Actor-Agent Communities: Design Approaches

S. M. Iacob, C. H. M. Nieuwenhuis, N. J. E. Wijngaards, G. Pavlin, J. B. van Veelen

Abstract We describe several principles for designing Actor-Agent Communities (AAC) as collectives of autonomous problem solving entities (software agents and human experts) that self-organize and collaborate at solving complex problems. One of the main distinctive aspects of the AAC is their ability to integrate in a meaningful way the expertise and reasoning of humans with different information processing algorithms performed by software agents, without requiring a unique and complete description of the problem and solution spaces.

1 Preliminaries

Many architectures for multi-agent systems have been proposed, each excelling in one or more functional or non-functional parameters, such as distributed processing, negotiation, response time, self-organization, etc. (see e.g. [2]). Nevertheless, if one abstracts from specific functionality and implementation platform, agents can be regarded as individual software programs which get activated by certain triggers in their environment (be it user commands, or some sensor values). A multi-agent system (MAS) can thus be regarded as a software program with different asynchronous threads (the agents), which can be coordinated either globally or locally. A distributed MAS is in principle only different form a MAS in that the individual agents may run on different physical platforms. At this abstraction level the different agent systems can be described by a single reference architecture, as proposed by FIPA [8]. From an application perspective, all agent-based systems have a request-response interaction model, where agents respond to requests from users (actors) or other agents, and possibly react to the environmental context (see Fig. 1). In general

S. M. Iacob, C. H. M. Nieuwenhuis, N. J. E. Wijngaards, G. Pavlin, J. B. van Veelen
Thales Research and Technology Netherlands, D-CIS Lab, Postbus 90, 2600 AB Delft, The Netherlands, e-mail: {sorin.iacob@icis, kees.nieuwenhuis, niek.wijngaards@icis, gregor.pavlin, bernard.vanveelen}@icis.decis.nl

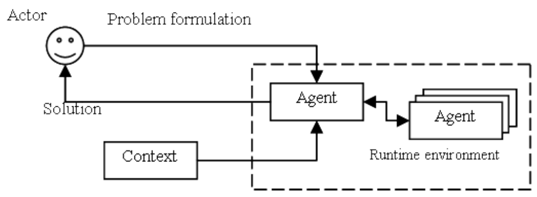

Fig. 1 Generic interaction model in MAS

this interaction can be replicated for a multitude of actors and agents. The autonomy of agents refers to their ability of performing some of the problem-solving steps without requiring explicit input from a human operator. We address the use of MAS for solving complex problems, where complexity refers to the solution process, and not to the computational complexity of the algorithms. Complex problems cannot be decomposed into a finite number of simpler problems that each be solved by an algorithm, either because of lacking adequate models, or because it contains uncomputable elements. The common approach to solving such problems relies on approximations through heuristics (or more generally, on computing in the limit [3]). In data-driven approaches these heuristics can in principle be found through training and learning algorithms, provided that enough training data is available [5]. However, the utility of these solutions is mostly limited to classification problems.

In this work we propose a new approach to autonomous cooperative problem-solving systems, where agents and humans form a problem-solving community. Complex problems are detected, formulated and solved jointly by humans and autonomous agents.

2 Actor-Agent Communities

Actor-Agent Communities (AAC) are socio-technical information systems that deliver a solution for otherwise intractable information processing problems. At the highest abstraction level an AAC can be regarded as a collection of autonomous problem-solving entities, with specific problem-solving capabilities resulting from their individual knowledge (or world models), interaction modalities, and (limited) capabilities and resources for information processing. A coherent behaviour can be induced on an amorphous collection of such autonomous entities by assigning them a common goal. This may be regarded as a set of quantities that describe a particular state of the environment (possibly including states of the entities themselves). Depending on the richness and complexity of their respective world models, each entity will maintain a particular (partial) representation of these goals. Problems can now be defined as mismatches between a target state of the environment and an observation thereof. The entity that discovers a problem may not be able to solve it all by itself, in which case other entities will be asked to contribute. Teams can thus be formed whose purpose is solving that particular problem. In order for two or more entities to be able to team up, their world model need to partially overlap (achieve

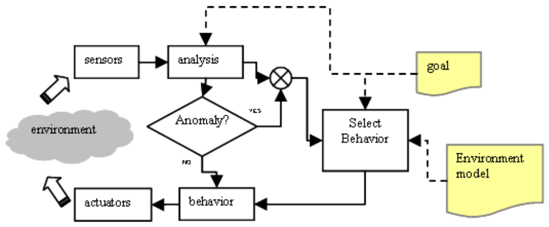

Fig. 2 A simplified functional model for autonomous problem-solving entities

a ”common ground”). The initiation and completion of such self-organization is only determined by the problem detected in the environment, the overlap in world models, and capabilities of the entities.

3 A Functional Model for Autonomous Problem-Solving Entities

An autonomous problem-solving entity is defined as a set of inputs (sensors) and outputs (actuators) and a set of states. A set of state transitions is defined as a function of a system goal, a system-specific world model, and the current inputs. The goals are regarded as static parameters that define a target state of the system (i.e. the environment and the entity itself). A state is characterized by a number of (heterogeneous) parameters, which the entity can measure or estimate, and whose meaning is described semantically in the entity’s world model. A heterogeneous distance function is defined that allows the entities to detect and react to anomalies. The entity tries to perform a sequence of predefined state transitions in an attempt to bring the system to one of the desired states. A problem solving entity makes use of an explicit world model that consists of a set of labels for the sensed data, possible data patterns, possible actions, etc., and the relationships between all these. In principle, such a world model could be regarded as an ontology fragment (in fact it also contains quantitative knowledge and complex functions). Both the goal of the entity and the state of the environment can be represented by arbitrary semantic constructs of tuples (concept, value) connected with any of the logical operators $\vee$, $\wedge$, or . Values are specified using relational operators (=, $<$, $>$, etc.).

Example: The input data for an agent is a video stream from a camera. The agent can measure the average luminance of the observed scene, can detect multiple instances of cars and people, and can map the apparent coordinates of the detected objects on absolute geographical coordinates. This agent’s ontology fragment can be described as in Fig. 3 by a minimal set of primary concepts (i.e. those that label the physical quantities or objects which the agent can detect) and some useful generalizations and abstractions. Some concepts are further generalized in some other ontology, which may remain unspecified (the “ext” connectors). The link between the primary concepts and the physical interfaces of the entity is performed through predefined algorithms (e.g. feature extraction, pattern recognition), which are regarded

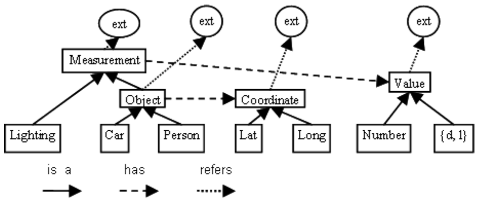

Fig. 3 Example of a local ontology fragment

as implicit (or procedural) knowledge of the entity. A goal can be represented in this case as a target state of the environment:

((Lighting, dark) $\wedge$ ((Car, <1) $\vee$ (Person, <1))) $\vee$ ((Lighting, light) $\wedge$ (Car, <10) $\wedge$ (Person, <20)).

Once an anomaly is detected the entity selects and executes one of the predefined algorithms for restoring the state of the system. From this perspective, the entity behaves in a purely reactive manner. However, for complex problems it may not be efficient, or even possible to define all the algorithms required for coping with all the anomalies that the entity can detect. Recall from above that an environment model describes not only relationships between the measured environment variables, but also how they can be acted upon. This latter part of the world model (i.e. the causal ontology, [7]) forms the basis on which the entity can select an appropriate behavior given a certain state of the system. Ideally, such a causal ontology would establish relationships between the available actuators of an entity and all its primary perceptual concepts, meaning that the entity is capable of influencing all the environmental parameters that it can measure. This, of course, is not always possible, so some actions may need to be specified in a causal ontology fragment that links to external ontologies.

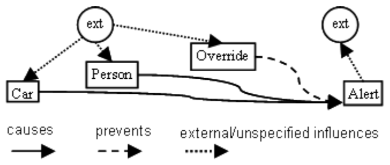

Fig. 4 A causal ontology fragment; external influences are only exerted on primary concepts (they interface with the environment). The internal causal relations are inherently nave given the limited ontologies. “Override” allows an external event to alter randomly the behavior of this entity.

4 AAC Self-Organization

As explained earlier, a goal can be formulated as a set of target states of a system (i.e. environment and AAC entities) expressed as (heterogeneous) vectors. If an entity has $m \leq n$ primary concepts (see section 3) that describe the same state parameters as m of the goal vector's components, then the entity will acquire a sub-goal which is a projection of the original set of target states on the m – dimensional subspace defined by the set of overlapping state parameters. However, the problem is more complex that just defining a linear map. Indeed, under the assumption that each entity has an incomplete and independently generated world model, a direct mapping between primary concepts is not a trivial task, as indicated by the ongoing research in the field of semantic matching and ontology alignment (see e.g. [1], [4]). Given the space limitations we do not extend the present discussion in this direction. Assume that suitable methods exist for estimating similarities or inclusion relations (e.g. subsumptions, intensions) between concepts. Obviously, finding these relations is only possible when the local ontologies of the entities in a given community partially overlap. For this reason this should be regarded as a hard requirement for the design of an AAC. Assume therefore that the community goal, expressed as a set of vector values indicating the desired states of the system, can be measured by two entities (see Fig. 5). The state vectors of these entities may contain additional parameters, and some other parameters may need to be provided by other entities (indicated by the grayed boxes). Nevertheless, a community goal can in principle be fulfilled when all state parameters can be measured, which means that all leaf nodes in the goal splitting multi-tree represent primary concepts for those entities. In such a case the community is perceptually complete. However, this does not mean that the goal can be effectively fulfilled. In order for this to happen, it is necessary that the participating entities possess the effectual capabilities required for bringing each of the state parameters to a desired value. The analysis of the requirements for effectual completeness is similar to that for perceptual completeness. The problem that

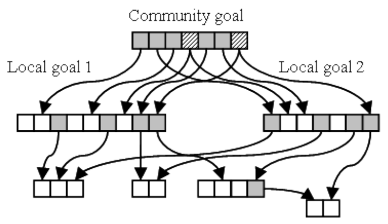

Fig. 5 Possible mappings between state vectors

remains is how to cope with the absence of a unique ontology for the whole community. Although we have assumed earlier that bilateral semantic similarities can be evaluated with existing techniques, it is still not obvious how a large number of entities can meaningfully work together at solving a complex problem. To explain this, we start by recalling that the goal vector is just a set of values corresponding to a semantic construct within the ontology of a certain entity. When a different entity tries to interpret this goal vector it actually parses the ontology of the originating entity, and tries to match some of the concepts in the source ontology with concepts in its own ontology. This second entity can derive a sub goal that it can fulfil either by its own, or with the help of some other entities. In the latter case it generates an additional goal containing those elements of its local goal which it cannot directly measure (i.e. are not primary concepts) and posts it as a new community goal. Eventually, if the community is perceptually complete, an entity will be able to fulfil a sub-goal all by itself. Then the entity which formulated the higher level goal can also fulfil a higher-level goal, and so on, up to the level of the original goal.

5 Discussion

The idea of integrating human users and software agents into a team is not completely new. In [6] Sycara and Lewis proposed a solution based on a set of specialized agents and a coordination framework where agents coordinate the communication between humans, but also contribute to the problem solving process. This is achieved through a common knowledge model shared by all agents. Problem solving capabilities and resources, task decomposition and behaviours are all defined at design time. An advertisement mechanism allows agents to find each other at runtime and respond to collaboration requests. The system is very efficient in coordinating multiple tasks and in coping with resource limitations, but requires a full definition of a common knowledge model, tasks sets, and behaviours.

The AAC approach proposed here provides a truly decentralized solution for a meaningful integration of human reasoning and software algorithms. The self-organization is based on goal decomposition and partial overlaps of the world models of the AAC entities.

6 References

[1] Doan, A., Halevy, A. (2005) Semantic Integration Research in the Database Community: A Brief Survey. AI Magazine, Vol. 26, No. 1, pp. 83-94.
[2] Ferber, J. (1999) Multi-Agent Systems, an introduction to distributed artificial intelligence. Addison Wesley.
[3] Gold, M. E. (1965) Limiting Recursion. Journal of Symbolic Logic, Vol. 30, No. 1, pp. 28-48.
[4] Kalfoglou, Y., Schorlemmer, M. (2003) Ontology Mapping: The State of the Art. The Knowledge Engineering Review Journal. Vol. 18:1, pp. 1–31.
[5] Lathrop, R. H. (1996) On the Learnability of the Uncomputable. Proc 13th Intl. Conf. on Machine Learning, pp. 302 – 309.
[6] Sycara, K., Lewis, M. (2007) Integrating Agents into Human Teams. Human Factors and Ergonomics Society Annual Meeting Proceedings, Cognitive Engineering and Decision Making , pp. 413-417.
[7] Terenziani, P. (1995) Towards a Causal Ontology Coping with the Temporal Constraints between Causes and Effects. Int. J. Human-Computer Studies, 43, pp. 847-863.
[8] Zhou, B.-H., Li, C.-C., Zhao, X. (2007) FIPA agent-based control system design for FMS. Int J Adv Manuf Technol 31, pp. 969–97.