Two complementary patterns to build multi-expert systems

Philippe Lalanda
Thomson-CSF Corporate Research Laboratory
Phone: 33 1 69 33 92 90
E-mail: lalanda@thomson-lcr.fr
Domaine de Corbeville
F-91404 Orsay, France

Abstract

The purpose of this paper is to present two complementary patterns to build multi-expert systems, that is systems that need to integrate large, heterogeneous specialized modules, and implement complex, non deterministic control strategies.

The first pattern summarizes the blackboard pattern presented in [1]. The second pattern builds on the model of dynamic control where a blackboard system has a repertoire of independent domain and control knowledge sources, a control plan expressing the system's desirable behavior, and a meta-controller that chooses at each point in time the currently enabled knowledge source that best matches the control plan. A system implementing this architectural pattern is able to dynamically change its control strategies as a response to unpredictable events.

Area: Control in complex, dynamic systems

Keywords: Architectural pattern, blackboard systems, dynamic control

1 Introduction

The blackboard architectural pattern provides a computational framework for the design and implementation of systems that need to integrate large and diverse specialized modules, and implement complex, non deterministic control strategies. This pattern, described in [1], defines several components, of which the control component is the most important. Several models have been proposed to support the definition of this component that is key for the system's global properties.

The purpose of this paper is to present two complementary patterns. The first pattern summarizes the blackboard pattern presented in [1]. The second pattern is concerned with the blackboard pattern's control component. It presents an approach building on a model of dynamic control that allows control strategies to be changed dynamically in response to unpredictable internal and external events.

2 Blackboard architectural pattern

Name

Blackboard

Example

Consider an autonomous robot running in an office environment. It performs tasks like fetching objects, sending messages, guiding visitors, and night-time surveillance. In so doing, it interacts with human beings and other dynamic entities and acquires potentially useful knowledge of its environment.

The robot has to continuously perform three distinct functions:

- It perceives its dynamic environment in order to move and to get orders from human beings,

- It reasons in order to interpret the perceived data, solve problems and determine actions to be triggered,
- It acts on the environment.

Different functions to be performed require various expertise. Modules specialized in various domains such as data interpretation or route planning have to be dynamically combined so that the robot can achieve its task.

Context

Software systems that need to integrate large and diverse specialized modules, and implement complex, non deterministic control strategies.

Problem

Classical knowledge-based systems like expert systems tackle problems that do not have a feasible, deterministic solution. However, these systems are not able to deal with complex applications such as signal interpretation, process control, or autonomous mobile robots control. The main reason is that such applications have very stringent requirements including:

- The problem under consideration spans several fields of expertise
- Intermediate solutions require different representations and paradigms
- The control strategy is complex and cannot be determined statically.

Solution

The blackboard paradigm defines heterogeneous problem solving representations as independent modules called knowledge sources. Knowledge sources can be seen as specialists in sub-fields of the global application and are only able to solve sub-problems. They read and write relevant data in a *blackboard* which is a structured global memory where a solution to the problem under consideration is incrementally constructed. When a knowledge source produces a significant change in the blackboard, it generates an *event*.

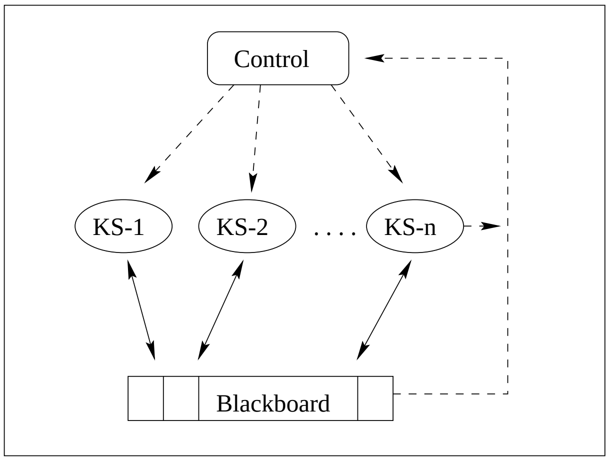

Figure 1: The blackboard model

Each knowledge source has a set of triggering conditions that can be satisfied by particular kinds of events, that is global changes in the blackboard resulting from external inputs or previously executed knowledge sources. When an event satisfies a knowledge source's triggering conditions, the knowledge source is enabled and its parameters bound to variable values from the triggering situation. A given knowledge source will be enabled, and therefore executable, whenever events satisfying its triggering conditions occur, regardless of its relative utility in achieving the current goals.

At each point in time, many competing knowledge sources may be enabled. It is the purpose of the control component to select the best one for immediate execution. The problem solving process is *opportunistic* in that sense that the activations of knowledge sources are not scheduled in advance, but determined at every control cycle depending on the current situation. In the basic blackboard model, control strategies are fixed.

The general organization of a blackboard system is illustrated by figure 1. Dash lines represent control flux whereas solid lines stand for read/write accesses.

Robot example: Software systems controling an autonomous robot have to perform cognitive and physical tasks, including:

- Navigation
- Environment learning
- Reasoning (about the goals that have to be achieved)
- Destination planning
- Route planning
- Execution monitoring

These various tasks can be achieved by one or several specialized knowledge sources. For example, a system usually includes different navigation routines making use of various combinations of sensors, various strategies (*wall following* is an example of strategy), and various properties such as safety or speed.

The blackboard contains global information such as the current goals, destination, route and constraints. It also includes a map which stores static information about the environment and dynamic information related to the robot knowledge and previous activities (level of knowledge of a given zone, routes already taken, *etc...*).

In the basic approach, the control component is a complex scheduler implementing a fixed strategy. At each control cycle, that is after each knowledge source execution, it computes a priority (a rating) for every enabled knowledge source. The rating is computed using a set of *heuristics*. Examples of heuristics are *a navigation routine always gets a better rating than a learning routine*, or *when a "wall following" navigation routine is executable, it reveives the highest rating among navigation routines.*

Structure

The blackboard model defines three main components:

- The blackboard is a structured global memory containing objects from the solution space. These objects, also called blackboard nodes (or hypotheses), are hierarchically organized into levels of analysis and can be linked with each other.
- Knowledge sources can be seen as highly specialized modules with their own representation. They are characterized by a set of triggering conditions and an executable code that retrieves data from the blackboard and adds its contribution to the problem solving process.
- The control component selects, configures and executes knowledge sources. Determination of executable knowledge sources is based on the state of the problem solving process as expressed in the blackboard.

Structural relationships between these components are summarized in Figure 2.

Implementation

The first step when designing a blackboard-based system is to define the solution space, that is the various intermediate solutions and their representation. This activity leads to the definition of the blackboard structure. It is then necessary to identify the knowledge sources that can provide these solutions. Obviously, these two activities are very related and influence each other.

Knowledge sources may correspond to existing modules. In that case, modules have to be put in the form of knowledge sources. That means that triggering conditions have to be added, input variables have to be linked to blackboard data so that they can be bound at run time, and results have to put in the blackboard. This last adaptation is generally made by simple code wrapping.

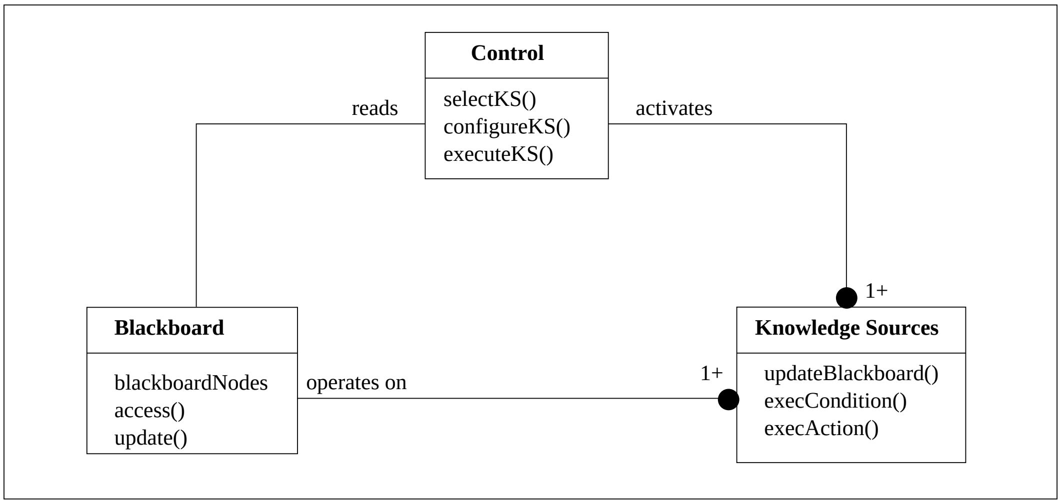

Figure 2: System's structure (adapted from [1])

The next step is to specify the control component. It generally takes the form of a complex scheduler that makes use of a set of domain-specific heuristics to rate the relevance of executable knowledge sources. Tuning heuristics is a very time-consuming activity. It is however a crucial setting that determines the system's suitability.

Known Uses

Blackboard-based systems have been used in numerous domains, including:

- Speech recognition [2]
- Vehicle identification and tracking [3]
- Identification of the structure of protein molecules [4]
- Sonar signals interpretation [5]

Consequences

The blackboard model provides effective solution to the design and implementation of complex systems where heterogeneous modules have to be dynamically combined to solve a problem. It also ensures very important non functional properties such as:

- Reusability. Knowledge sources are independent specialists that can be reused in different projects. Reuse is made easier by the fact that there is no direct communication between knowledge sources.
- Changeability and maintainability. High level of modularization and clear separation between control and domain (*i.e.* the knowledge sources) makes the maintenance phase easier.
- Robustness. The model naturally leads to the definition of alternative knowledge sources to solve a given sub-problem.

The blackboard architectural pattern has also some weaknesses. The main one is certainly the difficulty of defining a good control strategy. Control is based on heuristics tuning that can only be obtained through experiments.

Credits

The blackboard model has been defined by the members of the HEARSAY-II project [2], and first applied on speech recognition. Comprehensive presentation of the blackboard paradigm and existing systems can be found in [5] [6].

The first pattern description of the blackboard model has been presented in [1].

3 Blackboard based control pattern

Name

Blackboard-based control

Example

Consider the autonomous robot running in an office environment introduced in the example section of the blackboard pattern.

Some autonomous robots need to interleave activities depending on the external situation and their current state in terms of goals, resources, information available. For example, suppose a robot is on its way to fetch a book with no hard deadline. The robot may decide to follow a route it has not taken for a while in order to update its knowledge about it. Suppose now that it receives a request to go to a given room as soon as possible. It will then drop its information collecting task and take the quicker and safer route to reach the room. This has an effect on the navigation routines it selects to move, the data it perceives, and the reasoning tasks it performs. More generally, it influences its whole control strategy.

The office robot illustrates a class of systems that have to ensure the cooperation of a set of components in order to solve various problems and that must adapt dynamically their control strategies.

Context

A system that integrates heterogeneous specialized modules and that needs to dynamically change its control strategies as a response to unpredictable events (internal or external).

Problem

The blackboard model is a problem solving model that separates domain knowledge into heterogeneous modules called knowledge sources. A control component provides mechanisms to activate knowledge sources at the right time in order to ensure effective cooperation between them. Selection of the best enabled knowledge source is based on specific criteria and generally demands very precise tuning. In most blackboard systems, these criteria cannot be changed dynamically.

Changing these criteria at run-time is however necessary in many applications like robot motion in uncertain environment, aircraft pilot advising, process control, or intensive care monitoring. These applications require highly adaptive systems that are able to adapt their meta-control strategies to dynamic configuration of demands, opportunities, and resources for behavior.

Solution

The proposed solution builds on the model of dynamic control where a blackboard system has (a) a repertoire of independent domain and control knowledge sources that are described in term of their resource requirements and result properties; (b) a control plan expressing its desirable behavior; (c) a meta-controller that chooses at each point in time the currently enabled knowledge source, domain or control, that best matches the current control plan.

In this approach, criteria guiding the selection of executable knowledge sources are encapsulated in a control plan. A control plan describes the system's intended behavior as a temporal graph of plan steps, each of which comprises a start condition, a stop condition, and an intended activity in the form of a tuple (task, parameters, constraints). Control plans do not refer explicitly to any particular knowledge source in the system's repertoire. They only describe intended behaviors in terms of the desired task, parameter values, and constraints. Thus, at each point in time, the system has a plan of intended action, which intentionally describes an equivalence class of desirable behaviors and in which currently enabled specific knowledge sources may have graded degrees of memberships.

Control knowledge sources are described and managed in the same way as domain knowledge sources. Each knowledge source has an interface that describes the kinds of events that enable it, but also the variables to be

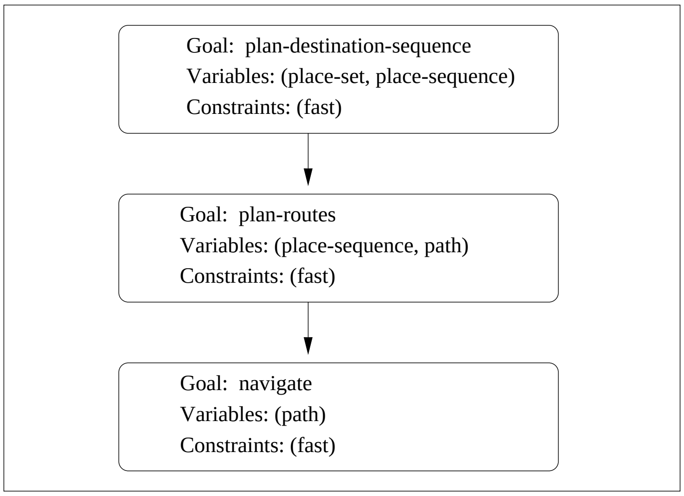

Figure 3: Example of control plan

bound in its enabling context, the task it performs, the type of method it applies, its required resources (e.g, computation, perceptual data), its execution properties (e.g, speed, complexity, completeness), and its results properties.
Domain and control knowledge sources do not access and update the same knowledge base. Domain knowledge sources are concerned with the solving of a particular problem. Control knowledge sources deal with the management of the system control plan. They can replace the current plan, postpone it, refine it, *etc...*

The meta-controller attempts to follow the current control plan by executing the most appropriate enabled knowledge sources. Specifically, at each point in time, the meta-controller configures and executes the enabled knowledge source that: (a) is capable of performing the currently planned task with the specified parameterization; and (b) has a description that matches the specified constraints better than any other enabled knowledge source that also satisfy (a). If the selected enabled knowledge source is a control knowledge source, the control strategy is updated. Otherwise, a domain knowledge source is executed in order to contribute to the problem solving process.

Robot example: Figure 3 provides a simple example of control plan. This plan corresponds to a situation where the robot's high level goal is to visit a set of places as soon as possible. The successive sub-goals that govern the robot's actions are to decide on a sequence of places to visit, to compute the best route, and to navigate with a constraint of rapidity. Note that, for the sake of clarity, we did not include the actions to be undertaken by the robot at the visited places in the control plan (this requires to add a loop navigate / check in the control plan).

Such plan allows the meta-controller to select the best enabled knowledge source at each point in time. It is set by a control knowledge source that made it up or instanciated it from a library of *skeletal plans*. When the current goal is achieved (here, all the scheduled places have been visited) or new conditions appear (new goal, important change in the environment, *etc...*), another control knowledge source can be enabled in order to post a more appropriate control plan.
For example, in a less stressed situation, a new plan could insist on the importance of learning the environment while moving. This would result in the selection of different paths, and different navigation routines.

Structure

The system is divided into five major components:

- The blackboard is the shared data structure where solutions are built
- The control plan encapsulates meta-control information necessary to run the system under the form of a temporal graph of plan steps. It is accessed and updated by control knowledge sources.
- Domain knowledge sources are concerned with the solving of domain-specific problems.
- Control knowledge sources adapt the current control plan to the current situation. Modifications expressing new system's needs range from minor changes to complete replacement of the control plan.

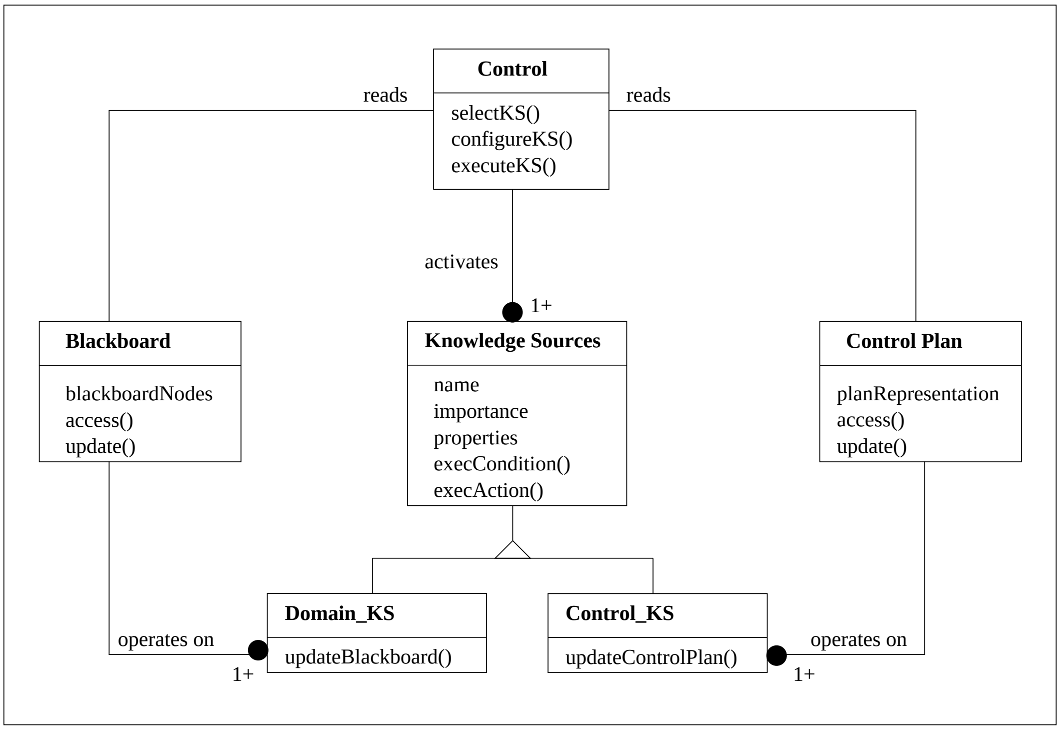

Figure 4: System's structure

- The control component selects, configures and executes knowledge sources. Selection of the most appropriate knowledge sources for execution is based on meta-control criteria contained in the control plan.

Structural relationships between these components are summarized in Figure 4.

Implementation

The blackboard-based control model brings new components that are not considered in the general approach. In particular, developers have to deal with the implementation of control plans.

Control plans can be implemented in many ways. The most general approach is to implement a control plan as a graph. A node represents a step in the problem solving process and consists of an intended activity in the form of a tuple (task, parameters, constraints). Steps are linked to each other through arcs labeled with conditions about the problem solving state. The solving path is determined dynamically, considering the conditions labeling the arcs of the graph.

Control knowledge sources are implemented as domain knowledge sources in terms of interfaces. Interfaces comprise:

- A condition of activation describing, among other possible things, the events that enable the knowledge source,
- Variables to be bound in the enabling context,
- Required resources,
- Non functional properties like safety or efficiency characterizing the execution,
- Results properties.

The action part of a control knowledge source contains either a new plan that has to replace or precede the current one when the knowledge source is executed, or a set of modifications to be brought to the current control plan.

Known Uses

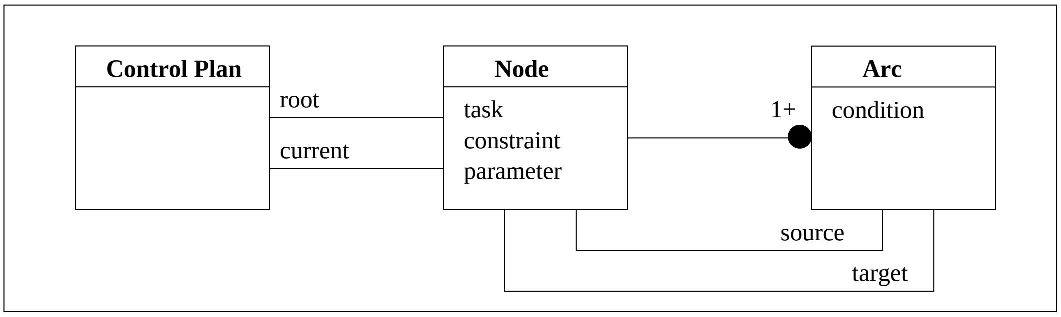

Figure 5: Control plan

This pattern for control has been used for ten years in numerous domains, including autonomous robots [7], systems for monitoring medical patients [8], and pilot's associate systems.

Consequences

This pattern presents the usual properties of blackboard systems as given in [1]. In particular, it provides a framework in which appropriate sets of knowledge sources can be selected and configured at both design time and run time. The integration of knowledge sources is actually very simple since knowledge sources are considered independently and are only characterized by their own properties.
At run-time, if useful new application-relevant knowledge sources should become available, the new knowledge sources can be substituted for old ones or added to the knowledge base alongside the old ones, without interrupting system operations. The architecture's event-based enabling of knowledge sources, its plan-based meta-control choices among competing knowledge sources, and its effort to retrieve necessary knowledge from the shared blackboard are not preprogrammed to require any particular knowledge sources. They operate on whatever knowledge sources are available in the knowledge source library at run-time. In the other hand, the approach does not guarantee the delivering of a solution to a given problem, since appropriate knowledge sources might be missing.

Encapsulating meta-control elements in a single changeable object (*i.e.* the control plan) brings considerable flexibility in the system. Control strategies can be adapted at run time to dynamic conditions. Thus, not only domain components can be plugged in the system both at run time and design time, but control strategies as well.

Finally, this pattern is a valuable basis for the design and development of application frameworks. Application frameworks are semi-complete applications providing architectural foundations to solve a range of domain-specific problems and that need to be specialized to meet particular applications' requirements. Such specializations can be done by plugging the adequate domain knowledge sources and control strategies in control knowledge sources.

See Also

The PRS system (*Procedural Reasoning System*) [9] implements a view of dynamic control very close to the one presented in this paper. PRS is a generic architecture that can both react rapidly to unanticipated changes in the environment and perform goal-directed reasoning. To that end, PRS uses special plans, called *Knowledge Areas* (KA), that include an invocation condition which specifies whether the KA is useful and a body which describes the sequence of subgoals which constitutes the procedure. A subgoal can result in a primitive directly executable or in another KA invocation. A KA expresses several ways to solve a problem, depending on environmental conditions and on the exact problem's nature. At each execution cycle, PRS checks all the KAs for applicability and selects one for execution.

The meta-control approach of PRS is similar to the blackboard-based control and provides flexibility and reactivity. Formalism for knowledge sources is restricted to the graph-based KA. Control plan corresponds to the goal under consideration at each control cycle of PRS.

PRS has been applied in many fields, including telecommunication applications.

Credits

The blackboard-based control model has been designed and developed at Stanford University by Barbara Hayes-Roth [10]. Over the years, several experiments on real-world applications have been conducted in order to validate the model and to show the wide variety of domains that can be tackled with this approach.

4 Conclusion

This paper presents a very first step towards a pattern language to define multi-expert systems. Additional patterns would be needed to describe other existing implementations of the control component in a blackboard-based system like the hierarchical control. Patterns guiding the design of knowledge sources and the selection of appropriate techniques in a given situation would be very useful as well.

5 References

[1] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, *Pattern-oriented Software Architecture: A System of Pattern*, Wiley & Sons.

[2] L. Erman, F. Hayes-Roth, V. Lesser, R. Reddy, *The Hearsay-II speech understanding system: Integrating knowledge to resolve uncertainty*, ACM Computing Surveys 12 (2), 1980.

[3] V. Lesser and D. Corkill, *The Distributed Vehicle Monitoring Testbed: A tool for investigating distributed problem solving networks*, Artificial Intelligence Magazine, vol. 4, nos 3, 1983.

[4] A. Terry, *Using explicit strategic knowledge to control expert systems*, in Blackboard Systems, Addison-Wesley, 1988.

[5] P. Nii, *Blackboard systems: Part I and II*, The AI Magazine, vol. 7, nos 2 and 3.

[6] R. Engelmore, T. Morgan (Editors), *Blackboard systems*, Addison-Wesley, 1988.

[7] B. Hayes-Roth, P. Lalanda, P. Morignot, M. Balanovic and K. Pfleger, *A domain-specific software architecture for adaptative intelligent system*, IEEE Transactions on Software Engineering, 1995.

[8] B. Hayes-Roth and J.E. Larsson, *A domain-specific software architecture for a class of intelligent patient monitoring systems*, Journal of Experimental and Theoretical Artificial Intelligence, 8(2), 1996.

[9] M. Georgeff and F. Ingrand, *Decision making in an embedded reasoning system*, International Conference on Artificial Intelligence, IJCAI-89, 1989.

[10] B. Hayes-Roth, *A blackboard architecture for control*, Artificial Intelligence, num. 26, 1985.