Knowledge-Based Systems 67 (2014) 143–161

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A hybrid approach to self-management in a pervasive service middleware

Weishan Zhang [a,*], Klaus Marius Hansen [b], Mads Ingstrup [c]

[a] Department of Software Engineering, China University of Petroleum, No. 66 Changjiang West Road, Qingdao 266580, China
[b] University of Copenhagen, Njalsgade 128, 2300 Copenhagen S, Denmark
[c] No Magic, Bangkok, Thailand

ARTICLE INFO

Article history:
Received 3 April 2013
Received in revised form 28 May 2014
Accepted 1 June 2014
Available online 14 June 2014

Keywords:
Self-management
Architectural styles
Component control
Goal Management
Change Management

ABSTRACT

Self-management capabilities for pervasive computing systems are critical in improving dependability, usability, and autonomicity. However, realizing self-management is not an easy task due to complexities of implementing autonomous behaviors. It has been recognized that a single autonomicity handling mechanism is not sufficient to realize comprehensive self-management capabilities when different technologies are involved. Therefore, we propose a hybrid approach, the 'LinkSmart Three Layered architectural (LinkSmart-3L) style', in which different architecture styles are incorporated. The LinkSmart-3L style enables self-management at an architectural level. In our approach, semantic web technologies are used to achieve comprehensive context-awareness and extensibility of self-management capabilities, genetic algorithms are used to achieve configuration optimizations, and a planner is used to compute planning procedures on how to arrive at an optimum system configuration based on current architectural structure of the underlying system using an architectural query language. These technologies are integrated seamlessly based on the service oriented computing (SoC) paradigm. We have extensively evaluated both runtime and development time qualities of our implementation of the style. These evaluations can serve as guidelines for evaluating other middleware systems. We conclude that our approach is usable and effective in achieving these quality attributes.

1. Introduction

Self-management capabilities are attractive for pervasive computing systems as these systems are becoming more widely deployed, and requirements on dependability increase. Such systems are operated as open systems undergoing dynamic changes in which services and devices may join or leave at any time anywhere, and system resources, e.g., battery levels may fluctuate dynamically. Self-management capabilities as in autonomic computing [1] can potentially make these pervasive systems more user-friendly and energy efficient. For example, to diagnose the source of a failure and then recover to a normal operating state, or to make full use of remaining energy to prolong the runtime of a system. Because of these potential benefits, many efforts have been put into the realization of self-management capabilities [1]. These self-management capabilities are considered as the most important features of future network applications and should be given priority in future research [2].

Although there is substantial pervasive middleware research such as EU MUSIC [3] and Rainbow [4,5], these efforts have primarily used a single approach to or focused on a single aspect of self-management. For example, the MUSIC project explored architecture based self-adaptation, while paying little attention to other self-management features such as self-protection, and self-optimization. For architecture-based self-adaptation, a critical issue is the choice of an architectural style to represent a target system [5], but the MUSIC project did not focus on this either. Kramer and Magee [6] observe that many early self-managing systems followed the 'sense-plan-act' architecture (e.g., Rainbow by Garlan et al. [4]) in which a system conceptually cycles through sense-plan-act phases. Kramer and Magee proposed a three layered architecture style that incorporate these three phases, but this is only an abstract reference model that needs to be verified in practice, and the problem of how to actually derive a software architecture style that can be implemented and used in practice for self-management systems is not addressed.

On the other hand, implementing and achieving self-management are not easy tasks [7]. There are many aspects that should be considered in a self-management solution, such as effective

* Corresponding author. Tel.: +86 532 86981972.
E-mail address: zwssky@gmail.com (W. Zhang).

http://dx.doi.org/10.1016/j.knosys.2014.06.002
0950-7051/

sensors to detect system status, actuators to accomplish needed changes, change management schemes to react to self-management changes, and reasoning on which actions to take. Furthermore, if an existing scheme cannot fulfill a Quality of Service (QoS) requirement, planning mechanisms should be used to help to find a corresponding (near) optimal configuration, and then this configuration should be enabled dynamically. Thus, the whole self-management process involves quite a number of different tasks which can be realized with different techniques. Correspondingly, a self-management solution should support a hybrid of different techniques technologies at an architecture level in order to integrate these technologies seamlessly. A single mechanism for realizing self-management may be considered as too limited to realize comprehensive autonomic features [7].

In the process of reaching a hybrid solution for self-management, there exist a number of challenges. These include:

1. How to define an architectural style that can be used to achieve self-management at an architectural level [6].
2. How to verify this architectural style by a practical implementation in order to show usefulness.
3. How to achieve interoperability, modifiability, and extensibility in this architectural style to easily integrate various self-management features including self-protection, self-protection, self-optimization, and self-configuration. Preferably it should be simple to add new self-management features that do not exist.
4. How to make sure that technologies used for various purposes can be integrated in this architectural style.

To address these challenges, we propose the LinkSmart Three Layered architectural (LinkSmart-3L) style that features a service-oriented architecture and seamlessly integrates several technologies to facilitate self-management. In our practical implementation and evaluation, we use pervasive web services [8] and OSGi Declarative Services [9]. The main contributions of our work are:

1. A 'LinkSmart Three Layered architectural (LinkSmart-3L) style' is proposed and exemplified. This style combines different architectural styles to enable architecture-level self-management. This makes it possible to bring together benefits from those underlying architectural styles.
2. Our approach is realized such that different self-management technologies can be integrated seamlessly based on the service-oriented computing paradigm. In our implementation, we have integrated Semantic Web technologies, genetic algorithms, an architectural scripting language, and an architectural query language.
3. Our middleware is evaluated in terms of runtime qualities including interoperability, performance, scalability, and development-time qualities including modifiability, testability and, usability, following the quality attributes framework proposed by Bass et al. [10]. We have performed evaluations which can serve as guidelines for evaluating other middleware systems, as currently there are no unified evaluation criteria or methodology.
4. Following the proposed evaluation quality attributes, our approach has been evaluated extensively in the LinkSmart middleware. For interoperability, both the .NET and Java environments are tested. This includes tests of using various available wired and wireless communication protocols. The performance has been tested extensively for various components and a whole self-management life cycle takes less than 4s which is acceptable in a distributed pervasive service environment. The scalability tests show that the needed time to process self-management actions is in linear with the number of events for our tests. The modifiability tests show that the middleware can be extended with relative ease to add new self-management functions. The testability and usability tests show that the proposed approach is practical in use.
5. To promote the usage of self-management technologies and encourage research in this area, we have released our work including a set of supporting ontologies as open source such that other researchers can explore and potentially further our work.

The remainder of this article is organized as follows. First, we present background on self-management, especially architectures for self-management in Section 2. In order to address challenge number 1, we define an architectural style (LinkSmart-3L) based on a set of objectives and constraints for self-management (Section 3) and we discuss key design choices of the LinkSmart-3L style. We then present the LinkSmart-3L architecture that follows a three layered self-management architectural model (Section 4) and we give design details of each layer. We show how to develop a self-managed application in B using a simple case study. Comprehensive evaluations regarding runtime and development time qualities of our approach are presented in Section 5. We discuss related work in Section 6. Conclusions and future work end this paper in Section 7.

2. Software architectures for self-management

The task of architecting self-managing systems can be approached from several conceptual perspectives, which lead to different architectures.

In one approach, inspiration is sought from nature to build systems with no explicit locus of control. An example is division of labor in a group of robots inspired by the decentralized organization of an ant colony [11]. This approach has particularly been applied to autonomic communications [12], but systems based on it are arguably difficult to engineer [13].

Another conceptual approach is to leverage traditional Artificial Intelligence (AI) by relying on explicit representation of plans as a basis for action. Although this might lead to a system with a central control unit (e.g., an AI planner as in [14]), control can also be distributed, e.g., by using Belief-Desire-Intention (BDI) agents [15].

A third conceptual approach is inspired by the model used to engineer control systems [16]. While it is orthogonal to the first two approaches in that it assumes nothing about representation of plans, it does require a certain level of centralized control in so far as *measured* system output must be compared with the *desired* output in order to compute the control measure needed to align the two.

Many early architectures for autonomous robotics [17] follow the "sense-plan-act" architecture, which suffers from difficulties in maintaining precise "world models". Brook's subsumption architecture [18] avoided this by following the slogan "the world is its own best model" and relying extensively on sensors, but it did not provide sufficient means to handle complexity. The three layer (3L) architecture described by Gat [17] combines ideas from both, in that the stateless and low complexity online control algorithms reside in the bottom layer, while the top layer employs traditional modeling and high-complexity planning algorithms, with the middle layer acting as interface between the two. This architecture has become a de facto standard architecture in autonomous robotics.

The self-management approach of LinkSmart follows the three layered reference model proposed by Kramer and Magee [6] which is adapted from Gat's 3L architecture. An overview is shown in Fig. 1.