On Using i^* for Modeling Autonomy, Reasoning, and Planning in Adaptive Systems

Yves Lespérance[1] and Alexei Lapouchnian[2]

[1] Department of Computer Science and Engineering, York University, Toronto, ON, Canada, `lesperan@cse.yorku.ca`
[2] Department of Computer Science, University of Toronto, Toronto, ON, Canada, `alexei@cs.toronto.edu`

Abstract. Many systems are being developed where components have some form of autonomy and can adapt to change. However, it is not clear how these aspects of autonomous systems can be modeled by existing software development frameworks. How can we model how much autonomy an agent/component has? How can we specify when autonomous behavior, reconfiguration, or reasoning/planning is triggered? How can we model the constraints that control autonomy? We need a framework and process for developing such systems. In this paper, these questions are discussed and some possible approaches are outlined.

1 Introduction

More and more systems are being developed where components have some form of autonomy, be it autonomic computing systems that reconfigure themselves in response to changing conditions, workflow systems that adapt, multiagent systems (MAS) that coordinate, or individual agents that perform planning, all to better achieve their goals. In most of these systems however, when dynamic reconfiguration or reasoning/planning occurs and what autonomy the system has is constrained by the designer or user. For instance, planning is usually constrained by control knowledge and behavior in MAS is usually constrained by social rules. It is not clear how these aspects of autonomous/adaptive systems, can be modeled by frameworks such as i^* [1], Tropos [2], Gaia [3], etc., either at the requirements or design stages. How can we model how much autonomy an agent has? How can we specify when autonomous behavior, reconfiguration, or reasoning/planning is triggered? How can we model the constraints that control autonomy? In this paper, these issues and questions are discussed and some possible approaches are outlined.

In the development of such systems, a *requirements-driven* process as in Tropos should be followed. The choice of how much autonomy and adaptiveness an agent should have and constraints on it should be evaluated with respect to the system requirements, functional and non-functional. The objectives of individual agents should be derived from the system requirements, something that existing approaches to adaptive/autonomous systems development, e.g. [4], often fail to address. The model should support requirements traceability. In this paper, we discuss these questions and make some proposals for an appropriate development process and associated models.

2 Background

Wooldridge [5] defines the notion of an autonomous agent as follows: “An agent is a computer system that is capable of independent action on behalf of its user or owner. In other words, an agent can figure out for itself what it needs to do in order to satisfy its design objectives rather than having to be told explicitly what to do at any given moment.” Adaptiveness is closely related to “reactivity”, an attribute of agents that perceive their environment and respond in a timely manner to changes. Autonomous action and adaptation may require activities such as reasoning, planning, scheduling, optimization, coordination, and negotiation, especially if the set of objectives and environment conditions that the agents have to deal with cannot be enumerated in advance.

Some work has tried to identify conditions under which using an agent architecture is appropriate for an application [5, 6]. There have been many proposals for agent-oriented software engineering methods, e.g. Gaia [3], Prometheus [7], Tropos [2], etc. Some of these are closely tied to a particular type of agent programming framework (e.g. BDI (Belief-Desire-Intention) agent programming languages) or even a particular programming platform. Such models may implicitly assume a particular control regime, e.g. models of goal decomposition rules in BDI agent programming languages.

i^* already has some features that support modeling autonomous agents. The distinction between a goal dependency and a task dependency turns on whether the delegatee has the autonomy to select the means to achieve a goal (rather than being required perform a specific task). The presence of a goal in an SR diagram may mean that the agent has the choice of how to achieve it. However, it may be expected that the agent will use one of the tasks specified as means in the model to achieve it. Generally, SR diagrams are not assumed to be complete, and additional means to achieving a goal may be derived, even at runtime. But there is often an implicit assumption that one of the specified means will be used. It seems clear that in some cases, the modeler takes the intentional stance [8] towards an agent, ascribing some goals (and beliefs) to it and expecting it to behave rationally and attempt to achieve these goals. In other cases, the modeler takes a design stance, expecting the agent to make decisions and act according to the way it was designed. Which of these stances is taken typically depends on how much of the agent’s design is known.

In i^*, an agent is viewed as capable of achieving a goal if it has a “routine” for achieving it. A routine is “an interconnected collection of process elements serving some purpose for an agent” [1], i.e. a plan skeleton. It seems clear that to support the specification of truly autonomous and adaptive agents and MAS, one needs models that are much less restrictive.

In previous work, our group has shown how i^* can be combined with ConGolog [9], a formal MAS specification/programming language, to support formal analysis/verification. Complete ConGolog models are executable and can also be validated by performing simulation. In this approach, i^* models are mapped into ConGolog by using an intermediate notation, annotated SR (ASR) diagrams [10], where process specification annotations are used to increase the precision and level of details of SR models.

Ordinary ConGolog does not support the specification of the intentional features of i^* models, i.e., the mental states of the agents in the system/organization modeled; these must be operationalized before they are mapped into ConGolog. But there is an extension of ConGolog called the Cognitive Agents Specification Language (CASL) [11] that supports formal modeling of agent mental states, incomplete agent knowledge, etc. Mapping i^* models into CASL gives the modeler the flexibility and intuitiveness of the i^* notation as well as the powerful formal analysis capabilities of CASL. We have extended the i^*-ConGolog approach to combine i^* with CASL and accommodate formal models of agents' mental states. Our intermediate notation has been generalized to support the intentional/mental state modeling features of CASL [12], in what we call intentional annotated SR (iASR) diagrams. With our i^*-CASL-based approach, a CASL model can be used both as a requirements analysis tool and as a formal high-level specification for a MAS that satisfies the requirements. This model can be formally analyzed using the CASLve [11] verification tool or other tools and the results can be fed back into the requirements model. One of the main features of this approach is that goals (and knowledge) are assigned to particular agents thus becoming their subjective attributes as opposed to being objective system properties as in many other approaches, e.g., Tropos [2] and KAOS [13]. This allows for the modeling of conflicting goals, agent negotiation, information exchange, complex agent interaction protocols, etc. However, this work does not support the modeling and analysis of many aspects of autonomous agent behavior, such as planning and reasoning.

Lapouchnian et al. [14] proposed a requirements-driven approach for designing adaptive systems with KAOS-like goal models (enriched with control flow annotations) that captured the variability in the way high-level system goals could be achieved. Various alternative ways of attaining these goals were analyzed with respect to their contribution to important quality criteria represented by softgoals. The system at runtime supports some or all of these alternatives and is able to switch from one alternative to another a) in response to changing user preferences over softgoals; b) in attempt to improve quality of service; c) as a result of a failure. This idea was further applied to the design and configuration of business processes (BPs) [15]. In this approach, the adaptivity of systems is limited to the alternative behaviors specified in the goal model. This favors predictability and trust in the system over adaptivity and autonomy. This technique may be a sensible choice for BP management and other applications where limited adaptivity may suffice, but it is not flexible enough to be used in a wide variety of adaptive systems.

3 Objectives

Here are some objectives that a framework for developing autonomous/adaptive systems should satisfy. First, it should support the specification of a wide range of types of autonomy/adaptivity (or lack thereof) that agents may possess, of what is known (and not known) about their design and decision making process, of the conditions under which reasoning or adaptation is triggered, and of what constraints apply to their decisions/behavior. This should rely on various models/stances that one can use to specify behavior while abstracting over design details.

Secondly, it should support analysis, allow predictions about agent behavior, and with a sufficiently detailed specification, formal verification, while remaining abstract. For instance, BDI-style specifications of agents together with specifications of their capabilities should allow reasoning about what goals will eventually be achieved under various conditions, even when the means cannot be specified in advance.

Thirdly, the framework should support the analysis of the merit of various alternative architecture designs with or without runtime reasoning and/or adaptation given the functional and non-functional requirements on the system. It should be possible to understand the benefits of doing more runtime reasoning in terms of increased robustness and improved solution quality, and its costs in terms of increased reaction time and unpredictability. The method should be requirements-driven. The objectives of individual agents and of the MAS itself should come from system requirements and the method should support requirements traceability. This is where using an i^*-based approach can help.

4 Some Proposals

There are two types of autonomy/adaptation that need to be modeled and analyzed: autonomy in individual agents, for instance through planning and reasoning, and adaptation in groups of agents, for instance through negotiation and coordination. Note that even the latter has an individual component as the negotiating/coordinating agents generally make individual decisions.

Here are some modeling and analysis techniques that could be exploited in a framework to achieve the objectives described earlier:

1. Tropos-style analysis of which parts of the system should be specified in advance and which should be left to be reasoned about at runtime; there are tradeoffs involved in making decisions about how much to specify, with effects on quality attributes such as predictability, trust, responsiveness, robustness, adaptiveness, autonomy, etc.;
2. extensions to the modeling language to support specification of actors or components as black boxes with behavioral constraints derived from high-level softgoals, and triggering conditions for reasoning and adaptation;

3. modeling extensions to support incomplete system specifications, such as weak constraints on the number of instances of an agent type and optional agent types/roles, and coordinator actors that constrain autonomy;
4. modeling extensions to support specification of the information that agents use to reason and make decisions.

In working on the design of the methodology/framework and its validation, it would be useful to model and analyze the use of various existing platforms for the implementation of autonomous agents, for instance:

- BDI agent programming languages that select plans to execute at runtime;
- agent programming languages that support runtime plan generation, such as IndiGolog [16] and CanPlan [17];
- other agent programming frameworks that support decision-theoretic planning, game-theoretic planning, or the use of deontic rules to constrain behavior;
- negotiating agents frameworks.

Moreover, there are many common applications where autonomous agents have been exploited that could be used to experiment with the framework:

- meeting scheduling systems;
- travel planning systems;
- systems that perform server load balancing or dynamic task allocation.

5 Conclusion

Autonomy and adaptiveness are qualities that are often required in state-of-art computer systems. Agent technology has been used to implement such systems. Agent-oriented software engineering methods have been proposed to help in designing them. Requirements engineering frameworks have also addressed the specification of such systems, while also incorporating agent notions such as "goals". Yet these development frameworks remain inadequate for modeling the features associated with autonomy and adaptiveness in systems. What is needed is a framework where these features and their connection with system requirements can modeled and analyzed. In this paper we have sketched how one might try to address this problem.

References

1. Yu, E.S.: Modelling Strategic Relationships for Process Reengineering. PhD thesis, Dept. of Computer Science, University of Toronto (1995)
2. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems **8**(3) (2004) 203–236
3. Wooldridge, M., Jennings, N.R., Kinny, D.: The Gaia methodology for agent-oriented analysis and design. Autonomous Agents and Multi-Agent Systems **3**(3) (2000) 285–312

4. Kephart, J., Chess, D.: The vision of autonomic computing. Computer **36**(1) (2003) 41–50
5. Wooldridge, M.: An Introduction to MultiAgent Systems. Wiley (2002)
6. Wooldridge, M., Jennings, N.R.: Pitfalls of agent-oriented development. In: Proc. of the 2nd Int. Conf. on Autonomous Agents (Agents 98), Minneapolis/St. Paul, MN, USA (1998) 385–391
7. Padgham, L., Winikoff, M.: Developing Intelligent Agent Systems: A Practical Guide. Wiley (2004)
8. Dennett, D.C.: The Intentional Stance. MIT Press, Cambridge, MA, USA (1987)
9. De Giacomo, G., Lespérance, Y., Levesque, H.J.: ConGolog, a concurrent programming language based on the situation calculus. Artificial Intelligence **121** (2000) 109–169
10. Wang, X., Lespérance, Y.: Agent-oriented requirements engineering using congolog and i^*. In Wagner, G., Karlapalem, K., Lespérance, Y., Yu, E., eds.: Agent-Oriented Information Systems 2001, Proceedings of the 3rd International Bi-Conference Workshop AOIS-2001, Berlin, iCue Publishing (2001) 59–78
11. Shapiro, S., Lespérance, Y., Levesque, H.J.: The cognitive agents specification language and verification environment for multiagent systems. In Castelfranchi, C., Johnson, W.L., eds.: Proc. of the 1st Int. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002), Bologna, Italy (July 2002) 19–26
12. Lapouchnian, A., Lespérance, Y.: Modeling mental states in agent-oriented requirements engineering. In Dubois, E., Pohl, K., eds.: Advanced Information Systems Engineering, 18th International Conference, CAiSE 2006, Luxembourg, Luxembourg, June 5-9, 2006, Proceedings. Volume 4001 of LNCS., Springer (2006) 480–494
13. Dardenne, A., Fickas, S., van Lamsweerde, A.: Goal-directed requirements acquisition. Science of Computer Programming **20** (1993) 3–50
14. Lapouchnian, A., Yu, Y., Liaskos, S., Mylopoulos, J.: Requirements-driven design of autonomic application software. In: Proc. of 16th Annual International Conference on Computer Science and Software Engineering CASCON 2006, Toronto, Canada, Oct 16-19. (2006)
15. Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-driven design and configuration management of business processes. In Alonso, G., Dadam, P., Rosemann, M., eds.: Proc. 5th International Conference on Business Process Management (BPM 2007), Brisbane, Australia, Sep 24-28. Volume 4714 of LNCS., Springer (2007) 246–261
16. De Giacomo, G., Levesque, H.J.: An incremental interpreter for high-level programs with sensing. In Levesque, H.J., Pirri, F., eds.: Logical Foundations for Cognitive Agents. Springer-Verlag (1999) 86–102
17. Sardiña, S., de Silva, L., Padgham, L.: Hierarchical planning in BDI agent programming languages: A formal approach. In: Proc. of the 5th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS'06), Hakodate, Japan, ACM Press (May 2006) 1001–1008