Future Generation Computer Systems 56 (2016) 140–152

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Concurrency of self-*in autonomic systems

Phan Cong Vinh

Faculty of Information Technology, Nguyen Tat Thanh University, 300A Nguyen Tat Thanh street, Ward 13, District 4, Ho Chi Minh City, Viet Nam

ARTICLE INFO

Article history:
Received 30 March 2015
Accepted 29 April 2015
Available online 21 May 2015

Keywords:
Autonomic computing
Autonomic systems
Self-*
Categorical approach
Concurrency

ABSTRACT

New computing systems are currently at crucial point in their evolution: autonomic systems (ASs), which are inspired by the human autonomic nervous system. Autonomic computing (AC) is characterized by self-*such as self-configuration, self-healing, self-optimization, self-protection and more which run simultaneously in ASs. Hence, self-*is a form of concurrent processing in ASs. Taking advantage of categorical structures we establish, in this paper, a firm formal basis for specifying concurrency of self-*in ASs.

1. Introduction

Autonomic computing (AC) essentially refers to the self-managing characteristics of distributed computing resources, adapting to unpredictable changes while hiding intrinsic complexity to users. Thus, AC is characterized by self-*such as self-configuration, self-healing, self-optimization, self-protection and more which run simultaneously in ASs [1–3]. In a sense, self-*is a form of concurrent processing in ASs.

Although the purpose and thus the behavior of ASs vary from system to system, every autonomic system (AS) is able to exhibit a minimum set of properties to achieve its purpose including aware, adaptive and automatic properties [1–4].

Aware property means that an AS must be able to monitor its operational context as well as its internal state in order to be able to assess if its current operation serves its purpose. Awareness will control adaptation of its operational behavior in response to context or state changes.

Adaptive property means that an AS must be able to change its operation (i.e., its configuration, state and functions). This will allow the system to cope with temporal and spatial changes in its operational context either long term (environment customization/optimization) or short term (exceptional conditions such as malicious attacks, faults and so on).

Automatic property essentially means being able to self-control its internal functions and operations. As such, an AS must be self-contained and able to start-up and operate without any manual intervention or external help. Again, the knowledge required to bootstrap the system must be inherent to the system.

ASs enable to execute concurrently self-*. Hence, one of major challenges is how to support self-*concurrency in the face of changing user needs, heterogeneous environments, and computation objectives. In other words, how can an AS perform concurrently self-*in accordance with context-awareness and goal-driven computational mechanisms?

Dealing with this grand challenge of ASs requires a well-founded specification and in-depth analysis on concurrency of self-*. With this aim, we develop a firm formal approach in which the notion of self-*concurrency is formulated and specified in categorical structures known as a categorical approach of autonomic computing.

The major contribution of the paper is to propose some applied categorical structures of self-*concurrency for autonomic computing in ASs which, to the best of our knowledge, have never been tackled thoroughly in this emerging field. In fact, by this approach, category theory is applied to deal in an abstract way with algebraic objects (i.e., self-*) and concurrent relationships between them for specifying self-*concurrency in ASs.

From an applicative aspect of the approach, moreover, formalizing self-*concurrency in ASs is a categorical specification of middleware that can be used to develop implementations for autonomic computing. The formalization describes what the middleware system should

E-mail address: pcvinh@ntt.edu.vn.

http://dx.doi.org/10.1016/j.future.2015.04.017
0167-739X/

do, not how such middleware system should do it. For autonomic computing, a middleware system design cannot ever be developed in isolation, but only with respect to a given specification. Therefore, this formalization is applicable to the following:

Given a specification, verification techniques (such as model checking, equivalence checking, satisfiability checking, theorem proving, constraints solving, and more) can be used to justify that a candidate middleware system design is correct with respect to the specification. This has the benefit that incorrect candidate middleware system designs can be revised before any further endeavor is made to implement the design actually.

Another application is to use provably stepwise refinement to transform, in verifiable stages, a specification into a middleware system design, and finally into an actual implementation, that is correct by construction.

Apparently, it is able to validate a specification by proving theorems about properties that the specification is expected to expose. If correct, these theorems increase our knowledge over the specification and its relationship with the underlying problem domain of ASs. Otherwise, the specification needs to be adapted to reflect better our insight over ASs including implementing the specification.

Furthermore, there are the mutual applications of AC and formal verification. On the one hand, AC provides suitable abstractions and platforms to support the construction of verification tools. On the other hand, the verification techniques support the design and analysis of the AC processes needed to reach more dependable robust and predictable behavior of ASs.

2. Outline

The paper is a reference material for readers who already have a basic understanding of ASs and are now ready to know the novel approach for formalizing concurrency of self-*in ASs using categorical language.

Formalization is presented in a straightforward fashion by discussing in detail the necessary components and briefly touching on the more advanced components. Several notes explaining how to use the formal aspects, including justifications needed in order to achieve the particular results, are presented.

We attempt to make the presentation as self-contained as possible, although familiarity with the notions of self-*and concurrency in ASs is assumed. Acquaintance with the algebra and the associated notion of categorical language is useful for recognizing the results, but is almost everywhere not strictly necessary.

The rest of this paper is organized as follows: Section 3 includes some major work related directly to the content of the paper. In Section 4, AC is described as self-*. Section 5 presents self-*concurrency and concurrent composition of self-*in ASs. In Section 6, we construct a category of self-*facets. Section 7 presents commutative diagrams of self-*facets. Section 8 specifies transformation on ASs. In Section 9, we construct category of ASs together with categorical aspects of transformation. Section 10 examines category of ASs as extensional monoidal category. Section 11 presents categorical characteristic of ASs. In Section 12, we briefly discuss a direction of further developments in the future. A short summary is given in Section 13. Finally, in appendix, we recall some major concepts of category theory used in this paper.

3. Related work

The topic of AC has seen a number of developments through various research investigations following the IBM initiative such as AC paradigm in [5–9]; different approaches and infrastructures in [10–14] for enabling autonomic behaviors [15–18]; core enabling systems, technologies, and services in [19–24] to support the realization of self-*properties in autonomic systems and applications; specific realizations of self-*properties in autonomic systems and applications in [25–31]; architectures and modeling strategies of autonomic networks in [32–34]; middleware and service infrastructure as facilitators of autonomic communications in [35–37]; approaches in [38–40] to equipping current networks with autonomic functionality for migrating this type of networks to autonomic networks.

Moreover, AC has also been intensely studied by various areas of engineering including artificial intelligence, control systems and human orientated systems [4,41–43]. Autonomic computing has been set as an important requirement for systems devised to work in new generation global networked and distributed environments like wireless networks, P2P networks, Web systems, multi-agent systems, grids, and so on [44–48]. Such systems pose new challenges for the development and application of autonomic computing techniques, due to their special characteristics including: *nondeterminism, context-awareness and goal- and inference-driven adaptability* [1–4].

Finally, the choice of the underlying formalization requires a close look at models for AC. Hence, our interest centers on formal approach to AC taking advantage of category theory [1]. In fact, categories were first described by Samuel Eilenberg and Saunders Mac Lane in 1945 [49], but have since grown substantially to become a branch of modern mathematics. Category theory spreads its influence over the development of both mathematics and theoretical computer science. The categorical structures themselves are still the subject of active research, including work to increase their range of practical applicability.

4. Autonomic computing as self-*

Autonomic computing (AC) imitates and simulates the natural intelligence possessed by the human autonomic nervous system using generic computers. This indicates that the nature of software in AC is the simulation and embodiment of human behaviors, and the extension of human capability, reachability, persistency, memory, and information processing speed [50]. AC was first proposed by IBM in 2001 where it is defined as

> "*Autonomic computing is an approach to self-managed computing systems with a minimum of human interference. The term derives from the body's autonomic nervous system, which controls key functions without conscious awareness or involvement*" [51].

AC is generally described as self-*. Formally, let self-*be the set of self-_'s. Each self-_ to be an element in self-*is called a *self-*facet*. That is,

$$\text{self-}^* = \{\text{self-}\_ \mid \text{self-}\_ \text{ is a self-}^* \text{ facet}\}. \tag{1}$$

We see that self-CHOP is composed of four self-*facets of self-configuration, self-healing, self-optimization and self-protection. Hence, self-CHOP is a subset of self-*. That is, self-CHOP $=$ {self-configuration, self-healing, self-optimization, self-protection} $\subset$ self-*. Every