Autonomic Self-Healing Systems in a Cross-Product IT Environment

Gary Dudley, Neeraj Joshi, David M. Ogle, Balan Subramanian, Brad B. Topol
IBM Corp.
{dudleyg,jneeraj,daveogle,bsubram,btopol}@us.ibm.com

Abstract

This paper presents the architecture and implementation of a prototype system developed to investigate fixing real-world problems with self-healing systems. Case studies performed to evaluate the efficacy of this approach are described along with the issues encountered and lessons learned.

1. Introduction

Diagnosing and fixing problems in information-technology (IT) systems is labor intensive. Moreover, problem-determination (PD) tools are unable to view an overall system as they are focused on individual products and produce data in product-specific formats.

This paper presents the architecture and implementation of a prototype system developed to investigate fixing real-world problems with self-healing systems in multi-product IT environments. This paper also discusses case studies performed to evaluate the efficacy of this approach, highlighting the issues encountered and lessons learned.

2. Architecture/Methodology

The autonomic-computing architecture [1] [2] defines an atomic Autonomic Element. The structure of an Autonomic Element includes an autonomic manager and managed elements [2]. The manager has monitor, analyze, plan, and execute elements collectively known as the MAPE loop. The monitor element collects, aggregates, and filters events from the managed elements. The analyze element compares these artifacts against a symptom database. The analyze element outputs an indication of any problematic patterns found and a set of possible solutions. The plan element, based on policy data, selects one of the solutions. The execute element carries out the actions for that solution.

A managed element is an entity that exists in the runtime environment of an IT system and has sensors and effectors. A sensor exposes information about the state of a managed element, and an effector changes the state of a managed element.

Managed elements produce artifacts that describe significant state-change events. A common schema for artifacts is required to facilitate PD. Such a schema, called the common base event [3], was used for the prototype. Certain events are categorized as pre-defined situations (e.g., memory-allocation failure). Situations are formatted as enhanced common base events specifying a category.

3. Implementation Technologies

The following products and Operating System (OS) technologies were used in the prototype system: AIX*, Cisco Catalyst 3550 Series Switch**, Generic Log Adapter for Autonomic Computing [4] [5], IBM DB2 Universal Database Version 8.1*, IBM Tivoli Monitoring (ITM) Version 5.1.1*, IBM WebSphere Studio Application Developer Integration Edition Version 5.0*, IBM WebSphere Application Server Version 5.0.2*, and Windows 2000 Professional**.

The prototype system makes use of a generic adapter for each managed element. An adapter is needed to convert existing log files into the canonical common-base-event format. The common base events are forwarded to ITM.

ITM provides a resource model (RM) for the managed elements. Incoming common base events are collected in a single uniform log file. The RM uses custom scripts to analyze these events in order to recognize known failures for which a set of actions can be proposed.

Web services are used for transporting common base events and for taking autonomic actions. The web service for transporting common base events improves performance by grouping multiple common base events. This web service also supports quality-of-service mechanisms such as immediately sending all buffered common base events when a priority event is received.

4. Case Studies

Three case studies were selected as representative of real-world customer problems. The objective of the DB2 deadlock case study is to demonstrate an autonomic solution that can detect...
failures from WebSphere Application Server (WAS) and DB2 log files, correlate these failures to recognize a problem, and instigate actions to correct the problem. The objective of the router-link failure case study is to demonstrate that hardware events can be monitored and hardware problems mitigated. Finally, the objective of the AIX low page memory case study is to demonstrate that operating-system logs can be monitored and used to diagnose problems. Details of the DB2 deadlock scenario are described in the following paragraphs.

The Trade3 benchmark application is installed on a Windows 2000 machine and runs as a WAS application with an associated local database. Two generic adapters are installed on this machine (one for the WAS activity log and the other for the DB2 db2diag log). ITM is installed on a second machine.

Parameters associated with the DB2 database lock size are set to low values to facilitate error introduction (simulating poor configuration choices by a database administrator.) A web-performance tool is used to stress the Trade3 application and induce a deadlock; the deadlock is identified by correlated situations from the WAS and DB2 logs. The log files are analyzed by the generic adapters, the exceptions are identified as situations, and common base events are passed to ITM.

Two actions are proposed to fix the deadlock problem: increase the DB2 lock size and restart the application. A policy engine is checked to see if the actions may be initiated, and the actions are started via web services. Access to the application is restored with minimal downtime.

5. Lessons Learned

During the development of the case studies, the following lessons were learned:

- Using system resources to administer autonomic fixes may cause resource-management problems. For example, making a web-services connection to a machine with a stress-related problem is often difficult without mechanisms to ensure that the connection has adequate priority.
- Autonomic actions can temporarily remove mission-critical applications from service. Therefore, it is necessary that customers define policies regarding what actions can be taken, when such actions can take place, and how long the system can wait for the actions to have the desired effect.
- Study of failure mechanisms can be used to designate product events as failure situations.

Combinations of situations from one or more products can be used to diagnose problems.
- The performance of autonomic systems can be enhanced by reducing the load on the autonomic manager.
- Mechanisms to handle issues with event correlation are required.

6. Conclusions and Future Work

We have presented the architecture and implementation of a prototype system used to investigate problem determination and self-healing for cross-product IT systems. We demonstrate, through the use of case studies how the architecture and mechanisms presented in this paper can be used to convert operating-system logs, hardware logs, and software-product logs to a common canonical event schema; to categorize these events as pre-defined situations; to characterize combinations of events as a fixable problem; and to prescribe a set of actions to fix the problem, initiate the prescribed actions, and monitor the resolution of the problem. Future work will focus on symptom database development and advanced event correlation technologies.

*Trademark or registered trademark of the International Business Machines Corporation.
**Trademark or registered trademark of Microsoft Corporation, Sun Microsystems, or Cisco Systems.

7. References