Chapter 1

Formal Methods for the Development and Verification of Autonomic IT Systems

Radu Calinescu
Aston University, UK

Shinji Kikuchi
Fujitsu Laboratories Limited, Japan

Marta Kwiatkowska
Oxford University Computing Laboratory, UK

ABSTRACT

This chapter explores ways in which rigorous mathematical techniques, termed formal methods, can be employed to improve the predictability and dependability of autonomic computing. Model checking, formal specification, and quantitative verification are presented in the contexts of conflict detection in autonomic computing policies, and of implementation of goal and utility-function policies in autonomic IT systems, respectively. Each of these techniques is illustrated using a detailed case study, and analysed to establish its merits and limitations. The analysis is then used as a basis for discussing the challenges and opportunities of this endeavour to transition the development of autonomic IT systems from the current practice of using ad-hoc methods and heuristic towards a more principled approach.

INTRODUCTION

The development of IT systems with self-managing capabilities—termed autonomic computing—is a relatively young area of research (Kephart & Chess, 2003; Murch, 2004). Now past the period of initial hype characteristic of any major new paradigm in computer science, autonomic computing looks set to become an established approach to addressing the continual increase in the scale and complexity of today’s IT systems. There are numerous indicators of this trend, including the
emergence of generic development platforms for autonomic computing (Calinescu, 2009a; Garlan, Schmerl, & Cheng, 2009; Twidle, Dulay, Lupu, & Sloman, 2009; Vassev & Mokhov, 2009) and the use of autonomic IT systems across a wide range of application domains (Huebscher & McCann, 2008; Parashar & Hariri, 2006).

While this healthy pace of progress is well in line with the ambitious plan put forward by the autonomic computing manifesto (IBM Corporation, 2001), one concern remains. The vast majority of autonomic IT systems – whether under test in research labs or deployed in production – implement the high-level objectives that guide their operation using heuristics derived from and validated through a combination of experimentation, simulation and testing. As it is well known from more established areas of computer science, this is insufficient for developing IT systems that are highly predictable and dependable. Yet, autonomic IT systems are required to excel in precisely these characteristics (Dai, 2005; Sterritt, 2003; Sterritt & Bustard, 2003).

This chapter proposes that the major problem identified above is addressed by using rigorous mathematical techniques termed formal methods (Boca, Bowen, & Siddiqi, 2009). Building on the authors’ previous work in the area (Calinescu & Kwiatkowska, 2009a, 2009b; Kikuchi, Tsuchiya, Adachi, & Katsuyama, 2007), the chapter explores ways in which formal methods can help overcome the discrepancy between what autonomic IT systems can deliver in terms of predictability and dependability, and what is expected of them. The next three sections look in turn at several aspects of autonomic computing that can benefit from the use of existing or enhanced techniques from the area of formal methods.

First, next section describes the use of model checking (E. M. Clarke, Grumberg, & Peled, 2000) to detect policy conflicts in autonomic computing systems. Given the potentially significant damage that conflicting policies can cause to autonomic systems, it is critical to ensure that policies expressing different system objectives do not interfere with each other. This section explains why model checking represents a better-suited technique for conflict detection than alternative approaches such as testing or simulation.

This is followed by two sections that present formal approaches to implementing two important classes of autonomic computing policies: goal policies and utility-function policies. Goal policies describe constraints that an autonomic system needs to observe at all times. Together, these constraints provide a formal specification (Abrial, 1996; Woodcock & Davies, 1996) for the system, and a technique termed model synthesis (Jackson, 2006) can be employed to update the system configuration in response to changes in its environment in ways that comply with this specification. This approach is presented in detail in the chapter. Utility-function policies provide a quantitative measure of the degree to which an autonomic system achieves its high-level objective, and request that the system configuration is adjusted automatically so that maximum utility is obtained in the presence of changes in the system state, workload and environment. The chapter describes the use of quantitative verification (Kwiatkowska, 2007) to implement this type of autonomic computing policy.

The final section concludes the chapter with a brief summary, and discusses the directions in which the field of formal methods for autonomic computing is headed. This discussion includes overviews of two emerging platforms that aim to employ formal techniques for the end-to-end development of autonomic IT systems. One development platform involves the model-driven, automatic generation of many of the components of an autonomic IT system starting from a quantitative model that defines the behaviour of its components. The other platform requires that the development process starts with the definition of a formal specification of the autonomic system, and generates its parts in a series of automated development steps.