http://www.diva-portal.org

Postprint

This is the accepted version of a paper presented at *12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), May, 6–10, 2013, Saint Paul.*

Citation for the original published paper:

Gil de la Iglesia, D., Weyns, D. (2013)
SA-MAS: Self-Adaptation to Enhance Software Qualities in Multi-Agent Systems.
In: Ito, Jonker, Gini, and Shehory (ed.), *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems (AAMAS 2013)* (pp. 1159-1160). The International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS)

N.B. When citing this work, cite the original published paper.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:lnu:diva-25909

SA-MAS: Self-Adaptation to Enhance Software Qualities in Multi-Agent Systems

(Extended Abstract)

Didac Gil de la Iglesia
CeLeKT, Linnaeus University, Sweden
didac.gil-de-la-iglesia@lnu.se

Danny Weyns
DFM, Linnaeus University, Sweden
danny.weyns@lnu.se

ABSTRACT

Engineering multi-agent systems (MAS) is known to be a complex task. One of the reasons lays in the complexity to combine multiple concerns that a MAS has to address, such as system functionality, coordination, robustness, etc. A well-recognized approach to manage system complexity is the use of self-adaptation (SA). Self-adaptation extends a system with support to monitor and adapt *itself* to realize a concern of interest (optimization, fault-tolerance, etc.). We present SA-MAS, an architectural approach that integrates MAS with SA. We present a reference model for SA-MAS and illustrate it with an excerpt from our research.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures

Keywords

Multi-agent systems, self-adaptation, reference model

1. COMPLEXITY IN ENGINEERING MAS

Multi-agent systems (MAS) have been applied in a broad number of fields, such as e-commerce, traffic, robotics, learning applications, etc. Among the benefits of these systems are autonomy of interacting entities, flexibility of entities that can come and go at will, efficiency as a result of local decision making, etc. Despite the benefits of MAS, engineering such systems pose huge engineering challenges because of their distributed behavior and the need to deal with multiple concerns [1]. One recognized approach to manage complexity in MAS is by means of employing domain-specific middleware, where (parts of) the coordination is separated from agents that realize the system functions [2].

Fig. 1 shows the traditional design of a MAS on top of a communication middleware. The agent behavior provides the required domain functions for the system at hand, while the (optional) coordination module deals with coordination concerns. The coordination module is optional as a MAS may be designed with purely agents that communicate via exchanging messages. Established approaches for coordination are electronic institutions, stigmergic coordination, and middleware support for organization management. Providing distinct modules for agent behavior and coordination offers a separation of concerns. However, this separation does not consider quality properties as first class concerns.

For example, a common approach to improve performance in the face of uncertainty is using a learning approach. However, when it comes to engineering, support for learning is often interwoven with the regular agent behavior, which may lead to complex agent architectures. [3] proposes a basic architecture for a learning agent that contains separate components that allow an agent to observe its behavior and the environment and make improvements on its behavior.

As another example, consider a situation where a deployed MAS has initially been designed without taking into account particular types of faults. A typical way to deal with such problem would be to extend the agent behavior and possibly the coordination module making the system fault tolerant. However, scattering support over agents and coordination modules, is error prone and may lead to complex designs that are difficult to understand and maintain.

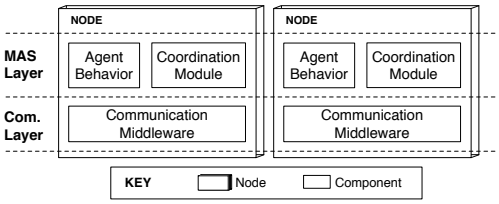

Figure 1: Traditional agent structure

2. SA-MAS

Self-adaptation (SA) is a well recognized approach for managing system complexity [4, 5, 6]. SA is used for adding so called self-* properties (self-configuration, self-optimization, self-healing, self-protection) to a system to address changing operating conditions in the system or its environment. SA is based on the design principle of *separation of concerns*. SA is a promising approach to tackle the complexity of engineering MAS by separating the logic that deals with selected quality concerns from the domain functions provided by agents and supporting coordination modules. SA can be either used as a principle for greenfield design of MAS, or as an approach to extend a legacy MAS.

Self-adaptation has been applied to MAS in a number of research and engineering efforts, some recent examples are [7, 8, 9]. From these and other studies, we present a reference model for a SA-MAS. Fig. 2 shows the model that

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.
Copyright © 2013, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

is conceived as a modular architecture, where different concerns of the MAS are clearly separated.

The *Agent Behavior* encapsulates the functionality of the domain at hand. The *Coordination Module* encapsulates coordination concerns, as explained in the previous section. The *Self-Adaptation Module* encapsulates the logic to deal with a particular quality concern of interest, e.g., enable to learn over time, deal with particular types of errors, handle security vulnerabilities, etc. Together these three modules comprise the logic of an agent that provides the required functions (agent behavior and coordination module) and is able to monitor and adapt itself (self-adaptation module) to enhance it with a particular quality concern. Note that the reference model does not impose any particular agent architecture or type of coordination support.

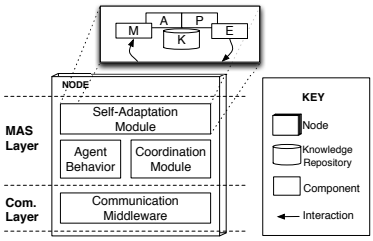

Figure 2: Reference model for SA-MAS

A common approach to realize SA is by a MAPE feedback loop [4], as shown in Fig. 2. A knowledge component (K) maintains relevant knowledge w.r.t. the agent behavior and coordination, as well as the adaptation goals. A monitor component (M) gathers particular data from the underlying components and possibly the system's environment in order to update the K component, providing the subsequent computations of the MAPE loop with the necessary data. An analyze component (A) examines the knowledge gathered by the monitor, and based on the adaptation goals draws conclusions on which actions should be undertaken by the SA module. A plan component (P) puts together a series of adaptation actions to resolve the problem identified by the analyzer. These actions are then carried out by an execution component (E), allowing adaptation of the agent behavior and coordination module to realize the adaptation goals.

Sometimes, multiple quality concerns need to be considered when designing a MAS, e.g., performance and robustness. This requires several MAPE loops to address the different concerns. As these quality concerns may be crosscutting, it is important to keep the self-adaptation components separated to reduce their complexity and improve scalability and reuse. [8] discusses inter-loop and intra-loop coordination in a SA-MAS application.

3. A MOBILE LEARNING EXAMPLE

In previous work [10], we developed a MAS application that provides pupils with GPS-enabled mobile devices to do outdoor learning activities. For example, three students have to work together to calculate distances using triangulation techniques. From experience, we learned that the GPS quality is critical to avoid misleading conclusions from the pupils. However, the initial design did not consider varying GPS sensitivity over time. To deal with this problem, we extended the MAS application with support for self-adaptation. Fig. 3 shows the top-level design of one mobile device of the distributed system.

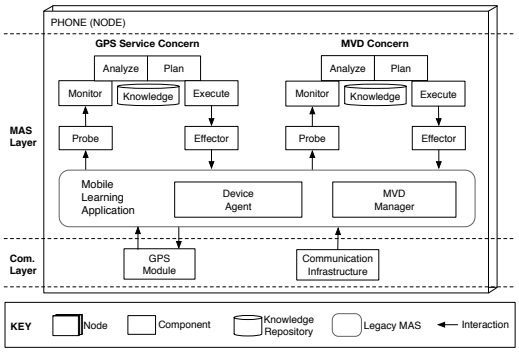

Figure 3: Example instance of SA-MAS architecture

The device agent is responsible for managing the tasks pupils have to perform (interacting with a server and devices of team members) and for performing measurements (using the GPS module). The MVD manager deals with organization management, that is, devices of a team are grouped as an organization, what we call a Mobile Virtual Device. Self-adaptation to handle robustness of the GPS module is conceived as two MAPE loops. The first loop (GPS service concern) monitors the quality of the local GPS module, compares it with the required quality, and based on that, activates or deactivates the GPS service. When a GPS service is deactivated, it triggers the second MAPE loop (MVD concern) to start a self-healing process, that is, find a new device and add this to the MVD. Probes and effectors enable the MAPE loops gathering the relevant data of the underlying system and applying the planned adaptations actions.

In our ongoing research, we study the use of formal methods for the self-adaptation design and apply model checking to verify the required properties.

4. REFERENCES

[1] D. Weyns and M. Georgeff, "Self-Adaptation using Multiagent Systems," in *IEEE Software*, 2010.

[2] D. Weyns, A. Omicini, and J. Odell, "Environment as a first-class abstraction in multiagent systems," *JAAMAS*, vol. 14, no. 1, 2007.

[3] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*, 2003.

[4] J. Kephart and D. Chess, "The vision of autonomic computing," *IEEE Comp. Soc.*, vol. 36, no. 1, 2003.

[5] J. Kramer and J. Magee, "Self-managed systems: an architectural challenge," *FOSE*, pp. 259–268, 2007.

[6] D. Weyns, S. Malek *et al.*, "FORMS: a formal reference model for self-adaptation," in *ICAC*, 2010.

[7] S. Lynch, "Using Meta-Agents to Build MAS Platforms and Middleware," in *ICAART*, 2011.

[8] P. Vromant *et al.*, "On Interacting Control Loops in Self-Adaptive Systems," in *SEAMS*, 2011.

[9] M. U. Iftikhar and D. Weyns, "A Case Study on Formal Verification of Self-Adaptive Behaviors in a Decentralized System," *EPTCS*, vol. 91, 2012.

[10] D. Gil de la Iglesia, *Uncertainties in Mobile Learning applications: Software Architecture Challenges.* Linnaeus University, 2012.