the agent must happen through this method. Surprisingly,

none of the major multi-agent frameworks, such as [3][4], defines the method `perceive()` or anything alike.

To be considered intelligent, an agent must be able to "acquire and apply knowledge". To acquire the information about the environment and then apply it, modifying the environment, the agent needs sensors and actuators. While the body is responsible for gathering and representing the information, the mind (the agent) must be able to store (`set()`) and retrieve (`get()`) the information represented by the body. Figure 3 shows the UML class diagram of the **MInD**, a **M**odel for **In**telligence **D**evelopment.

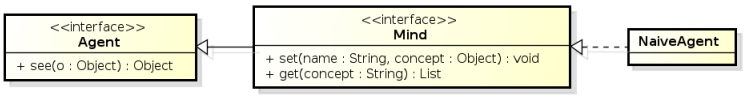

Fig. 3. The UML class diagram for MInD

To test the model, the `NaiveAgent` class provides a flexible simple Java implementation of the `Mind` interface. It describes an initial basic cognitive cycle that cannot solve any problem, that cannot communicate, that does not move or is proactive in any way. However, an instance of `NaiveAgent` represents the mind of a live software ready to learn whatever behaviour imagined. It is programmed to `see()` symbols that represent orders the agent should try to follow. The agent searches in its memory for possible actions. If it find one or more actions, it will choose the last defined action and act. Else, it does nothing. For the agent to find something in its memory, the knowledge must be somehow acquired by a sensor and `set()` to the agent's memory. The body of the agent is the responsible for the knowledge definitions and updates, consequently determining the agent's behaviour as suggested in [5][6]. That is because the agent acts based on its own experience, and this experience is provided by the agent's sensors. The agent's body determines what the agent perceives, consequently determining what it learns and, thereafter, how it behaves.

Because we want to test the mind separately, let us simulate the body's impulses and see what we can do with a `NaiveAgent`. The code in Listing 1.1 shows, in Lines 3 and 10, the simulation of the same perception by the agent resulting in different outputs. At the first time, the agent does not know what to do, so it does nothing. In Lines 4 to 9, the agent is arbitrarily taught how to write, producing, from the second perception, the output of Line 1 of the Listing 1.2. In Line 12, we try to ask the agent to write a sum of numbers. Because it does not know how to sum, it writes "null" (Line 2 of Listing 1.2). The same perception at Line 23, after the agent has learned how to sum (Lines 13 to 22), produces the desired output (Line 3 of Listing 1.2).

```
1  Mind a = new NaiveAgent();
2
3  a.see(new Symbol("write", "hi"));
4  a.set("write", new AbstractAction() {
```

```
5           public Object act(Object object) {
6                   System.out.println(object);
7                   return null;
8           }
9   });
10  a.see(new Symbol("write", "hi"));
11
12  a.see(new Symbol("write", a.see(new Symbol("sum", new int[] {2,-4,5}))));
13  a.set("sum", new AbstractAction() {
14          public Object act(Object object) {
15                  int sum = 0;
16                  int [] numbers = (int []) object;
17                  for (int n: numbers) {
18                          sum += n;
19                  }
20                  return sum;
21          }
22  });
23  a.see(new Symbol("write", a.see(new Symbol("sum", new int[] {2,-4,5}))));
```

Listing 1.1. Test of NaiveAgent

Output:

```
1  hi
2  null
3  3
```

Listing 1.2. Output of the test of NaiveAgent

Teaching an agent to sum and to write is not very impressive, but it demonstrates the agent's capacity to acquire and apply knowledge and doing so in different domains. If one is not satisfied with this definition of intelligence, any other behaviour considered necessary to achieve intelligence can be learned by the agent in execution time. Furthermore, most, if not all, AI's techniques can be implemented using OO classes and objects. In practice, the `NaiveAgent` can be seen as a dynamic object with its attributes and methods defined at runtime.

The `NaiveAgent` reproduces the desired intelligent behaviour, that is, acquiring and applying knowledge. But that is not good enough if we are going to use agents as objects. Wooldridge [7] explains that agents should not invoke each others methods in agent-oriented world. A method invocation is like pushing a button: the corresponding behaviour is triggered without the object's decision. Instead, agents should communicate with each other "asking" for a desired operation. Because of that, multi-agent frameworks focus on the communication between the agents, defining specific communication methods and protocols. This means that the agent is not forced to execute the desired action, but it is forced to communicate. Also the communication language is "hard-coded".

In MInD, the `Agent` interface has only one method: the `see()` method. All interactions with agents, including attempts of communication, should be through a `see()` method. Invoking any other different method is like using the agent as an object, triggering behaviours without the agent's permission. The `Mind` interface abstracts the necessary methods to achieve intelligence. It means that an agent will probably need to implement and use these methods. However, invoking them is forcing the agent to act without asking. The body of the agent supposedly has a close relationship with the agent's mind and could arbitrarily manipulate it through the `Mind` interface. Even so, to maximize the agent's

choice and flexibility, the `Mind` interface should be avoided and its superinterface `Agent` should be used instead.

Agents should not used nor built as objects. Several frameworks offer basic functionality to support multi-agent systems development. Extending one of the provided basic implementations, an agent can be programmed to learn a specific task following some specific approach. After the programming is finished, the agent is put to life and tested if it does the right thing. If the test does not succeed, the agents must be stopped. To use the actual terms, the process is killed. Another round of programming can bring some new agent to life. Is that how it works with intelligent beings? If your kid fails a math exam do you kill him and try to make a smarter kid? The basic implementation provided by the MInD framework is not meant to be extended. The development starts by instantiating a `NaiveAgent`. The desired functionalities are defined at runtime, allowing the modification of the software without the hideous cycle of "edit-refresh-save". In analogy with Web 2.0, MInD allows a single method to be updated without the need to restart the entire software.

The test of Listing 1.1 uses the `Mind` interface and thus uses the agent as an object, arbitrarily manipulating its knowledge. As discussed, only the agent's body should use the `Mind` interface. Imagine how it would be easy for a teacher to just press the "set" button inside the student's head. In further analysis, it is also not so easy to even get the student's attention, to make him really "perceive". In a more realistic agent-oriented world, the environment should not have access to the agent at all. Indeed, as can be noticed in the Figure 2, the environment only interacts with the agent's body (its sensors and actuators), but never with its mind (represented by the `Agent` interface). From the point of view of the environment, the `Agent` and `Mind` interfaces should not even exist. In fact, there is no way of proving that minds exist, we just suppose they are in control of the bodies. Because if not a mind, what is? Yet, has anyone seen a mind? If we open a human's head we will find a brain, but can we find a mind?

References

1. Pfiefer, R., et al.: Understanding intelligence. The MIT Press (2001)
2. Russell, S.J., Norvig, P., Canny, J.F., Malik, J.M., Edwards, D.D.: Artificial intelligence: a modern approach. Volume 2. Prentice hall Englewood Cliffs (1995)
3. Bellifemine, F., Caire, G., Rimassa, G., Poggi, A., Trucco, T., Cortese, E., Quarta, F., Vitaglione, G., Lhuillier, N., Picault, J.: Java agent development framework. TILAB Italia, http://jade. cselt. it, status **10** (2002) 2002
4. Baumer, C., Breugst, M., Choy, S., Magedanz, T.: Grasshopper—a universal agent platform based on omg masif and fipa standards. In: First International Workshop on Mobile Agents for Telecommunication Applications (MATA'99), Citeseer (1999) 1–18
5. Brooks, R.A.: Elephants don't play chess. Robotics and autonomous systems **6**(1) (1990) 3–15
6. Lakoff, G., Johnson, M.: Philosophy in the flesh: The embodied mind and its challenge to western thought. Basic books (1999)
7. Wooldridge, M.: An introduction to multiagent systems. John Wiley & Sons (2009)