Autonomic Computing: 
Characteristics of 
Self-Managing IT Systems

A Special Report From:

Grid Technology Partners
Autonomic Computing:
Characteristics of Self-Managing IT Systems

TABLE OF CONTENTS

1.0 Executive Summary
2.0 Autonomic Computing Definition
   2.1 Characteristics of an AC System
      2.1.1 Self Monitoring
      2.1.2 Self Configuration
      2.1.3 Self Optimization
      2.1.4 Self Healing

About the Author
Ahmar Abbas is Managing Director of Grid Technology Partners a vanguard market research and consulting
firm focusing on global utility and grid computing opportunities. Grid Technology Partners' clients include
Global 100 companies in North America, Asia and Europe. Prior to founding GTP, Abbas was the Director of
Product Marketing at ONI Systems, a leading provider of next-generation optical networking solutions for
regional and metropolitan markets. Prior to that, he was Director of Business Development at San Jose-based
Zaffire. Abbas also served as Senior Manager for Asia-Pacific and Latin America Product Management at
UUNET, a Worldcom company. He also served as Manager of the Network and Systems Management group at
UUNET and was on the technical staff at Salomon Brothers.
as the forthcoming book Grid Computing - A Practical Guide to Technology and Applications (Charles River
Media, Summer 2003).

About This Report
This special report was excerpted from the Autonomic Computing Report 2003. For information on the full
report, please contact Ahmar Abbas at aabbas@gridpartners.com.

All rights reserved. Copyright © Grid Technology Partners, 2003.
1.0 Executive Summary:

Over the course of the last two decades, as technology has proliferated into every aspect of corporate life, it has become increasingly difficult to determine whether it has brought along any lasting and across the board productivity gains.

Yet, technology deployments have continued unabated. A typical data center today contains a wide array of applications, computers, storage devices and networking equipment. The management of this infrastructure has become extremely difficult – with many inefficient human touch points. The costs associated with managing and maintaining this infrastructure has also increased tremendously. But the worst is yet to come. It is expected if data centers continue to grow at the current pace, they will become extremely difficult and costly to manage. Large vendors such as IBM, HP and SUN have
all started initiatives at tackling the problem of managing complex data centers. The emphasis has been on creating software and hardware components that are self-managing and self healing. In other, words, using technology to automate technology. In this report we look at the motivation, technology and various vendor initiatives to reduce IT infrastructure complexity.

2.0 Autonomic Computing:

Autonomic Computing (AC) is an approach to managing a range of different aspects of computer systems, based on the premise that self-managing systems are required to deal with an increase in system complexity, and the need for a broader audience for such systems. This is necessary because the connectivity and complexity of systems has increased so significantly, that finding ways to optimise interactions between such systems, and managing individual components has become a daunting task. With an increase in complexity, it is also likely that system administrators may find it difficult to see the “big picture”, and may be unable to take advantage of changes in individual components. Furthermore, system administrators may often prefer to utilise their prior experience and techniques for managing systems – and may not be effectively utilising improvements in individual components. Commercially, the AC vision is a significant change in emphasis in enabling systems to automatically “adjust” their behaviour based on conditions in which they operate. From a research perspective, it brings together a number of different areas in artificial intelligence, self-organisation/emergent behaviour, distributed computing, and dependability. An AC system is required to be self-managing, with a “minimum of human interference” [18], and is seen by IBM to be a Grand Challenge for IT infrastructure. Paul Horn [16] provides a convincing case of why such changes are necessary, and defines this emphasis as a necessary measure to deal with a variety of devices that must be managed within a corporate IT environment (such as desktops, workstations and PCs, Personal Digital Assistants, cell phones, pagers etc). According to him,
“annual compound growth of these devices is expected to be 38 percent over the next three years”.

![Autonomic Computing Process](image)

**Figure 2: Autonomic Computing Process**

The AC vision runs throughout various IBM hardware and software products – and is very much intended to be a unifying concept across the different systems it provides – an “end-to-end infrastructure” as IBM calls it. These range from IBM’s eServer and Storage, to specialised software such as Tivoli, Lotus, WebSphere and DB2 [19]. The AC concepts are intended to be expressed as high-level goals or policies that must be undertaken by a combination of infrastructure systems working collectively. IBM even undermines some of its own earlier efforts in providing automation such as the System Managed Storage effort, by stating that automating part of the system rather than the whole does not really help.
The IBM AC vision is based on Microsoft’s Joseph Barrera III, in his paper on Self-Tuning Systems Software [12], he outlines the activities necessary to construct such systems: (1) the expectations a user has of such a system – generally translated to the high-level policy being implemented using a particular system, (2) the measurements that must be made of the actual system in operation, (3) analysing these measurements, and comparing these with the requirements set out in the expectations, and (4) actions that need to be undertaken as a response – ranging from gathering of additional data to reconfiguring components of the system. A major emphasis in his paper is the need to provide separation between these characteristics, to enable system components to work together more effectively.

The need for AC is seen as being very urgent, for instance, it is estimated that 200 million information technology workers might be needed to support a billion people using computers at millions of businesses – connected via intranets, extranets, and the Internet [14]. Jim Gray of Microsoft [39] also agrees with the need to support autonomic computing, indicating that, “IBM’s autonomic goals are great; very similar to the goals that Microsoft has had for many years.” He characterises these goals for Microsoft as a “holistic approach to products”. According to Seth Grimes [39], many major platform vendors and academic research teams are attacking different aspects of this AC vision – which weaves together many significant technology initiatives, such as Web Services, Grid Computing, rule based systems, and agent technology – with an essential focus on “service-oriented” computing.

2.1 Characteristics of an AC System:

To realise the AC vision of building computing systems that can adjust to varying circumstances, and more efficiently handle the available workload – four generic principles are outlined as characterising such systems:

- **Self-configuring:** being able to modify interactions and behaviours based on changes in the environment
- **Self-healing:** being able to discover, diagnose and prevent disruptions
- **Self-optimizing**: being able to tune resource usage and improve workload balancing
- **Self-protecting**: being able to detect, identify and protect against failure and security attacks

To achieve these four principles, each component within a system that is autonomic must support additional intelligence – as part of the activities previously undertaken by some centralised system are now delegated to each individual component. Some of these four characteristics are provided – to a limited extent, in the IBM x,i,p,zSeries server products, as outlined on [27]. In order to achieve these, rather ambitious, objectives, the following essential features are necessary [20] (based on the IBM AC Manifesto [18]):

### 2.1.1 Self Monitoring

A system must be able to measure and monitor its own components. Since a system can exist at many levels, and comprise of a number of different elements, an autonomic system will need detailed knowledge of its components, current status, ultimate capacity, and connections to other systems that it influences. It will need to know the extent of its “owned” resources, and those it can borrow from other resource owners, and those that can be shared or should be isolated. Resource management and discovery therefore plays an important role in AC, as does the ability for a component (software or hardware) to know what resources it currently has, and how these are to be made available to others (an approach generally referred to as resource/service registration). Interactions between systems may be modulated through “Service Level Agreements” (SLAs), which outline what services a given component is to offer to others, and what it expects in return. SLAs go beyond the definition of communication protocols, as they also require such interactions to be monitored, and in some cases, for the SLAs themselves to be negotiated dynamically. For instance, a service may be able to reduce its performance demands on another, if not being able to meet its required performance means not having any results at all. In such cases, services are
able to adapt their behaviour to accept a low quality result from another service, rather than no result at all. Hence, the definition and adaptation of SLAs make interaction between services more agile and resilient to changes in the operating environment.

### 2.1.2 Self Configuration:

An AC system must be able to configure, and re-configure itself under varying (and sometimes “unpredictable”) conditions – either at boot time or at run time. System configuration or set-up must occur automatically, as well as dynamic adjustments to the configuration to best handle a changing operating environment. Such dynamic adjustments may also mean the addition of new hardware resources in response to management software or policies defined by a systems administrator. Currently, configuration of resources and services is undertaken using text files or configuration wizards. Deciding which parameters are significant and can be dynamically altered is a complex task, and very few existing systems achieve this. Self-configuration can also be supported through the recording of system activity – i.e. a collection of log files which record system runs/activity from previous use. Such log files may be used as a first step to support reconfiguration.

### 2.1.3 Self Optimization:

An AC system should be “goal” oriented, i.e. it should pro-actively look for opportunities to optimise its use. Existing work in pro-active computing [21] plays a significant role in this context. Pro-active systems will monitor events in their environment, and trigger pre-defined activities based on the detection of particular events. The complexity of planning undertaken by such a system determines how successfully it can exploit events that it detects. Greater complexity also impinges, however, on aspects such as reliability, computational cost, access time etc, and a balance is necessary to enable both pro-active and re-active behaviours from components.
2.1.4 Self Healing:

An AC must be able to recover from routine and extraordinary events that might cause some parts of it to malfunction. Such malfunctions can either be detected by the system, or they may be predicted to support automated re-configuration. Such fault tolerance may be supported through replication, or by looking for patterns of activity that indicate a likely failure in the future. Once again, recorded data from previous system activity may be used to support such predictions. Similarly, an AC must be able to detect, identify and protect itself again various types of security attacks. Generally access rights are specified based on a policy, and the adaptability of the system is based on how constrained such an access policy can be. Another aspect of fault tolerance is the graceful degradation of performance as parts of a system fail, and require the balancing of workload over the available (working) components. Such a load distribution can either be based on a static analysis of the system (using suitable monitoring techniques), or a predictive model which also accounts for future workload the system is likely to generate. There is unlikely to be a single load manager within an autonomic system, requiring each component to interact with its peers to share and negotiate the distribution of workload. Both failure management and security are subject to a risk management strategy that the system developers of AC should identify. Each component within the system should then reflect such a policy, and enable the operations of the components to be modified within the constraints of the policy. Risk management policies may be aimed at containing the effects of faults within defined limits, to enable a component to be made inactive, for instance, or its workload migrated to another component. An example is the ability of a resource to manage power and migrate workload when power falls below a threshold.
EXECUTIVE SUMMARY

SECTION I: CONTEXT & TRENDS

1.0 Introduction
1.1 Productivity Paradox
1.2 Return on Investments
1.3 Multi-Story Bureaucracy
1.4 Information Technology Straightjacket
1.5 Consolidation
   1.5.1 Server Consolidation
   1.5.2 Application Consolidation
   1.5.3 Storage Consolidation
   1.5.4 Network Consolidation
1.6 Virtualization
1.8 Outsourcing
   1.8.1 Application Service Providers
   1.8.2 Turnkey
   1.8.3 Utility Computing
1.7 Real Time Enterprise

SECTION II: TECHNOLOGY

2.0 Introduction
2.1 Anthropomorphising of Information Technology
2.2 Autonomic Computing Definition
2.3 Characteristics of an AC System
   2.3.1 Self Monitoring
   2.3.2 Self Configuration
   2.1.3 Self Optimization
   2.1.4 Self Healing
2.4 Component Technologies
   2.4.1 Grid Computing
   2.4.2 Web Services
   2.2.3 OGSA
   2.2.4 Jini and JXTA
   2.2.5 Agents
2.5 Role of Adaptivity in AC Systems
   2.5.1 Reconfiguration
   2.5.2 Fault Tolerance
2.6 Provenance
2.7 Service Levels and Quality of Service (QoS)
2.8 Emergent Behavior
2.9 Standards
2.10 Users of Technology
2.11 Applications Changes

SECTION III: MARKET

3.0 Autonomic Computing: A Market Perspective
3.1 Market Taxonomy
   3.1.1 Embedded vs. Management Domain Taxonomy
   3.1.2 AC Characteristics Based Taxonomy
3.2 Business Process Integration
3.3 “Grand” Initiatives
   3.3.1 IBM Autonomic Computing
      3.3.1.1 Project Eliza
      3.3.1.2 Project Oceano
      3.3.1.3 eWLM
      3.3.1.4 SMART
      3.3.1.5 ABLE
      3.3.1.6 Blue Gene
      3.3.1.7 Future
   3.3.2 HP Planetary-Scale Computing
3.3.2.1 Adaptive Internet Data Center
3.3.2.2 iShadow
3.3.2.3 Future
3.3.3 Sun N1
3.4 Vendor Discussion
  3.4.1 Ejasent
  3.4.2 Jareva
  3.4.3 Terraspring
  3.4.4 Others
3.5 State of the Market
3.6 Band Brands

REFERENCES