From Adaptive Systems Design to Autonomous Agent Design

Alexei Lapouchnian[1] Yves Lespérance[2]

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

Abstract. Interest in adaptive systems design has been steadily growing in the SE community, in part due to the ever-increasing complexity of modern software-intensive systems. Inspired by control theory, various types of controllers (e.g., feedback) are beginning to appear in software architectures for many applications. Within those controllers, distinct activities such as monitoring, analysis/diagnosis, planning, and execution are present. Approaches for requirements engineering, software architectures, and the design of such systems are beginning to emerge. In the case of agent-oriented systems, however, their hallmark is the ability to operate in highly dynamic and incompletely specified environments, handle tasks that may not be known a priori, etc. In this position paper, we look into what needs to be included in requirements-driven methods for designing agent-oriented systems, which, while drawing on ideas from control theory and adaptive systems design, would support much more autonomy, flexibility, and dynamism.

1 Introduction

As the complexity of software-intensive systems increases, more research in software engineering (SE) is being dedicated to the modeling, analysis, and implementation of adaptive systems, which promise to take on a certain range of tasks related to self-maintenance and self-adaptation. The main motivation for this is to reduce the maintenance overhead for these systems, to allow them to adapt to changing environments and user needs while continuing to deliver their functionality.

There are a number of ways to implement self-adaptation in software. A recent paper [1] identified several of the most common approaches for adaptive systems design. One of these approaches, which advocates the use of control loops, has roots in control theory, while another, which uses agents and multiagent systems, in artificial intelligence. There is a certain overlap between control loops and agents since in both paradigms, monitoring/sensing of the environment is followed by some analysis/reasoning and the enactment of the appropriate behaviour, both agents and control loops can be organized hierarchically, etc. The emerging control loop-based approaches for developing adaptive systems attempt to use well-founded systematic techniques to specify the details of self-adaptive systems. However, what distin-

guishes the agent-based approach is that it can support a higher degree of autonomy and distributed decision making, as well as be effective in highly dynamic and incompletely known environments, among other things. In this position paper, we consider several problems. Firstly, we look into which adaptation scenarios warrant the selection of agent-based approaches over control loop-based ones. Secondly, we look into what needs to be included in requirements-driven methods for designing agent-oriented systems in a more transparent and predictable way, which, while drawing on ideas from control theory and adaptive systems design, would achieve greater autonomy, flexibility, etc.

2 Background and Related Work

Control loops and especially feedback loops, have a long and successful history in engineering, and have recently been promoted as a promising way to implement self-adaptive software systems [2, 3]. Moreover, it is argued that adaptation concerns should be modeled and designed separately from the main functionality of the system (e.g., [3]), which fits nicely with the control loop approach. Feedback loops provide a generic mechanism for self-adaptation. To realize self-adaptive behaviour, systems typically have a number of controllers that can be organized into hierarchies. The main idea of feedback control is to measure the system output and achieve control objectives (e.g., maintaining a CPU utilization rate for a server) by adjusting system parameters. Thus, feedback controllers can be viewed as having to monitor the system, analyze the captured data, perform diagnosis, and plan and execute a course of action. One can then concentrate on aspects of these specific activities within the feedback controller. However, while there have been numerous advances in requirements engineering for this, little attention has so far been paid to the elicitation and analysis of the adaptation requirements such as deciding what to monitor, how to perform the analysis of the monitored data and the system diagnosis, as well as when and how to do compensation.

There are a number of recent agent-oriented approaches (e.g., [6]) that attempt to extend the Tropos requirements-driven approach [7] to support the design of adaptive systems. In [6], an approach for Belief-Desire-Intention (BDI) agent systems is proposed. Tropos extensions include the modeling of failure symptoms, possible causes, and compensations. This significantly constrains the amount of autonomy that the agents have in dealing with dynamic and incompletely known environments.

3 Research Objectives

One of the key questions for the research proposed in this position paper is to determine, given the benefits of agents and multiagent systems (MAS) as well as their costs and limitations, in which circumstances the use of agent technology is warranted in the design of adaptive systems. Also, this may help in determining the complexity and architecture of the agents needed. We propose to look at the recent research on the dimensions of self-adaptive software systems [8] and identify particular dimen-

sions, and values within those dimensions, that demand flexibility and autonomy that can be provided by the use of agents, thus indicating in what circumstances the use of an agent-based adaptive system is most promising. Here, we list some of the dimensions taken from [8] that seem like the most relevant for the use of agents:

- Goal *flexibility* is related to the level of uncertainty and flexibility in goal specifications. The values in [8] for this dimension are: rigid, constrained, and unconstrained. Common SE approaches are mostly applicable to rigidly specified goals. However, agents are capable of handling much more uncertainty in goal specifications up to the point where goals may not be known at design time. Thus, the need to handle unconstrained goals is pointing to planning agent-based solutions.
- *Anticipation* of change captures whether change (i.e., the cause of adaptation) can be predicted. The values are foreseen (taken care of), foreseeable (planned for), and unforeseen (not planned for). Clearly, the need to handle unforeseen changes may require the use of planning/reasoning agents capable of selecting and/or constructing plans (albeit in particular pre-analyzed domains) at runtime.
- *Autonomy* of adaptation mechanisms identifies the degree of outside intervention during adaptation (from autonomous to assisted – by a human or another system). While both control loop-based and agent-based approaches provide some degree of autonomy, in the context of unforeseen changes, reasoning and social agents will generally be more autonomous and thus more applicable. In MAS, the notion of a system may be quite fluid, since agents may be joining and leaving the MAS.
- *Organization* of adaptation – centralized vs. decentralized. Agents naturally support decentralized adaptation, which is especially useful when information is distributed. The challenge here is to integrate adaptations addressing different concerns that may be implemented by different agents and manage conflicts between them. Approaches to this include social laws and mechanism design. Thus, multiagent systems can provide another level of control and governance to distributed adaptive systems. Another challenge is integrating agent-based adaptation mechanisms with other approaches, notably control loop-based methods.
- *Effect* (*predictability*) of adaptation – whether the consequences of self-adaptation can be predicted both in their nature and temporal extent. Degrees of predictability range from non-deterministic to deterministic. This predictability is associated with guarantees. In control loop-based approaches, the effects of adaptations are systematically studied and generally are predictable (still, external *disturbances* can make things less predictable). In agent-based systems, especially in complex ones, predicting *emergent behaviour* can be difficult. Careful derivation of constraints on reasoning components and other advanced agent features (e.g. using "anytime" methods) may improve predictability.

Another question that we propose to address is the following. On the one hand, agents and MAS offer a lot of power and flexibility when autonomy is needed, when dealing with dynamic and incompletely known environments, goals that are unknown at design time, etc. On the other, these advanced features frequently are not supported by systematic requirements-driven engineering approaches, are hard to represent visually in modeling notations, and may not provide enough predictability and transparency for some domains. Moreover, there is a variety of agent technology/architectures, ranging from simple rule-based reactive agents to planning and decision-theoretic agents. While the former can be seen as variations of feedback

loops and thus can use, e.g., the ideas in [4], the latter seem to require quite different modeling and analysis techniques and cannot be easily dealt with by existing requirements-driven approaches such as Tropos.

Thus, the challenge is to allow the use of advanced agent techniques as needed, while improving the transparency and predictability of agent-based adaptive systems. We need more systematic requirements-driven agent-oriented software engineering approaches. Here, we can treat agents as feedback controllers with distinct monitoring, reasoning, planning, and execution activities and then use ideas from feedback control-based adaptive systems and proposals such as [4] to systematically derive not only the functional requirements for the system, but also the adaptation requirements – for monitoring, reasoning, etc. The method should support explicit representation of reasoning and (classical or decision-theoretic) planning capabilities within agents. The requirements for these agent features can be identified by looking at the relevant adaptation dimensions as described above. Constraints on the behaviour of these agent components should also be elicited and represented. Declarative specifications of at least certain parts of agents will support their evolvability and help in avoiding the need to explicitly and exhaustively capture, e.g., situations requiring adaptation. These agent features support shifting goal refinement from design time to runtime.

One possible difficulty with the above approach is that to improve the predictability and transparency of agent-based adaptive systems, the proposed method needs to separate the specification of the adaptive functionality from the main system functionality. For example, if an agent has a component capable of constructing plans to achieve goals in a certain domain as well as to adjust these plans as they are being executed, not only does the component address the functional requirements, but the adaptation requirements as well, since it has to monitor plan execution and the state of the environment, compute diagnosis, and provide compensations/do replanning. It remains to be seen whether this idea has limitations.

4 Ongoing and Future Work

Many of the ideas suggested in the previous sections are for future work. We are performing a thorough analysis of the dimensions of self-adaptive software systems and the identification of the ones that warrant the use of agent-based adaptive systems approaches. Moreover, we are interested in identifying which adaptive system requirements and which values for adaptive systems dimensions can help us with the selection of a particular agent type/architecture (e.g., simple rule-based reactive agent vs. BDI agent vs. classical planning agent vs. decision-theoretic planning agent).

In [4], an attempt is made at deriving monitoring and analysis requirements for feedback loops given a particular class of meta-requirements (awareness requirements). These meta-requirements are captured using goal models *in addition* to the usual functional and non-functional requirements for the system. These models represent the requirements of *meta-processes* responsible for the adaptive behaviour of the system. Contexts [5] are used to model situations requiring adaptation (e.g., failures), while compensation goals are explicitly represented and refined. However, this approach involves explicit modeling of situations that require changing the behaviour of

the system as well as explicit specifications of adaptations/compensations. One cannot say in this approach that decisions about when and how to change the behaviour are to be made at runtime. We plan to use the approach of [4] as a starting point to integrate ideas from control loop-based approaches with relevant agent techniques to support the analysis and design of agent-based adaptive systems.

Integrating centralized control loop approaches with the distributed agent-based approaches is also a challenge. How can we seamlessly integrate these adaptation mechanisms? Can they be used in a hierarchical fashion (e.g., low-level feedback loops being controlled by higher-level goal-driven agents) or at the same level, each controlling a particular aspect of system adaptation?

5 Conclusions

Research in self-adaptive systems is growing in importance driven by the increasing complexity of software systems. While control loop-based approaches for engineering adaptive systems look promising, they lack support for distributed adaptive behaviour that supports dynamic and incompletely known domains. On the other hand, agent-based approaches, while being powerful in their flexibility, support for dynamic goals, etc., may lack predictability and transparency. In this position paper, we argue for a requirements-driven design approach that builds on control loop-based approaches to support more flexibility, autonomy, as well as transparency and predictability, in agent-based adaptive systems.

References

1. Cheng, B.H.C., et al. Software Engineering for self-adaptive systems: a research roadmap. *Software Engineering for Self-Adaptive Systems*, LNCS Vol. 5525, 2009, pp. 1–26.
2. Hellerstein, J.L., Dao, Y., Parekh, S., Tilbury, D.M. Feedback control of computing systems. Wiley, 2004
3. Brun, Y., et al. Engineering self-adaptive systems through feedback loops. LNCS Col. 5525, 2009, pp. 48–70.
4. Lapouchnian, A., Souza, V.E.S., Mylopoulos, J. Awareness requirements for adaptive systems. Submitted for publication.
5. Lapouchnian, A. and Mylopoulos, J. Modeling domain variability in requirements engineering with contexts. In Proc. *ER '09*, LNCS Vol. 5829, 2009, pp. 115–130.
6. Morandini, M., Penserini, L., Perini, A. Towards goal-oriented development of self-adaptive systems. In Proc. *SEAMS '08*, pp. 9–16.
7. 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):203–236, 2004.
8. Andersson, J., de Lemos, R., Malek, S., Weyns, D. Modeling dimensions of self-adaptive software systems. *Software Engineering for Self-Adaptive Systems*, LNCS Vol. 5525, 2009, pp. 27–47.