Declarative Technologies for Open Agent Systems and Beyond

Federico Chesani, Paola Mello, Marco Montali, and Paolo Torroni

DEIS, University of Bologna
V.le Risorgimento 2
40136 Bologna, Italy
{Federico.Chesani,Paola.Mello,Marco.Montali,Paolo.Torroni}@unibo.it

Abstract. Open systems are complex, heterogeneous systems whose complexity is often handled by component-based approaches. While the internal functioning of such components is invisible to the outside, a great emphasis is put on the verification of properties that ensure a safe, predictable, and understandable external behaviour. At the same time, the rules that govern and describe the system behaviour must guarantee maximum flexibility. Developing technologies that can satisfy the requirements of open multi-agent systems poses an important and difficult challenge. Here, declarative approaches have the potential of offering solutions satisfying the needs for both specifying and developing open multi-agent systems and other open systems for diverse applications such as business processes, clinical guidelines and cloud computing.

Extended Abstract

"Open" is a recurring term in multi-agent research, commonly used to give a flavour of flexibility and unpredictability at the same time. Davidsson calls "open societies" those in which agents can join in without restrictions [4]. Along the same line, Dastani et al. write that open systems are characterized by heterogeneous participants which can enter or leave the system at will [3].

In the transition phases between "distributed AI" to "actors" and "multi-agent systems", definitions of open systems such as those proposed by Hewitt emphasized that such systems are open-ended and incremental, subject to continuous evolution [7], and that they are subject to unanticipated outcomes in their operation and can receive new information from outside themselves at any time [6].

This understanding of openness, which is more general than the one above, is still very influential in multi-agent research. According to Artikis and Pitt, "open agent systems" are those whose members have possibly competing interests, behave and interact in unpredictable ways, and do not have a publicly known internal architecture [2]. A great emphasis on unpredictability is also present in Fornara et al.'s view of open systems, by which interacting agents may not conform to predefined specifications [5].

P. Jędrzejowicz et al. (Eds.): KES-AMSTA 2010, Part I, LNAI 6070, pp. 1–5, 2010.

We give the term "open" the meaning adopted by Hewitt, because we are interested not only in the process of entities joining/leaving a systems, but also in the heterogeneity and autonomy of such entities. By "open systems" we thus mean heterogeneous systems whose complexity is handled by autonomous, distributed entities, whose internal architecture may be unknown.

Such systems are nowadays ubiquitous. They can be recognized in business process management, computerized clinical guidelines, cloud computing, and of course in open multi-agent systems. In all these systems, the achievement of strategic goals, such as a product's delivery or a service supply to the customer, emerges from the interaction between the entities composing the system. Therefore, while the internal functioning of their components is invisible to the outside, a great emphasis is put on the interactions among such components, and on verification of properties that ensure a safe, predictable, and understandable external behaviour, while guaranteeing flexibility.

Modeling and managing the interaction dimension is a challenging task, which can hardly be accomplished by using procedural specification languages. A major drawback of procedural, "closed" approaches is that they tend to force unnecessary rigidness on the way the system$\tilde{\text{O}}$s sub-parts coordinate to accomplish the desired strategic goals. For instance, in the context of business processes, procedural languages such as BPEL impose to explicitly enumerate all the ordering constraints among the activities carried out by the interacting entities [12]. This is unreasonable when the system must be able to cope with an unpredictable, changing and dynamic environment, where the autonomy of interacting entities and the possibility of exploiting new opportunities must be preserved as much as possible [13] or the expertise of workers must be exploited at best [9].

To overcome these issues, declarative approaches are gaining consensus as a way to capture regulations, constraints, policies and best practices involved in the open domain under study in a flexible way.

Open declarative interaction models are interaction models which provide openness and guarantee a high level of flexibility. They allow us to focus on the domain and the problem under study, abstracting away from specific execution flows. Suitable execution flows are synthesized or chosen by the interacting entities, while the model is completely focused on eliciting the set of constraints, regulations, norms, and patterns extracted from the domain and the problem.

Two complementary aspects of open systems are thus flexibility and compliance. Flexibility tends to widen the number of accepted system executions, leaving interacting entities free to exploit their own expertise in order to dynamically choose the best way to interact with one another. Flexibility favours usability and adaptability.

Compliance (also known as support) aims instead at guaranteeing adherence to external regulations/norms, internal policies, business rules, and so on. Compliance constrains the behavior of interacting entities, and it therefore restricts the number of accepted system executions.

Declarative open interaction models call for suitable frameworks able to provide support for their entire life cycle. Such frameworks must be able to keep an adequate level of flexibility to ensure usability and adaptability. At the same time, they must accommodate formal verification of the developed models and their executions, to support compliance and ensure trustworthiness and reliability.

If we look at the different stages of the specification's life cycle, we can single out the following key phases and desiderata [8]:

Design phase. Specification languages should accommodate openness and declarativeness. Moreover, the model under design should lend itself to verification. In particular, it should be possible to check whether the desired properties will be satisfied during some/all executions of the system, and to guarantee that regulations and norms are respected.

Execution phase. An enactment engine should be provided, to support the interacting entities during the execution. While procedural, closed models are usually enacted by presenting a sort of a "to-do-list" (i.e., a list of activities that must be executed next), the execution of declarative open models should be guided instead by the knowledge of which behaviours or actions are expected, preferred, forbidden, and so on, at all times.

Analysis phase. It should be possible to analyze the execution traces collected during different executions of the system, to compare the real, effective behaviour of interacting entities with the intended model. In particular, it should be possible to assess whether the business goals have been achieved, or to extract (discover) new models from the real behaviour.

A declarative open interaction model could also be used to verify third-party interaction models, either during the development phase (provided that the third-party model is accessible), or during/after the execution. Different possible verification tasks are:

Design phase. Let *IM* denote an internal model and *EM* a third-party model. A static verification task could be carried out to see whether *EM* complies with *IM* (which represents, in this case, a regulatory/prescriptive model). Alternatively, *IM* could represent a partial/local model, which must be composed with *EM* for achieving a complex strategic goal.

Execution phase. If *EM* is unaccessible (e.g. due to openness), the only possibility to address compliance is to monitor its executions, and to verify at run-time if it adheres to the prescriptions of *IM*. In some settings, such as electronic institutions, the output of run-time verification processes is used to generate repair actions aimed at influencing the system evolution towards the achievement a social goal (e.g., increased social welfare, or better resilience).

Analysis phase. If the execution traces produced by the execution of *EM* are stored and made available, then all the reasoning tasks introduced for the internal life cycle can be applied.

To meet the desiderata above, we propose to combine three complementary approaches, providing different abstractions to deal with open declarative interaction models. The first one is the constraint-based, declarative and graphical ConDec language proposed by Pesic and van der Aalst for modeling the acceptable courses of interaction in a graphical and intuitive manner [9]. ConDec was proposed in the Business Process context, but it can be used for other kinds of open systems too.

The second ingredient are social commitments: a well-known concept in multiagent research [10,11]. We use them for linking events and their effects to the mutual obligations established between the interacting entities during the execution, characterizing the (un)desired and exceptional relevant states of affairs.

The third element is the SCIFF computational logic framework [1] to formalize commitments and ConDec constraints, providing at the same time the verification capabilities for reasoning on the developed interaction models during their whole life cycle. The SCIFF framework encompasses a rule-based language for the declarative specification of interaction models, and a corresponding family of proof procedures that enable reasoning from specifications and system executions.

References

1. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: The sciff framework. ACM Trans. Comput. Log. 9(4) (2008)
2. Artikis, A., Pitt, J.V.: Specifying open agent systems: A survey. In: Artikis, A., Picard, G., Vercouter, L. (eds.) ESAW 2008. LNCS, vol. 5485, pp. 29–45. Springer, Heidelberg (2008)
3. Dastani, M., Dignum, V., Dignum, F.: Role-assignment in open agent societies. In: AAMAS 2003: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pp. 489–496. ACM, New York (2003)
4. Davidsson, P.: Categories of artificial societies. In: Omicini, A., Petta, P., Tolksdorf, R. (eds.) ESAW 2001. LNCS (LNAI), vol. 2203, pp. 1–9. Springer, Heidelberg (2001)
5. Fornara, N., Viganò, F., Verdicchio, M., Colombetti, M.: Artificial institutions: a model of institutional reality for open multiagent systems. Artif. Intell. Law 16(1), 89–105 (2008)
6. Hewitt, C.: Open information systems semantics for distributed artificial intelligence. Artif. Intell. 47(1-3), 79–106 (1991)
7. Hewitt, C., de Jong, P.: Open systems. In: On Conceptual Modelling (Intervale), pp. 147–164 (1982)
8. Montali, M.: Specification and Verification of Open Declarative Interaction Models: a Logic-Based Framework. PhD thesis, Department of Electronics, Computer Sciences and Systems, University of Bologna, Italy (2009)
9. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)
10. Singh, M.P.: An ontology for commitments in multiagent systems. Artif. Intell. Law 7, 97–113 (1999)

11. Torroni, P., Yolum, P., Singh, M., Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P.: Modelling interactions via commitments and expectations, ch. XI. IGI Global, Hershey, PA, US (2009)
12. van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Russell, N., Verbeek, H.M.W., Wohed, P.: Life after BPEL? In: Bravetti, M., Kloul, L., Zavattaro, G. (eds.) EPEW/WS-EM 2005. LNCS, vol. 3670, pp. 35–50. Springer, Heidelberg (2005) (invited speaker)
13. Yolum, P., Singh, M.P.: Flexible protocol specification and execution: applying event calculus planning using commitments. In: AAMAS, pp. 527–534. ACM, New York (2002)