Chapter 4

Formal Stepwise Development of Scalable and Reliable Multiagent Systems

Denis Grotsev
Kazakh National University, Kazakhstan

Alexei Iliasov
Newcastle University, UK

Alexander Romanovsky
Newcastle University, UK

ABSTRACT

This chapter considers the coordination aspect of large-scale dynamically-reconfigurable multi-agent systems in which agents cooperate to achieve a common goal. The agents reside on distributed nodes and collectively represent a distributed system capable of executing tasks that cannot be effectively executed by an individual node. The two key requirements to be met when designing such a system are scalability and reliability. Scalability ensures that a large number of agents can participate in computation without overwhelming the system management facilities and thus allows agents to join and leave the system without affecting its performance. Meeting the reliability requirement guarantees that the system has enough redundancy to transparently tolerate a number of node crashes and agent failures, and is therefore free from single points of failures. The Event B formal method is used to validate the design formally and to ensure system scalability and reliability.

INTRODUCTION

The variety and ubiquity of modern computational devices raise the problem (and create the opportunity) of utilizing and orchestrating their processing capabilities within an integral approach which would ensure that the system using them is scalable and reliable. In our work we refer to such computational resources as system nodes. Our solution is based on the universal principle of dealing with complexity by introducing a particular level of abstraction that allows us to
focus on achieving certain system properties. In particular, our aim is to demonstrate how properties of solutions can be formally reasoned about at various levels of abstraction.

Examples of such abstraction levels that allow developers to support integration of many nodes can be found in peer networks (BitTorrent), cloud platforms (Google App Engine) and distributed file systems. Such systems are designed to achieve required system properties. The most important one is scalability, which ensures a linear or almost linear increase in system performance with the increase in the number of nodes. Another critical system property is reliability, which allows clients to see the system as if it was realised on a single fault-free node. Due to the nature of these systems, node failures are not uncommon and should not normally lead to an overall failure or require explicit actions at the level of applications deployed on the system. In other words, within certain limits, node failures should be masked. This is typically achieved through node and application redundancy, whereby the same activity is executed on several nodes. Crucially, in case of node failures the system is automatically reconfigured.

Our work proposes a formal step-wise development model which allows us to prove the scalability and reliability of the solutions using the Event-B method. As part of our rigorous system development, we demonstrate how to formally specify a reconfiguration of the system topology performed as a response to a change in the number of nodes. We apply a multiagent approach in which a special programming unit, an agent, resides on every node and reacts to node failures and system changes in such a way as to automatically reconfigure the system to an acceptable state.

BACKGROUND: EVENT-B

Event-B (Abrial, 2010) is a state-based formal method inherited from Classical B (Abrial, 1996). It is an approach for realising industrial-scale developments of highly dependable software. The method has been successfully used in the development of several real-life applications. An Event-B development starts from creating a formal system specification. The basic idea underlying stepwise development in Event-B is to design the system implementation gradually, by a number of correctness preserving steps called refinements.

The unit of a development is a model. An Event-B model is made of the static part, called a context, and the dynamic part, called a machine. A context defines constants \(c\), sets (user defined types) \(s\), and declares their properties in axioms \(P\) and theorems \(T\):

\[
\begin{align*}
\text{context } C \\
\text{sets } s \\
\text{constants } c \\
\text{axioms } P(c, s) \\
\text{theorems } T(c, s)
\end{align*}
\]

A machine is described by a collection of variables \(v\), invariants \(I(c, s, v)\), an initialisation event \(RI(c, s, v')\) and a set of machine events \(E\):

\[
\begin{align*}
\text{machine } M \\
\text{sees } C \\
\text{variables } v \\
\text{invariants } I(c, s, v) \\
\text{events } E
\end{align*}
\]

In the above, construct \(\text{sees} C\) makes context \(C\) declarations available to machine \(M\). The model invariants specify safe model states and also define variable types. An event is a named entity made of a guard predicate and a list of actions and has the following syntax:

\[
\begin{align*}
\text{name } \text{= any p where } G(c, s, p, v) \text{ then } R(c, s, p, v, v')
\end{align*}
\]

where \(p\) is a vector of parameters, \(G(c, s, p, v)\) is a guard and \(R(c, s, p, v, v')\) is a list of actions.
Related Content

Shifting Legitimation along Information Infrastructures Growth: Local Social Embeddedness, Global Organizational Fields, and Full Scale Coverage
www.igi-global.com/chapter/shifting-legitimation-along-information-infrastructures/62546?camid=4v1a

The State of Development of CSE
www.igi-global.com/chapter/state-development-cse/60372?camid=4v1a

OpenLaszlo: Developing Open Rich Internet Applications for Mobile Learning Environments
www.igi-global.com/chapter/openlaszlo-developing-open-rich-internet/62496?camid=4v1a

Secure Key Establishment in Wireless Sensor Networks
www.igi-global.com/chapter/secure-key-establishment-in-wireless-sensor-networks/203539?camid=4v1a