Chapter 2
Theoretical and Practical Aspects of Developing Autonomic Systems with ASSL

Emil Vassev
Lero at University College Dublin, Ireland

ABSTRACT

ASSL (Autonomic System Specification Language) is an initiative for self-management of complex systems whereby the problem of formal specification, validation, and code generation of autonomic systems is approached within a framework. Being a formal method dedicated to autonomic computing, ASSL helps developers with problem formation, system design, system analysis and evaluation, and system implementation. The framework provides a powerful formal notation and suitable mature tool support that allow ASSL specifications to be edited and validated and Java code to be generated from any valid specification. As part of the framework’s proof-of-concept strategy, ASSL has been used to make a variety of existing and prospective systems autonomic. This entry presents the ASSL formal specification model and tools. Moreover, two case studies are presented to reveal practical aspects of using ASSL for the development of prototypes of prospective space exploration systems incorporating autonomic features.

1. INTRODUCTION

It is widely recognized that high software complexity is a source of software failures that may have a disastrous effect, especially in safety-critical systems. This makes complexity one of the biggest challenges software producers are facing today. To respond to this threat, many initiatives such as Autonomic Computing (AC) (IBM Corporation, 2006), (Horn, 2001), (Kephart & Chess, 2003) have been started to deal with complexity in contemporary software systems. AC has emerged as a paradigm and research field tackling the development of complex large-scale systems by transforming them into special self-managing autonomic systems (ASs). Conceptually, ASs are intrinsically intended to reduce complexity through automation by applying principles of self-regulation from biology. In 2001, IBM Research introduced the term autonomic computing to draw...
an analogy between the computer systems and the human body’s Autonomic Nervous System (Horn, 2001). The idea behind this is that computer systems must manage themselves, as the human body does, or they risk being crushed under their own complexity.

Although AC has recently inspired a tremendous number of initiatives for self-management of complex systems (note that company like IBM, Microsoft, Oracle and HP started AC-based programs), it still is not pervasive across the IT industry. The problem is that ASs cannot be developed successfully with the traditional software-development approaches, because these pay scant attention to many of the features of an AS and the very complexity inherent in many systems that lend themselves well to AC can often cause difficulty in designing that same ASs. Therefore, in order to avoid the threat of exploding complexity, we need to reconsider fundamentally the way we build AC software. However, although it is clear that new development approaches are needed to make AC take hold throughout the industry, the vast majority of IT companies is reluctant to invest in such development approaches. This is due mainly to the fact that traditional software development techniques (e.g., object-oriented programming) have proven their efficiency in practice as reliable approaches that guarantee low risk and high rate of return of investments.

This entry presents an approach towards building ASs with the Autonomic System Specification Language (ASSL) (Vassev, 2008), (Vassev, 2009), (Vassev & Hinchey, 2009), a formal method dedicated to AC. Conceptually, ASSL have been intended to help developers make the real transition to an “autonomic culture” by connecting AC with formal methods. Despite being a subject of controversy for decades, over the last decade (Bowen & Hinchey, 2004), formal methods have regained confidence and have proven to be extremely useful in the development of reliable software for safety-critical systems such as modern avionics systems and nuclear plants (Amey, 2002), (Beveniste et al., 2003), where software failures easily emerge to safety hazards. The provided high level of abstraction and the formal treatment of the problems have motivated this success. ASSL builds on this by adding an AC domain-specific formal notation and tools for AS specification, validation, and code generation.

This entry presents ASSL from two perspectives – theoretical and practical. The entry introduces first both the ASSL specification model and tools as a theoretical background needed for understanding the following section where two case studies are presented. These case studies describe practical experience of using ASSL in the development of autonomic features for AC prototypes of space exploration missions based on swarm intelligence and prospective autonomic Voyager-like missions.

2. ASSL

Initially developed at Concordia University, Montreal, Canada, theAutonomic System Specification Language (ASSL) (Vassev, 2008), (Vassev, 2009), (Vassev & Hinchey, 2009) is a domain-specific formal tool whereby the problem of formal specification, validation, and code generation of ASs is approached within a framework. Being a formal method dedicated to AC, ASSL helps AC researchers with problem formation, system design, system analysis and evaluation, and system implementation. The framework provides a powerful formal notation and suitable mature tool support that allow ASSL specifications to be edited and validated and Java code to be generated from any valid specification.