Agent-Based Modeling for Holonic Manufacturing Systems with Fuzzy Control

Stephan Flake*, Christian Geiger*, Georg Lehrenfeld**, Wolfgang Mueller*, Volker Paelke*

C–LAB*, Heinz-Nixdorf-Institut**
Fuerstenallee 11
33102 Paderborn, Germany

Abstract

Agent-based systems technologies are of emerging interest in the specification and implementation of complex systems. This article introduces the CASA agent development system which seamlessly combines the BDI (Belief Desire Intention) approach with the FIPA agent communication language standard and an integrated specification of fuzzy controllers. The behavior of agents is defined by strategies which basically correspond to extended guarded horn clauses with priorities. The presented concepts are introduced by an example from Computer Integrated Manufacturing (CIM). The example gives the specification of a fuzzy controller for a manufacturing station in the context of a holonic manufacturing system (HMS).

1 Introduction

Agent-based systems technologies in the sense of distributed computing is an area of emerging interest in the domain of complex systems design. The agent-based paradigm can be seen as a real enhancement of the object-oriented paradigm where objects become autonomous, proactive, and perceptive with respect to their environments. Meanwhile, a large number of commercial agent management systems are available or are currently under development. Examples are Odysee, Aglets, Voyager[7], Mecca [1]. They support various facilities like implementation of mobile and secure agents mostly dedicated to specific applications such as e-commerce and travel assistance.

In the general field of agent-based systems we can identify two major diverging directions, agent theory and industrial applications. On the one hand there is considerable work on formalization of multi-agent systems, e.g., commitments, capabilities, know-how. Advanced logics which capture essential properties of agents like concurrency, time dependent behavior or inconsistent states are under development. Formal specification techniques like Z are applied to formally capture those systems. In addition, as reported by Nwana in [9], a couple of new concepts in recent multi-agent system research seem to reinvent the wheel. There is already a large amount of relevant and first class literature published in traditional AI, soft computing, parallel computing, and software engineering which directly apply to the principles of agent-based systems. On the other hand, there exists a number of successful implementations dedicated to tackle real-world problems like flight control, manufacturing resource allocation, and information retrieval in large databases. Latter ones mostly follow an ad-hoc approach building systems from scratch. Due to the complexity of the developed code implemented systems often lack an underlying sound theory. Recent work based on Rao's and Georgeff's BDI architecture seems to successfully bridge the gap between industry and theory by providing a formal agent specification language for an already implemented multi-agent systems. Based on this cross fertilization a sound combination of a formal description and an implementation framework has been developed by AgentSpeak(L) [11].

Agent-based systems literature distinguishes micro and macro views. The micro view considers the local behavior of an individual agent whereas its environment and interaction is investigated in the macro view. This article presents CASA. CASA is a multi-agent system based on the BDI architecture and AgentSpeak(L) [10, 11] extented by priorities, hierarchical rules with speculative computations, deep guards, and the integration of fuzzy controllers. The CASA specification language defines a multi-agent systems by means of goals, plans, local knowledge, and meta knowledge. In this article, we focus our investigation on the CASA micro view, i.e., modeling and execution of

agent strategies. Strategies are defined by rules with complex context-sensitive tests. The parallel, event-based evaluation of strategies with priorities is defined by extended guarded horn clauses. Fuzzy controllers are integrated in the definition of local beliefs (facts). We present our approach by means of a realistic example, a so–called holonic manufacturing system (HMS) [14]. The example investigates the interaction of autonomous robotic vehicles moving parts between different manufacturing stations (milling, drilling, washing). We have chosen this example since it gives an ideal application scenario for our purpose combining agent-oriented behavior with fuzzy strategies. Existing works [6, 15] have already demonstrated the applicability of fuzzy technologies (fuzzy linear programming, fuzzy Petri-Nets, fuzzy rule bases, etc) in this area.

In the remainder of this paper we first introduce the basic principles of holonic manufacturing systems with an example. For the example we give a specification of the basic behavior of the main components with an integrated fuzzy controller. Thereafter, we outline the basic concepts of the CASA system and sketch the definition of the fuzzy controller by means of the CASA specification language. The paper closes with a brief overview of the implementation and a short summary.

2 Holonic Manufacturing with CASA

This section first introduces the example of a holonic manufacturing system. Thereafter, we outline the CASA system and the embedding of a fuzzy controller specification on the basis of that example.

2.1 Holonic Manufacturing

Our example specifies the holonic manufacturing system (HMS) introduced by the IMS Initiative[1] TC 5 [14]. An HMS is based on the notion of a holon which denotes a building block of a system for complex production lines. Hierarchically composed holons cooperate along the lines of basic rules to achieve a goal or objective. A single holon is understood as an autonomous co-operative building block which mostly consists of a information processing and a physical part. Holonic system structure is based on the principles of self-similarity and self-configuration which form self-adaptable and thus fault-tolerant networks of holons.

Our HMS example is composed of a set of different stations and a transport system as it is given by the virtual 3D model in Figure 1. The different manufacturing stations transform parts: milling, drilling, and washing. Additional input and output stations are for primary system input and output. The complete transport system consists of a set of HTSs (Holonic Transport Systems). An HTS is an autonomous vehicle which moves parts between stations. An HTS

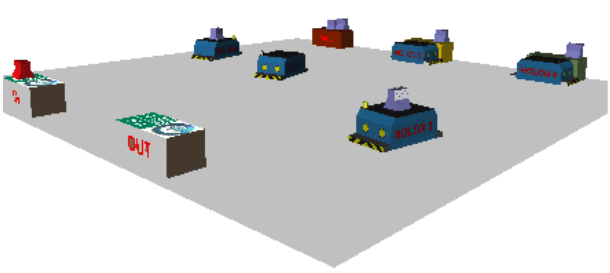

Figure 1. Screenshot of an HMS 3D Simulation

1. is idle until it receives a request for delivery from a station s_i
2. sends the distance d_i from its current position to s_i
3. moves to s_i on notification of acceptance from s_i
4. takes the part from the output and moves it to its destination[2]
5. moves to a parking position and returns to Step 1.

Stations have an input and output buffer for incoming and outgoing part. Once having located one part at the output a station

1. broadcasts a request for delivery to all HTSs
2. receives distances d_i from idle HTSs for a specific time period
3. returns to Step 1 if no HTS replies
4. computes the most appropriate HTS from all received distance values d_i and their reliability factor r_i
5. notifies that HTS for its acceptance and notifies other HTSs for their rejection

The distance d_i is computed as the sum of subpaths from HTS i to the calling station. The reliability factor r_i is a cumulative value which is continuously computed from the estimated arrival time and the real arrival time.

In our example, we implement the computation of Step 4 by means of a fuzzy controller which is given by the following table. The table defines the acceptance for delivery as a function of distance d_i and reliability r_i for an HTS_i.

$d_i \backslash r_i$	low	medium	high
near	medium	high	high
close	medium	medium	high
reachable	low	medium	high
far	low	low	high

[1] IMS (Intelligent Manufacturing Systems) is an industry-led international research & development program to develop the next generation of manufacturing and processing technologies with currently over 250 companies and over 200 research institutions participating.

[2] Without loss of generality we can presume that the destination for the part can be easily retrieved from the part itself, i.e., weight, shape, etc.

2.2 CASA

CASA is a multi-agent system combining the BDI[3] agent micro view with the FIPA ACL [2] macro view for specification, prototyping, and validation of agents [5].

CASA is based on three design principles. Firstly, we considered well known approaches in multi-agent systems which seamlessly combine theory and practice. We therefore selected Rao's AgentSpeak(L) [11] as a semi-formal base for micro level specification. Secondly, we tried to reuse well known existing concepts from other domains as far as possible, i.e., concurrent logic programming [12] and fuzzy logic [4]. Lastly, we implemented our systems by the reuse of available libraries and frameworks. The current implementation intergrates parts of the JAM! agent architecture [8] and the MECCA agent management frame work [1] and combines it with our internal fuzzy library.

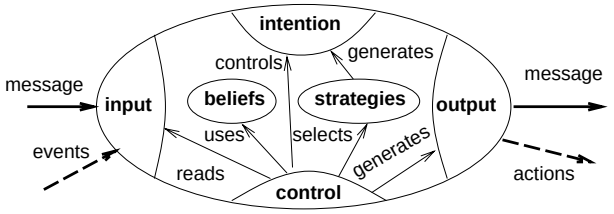

Figure 2. Micro View of an Agent

The specification of the behavior of a CASA agent is formally based on guarded horn clauses [13] which are extended by concepts for events and actions derived from the BDI (Beliefs Desire Intention) approach. A CASA agent has individual control over its behavior. Its internal state is composed of current sets of beliefs, strategies, and intentions (cf. Figure 2). Beliefs are a representation of the agent's local world model and basically compare to facts in logic programming. Intentions are partially instantiated strategies which are currently executed by an agent.

A CASA agent continuously observes its environment. Based on this observations it performs actions which send messages and modify beliefs. The behavior of an agent is basically defined by specifying strategies (resp. plans). Strategies compare to clauses in logic programming. A CASA agent strategy is formally defined as an extended guarded horn clause of form

$$H \rightarrow G_1 ... G_n \mid B_1 ... B_m(p)\,.$$

where H is the head, G_i are guards, B_j are predicates as body goals, and p defines a priority.

The head of a CASA strategy describes the event the agent must perceive in order to execute the plan. The guard elements may consist of any number of predicates and messages which are processed sequentially or in parallel. Additionally, a priority (or weight) allows to choose between different applicable strategies.

Based on the type of the guards different strategies are distinguished. If all guards are simple tests, the strategy is considered as reactive. If goals are also elements in the strategy's guards the strategy is deliberative because the evaluation of the guard requires a speculative computation which evaluates other strategies in order to reduce the goal (multi level plans). If the context test also requires the communication between agents, the strategy is described as communicative. Actions are not allowed in guards. If multiple strategies can be applied for a given event reactive strategies have a higher priority than deliberative strategies. Communicative strategies have lowest priority when the agent chooses an applicable strategy.

During execution, an agent can suspend currently executing strategies and resume suspended ones by using special operations for suspending / resuming.

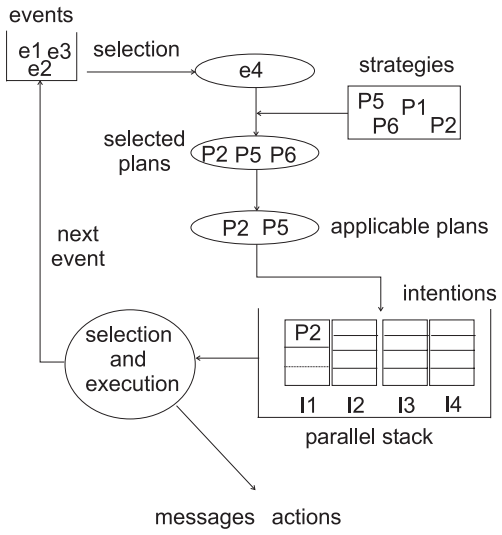

Figure 3. CASA Execution Cycle

The operational semantics of a CASA agent can be best described by means of an abstract interpreter as it is given in Figure 3. The interpreter manages the execution of all agent activities in an interpretation loop. The operation of the agent interpreter is controlled by three functions that control event selection, plan selection, and intention selection. By modifying their implementation, the system can be easily tailored the different operational semantics for various other applications.

The interpretation starts with the selection of an incoming event. In a second step, a set of relevant plans which are appropriate for processing the selected event are identified.

[3] Beliefs Desire Intention

The preconditions of all relevant plans are checked against the facts stored in the agent's beliefs to extract the set of applicable plans, i.e., plans whose preconditions are satisfied. One applicable plan from that set as the pursued strategy is selected. That plan becomes instantiated as an intention on the multistack. The multistack concept allows each agent to investigate several plans in parallel and to instantiate new (sub)intention. Finally, the interpreter selects an intention from the multistack and executes it starting from the top element. Execution can result in either a direct action, the generation of an event, or the instantiation of new (sub)intentions. Thereafter, the interpreter advances to process the next event.

The definition of a CASA agent is implemented by the means of the CASA specification language. In this article, we focus our interest on the specification of facts and plans. They are defined along the lines of the following patterns.

```
FACTS:
FACT   <id> <list of values>;
...

PLAN: {
NAME:          <string>;
DESCRIPTION:   <string>;
GOAL:          ACHIEVE <relation>;
TYPE:          <REACTIVE
                | DELIBERATIVE
                | COMMUNICATIVE>;
PRECONDITION: <list of conditions>;
BODY:          <list of actions>;
FAILURE:       <list of actions>;
PRIORITY:      <numeric value>;
}
```

Facts simply associate list of data values with an identifier. A plan basically gives a definition of a guarded horn clause. In addition to the definition of the body a list of failures can be defined which are elaborated when the evaluation of the body fails. The individual plan elements goals, preconditions, the body and failures are given by the specification of control structures like par, seq, if-then-else, wait. Definition of messages and new subgoals are supported.

For CASA fuzzy controllers we extended the syntax of the fact definition section. Controller definitions are preceded by keyword FUZZY. A controller definition covers the specification of input and output variables, fuzzy sets, and rules over the specified fuzzy set. The following description sketches the definition of the fuzzy controller of the holonic transport system which was introduced in the previous subsection.

```
FUZZY controller {
INPUT:  in1, in2;
OUTPUT: out;

  SETDEFINITION{
  set: dist_near;      0; 0;  0;  2;
  set: dist_close;     0; 2;  5;  7;
  set: dist_reachable;5; 7;  8; 10;
  set: dist_far;       8;10;max;max;
  set: reliab_low;     ...
  ...
  }
  RULEDEFINITION {
  IF in1 == dist_near & in2 == reliab_low
    THEN out = accept_medium;
  IF in1 == dist_near & in2 == reliab_medium
    THEN out = accept_high;
  IF in1 == dist_near & in2 == reliab_high
    THEN out = accept_high;
  ...
  }
}
```

The controller has two input variables for crisp data. The fuzzy sets define functions for mapping of crisp to fuzzy data which are applied in the rule definitions. The above rules directly correspond to the table of the previous subsection which defines the HMS station fuzzy controller. The fuzzy controller is invoked by a station agent with the identifier and two crisp values as it is given in the following example.

```
$ACCEPT = DEFUZZIFICATE controller, 15.4, 0.8;
```

The crisp values are defuzzified by the sets, rules are evaluated, a fuzzified value between 0 and 1 is returned and assigned to the variable ACCEPT. In our example, the station calls the defuzzification with crisp values for each HTS. The HTS with the highest ACCEPT value is finally notified for acceptance of the delivery.

3 Implementation

CASA's micro level view is implemented in Java with JDK 1.1.2 integrating modules of the JAM! library [8]. A compiler written in JavaCC parses CASA programs and executes them on the CASA interpreter. CASA agents are integrated in the MECCA framework [1]. MECCA is an agent mangement system which implements the FIPA ACL (Agent Communication Language) standard [2]. An integrated CASA agent reads and writes messages/events through a specific communication adaptor from/to the internal message transport channel of the MECCA system. This allows CASA agents to communicate with any FIPA compliant agent via the MECCA framework as it shown in Figure 4.

4 Summary and Outlook

We have introduced the agent-based modeling and validation environment CASA for complex systems design. CASA combines the BDI agent approach with the FIPA ACL standard. It seamlessly integrates specification and invocation of fuzzy controllers in an agent-based systems. We

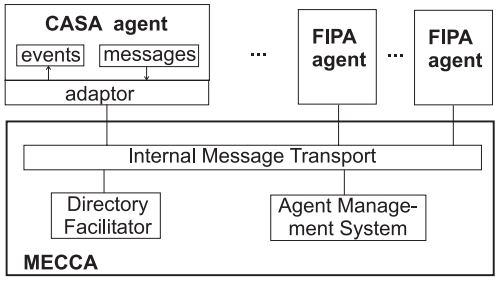

Figure 4. CASA Architecture

have introduced CASA by the example of a holonic manufacturing system which addresses future concepts in the field of complex CIM systems. Since holonic manufacturing systems are based on the principles of self-configurable networks with self-similar components this example provides an ideal application for our purpose. Additionally, it serves well for the application of fuzzy technologies.

This article gives our preliminary results and first experience in the application of CASA. This work presented herein provides a promising starting point for future investigation in this field. Future works will additionally focus on the completion of the CASA implementation as well as on other applications in the fields of manufacturing systems and 3D computer graphics.

Acknowledgements

The work described in this article is funded by the DFG-Schwerpunktprogramm 1064 “Integration von Techniken der Softwarespezifikation fuer Ingenieurwissenschaftliche Anwendungen”. We would like to thanks Christoph Schaeffer, FhG IPA Stuttgart, for his cooperation with the 3D simulation of the holonic transport system.

References

[1] B. Bauer, D. Steiner. MECCA - System Reference Manual. (*Internal Documentation*) Siemens, Munich, 1998.

[2] FIPA 97 Specification - Part 2: Agent Communication Language. *FIPA - Foundation for Intelligent Physical Agents*, Geneva, Switzerland, 1997.

[3] C. Geiger, G. Lehrenfeld. The Application of Concurrent Fuzzy Prolog in the Field of Flexible Manufacturing Systems. *Proc. of the PAP, Practical Applications of Prolog*, London, UK, 1994

[4] C. Geiger, G. Lehrenfeld, A. Weber. Concurrent Object Oriented Modeling of Fuzzy Strategies. *Proc. of the IEEE Conference on Fuzzy Systems*, Orlando, USA, 1996.

[5] C. Geiger. Rapid Prototyping of interactive 3D animations. *PhD Thesis*, Paderborn University, September 1998 (in german).

[6] B. Grabot. A decision support system for variable routings in manufacturing systems. *Fuzzy Set and Systems*, 58:87 - 104, 1993.

[7] J. Kiniry, D. Zimmerman.A Hands-On Look at Java Mobile Agents. *IEEE Internet Computing*, Vol. 1, No. 4, July 1997.

[8] J.M. Huber. JAM - A BDI-theoretic Mobile Agent Architecture. *Proceedings of the Third International Conference on Autonomous Agents*, Seattle, Washington, May 1-5, 1999.

[9] H. Nwana, D. Ndumu. A Perspective on Software Agents Research. (To appear). *Knowledge Engineering Review*, Cambridge University Press, Cambridge, 1999.

[10] A.S. Rao, M.P. Georgeff. Modeling Rational Agents within BDI-Architecture. *Tech. Report 64*, Australian Artificial Intelligence Institute, Melbourne, Australia, February 1996.

[11] A.S. Rao. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. *7th European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands, 1996.

[12] E. Shapiro. Concurrent Prolog - A Progess Report. *IEEE Computer*, 19 (5), August, 1986.

[13] E. Shapiro. The Family of Concurrent Logic Programming Languages. *ACM Surveys*, 21(3) 1989.

[14] E. Westkaemper, M. Hoepf, C. Schaeffer. Holonic Manufacturing Systems (HMS) - Test Case 5. *Proceedings of Holonic Manufacturing Systems* Lake Tahoe, CA, 9-16 February 1994.

[15] H. Zimmermann. *Fuzzy Set Theory - and its Applications*. Kluwer Academic Publishers, 2nd, revised edition, 1991