Proceedings of 3-rd Workshop on Enabling Technologies - Infrastructures of Collborative Enterprises 94, IEEE Computer Society Press, 1994

The Information Agent: An Infrastructure Agent Supporting Collaborative Enterprise Architectures

Mihai Barbuceanu and Mark S. Fox

Department of Industrial Engineering, University of Toronto, email:{mihai, msf}@ie.utoronto.ca

We introduce the Information Agent as a component of the information infrastructure supporting collaborative computing environments. We discuss the functions of the Information Agent, describe an architecture based on an Agent Program and a Knowledge Management System and present our choices for these components. We show how the architecture can be designed and implemented using description logic representation systems and argue for the advantages of this approach. Then we show how services supporting time-map management, conflict resolution, deductive queries and change management have been introduced and how all of these support the functions of the Information Agent.

Keywords: *conflict management, description logics, information agents, information sharing, temporal reasoning*

1.0 Introduction

Our research approaches the construction of distributed collaborative environments for enterprise integration [11, 12]] by relying on an information sharing infrastructure consisting of a class of agents called Information Agents. The purpose of this paper is to present our concept of information agent and to show how this concept is realised in an architecture that integrates several recent approaches to knowledge representation and knowledge sharing.

First, our information agent relies on a description logic language as the underlying representational and inferential substrate. Second, it extends this substrate with a new conflict management model and with a specific temporal reasoning capability. Third, it integrates in this environment recent languages including KQML [7] for the high level communication protocol and KIF [9] as the interlingua for knowledge communication.

2.0 The Information Agent

Information Agents (IAs) are agentified [15] knowledge and data management systems that allow other agents from the computerized organization to be selectively aware of relevant information by providing communication and information services related to:

- Persistent storage of information to be shared among the multiple functional agents.
- Deductive capabilities allowing new information to be inferred from existing information and domain knowledge.
- Automatic, content-based routing and distribution of information to the agents that need it.
- Automatic retrieval, processing and integration of information that is relevant to agents.
- Checking and maintaining various forms of consistency of the information. We address terminological consistency, assertional consistency and temporal consistency.
- Supporting temporal reasoning.
- Providing change management services.

2.1 Architecture of the Information Agent

The IA is currently composed of two components: an Agent Program and a Knowledge Management System (fig. 1).

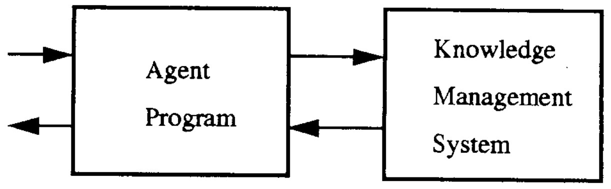


FIGURE 1. Architecture of the Information Agent.

The Agent Program is responsible for the social interaction function if the IA. It turns the IA into an autonomous agent supporting a protocol for interaction with the outer world. The IA currently supports the spech-act based Knowledge Query and Manipulation Language (KQML) [7] as the interaction language and the Knowledge Interchange Format (KIF) [9] as the content language in terms of which knowledge is communicated.

The Knowledge Managenment System has the goal of providing the common representational and reasoning substrate on top of which the other components and services of the IA are built. We have adopted a description logic language [3] and extended it with time map management and conflict management for this purpose.

3.0 The Knowledge Management System

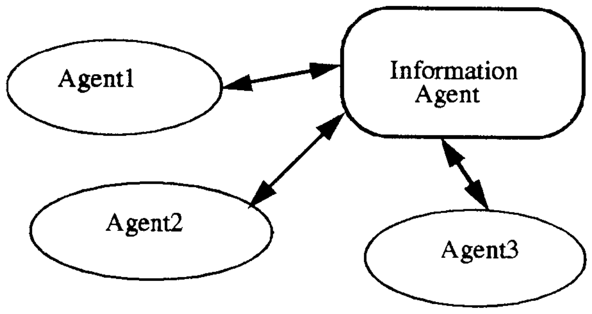
Description logic languages integrate aspects of object-oriented and logic representations. They express knowledge in a modular fashion, using inheritance and hierarchical organizations and rely on well-defined declarative semantics to describe the meaning of constructs. Well-known examples of such languages include KLONE [5], LOOM [10] or CLASSIC [4]. For the work reported herein, we use a description logic language that provides the usual *concept*-forming operators - *conjunction*, *value restrictions*, *number restrictions* - *roles* and *subroles*, *disjointness* declarations, *primitive* and *defined concept* specifications. The language T-Box provides the usual services of constructing the *complete form* of concepts and *automated classification* based on *subsumption* checking. The language A-Box is essentially a *constraint propagation* engine that makes instances conform to the various constraints asserted about them. It uses a propositional representation of instances and roles, a boolean constraint propagation TMS [2] and a number of non-standard services including rules, conjunctive patterns and demons.

4.0 Content-based information distribution: a scenario

Consider the situation in fig. 2-a where an IA services three other agents in an organization and assume that the topics of interest of two agents are as given in figure 2-b. Suppose first that Agent-1 informs the IA about a certain part p-111, by sending it the following information:

```
(part p-111)
(part-of p-111 c-12)
(part-of p-111 c-13)
(part c-12)
(part c-13)
(weight c-12 2700)
(weight c-13 3400).
```

Given its prior knowledge about parts and components, the IA will infer that (component p-111) and (weight p-111 6100), hence (heavy-component p-111), thus informing Agent-2 about some information relevant to its interests. The processes required for this inference use terminological knowledge, e.g. any part is a component and "aggregation" knowledge for weights, cost or other similar properties.



a - Information Agent servicing functional agents

Topic of interest of Agent-2:

```
(concept heavy-component
  (:and component (:gt weight 5000))),
```

that is "any component whose weight is greater than 5000"

Topic of interest of Agent-3:

```
(concept weight-change
  (:and change (:the changed Component)
        (:gt difference 100))),
```

that is "any change in weight such that the difference between the new and the old value is at least 100"

b - Topics of interest

FIGURE 2. Content-based information routing.

Later on, Agent-1 may modify the weight of c-12:

```
(retract (weight c-12 2700))
(assert (weight c-12 1000)).
```

As a result, the weight of p-111 becomes 4400, p-111 ceases to be a heavy-component of interest to Agent-2 and the IA informs Agent-2 about that. Also, using an event that builds an weight-change instance whenever a weight is changed, the IA realizes that it has two weight-change instances, both with a difference of 1000 and sends both to Agent-3 since both match its interest.

5.0 The Time Map Management System

Explicit support for reasoning with time allows agents to maintain models that reflect the temporal dimension of their activity. The IA supports temporal reasoning at the assertional level by extending the A-Box to a full fledged Time Map Management System [6] that: (i) records the time intervals [1] during which propositions are true or false, (ii) provides a query language that supports temporal interrogations, (iii) provides for the definition and application of rules that extend the information in the time mapped data base in both temporal directions and (iiii) provides a mechanism for recording dependencies amongst time-mapped propositions and supporting retraction of these propositions, by means of a temporally extended truth maintenance system.

As a result, the proposition database maintained by the assertional component is indexd by time intervals and the truth maintenance process infers time intervals for each proposition. For example, the assertional component represents the fact that John is a manager during 1993 as (manager John (at 1993)) and that Bill operates machine M1 during december 1993 as (operates M1 Bill (at december 1993)). The justifications propagated by the truth-maintenance system also propagate time intervals. Such a justification is, for example:

```
(failure-free M1 (at (+ ?int 1))
 <- (operates M1 Bill (at ?int)).
```

Assuming that we measure the time in months, when the above rule is matched with the proposition (operates M1 Bill (at december 1993)) it will justify the new proposition (failure-free M1 (at january 1993)).

As an example of a temporal query, to find out who operated all failure-free machines during december 1993 , the following query can be formulated:

```
and:((failure-free ?machine (at december 1993))
 (operates ?machine ?who)).
```

A query like (injection-molding-machine :anytime 28 february 1994) will be answered by all propositions about injection-molding-machine-s that are true during any interval included in the interval denoted by 28 february 1994. If the keyword is :alltime, then the response is composed of all propositions that are true during the entire interval 28 february 1994. Finally, if the keyword is :overlapping, the response will contain all propositions that intersect the interval 28 february 1994.

6.0 Query management

Description logic languages represent queries as concepts, the *extensional* response to such a query being the set of instances belonging to the concept, and the *intensional* one a relevant set of concepts subsumed by the query concept. Both responses are constructed through deduction, as they rely on the deductive subsumption operation.

When one is interested in collecting propositions related to several concepts among which arbitrary relations exist, our language supports a new pattern construct that is helpful. For example, suppose an agent is interested in all pairs of machines that are scheduled for maintenance activities during adjacent time intervals. The conjunctive pattern mechanism allows such queries to be formulated and answered, e.g. :

```
:and
((machine ?x)(machine ?y)(maintenance-time ?x ?t1)(maintenance-time ?y ?t2))
:such-that (meets ?t1 ?t2).
```

7.0 Conflict management with the authority/ deniability model

It is absurd to believe that in real organizations contradictions will never occur. For this reason, the IA supports conflict management at two levels. At the terminological level, the T-Box mechanisms ensure that all concepts can be satisfied, hence concepts like e.g. (and v6engine l4engine) will not occur. At the assertional level, contradictions can be caused by having diferent (locally consistent) agents assert contradictory information such as (v6engine e1(at march 94)) and (l4engine e1 (at march 94)).

The conflict management service helps remove such contradictions by determining which beliefs to retract in order to remove a contradiction. The IA maintains two kinds of propositions. *Premises* are propositions sent to the IA by other agents that consider them true. The IA has no access to whatever justification the sending agent may have for the proposition. *Derived* propositions (or simply propositions) are propositions inferred by the IA based on the available premises and on the IA's knowledge of the domain. Agents that supplied propositions to the IA are named *producers* of the information. Agents that have been supplied information by the IA are named *consumers* of the information.

Assume that the marketing agent may have determined that for a new automotive product a v6 engine would sell better. Hence marketing will sent the IA a message telling that the engine should be a v6: (v6engine e1(starting 13 march 94)). From different requirements, design may later determine that only a l4 engine can be used: (l4engine e1 (starting 14 march 94)). Using domain knowledge that the v6engine and l4engine concepts are disjoint, the IA will derive a contradiction (for the common time interval during which both beliefs are held):

```
(and
(v6engine e1(starting 13 march 94))
(l4engine e1(starting 14 march 94)) => false.
```

The conflict management service of the IA removes this contradiction by considering two properities of information, *authority* and *deniability*. Authority is defined as a kind of *priority* agents may have wrt the truth of the information they deliver. If marketing has established a clear trend in the market favoring v6 engines, then it may deliver this information to the IA specifying an increased authority, e.g.: ((v6engine e1(starting 13 march 94)) :authority 9). Our model assumes that agents will honestly assess their authority.

Now, if marketing was first to determine that the engine must be a v6, the IA sent this information to the purchasing agent (whose interest was matched by it). The purchasing agent used the information to order v6 engines from another company. Later, design discovered that the engine must be a l4. If the design view is accepted, purchasing will have troubles in cancelling the order (paying penalties, etc.). This shows that information that has been consumed may be *costly* to retract later. We define the *deniability* of consumed information as a measure of the *cost* to retract it. We often use *undeniability* as the inverse of deniability. Undeniability (or deniability) is determined by the consumers of information. The same assumption about honest assessment of undeniability is made.

7.1 The a-u space

Suppose we have a p&q=>false contradiction and we have determined a set {pi} of premise such that each pi is a premise that directly or indirectly implies either p or q. To each pi we can attach an authority measure - the authority of its producer - and an undeniability measure - derived from the sum of deniability costs of all propositions that would have to be retracted if pi is retracted. A high authority means that the proposition is more difficult to retract since a high authority has to be contradicted. A high undeniability means that the proposition is more difficult to retract because the costs of retraction incurred upon consumer agents will be great.

We can represent these two values of all {pi} premises as points in a diagram having authority on the x-axis and undeniability on the y-axis. Such a diagram is called an a-u space and is illustrated in fig. 3.

Propositions from the a-u space that have both low authority and low undeniability - as defined by the threshold values at and ut - are easy to retract because no significant authority is violated and no significant costs are incurred. Propositions that have high authority and high undeniability are hard to retract exactly for the opposite reasons. An aggregated measure of both authority and undeniability is the distance r to the origin. If a proposition with high autority or undeniability is considered for retraction (e.g. because low authority and undeniability propositions do not exist), the IA must negotiate retraction with the producer and/or consumers.

Figure 3 shows the regions defined by these thresholds in the a-u space. Examples of work exploring negotiation as a means to mediate among conflicting agents are in [16] and [18].

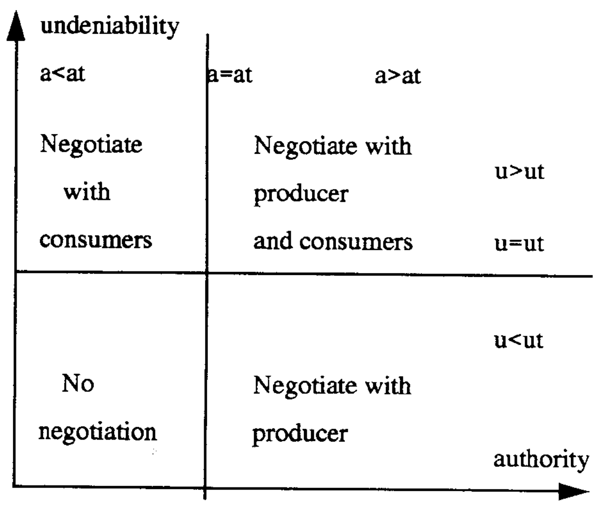


FIGURE 3. Negotiation regions in the a-u space.

This model has three main advantages. First, it is accurate because the selection of the retracted belief is based on the the views of all involved parties at the moment the contradiction is detected. This means that the selection implicitly relies on domain knowledge and on the current state of the global problem-solving effort. Second, by estimating costs and identifying negotiation regions, the model takes advantage from situations in which negotiation may not be required or in which a smaller amount of negotiation may suffice. Third, the model handles the new time-dependent inconsistencies that arise in time-mapped data-bases.

8.0 Change management

When the shared model changes, agents that used assumptions no longer valid must be notified. A typical example is selecting a machine for scheduling an operation. The scheduling agent may question the IA about all machines that are expected to be availabe during a specified period of time (e.g. for which no maintenance activities will be carried out). The scheduling agent will receive such a list and then will select one machine for the desired operation. Later on, the maintenance agent may change the schedule for maintenance activities for the selected machine. The scheduling agent needs to be informed about that in order to take appropriate action.

This kind of change management can be supported with the mechanisms discussed up to now. The critical step is to have the Scheduler notify to the IA the assumptions it uses by posting a rule that will be fired by the IA in due time:

```
(rule r0987
:if (maintenance-schedule Mj ((march 1993) (september 1993)))
:then (use OP-12 Mj)(time OP-12 (april 1993))(time OP-12
(august 1993)))
```

When the maintenance agent later changes the maintenance schedule by(retract (maintenance-schedule Mj (march 1993))), the propositions inferred by the rule become false and hence they are retracted by the IA who sends denial messages to all agents that used these assumptions:

```
(deny (use OP-12 Mj)(time OP-12 (april 1993))(time OP-12
(august 1993))).
```

9.0 Concluding remarks

Information Agents are a component of the information infrastructure supporting collaborative computing environments. This paper has identified the Information Agent notion and the functions it can play in a cooperative information system environment. We have proposed an architecture for the IA, composed of an Agent Program and a Knowledge Management System and we have discussed our choices for these components. We have shown that the choice of description logics as the representational substrate provides leverage for building Information Agents that can automatically determine the relevance of information, automatically distribute it, maintain terminological, assertional and temporal consistency, automatically organize the shared vocabulary cooperating agents use, apply deduction to respond queries, provide descriptive/intensional answers, reason with partial information and manage model change. To achieve these objectives we had to substantially extend this substrate with a new conflict management model and a new - in the context of description logics - time map management system. To achieve further objectives related to knowledge communication in distributed environments we integrated KQML and KIF.

While many of the above methods and approaches are known and have been applied elsewhere, we believe that their integration in an Information Agent is new. As all component languages are generic, the resulting IA is also a generic agent shell that can be used across many application domains just by loading specific ontologies, data bases and application models. Although the IA as described here has been implemented, it suffers from the lack of experimental results that might demonstrate that it is effective and can scale up in applications. We hope to improve on that in the near future when the Information Agent will be experimented within the TOVE project [8].

Acknowledgements

This research is supported, in part, by the Manufacturing Research Corporation of Ontario, Natural Science and Engineering Research Council, Digital Equipment Corp., Micro Electronics and Computer Research Corp., Spar Aerospace, Carnegie Group and Quintus Corp.

10.0 References

1. Allen, J., F. Maintaining Knowledge About Temporal Intervals, Communications of the ACM, 26 (11), pp 832-843, 1983.

2. McAllester, D. Truth Maintenance, Proceedings AAAI-90, pp. 1109-1116.

3. Barbuceanu, M. Models: Toward Integrated Knowledge Modeling Environments, Knowledge Acquisition 5, pp. 245-304, 1993.

4. Borgida, A., Brachman, R. J., McGuiness, D. L., Resnick. L. A. CLASSIC: A Structural Data Model for Objects, Proc. 1989 ACM SIGMOD International Conference on Management of Data, june 1989, pp. 59-67.

5. Brachman, R. J., Schmolze, J. G., An Overview of the KL-ONE Knowledge Representation System, Cognitive Science 9(2), April-June 1985, pp. 171-216.

6. Dean, T.L. and McDermott, D.V. Temporal data base management, Artificial Intelligence 32 (1987), pp-1-55.

7. Finin, T., et al. Specification of the KQML Agent Communication Language, The DARPA Knowledge Sharing Initiative, External Interfaces Working Group.

8. Fox, M. S. A Common-Sense Model of the Enterprise, Proc. of Industrial Engineering Research Conference, 1993.

9. Genesereth, M.R., FiKes, R.E. Knowledge Interchange Format, Version 3.0, Reference Manual, Computer Science Department, Stanford University, Technical Report Logic-92-1.

10. McGregor, R., Bates, R. The LOOM Knowledge Representation Language, ISI IRS-87-188, USC/ISI Marina del Rey, CA

11. Pan, J.Y.C., Tenenbaum, J. M. An Intelligent Agent Framework for Enterprise Integration, IEEE Transactions on Systems, Man and Cybernetics, 21, 6, pp. 1391-1408, 1991.

12. Petrie, C. Revised dependency-directed backtracking for default reasoning, Proceedings of AAAI-87, pp 167-172

13. Roboam, M., Fox, M. S. Enterprise Management Network Architecture, A Tool for Manufacturing Enterprise Integration, Artificial Intelligence Applications in Manufacturing, AAAI Press/ MIT Press, 1992.

14. Schmiedel, A. A Temporal Terminological Logic, Proceedings of AAAI-90, pp 640-645.

15. Shoham, Y. Agent-Oriented Programming, Artificial Intelligence 60 (1993) pp 51-92.

16. Sycara, K., Multi-agent compromise via negotiation, In Les Gasser and Michael N. Huhns, editors, Distributed Artificiall Intelligence, Volume II, pp. 119-137, Pitman Publishing, London, 1989

17. Tenenbaum, J. M., Gruber, T. R., Weber, J.C. Lessons from SHADE and PACT, Enterprise Modeling and Integration, C. Petrie (ed), McGraw-Hill 1992.

18. Zlotkin, G., Rosenschein, J.S. Negotiation and task sharing among autonomous agents in cooperative domains, Proceedings of IJACI-89, pp. 912-917, Detroit, MI, aug. 1989