Using Semantic Web Technology for Self-Management of Distributed Systems*†

A.R. Haydarlou M.A. Oey B.J. Overeinder F.M.T. Brazier

Vrije Universiteit Amsterdam, De Boelelaan 1081a,
1081 HV Amsterdam, The Netherlands
Email: {rezahay,michel,bjo,frances}@cs.vu.nl

1 Introduction

Automated support for management of complex distributed object-oriented systems is a challenge: self-management is the goal. A self-management system needs to reason about the behaviour of the distributed entities in a system, and act when necessary. The knowledge needed is multi-leveled: different levels of concepts and rules need to be represented. This paper explores the requirements that hold for representing this knowledge in self-managed distributed object-oriented systems, and explores the potential of Semantic Web technology in this context.

2 Self-Management Knowledge Representation Requirements

This section discusses a number of important requirements for representing self-management knowledge in distributed object-oriented environments, and the potential of the Semantic Web in relation to these requirements.

Knowledge locality and modularity - In a distributed system, consisting of multiple software components, the self-management knowledge of each component needs to be described and stored locally to facilitate the specification, distribution, migration, and reuse of these components. This implies that the knowledge representation needs to be able to describe and reason about distributed entities. This requirement is satisfied by the Semantic Web languages because they support the use of *Uniform Resource Identifiers (URI)*. URIs make it possible to identify and refer to resources stored on different locations.

Knowledge reasoning - Self-management knowledge does not only contain concepts, but also includes rules to reason about these concepts, and even meta-rules to reason about rules themselves. Rules are used in a running self-management system to analyse a system's current status in order to detect and repair any unwanted situation. Therefore, there is a need for a rule language that supports multi-level reasoning with distributed knowledge. This requirement is satisfied by the Semantic Web languages because SWRL is specifically designed to reason about OWL concepts. SWRL is closely integrated with OWL, and therefore, rules in SWRL can directly use OWL concepts.

Knowledge acquisition - Self-management knowledge is specified by many types of users across organisations. To support the creation and maintenance of self-management knowledge, two non-functional requirements hold. (1) **Tool availability**: in distributed environments, knowledge acquisition tools are essential for system developers to enter, visualise, and check the consistency and soundness of complex self-management concepts, and to execute logical rules to infer new facts. (2) **User acceptance**: given the diversity in the types of users and organisations, the language used for self-management knowledge specifications should be intuitive and preferably comply with standards. Both non-functional requirements, mentioned above, are satisfied by the Semantic Web languages. The **tool availability** requirement is satisfied, because the

*This research is supported by the NLnet Foundation, http://www.nlnet.nl, and Fortis Bank Netherlands, http://www.fortisbank.nl.
†The full version of this paper appeared in Proceedings of the International Conference on Web Intelligence (WI-06), 2006.

Semantic Web community provides various *Integrated Development Environments (IDEs)*, plugins to other existing IDEs, Java APIs, and tools and techniques for checking the syntax and consistency of OWL documents. The **user acceptance** requirement is satisfied, because the Semantic Web languages comply with W3C standards and have common and relevant features with *Unified Modeling Language (UML)*. UML is the de facto industrial standard used by software developers (users).

3 Self-Management Concepts in OWL

A model-based framework for self-management of distributed object-oriented systems has been designed [1] in which the three important modules, i.e., self-monitoring, self-diagnosis, and self-adaptation share the same model of the distributed object-oriented system. Figure 1 illustrates the high-level self-management concepts of the framework, related to each other by means of OWL object properties.

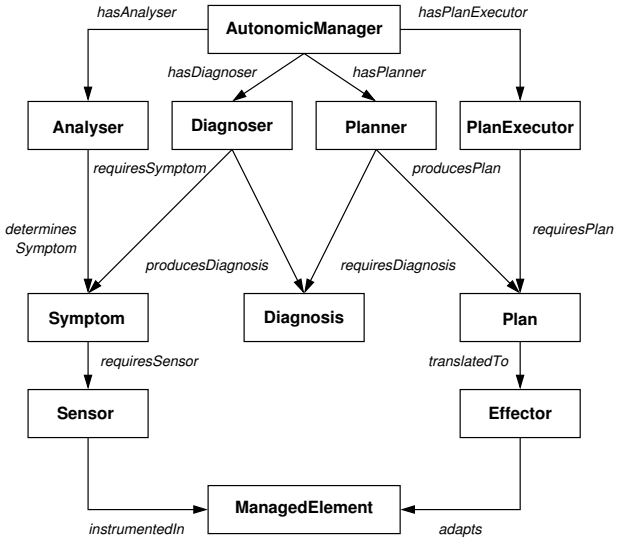

Figure 1: Self-management overview

The AutonomicManager monitors the execution of a system use-case. It has a containment relation with the concepts Analyser, Diagnoser, Planner, and PlanExecutor. The Analyser is responsible for detecting an incorrect system use-case realisation (i.e., abnormal system behaviour). A Symptom represents such abnormal system behaviour based on one or more sensor-values. A Sensor is instrumented in a software unit (ManagedElement) in a running application. The Diagnoser uses the determined Symptom, and its own meta-knowledge to produce a Diagnosis addressing the possible root-cause of application malfunctioning. The Planner constructs a Plan based on the Diagnosis. A Plan is an ordered set of actions (self-configuring, self-healing, or self-optimising actions) needed to repair the detected abnormal behaviour. PlanExecutor is responsible for translating Plans into Effectors which are instrumented in ManagedElements to adapt the system's behaviour by executing the plan.

In conclusion, OWL is used to express knowledge about the mentioned self-management concepts and concept hierarchies. SWRL is used to express system behaviour, multi-level diagnostic reasoning, and adaptation constraints. The ability of these languages to represent knowledge relates strongly to the requirements of representing self-management knowledge in distributed object-oriented systems: support for knowledge acquisition, local knowledge representation, and distributed reasoning.

References

[1] A. R. Haydarlou, B. J. Overeinder, M. A. Oey, and F. M. T. Brazier. Multi-level model-based self-diagnosis of distributed object-oriented systems. In *Proceedings of the 3rd IFIP International Conference on Autonomic and Trusted Computing (ATC-06)*, Wuhan, China, September 2006.