Towards a Generic Anticipatory Agent Architecture for Mobile Robots*

Noury Bouraqadi[1] and Serge Stinckwich[2]

[1] Dépt I.A. – Ecole des Mines de Douai – France

[2] GREYC – CNRS / Université de Caen – France

Abstract. An anticipatory agent [1] is a hybrid agent which is able to predict changes of itself and its environment. Such agents prove interesting [2] [3] [4] in embedded systems such mobile robots. Indeed, they combine a reactive fast layer with a cognitive layer capable to perform corrective actions to avoid undesired situations before they occur actually. We present in this paper a generic architecture, that we plan to use as a guideline for developing anticipatory agents embedded into robots for search & rescue missions. Our approach relies on software components in order to explicit the anticipatory mechanisms.

1 Davidsson Quasi-Anticipatory Agent Architecture

From the definition of Rosen [5], Davidsson defines a very simple class of anticipatory agent system: it contains a causal system S and a model M of this system that provides predictions of S. As the model M is not a perfect representation of the reactive system, this is called a quasi-anticipatory system. This architecture is rather coarse-grain. It is only composed of 5 parts:

- Sensors: provide information about the agent environment.
- Effectors: allow the agent to act upon its environment.
- Reactor: drives the effectors in reaction to latest information provided by sensors.
- World Model: is an abstract view of the agent's environment based on data collected using sensors.
- Anticipator: modifies the reactor to avoid undesirable predicted world state.

2 MALEVA: A Software Component Model Expliciting Data and Control Flows

Software component [6] is a programming paradigm that aims at going beyond Object-Oriented programming from the point of view of modularity, reuse and improvement of

* This work is partially supported by the CPER TAC 2004-2006 of the region Nord-Pas de Calais and the european fund FEDER.

Bouraqadi N. and Stinckwich S. (2007).
Towards a Generic Anticipatory Agent Architecture for Mobile Robots.
In *Proceedings of the 3rd International Workshop on Multi-Agent Robotic Systems*, pages 102-105

software quality. Indeed, a software component is a software entity which explicits its dependencies and interactions with other components and resources it relies on.

In this paper, we use the MALEVA hierarchical component model [7] in order to define and implement our anticipatory hybrid agent architecture. Indeed, MALEVA components are close to building blocks of the Brooks subsumption model in their encapsulation and interaction through data exchange [8].

A MALEVA component is a run-time software entity providing encapsulation like objects, while expliciting its interactions with other components. MALEVA components interact only through their interfaces. Interfaces can be of two kinds: data interfaces or control interfaces. Data interfaces are dedicated to data exchange, while control interfaces are dedicated to control flow.

A component can be either active or passive. A passive component is a component that does perform some computation only after being triggered through one of its control input interfaces. Once the component computation is over, it stops until being again triggered. Contrary to a passive one, an active component don't need to be triggered to act. It uses a thread in order to run autonomously.

3 Overview of our Generic Anticipatory Agent Architecture

As shown on figure 1, our agent architecture is an assembly of five components: sensors, effectors, reactor, reaction ticker and anticipator. The first three parts (namely: sensors, effectors and the reactor) are application specific. However, the reactor is instrumented in order to provide two generic interfaces for modifications input: one for modification data flow and the second for modification control flow. The former allows the anticipator to provide modifications to be performed on the reactor, while the latter allows the anticipator to trigger the modifications. These two interfaces can be viewed as the so-called "meta-interfaces" in the work on Open Implementation [9], since they allow a disciplined modification of the reactor.

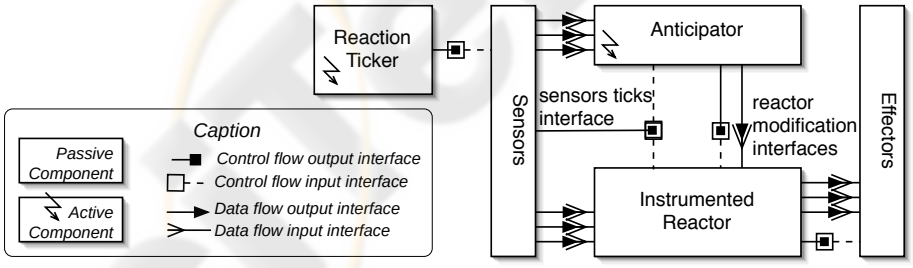

Fig. 1. Our Anticipatory Agent Architecture.

The "Reaction Ticker" is a generic active component that drives the agent's reaction. It defines the frequency at which the agent will sense its environment and react to changes. Indeed, the "Reaction Ticker" triggers the Sensors component every m milliseconds, where m is the duration between two ticks and depends on the application context.

The Sensors component[3] collects data from the agent's world and propagates it through its data interfaces to both the reactor and the anticipator. Then, it triggers the activity of both the anticipator and the reactor. When getting triggered, the reactor decides the appropriate reaction to perform and translates this decision into data propagated to the Effectors component[4].

4 The Anticipator Component

The Anticipator component is an active composite component (see figure 2). It is active since it includes its own ticker that allows it to run concurrently to the reactor. The ticking frequency is higher than the "Reaction Ticker" one, since the anticipator has to work faster than the reactor, in order to make useful predictions. By expliciting the ticking control through the "Anticipation Ticker", this frequency can be easily changed. This feature is very important since it allows to tune the anticipator consumption of resources (computing, energy, ...), particularly in case of embedded devices with low capabilities. This frequency can even be changed dynamically, according to resource evolutions, such as the battery level in a mobile robot.

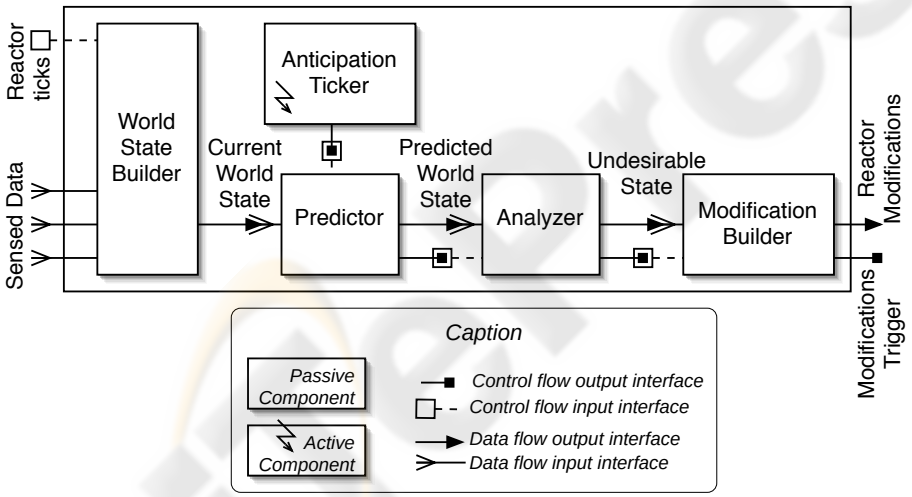

Fig. 2. The Anticipator Architecture.

Each tick of the "Anticipation Ticker" makes the Predictor component predict the next world state and the next reactor action. Then, the Analyzer component analyzes the

[3] Actually, the Sensors component can be a composite with multiple subcomponents corresponding to different sensors.

[4] Actually, the Effectors component can also be a composite with multiple subcomponents corresponding to different effectors.

predicted world state and identifies undesired situations. In case of undesired states, the "Modification Builder" component plans the appropriate modifications and transmits them to the reactor.

It worth noting that, except the data input interfaces connecting the "World State Builder" to sensors, all other interfaces are generic. Therefore this architecture can be reused in multiple contexts.

5 Conclusion and Ongoing Work

In this paper, we draw the foundations for a generic agent architecture based on the Davidsson anticipatory model. This architecture can be reused in multiple contexts and may also serve at the basis for a methodology to design an anticipatory agent

Implementing the examples ("bot in a maze") described in the Davidsson paper, enabled us to prove that it is possible to propose a sufficiently generic architecture for an anticipatory agent regarding the application domain. A more complete validation will be soon carried out with experiments under development of a vacuum cleaner robot simulation. We also plan to experiment our architecture on mobile robots in a search & rescue project.

Another question we would like to explore is how to instrument the reactor component and how to automate the transformation in order to introduce a modification interface. This point is rather complex and varies according to the reactor architecture and its properties. For example, in the case of the subsumption model [8], we need to establish the interaction between the modification interface and the reactor layer.

References

1. Davidsson, P.: A linearly quasi-anticipatory autonomous agent architecture : Some preliminary experiments. In: Distributed Artificial Intelligence Architecture and Modelling. Number 1087 in Lecture Notes in Artificial Intelligence, Springer Verlag (1996) 189–203
2. Stolzmann, W.: Anticipatory classifier systems. In Koza, J.R., Banzhaf, W., Chellapilla, K., Kalyanmoym, D., Dorigo, M., Fogel, D.B., Garzon, M.H., Goldberg, D.E., Iba, H., Riolo, R., eds.: Genetic Programming 3: Proceedings of the Third Annual Conference, University of Wisconsin, Madison, Morgan Kaufmann (1998) 658–664
3. Einarson, D.: Hierarchical models of anticipation. In: CASYS'2001. (2001)
4. Shang, F., Cheng, J.: Implementation issues of anticipatory reasoning-reacting systems. In: International Workshop on Research Directions and Challenge Problems in Advanced Information Systems Engineering. (2003)
5. Rosen, R.: Anticipatory Systems - Philosophical, Mathematical and Methodological Foundations. Pergamon Press (1985)
6. Szyperski, C.: Component Software - Beyond Object-Oriented Programming. 2nd edition. Addison-Wesley (2002)
7. Briot, J.P., Meurisse, T., Peschanski, F.: Une expérience de conception et de composition de comportements d'agents à l'aide de composants. L'Objet **11** (2006) 1–30
8. Brooks, R.A.: A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation **2** (1986) 14–23
9. Kiczales, G.: Beyond the black box: Open implementation. IEEE Software (1996)