HOLONS AND AGENTS IN ROBOTIC TEAMS. A SYNERGISTIC APPROACH.

Boldur BARBAT[1], Ciprian CANDEA[2] and Constantin ZAMFIRESCU[1]
[1]"Lucian Blaga" University of Sibiu, Dept. of Computer Science
4 Emil Cioran St., 2400 Sibiu, Romania
E-mail: bbarbat@sibnet.ro Phone: +40-69-217928 Fax: +40-69-212716
[2]Artificial Intelligence Research Group Sibiu
Reconstructiei Nr. 2A, 2400, Sibiu, Romania
E-mail: ciprianc@airg.verena.ro Phone: +40-69-224168

Abstract

Although established as wide-ranging trends in IT, some paradigms are not yet as pervasive, diversified, and – most of all – integrated (in both senses: *incorporated* as well as *intertwined*) in real-time control systems, as they should be. The paper considers three of them: a) *agent-orientation* – particularly, multi-agent systems (MAS) [10]; b) *holonic approach* to manufacturing systems [6]; c) *coarse-grain parallelism* [7] [11]. The state of the art in these fields is elaborated upon passing it through a threefold filter regarding real-time applications: *high complexity*, *vital robustness*, and *critical response time*.

1. Introduction

On the other hand, robotic soccer (RoboCup) proved itself as an outstanding test bed for innovative approaches. In implementing teams for RoboCup, MAS are widely used [8] [9]; moreover, a holonic-like approach has been recently proposed too [3]. Thus, while preparing a new team for the RoboCup competition, simulated soccer (RoboCup environment) is an adequate test bench for joining the two paradigms. As well, it let us use as "benchmarks" games played against our own team, exercised in 1999 in Stockholm [2].

2. Rational

Why agents? The paper supports the assertions that: a) MAS are crucial for robotic teams because, besides the individual goal of each agent, global objectives are established committing all or some agent groups to their completion; b) generic agent architectures like that proposed in [1] allow easy instantiation for similar – but distinct – entities, such as the players are (or should be); c) MAS are the natural means to design and implement holonic software systems [4] as well as a lot of related kinds of applications [14] offering the conceptual tools to tackle their complexity (decomposition, abstraction and organization) [N2]; d) MAS are also used in heterarchical control, and provide the software with opportunities for taking the initiative to take autonomous decisions.

Why holons? Starting from the fact that Holonic Manufacturing Systems (HMS) were set up as a new approach to the manufacturing control problem [13], the paper defends, among others, that:

a) Because of one of the main weaknesses of MAS – the practical impossibility to deal with more than two levels (the *agent* and the *system*) – the holonic paradigm allows better modelling of multilevel systems (including the player-team-coach ensemble).

b) It increases also the flexibility of decisional systems (as both HMS and robotic teams); indeed, a "holarchy" is "a hierarchy of self-regulating control building blocks (holons), which function (i) as autonomous wholes in supra-ordination to their parts, (ii) as dependent parts in subordination to controls on higher levels, (iii) in co-ordination with their local environment." [7].

c) One of the most important characteristics of holarchies is the capacity to modify themselves, i.e. to create temporary hierarchies [5] (like modern industry, soccer is very dynamic: not only that each team comes with its own style and game strategy, but also each game phase has a dose of novelty).

d) Holarchies offer a balance between the two usual approaches to the guided process: the *hierarchical* control (fixed, static, pre-established) and the *heterarchical* one (autonomous, decentralized, flexible).

e) Holons specialization and aggregation offer a great flexibility in testing the system at different levels of details during the fine-tuned phases. Aggregated holons are defined as a set of related holons that are clustered together and form in their turn a bigger holon with its own identity, so holons may belong to multiple aggregations at the same time. Aggregated holons can

dynamically change their contents depending on particular needs of the system (they may even emerge out of the self-organising interaction of holons).

As regards the third paradigm, after proposing to assign semantic value to some concurrent programming concepts (e.g., priorities), in order to enhance both flexibility and speed, the paper explains why, at this research stage, such mechanisms are only barely applied.

3. Architectural issues

Approach. Our idea is inspired by the PROSA architecture developed at PMA/KLeuven as a reference model for Holonic Manufacturing Systems [13]. The acronym PROSA came from **P**roduct-**R**esource-**O**rder-**S**taff **A**rchitecture, the holon types used. The resource holon contains a physical part namely a production resource of the manufacturing system, and an information processing part that controls the resource. The product holon holds the process and production knowledge to assure the correct making of the product with sufficient quality. The order holon represents a task in the manufacturing system. It is responsible for performing the assigned work correctly and on time. The staff holon is implemented in the idea to assist the other three holons in performing their work.

Based on this architecture we propose a similar approach for robotic teams (see Table 1, intended rather as a suggestion, than as a full chart). The holarchy is structured on five levels (RoboCup, Coach, Team, Player, Component), each of them containing the specific holons (Product, Resource, Order, Staff). Their role, functionality and cooperation mechanisms are described and discussed in the RoboCup context. A more detailed description is given in [3].

Table 1. Holon-like approach for robotic teams

	Holarchy Levels				
	RoboCup	**Coach**	**Team**	**Player**	**Component**
Product	▪ Championship ▪ Workshops, etc.	▪ Selection ▪ Training ▪ Strategies ▪ Tactics	▪ Game	▪ Skills acquired ▪ Tactics learned ▪ Implemented strategies ▪ Implemented tactics	▪ Applied skills
Resource	▪ Organisational ▪ Technical ▪ Financial	▪ Strategies ▪ Tactics ▪ Experience ▪ Rules	▪ Time ▪ Information ▪ Schemata	▪ Skill ▪ Schemata ▪ Stamina ▪ Components	▪ Head ▪ Right leg ▪ Left leg
Order	▪ Research ▪ Entertainment	▪ Building teams ▪ Testing teams ▪ Winning championships	▪ Win game	▪ Fulfil role ▪ Preservation ▪ Learn	▪ Execute
Staff	▪ All individuals involved	▪ Players ▪ Teams	▪ Spare players	To be adapted	Not applicable

Since MAS are the natural means to design and implement holonic software systems, a generic holon architecture is conceptualised and implemented as a three-layer agent architecture (i.e. holarchy, deliberative and reactive layer, respectively). The *holarchy layer* includes mechanisms for devising joint plans with other

holons/agents. To achieve the needed flexibility it was divided in three sub-layers: (i) integration (for vertical collaboration with adjacent layers, e.g., a holon/agent situated at the team layer shall collaborate with other holons/agents situated at the coach and/or player levels); (ii) cooperation (for horizontal integration of entities situated at the same level); and (iii) monitoring (for modifying the holarchy). The *deliberative layer* includes mechanisms able to deal with local plans and local goals. Finally, the *reactive layer* includes facts representing the world model as well as primitive if-then rules.

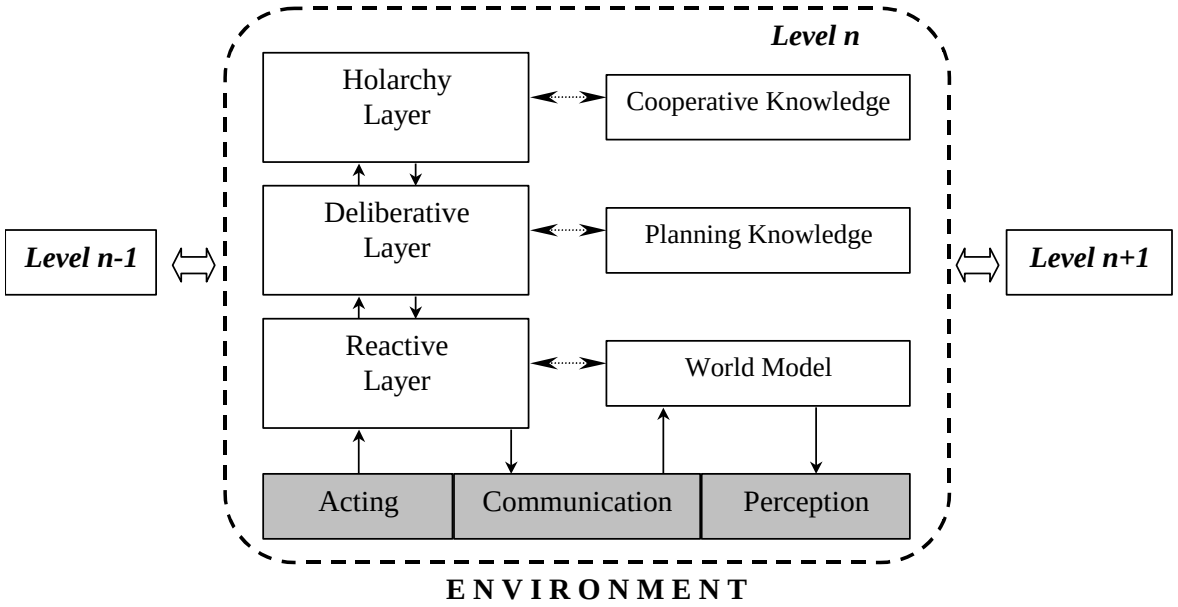

Figure 1.Generic holon architecture (adapted from [4])

In spite of their usefulness in the earlier development stages (e.g., conceptualisation, design) all concepts of functional entities, except execution threads, disappear at run-time – at least partially – insofar their rationale and functionality have been implanted into threads. On the contrary, the thread preserves full autonomy and operational ability, being the central dynamic entity accepted by the system. In order not to distort the semantics of cooperation/rivalry between agents/holons, their interaction has to remain operational at run-time. In other words, no matter how refined and sophisticated negotiation methods could be, to make any sense at run-time, they must reflect themselves in thread abilities. The main idea is to raise the importance of priorities from the minor significance of technological mechanism to the major conceptual role of expressing holarchies – and, perhaps as a "potentiometer" for building refined temporary holarchies. In order to carry out this macro-architectural role, some specific issues are proposed and elaborated upon: a) assigning high-level meaning to priorities (e.g. position of holons in holarchies); b) applying dynamic priorities to express more accurately generic architectures (ranging from quite simple agents to temporary holarchies); and c) refining the granularity of all architectural levels. So, without intending to propose a new model, the paper illustrates how minor theoretical changes of existing MAS models [12] are necessary in order to formalize the semantics of priorities. Specifically, the only adjustment proposed consists of replacing the Boolean characterization of agent state variables with a multi-valued one. All these aspects are outlined in the context of our Robocup team (entirely implemented in Java).

4. Implementing holons and agents

How we already see (Table 1) our proposed architecture is based on holons for RoboCup simulator league. Last year our team took part at the RoboCup 1999 competition event in Stockholm, where our team obtain good results at the first participation. Based on this experience we redesign our existing team to work on new approach. Our problem was to find the best solution to adapt our existing implementation in Java for this new approach. For this competition year RoboCup 2000 we chose to implement only few holons because the time necessary to be implemented and tested in

RoboCup domain is big. First of all we start to design in Java the generic holon (Figure 1). For that we chose to use a multi-threading approach because practically, how we early said, a holon we can identify with an agent and in one holon we have few agents. How results from Fig. 1 we have to implement for holon the capability of reaction but, in the same time, the deliberative process, that is a more complex task and also a holarchy level. For that we design a multi-threading architecture like in Figure 2.

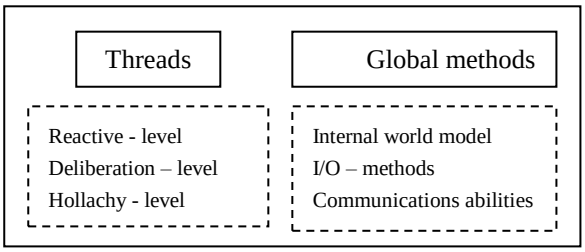

Figure 2.Generic holon implementation

We tried to respect all the desiderate from Figure 1 that means that our holon - agent must be capable to communicate with other holons but in the same time with other players. The solution find for this problem is very simple because the communication task with other holons is like to communicate with other agents. Here the algorithms used could be very complex, on other hand to communicate with other players (entities) is like to communicate with other holons the single difference is the method used. The rest of the functionalities are encapsulated in each thread, for example the deliberation level is in deliberation thread with specific implementation (algorithms).

At the player level (Table 1) are the holons who handle the resources (i.e. player stamina), implement the skills (i.e. pass, dribble, score), model the game tactic and coordinate the body parts in future. The resource holon handles the player stamina, simulation cycle and implement skills. In current implementation our skills used analytical methods.

At the product level we have more complex problem like tactics where we used a fixed decision tree obtained with an off-line algorithm, but also we have to implement the necessary knowledge to resolve the advice contains strategies request comes from other holons.

At the Order level we have more abstract and complex task like learn. At this level we implemented a neural network [16] that was used to obtain the strategic positioning when players don't have the ball control. This type of neural network we used also in 1999 competition with some promising results.

With this architecture we can control better our players because when a new situation appear immediately one of the holons will react at the environment changes. Is also possible that more than one holon to react at the changes; for example one react using reactive modules and the other one have some other plans generated by deliberative level and the negotiation process begin. In this moment we test only a very simple negotiation model based on fixed priorities.

To obtain a collaborative team was necessary to implement the team level where the biggest problem was at the resource level because here the information are shared with the rest of the team mates and the schemata selection is based on the previous off-line learned knowledge.

5. Conclusion and future work

The noticeable conclusions are: a) Despite being a conventional approach, not even the potential of con-current programming facilities offered by existing operating environments is fully used. b) The other two paradigms are even sparser represented in actual systems (perhaps, because of the dissimilar domains they come from – or of their relative novelty); especially the holonic approach is almost absent. c) Moreover, even if they are taken into consideration, they are taken isolated in every concern. d) It seems to be rather a gap between the architecture at the problem-solving level and the structure at the implementation one (thus, some relevant concepts have up to now, more or less, purely "technological" role)

Interaction between a large number of low level agents results in complex system behaviour that is difficult to understand, to control and to predict. Structuring the agents in a hierarchy is the appropriate solution to tackle this complexity. The necessity to deal with different levels of abstraction has been pointed out by several authors [10,N3,4]. For example, Scerri, Tambe and Pynadath [N3] layered the team agents on the basis of their adjustable autonomy, matching so the levels of autonomy existing in human organizations. Thus, the system that supports an adjustable autonomy is able to dynamically change the autonomy it has to make and carry out decisions. In this case, the agent's role in the team composition is mapped with a degree of autonomy necessary to achieve its responsibility.

The conclusions, yet partial and amendable, highlight the points where the approach seems positive; some of them are:

- A multilevel environment as Robocup is better assisted by a joint holon/agent model than by a monoparadigmatic one.

- Because of the significant differences between the paradigms, and the successful employment of MAS, it is better to start out from agents and add stepwise holonic functionality, than vice versa.

- The three-layer agent architecture allowed a sound and practical shift in this direction.

- In this "off-line context", the approach worked (the new team won all games played against the 1999 team); of course, a credible validation can be achieved only in a championship.

The still open aspects are underlined too (e.g., for the game strategy the order holon is insufficiently adapted; negotiation between holons is still to primitive; no matter the applied paradigm, skill remains of paramount importance).

From a quite general perspective, future objectives are: extending the holarchy with two new levels (team and player-components, respectively); moving further from agent to holon, by improving the architecture of individual entities, as well as their relationship (mainly interholonic negotiation/communication [15]; extending the approach to other domains. Furthermore, we plan to test our approach in more complex simulation environments [N1](i.e. RoboCup Rescue - a project focused on multiple agents collaborating to rescue civilians from a disaster area) in which high complexity, vital robustness, and response time become critical factors.

REFERENCES

1. Bărbat, B. and C.B. Zamfirescu (1999). A^{3}CKM: Anthropocentric Agent Architectures for complex knowledge management, *Acta Universitas Cibiniensis*, Vol. **XXXVIII**, Sibiu, pp. 23-28.
2. Candea, C., Oancea, M. and Volovici, D. (1999). Emulating real soccer, in *Proceedings of the International Conference Beyond 2000*, Sibiu, pp. 35 - 38
3. Candea, C., M. Staicu, and B. Bărbat (2000). Holon - Like Approach for Robotic Soccer. In: *Proceedings of the RoboCup European Workshop 2000*, Amsterdam
4. Fischer, K. (1998). An Agent-Based Approach to Holonic Manufacturing Systems. In: *Intelligent Systems for Manufacturing: Multi-Agent Systems and Virtual Organizations*. Proceedings of the BASYS'98 3rd IEEE/IFIP Int.Conf. Prague (L. Camarinha-Matos, H. Afsarmanesh, V. Marik, Eds.), pp. 3-12, Kluwer Academic Publishers, Boston.
5. Giebels, M., Kals, H. and Zijm, H. (1999). Building Holarchies for Concurrent Manufacturing Planning and Control. Proceedings of the second International Workshop on Intelligent Manufacturing Systems, Leuven, Belgium, pp.49-56
6. Hermans, K., Y. Berbers, B. Robben and H. Van Brussel (1999). A software framework for shop floor control systems using active objects. In: H. Van Brussel and P. Valckenaers (Eds.). *Proceedings of the Second International Workshop on Intelligent Manufacturing Systems*, Katholieke Universiteit Leuven, pp. 137-145.
7. Hino, R. and Moriwaki, T.(1999) Decentralized Scheduling in Holonic Manufacturing System. Proceedings of the second International Workshop on Intelligent Manufacturing Systems, Leuven, Belgium, pp.41-47
8. Iozon, G. and Candea, C. (1999). RoboCup '99. Level and Trend, in Proceedings of the International Conference Beyond 2000 Sibiu,pp. 61 - 64
9. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I. and Osaw, E. Robocup: The Robot World Cup Initiative {DOAR UN "AND" LA URMA}
10. Nwana, H.S. and M. Wooldridge (1997). Software Agent Technologies. In H. S. Nwana, N. Azarmi (Eds.). *Software Agents and Soft Computing. Towards Enhancing Machine Intelligence*, Springer-Verlag, Berlin, pp. 59-78.
11. Pallmann, D. (1999). *Programming Bots, Spiders, and Intelligent Agents in Microsoft Visual C++*. Microsoft Press, Redmond, WA.
12. Patriti V.F., K. Schäfer, P. Charpentier and P. Martin (1998). Analytical design methodology of agent oriented manufacturing systems. In L.M. Camarinha-Matos, Holon Afsarmanesh and V. Marik (Eds.). *Intelligent Systems for Manufacturing. Multi-Agent Systems and Virtual Organisations*, Kluwer Academic Press, Boston, pp. 99-108.
13. Van Brussel, H. (1994) Holonic Manufacturing Systems, the vision matching the problem. Proceedings of the First European Conference on Holonic Manufacturing Systems, Hannover, Germany, IFW-Hannover(ed),
14. Wyms, J. and Van Brussel, H. and Bogaerts, L. (1999) Design Pattern for Integrating centralised scheduling in distributed holonic manufacturing control systems. Proceedings of the second International Workshop on Intelligent Manufacturing Systems, Leuven, Belgium, pp.75-82
15. Zamfirescu, C.B. and F.G. Filip (1999). An agent-oriented approach to team-based manufacturing systems. In: H. Van Brussel and P. Valckenaers (Eds.).

Proceedings of the Second International Workshop on Intelligent Manufacturing Systems, Katholieke Universiteit Leuven, pp. 651-658.

[N1] Ranjit Nair, Takayuki Ito, Milind Tambe, and Stacy Marsella. "Robocup Rescue: A Proposal and Preliminary Experiences, *Workshop on Robocup Rescue at International Conference on Multiagent Systems* (ICMAS2000), 2000.

[N2] N. R. Jennings (1999) "Agent-Oriented Software Engineering" Proc. 12th Int Conf on Industrial and Engineering Applications of AI, Cairo, Egypt, 4-10.

[N3] Scerri, P., Tambe, M., Lee, H., Pynadath, D., et al 2000, Don't cancel my Barcelona trip: Adjusting the autonomy of agent proxies in human organizations. Proceedings of the AAAI Fall Symposium on Socially Intelligent Agent