Autonomic Computing: Towards a Self-Healing System

Sharee S. Laster, B.S., Ayodeji O. Olatunji, B.S.,
ssl15@ece.msstate.edu, dolatunji82@yahoo.com
Department of Computer Engineering, Jackson State University

Abstract

The increasing complexity of computer systems results in systems which are prone to errors and disruptions creating major problems for the user. In order to reduce the total cost of ownership and to cope with the rapidly growing complexity of integrating and managing today's computer-based systems, autonomic computing was introduced by IBM in 2001. Autonomic computing is a computer environment that can detect and adjust its system automatically to manage itself without the assistance of any human interaction. This paper introduces autonomic computing, a methodology that focuses on the management of system complexity through the use of methods providing for self-learning and self-healing operating systems.

Introduction

For decades, the advancement of technology and science has mirrored the increase of complexity in many computer environments. Recently, the advancement of computer technology and science has not been increasing at the same pace as complexity in computer environments. This unbalance in advancement creates a major obstacle. The major obstacle that concerns researchers is centered on complexity. However, as the scale and complexity of these systems and applications grow, their development, configuration and management challenges are beginning to break current paradigms, overwhelm the capabilities of existing tools and methodologies, and rapidly render the system and applications, brittle, unmanageable, and insecure [22]. As complexity increases, computer environments are being impacted with more failures and downtown. Most frequently cited outages included [18]:

- For systems: operational error, user error, third-party software error, internally developed software problem, inadequate change control, lack of automated processes
- For networks: performance overload, peak load problems, insufficient bandwidth
- For database: out of disk space, log file full, performance overload
- For applications: application error, inadequate change control, operational error, non automated application exceptions

Researchers became very concerned about the current epidemic of computer environments being destroyed due to complexity. For this purpose, researchers were faced with the task of finding an alternative approach to complexity. After exploring multiple methodologies, researchers finally

"Proceedings of the Spring 2007 American Society for Engineering Education Illinois-Indiana Section Conference.

developed a solution to the problem of complexity. The methodology researchers used to overcome the barrier of complexity is called autonomic computing.

Autonomic computing is a computer environment that can detect and adjust its system automatically to heal itself without the assistance of any human interaction. Figure 1 displays the typical procedures implemented in various IT organizations. Therefore, the IT industry was in need for a computer system that would foresee the users need and allow users to focus more on completing their work tasks and less on troubleshooting their computer system. Autonomic computing was conceived to lessen the spiraling demands for skilled IT resources, reduce complexity, and to drive computing into a new era that may better exploit its potential to support higher order thinking and decision making [23]. Implementing an autonomic computing system will help companies eliminate the increasing costs of restoring hardware and software failures. This methodology could help IT professionals develop more reliable and dependable systems within computer environments. Consequently, autonomic computing will effectively prevent downtimes and system failures. In addition to less downtimes and system failure, the production rate for computer environments controlled by autonomic systems will increase dramatically. For that reason, autonomic computing is emerging significantly in the IT industry.

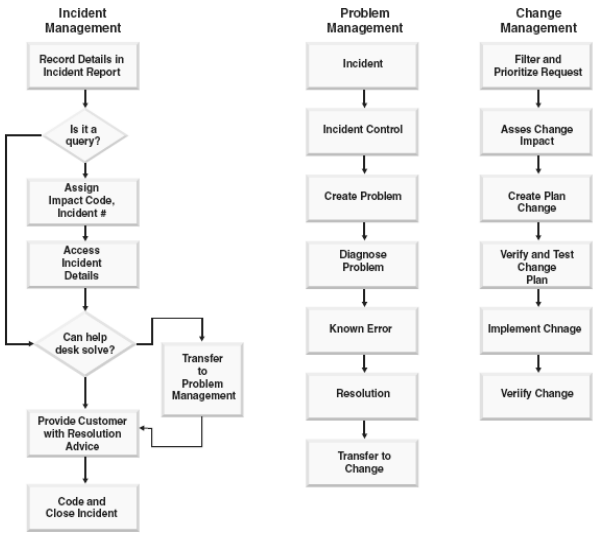


Figure 1: Typical I/T process [5]

History of Autonomic Computing

"It's time to design and build computing systems capable of running themselves, adjusting to varying circumstances, and preparing their resources to handle most effectively the workloads we put upon them. These autonomic systems must anticipate needs and allow users to concentrate on what they want to accomplish rather than figuring how to rig the computing systems to get them there."[6] (IBM's vision of autonomic computing, launched by IBM's senior vice president of research, Paul Horn). With the growth of computer industry, notable examples with highly efficient networking hardware and powerful CPUs, autonomic computing has become a means to cope with the rapidly growing complexity of integrating, managing, and operating computer systems [22]. Any

computer system should possess the ability of being effective and useful from the launched phase regardless of changes in the environment.

The name autonomic insinuates that it is a metaphor based on biology. The biological concept in which autonomic computing originated is the human body's autonomic nervous system. The autonomic nervous system consists of sensory neurons and motor neurons that run between the central nervous system (especially the hypothalamus and medulla oblongata) and various internal organs as illustrated in figure 2b. The autonomic nervous system tells your heart how fast to beat, check your blood's sugar and oxygen levels, and controls your pupils so the right amount of light reaches your eyes as you read these words. It monitors your temperature and adjusts your blood flow and skin functions to keep it at 98.6°F. It controls the digestion of you food and your reaction to stress-it can even make you hair stand on end if you're sufficiently frightened. It carries out these functions across a wide range of external conditions, always maintaining a steady internal state called homeostasis while readying your body for the task at hand. But most significantly, it does all this without any conscious recognition or effort from the human. Each Level maintains a measure of independence while contributing to a higher level of organization. The autonomic nervous system is divided into two subdivisions called the sympathetic nervous system and the parasympathetic nervous systems as shown in Figure 2a. The sympathetic nervous system and the parasympathetic nervous systems have parallel effects. The sympathetic nervous systems have fight and flight responses while the parasympathetic nervous system have the rest and digest response. For example, Table 1 exemplifies the parallelism of the two nervous systems. The aim of using this metaphor is to express the vision to enable something similar to be achieved in computing, in other words, to create the self-management of a substantial amount of computing functions to relieve users of low-level management activities, allowing them to place emphases on the higher-level concerns of running their business, their experiments or their entertainment [25].

Figure 2(a): Autonomic Nervous System [26]

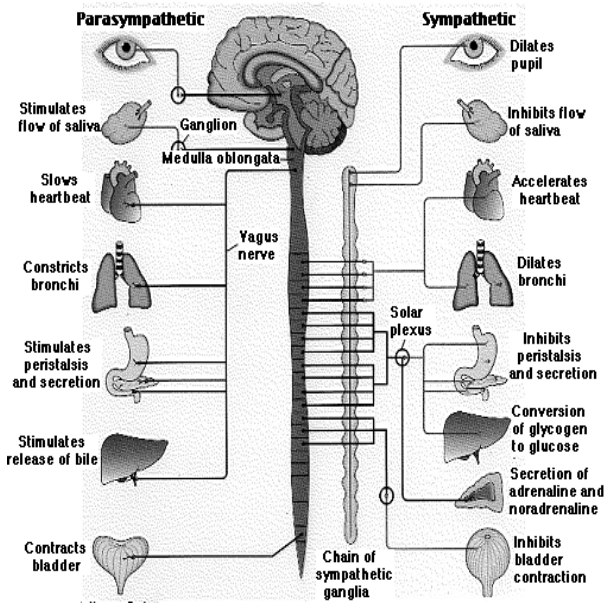


Figure 2(b): Sensory neurons & motor neurons [26]

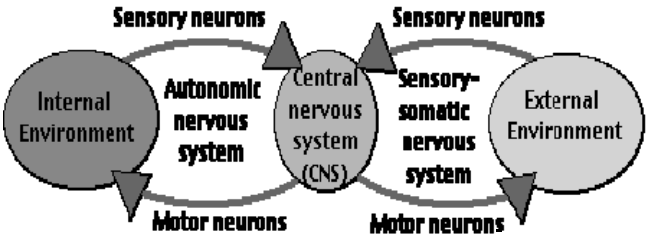


Table 1: Parallelisms of the Sympathetic Nervous System and the Parasympathetic Nervous System of the body's organs

Organ	Sympathetic Nervous System	Parasympathetic Nervous System
Eye (Iris)	Pupils dilate	Pupil constrict
Bladder	Walls relax	Walls constrict
Heart	Heart rate increase	Heart rate decrease
Kidney	Decrease urine secretion	Increase urine secretion
Lung	Bronchial muscle dilate	Bronchial Muscle contract
Stomach	Decrease secretion	Increase secretion
Male Sex Organs	Promotes erection	Promotes ejaculation

Autonomic Computing

An autonomic computing system consists of eight key characteristics as follows [6]:

- To be autonomic, a computing systems needs to "know itself"-and comprise components that also possess a system identity
- An autonomic computing system must configure and reconfigure itself under varying and unpredictable conditions
- An autonomic computing system never settles for the status quo-it always looks for ways to optimize its workings

- An autonomic computing system must perform something akin to healing-it must be able to recover from routine and extraordinary events that might cause some of its parts to malfunction
- A virtual world is no less dangerous than the physical one, so an autonomic computing system must be an expert in self-protection
- An autonomic computing system knows its environment and the context surrounding its activity, and acts accordingly
- An autonomic computing system cannot exits in a hermetic (protected from outside influence) environment
- Perhaps most critical for the user, an autonomic computing system will anticipate the optimized resources needed while keeping its complexity hidden

Four Fundamental Elements of Autonomic Computing

In addition to possessing the eight key characteristics, a computer system must possess at least one of the four fundamentals elements of the following self- managing properties:

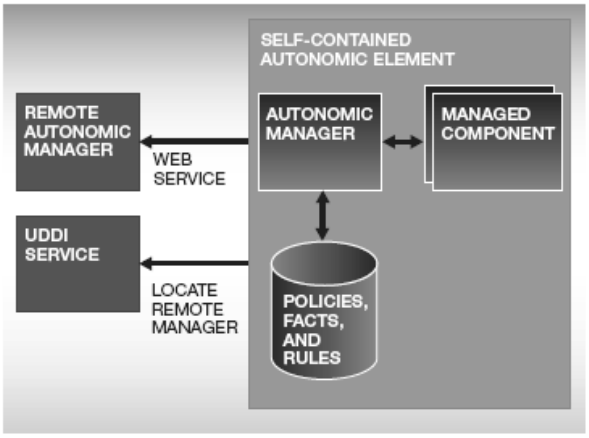
- Self-Healing: Systems discover, diagnose, and react to disruptions [18]. For a system to be self-healing, it must be able to recover from a failed component by first detecting and isolating the failed component, taking it off line, fixing or isolating the failed component, and reintroducing the fixed or replacement component into service without any apparent application disruption. System will need to predict problems and take actions to prevent the failure from having an impact on applications. The self-healing objective must be to minimize all outages in order to keep enterprise applications up and available at all times. Developers of system components need to focus on maximizing the reliability and availability design of each hardware and software product toward continuous availability [18].
- Self-Optimizing: Systems monitor and tune resources automatically [18]. The concept of self-optimization is when a computer environment manages resource allocations and workloads to meet the end-user needs without any or minimal human interference.
- Self-Configuring: Systems adapt automatically to dynamically changing environments [18]. The idea of self-configuring is for hardware and software systems to have the ability to define themselves at any given moment. Having the desired capabilities, allows new components to be dynamically added without any or minimal human interference.
- Self-Protecting: Systems anticipate, detect, identify, and protect themselves form the attacks from anywhere [18]. The ability to detect unauthorized access, eliminates intrusions, reports such activities for each occurrence, and secured backup capabilities as the original source manager system.

Autonomic Architecture

The building block of autonomic computing system is called an autonomic element. An autonomic element is an individual system constituent that contains resources and delivers services to human and other autonomic elements [9]. Autonomic elements learn from past experiences by managing their internal behavior and their relationship with other autonomic elements in accordance with guidelines that humans or other elements have established to manufacture and execute action

plans. The general structure of an autonomic element is depicted in Figure 3. The formation of an autonomic element consists of an autonomic manger and managed elements. According to *An architectural blueprint for autonomic computing*, a managed element is what the autonomic manager is controlling, and an autonomic manager is component that implements a particular control loop.

Figure 3: Architecture of an autonomic element [16]



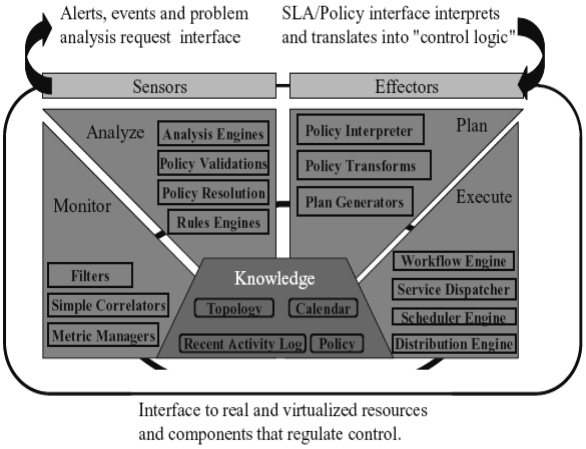
Autonomic Manager

An autonomic manager is a component that implements the control loop. The control loops can be executed through numerous permutations of management tools and products, or distributed by a resource provider in the runtime environment. The structure of an autonomic manager is partitioned into four sections that share the knowledge:

- The monitor part provides the mechanism that collects, aggregate, filter, manage, and report details (metrics and topologies) collected from an element [5, 7].
- The analyze part provides the mechanisms to correlate and model complex situations (time-series forecasting and queuing models, for example).These mechanisms allow the autonomic manger to learn more about the IT environment and help predict future situations. [5, 7].
- The plan part provides the mechanism to structure the action needed to achieve goals and objectives. The planning mechanism uses policy information to guide its work [5, 7].
- The execute part provides the mechanism that control the execution of a plan with considerations for on-the-fly updates [5, 7].

Figure 4 is an example of a more structural diagram rather than a control loop. The bold line connects the four parts should be thought of as a common messaging bus rather than a strict control flow [7]. For instance, the monitor function could signal for the plan function to create a new plan. Additionally, the plan function could signal for the monitor function to accumulate more or less information. The asynchronous collaboration technique, such as a messaging bus, is how the four functions collaborate with one another.

Figure 4: The functional details for an autonomic manager [7].



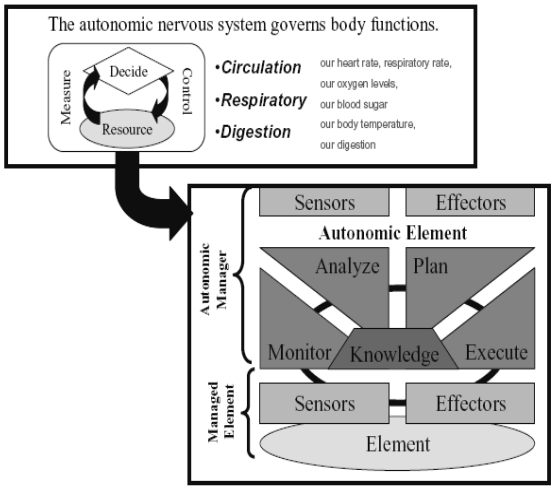
Managed Elements

A controlled system component is known as the manage element where the can be either a single resource (server or router) or a set of resources (cluster or pool of servers). The managed element is controlled through its sensors and effectors [7]:

- The sensors provide mechanisms to collect information about the state and state transition of an element. To implement the sensors, either uses a set of "get" operations to retrieve information about the current state, or a set of management events (unsolicited, asynchronous messages or notifications) that flow when the state of the element changes in a significant way.
- The effectors are mechanisms that change the state (configuration) of an element. In other words, the effectors are a collection of "set" commands or application programming interfaces (APIs) that change the configuration of the managed resource in some important way.

In order for autonomic computing environments to communicate, collaborate, and use management tools, the computing environments must organize their control loops into either autonomic manager or managed elements which are illustrated in figure 5. The manageability interface accessible to an autonomic manager is created due to the combination of sensors and effectors. As shown in the figure below, by the black lines connecting the elements on the sensors and effectors sides of the diagram, the architecture encourages the idea that sensors and effectors are linked together [7]. An instance of this is if any configuration change occurs by means of effectors this will be indicated as a configuration change reported through the sensor interface.

Figure 5: In an autonomic computing architecture, control loops facilitate system management [7].

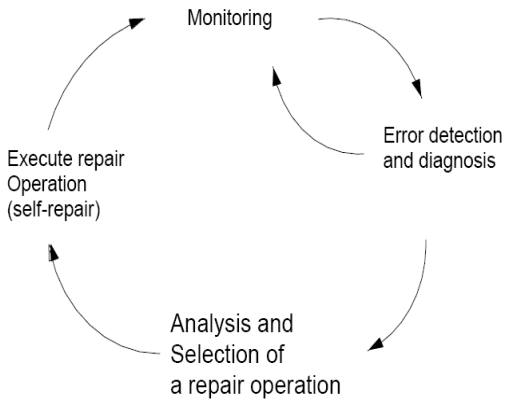


What is Self-Healing?

- **Self-Healing** denotes the system ability to examine find, diagnose and react to system malfunctions. Self-healing components or applications must be able to observe system failures, evaluating constraints imposed by the outside, and to apply appropriate corrections. In order to automatically discover system malfunctions or possible futures failures, it is needful to know the expected system behavior. Autonomic systems must have knowledge about own behavior then they must have a knowledge in order to determine if the actual behavior is consistent and expected in relation of the environment. In new contexts or in different scenarios, new system behaviors can be observed and the knowledge module must evolve with the environment [4].
- **Self-Healing** systems basically endure a process in order to maintain satisfactory quality of service of the principal system during runtime in the presence of any fault. The first cycle is called the monitoring cycle. During the monitoring cycle, the systems monitor will inspect the computer environment for any improper conduct. After the monitor's inspections are complete, it will send the data gathered through current observations to the next stage. The second phase of the cycle is called error detection and diagnosis; if the diagnosis reports that there is no fault in the system then it will loop back to the monitor for more observations. If there is an error detected by the monitor, the error detection cycle will report it to the next stage of the cycle. The third stage of the cycle is known as analysis and selection of a repair operation. At this stage, the fault is analyzed and a method of repairing is determined at this part of the cycle. After the repair operation is determined, the report is passed onto the finally phase of the cycle called execute repair and operation (self-repair). Any repairs that are needed are completed at this phase in the cycle. Once, the faulty areas are self-repaired the cycle begins all over

again. Since this cycle is a closed loop the, the process of self-healing environments will continuous heal itself as depicted in figure 6.

Figure 6 [28]: Self-healing System Process



Self-healing environments have comparable objectives to the common area of dependable computer environments. One of the fundamental tenets of dependable computing is that a fault hypothesis (often called a fault model) must be specified for any fault tolerant system [20].The fault hypothesis answers the question of what faults the system is to tolerate. Familiarly to dependable computer systems, self-healing systems must have a fault model in terms of what faults they are expected to be able to self-heal. Without a fault model, there is no way to assess whether a system actually can heal itself in situation of interest.

Fault Model Characteristics

The Following are typical fault model characteristics that seem relevant [20]:

- **Fault duration:** Faults can be permanent, intermittent(a fault that appears only occasionally), or transient (due to an environmental condition that appears only occasionally).Since it is widely believed that transient and intermittent faults outnumber permanent faults, it is important to state the fault duration assumption of a self-healing approach to understand what situations it addresses.
- **Fault manifestation:** Intuitively, not all faults are as severe as others. Beyond that, components themselves can be designed to exhibit specific characteristics when they encounter faults that can make system-level self-healing simpler. A common approach is to design components that are fail-fast, fail-silent. However, other systems must tolerate Byzantine faults which are considered “arbitrary” faults. (It is worth noting that Byzantine faults exclude systematic software defects that occur in all nodes of a system, so the meaning of “arbitrary” is only with respect to an assumption of fault independence.) Beyond the severity of the fault manifestation, there is the severity of how it affects the system in the absence of a self-healing response. Some faults cause immediate system crashes. But many faults cause less catastrophic consequences, such as system slow-down due to excessive CPU

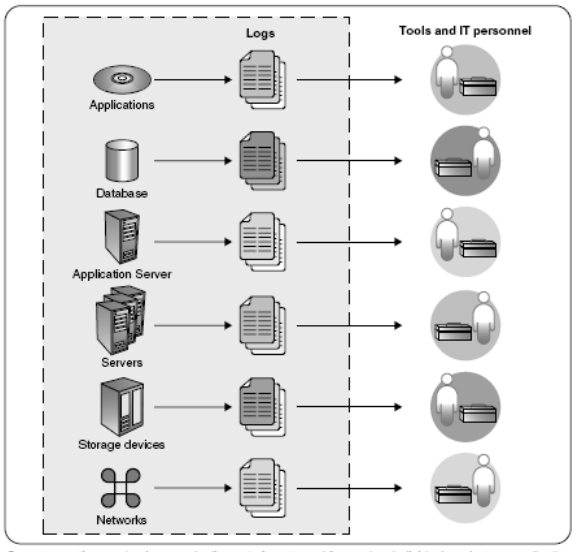
loads, thrashing due to memory hierarchy overloads, resource leakage, file system overflow, and so on.

- **Fault source:** Assumptions about the source of fault scan affect self-healing strategies. For example, faults can occur due to implementation defects, requirements defects, operational mistakes, and so on. Changes in operating environment can cause a previously working system to stop working, as can the onset of a malicious attack. While software is essentially deterministic, there are situations in which it can be argued that a random or "wear-out" model for failures is useful, suggesting techniques such as periodic rebooting as a self-healing mechanism. Finally, some self-healing software is designed only to withstand hardware failures such as loss of memory or CPU capacity, and not software failures.
- **Granularity:** The granularity of a failure is the size of the component that is compromised by that fault. (The related notion of the size of a fault containment region is a key design parameter in fault tolerant computers.) A fault can cause the failure of a software module (causing an exception), a task, an entire CPU's computational set, or an entire computing site. Different self-healing mechanisms are probably appropriate depending on the granularity of the failures and hence the granularity of recovery actions.
- **Fault profile expectations:** Beyond the source of the fault is the profile of fault occurrences that is expected. Faults considered for self-healing might be only expected faults (such as defined exceptions or historically observed faults), faults considered likely based on design analysis, or faults that are unexpected. Additionally, faults might be random and independent, might be correlated in space or time, or might even be intentional due to malicious intent.

Autonomic Computing Self-healing Systems vs. General Computing Systems

Complexity in problem determination is reducing the effectiveness of computing in many computing environments. One of the major factors contributing to the complexity in problem determination is the various ways that different parts of the system report events, conditions, errors, and alerts. For instance, examine the ways that general computers maintain logs for the system. These logs contain a variety of content in differing formats because solutions are built using disparate pieces and part, often with products form multiple vendors [21]. Figure 7a illustrates today's general computing environments and the obstacle of combing hardware and software components in a typical solution. Most of the logging done today is focusing on reporting data that a product developer considers important for debugging the problem in a single product, as opposed to providing data to debug a solution. This inconsistency in both the format and the content that is made available by products makes it more difficult to write management tools that might ease the complexity issues. To optimize the usefulness and business value of existing and future e-business solutions, major changes and improvements in problem determination must help businesses deal with complexity, ease cross-product problem determination and automate the process of identifying and fixing frequently occurring problems.

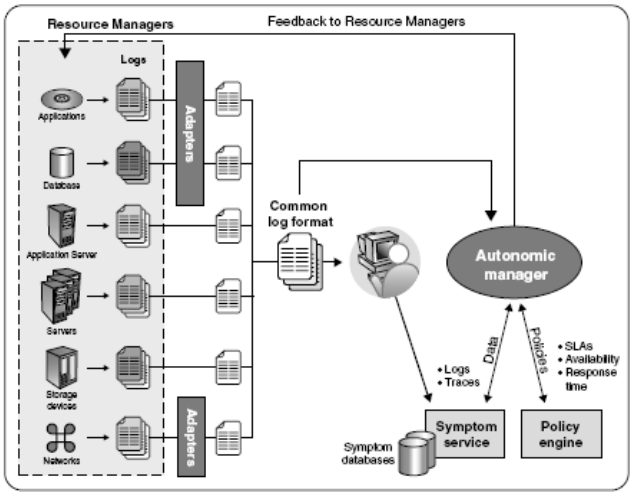
Figure 7a [21]: General Computing System



Autonomic computing systems are the solution to solving the problem of complexity. Refer to figure 7b on how autonomic computing is going to solve the problem determination of complexity. Let's begin with the resource manager. In the resource manager, various components such as applications, database, application server, servers, storage device, and networks are included. These components are apart of the troubleshooting process when a problem occurs. In figure 7b, each component produces multiple log files individually in its own format in various locations. Because there is no cost-effective way to change log files in legacy applications or solutions that have already been deployed, the IBM autonomic computing architecture includes adapters to translate disparate logs into the common format [21]. Adapters help keep implementation cost low, and adapters also allow business to use applications from independent software vendors that may not adopt common log formats. Log formats that are familiar enables specialist to easily look for problems in logs and take necessary actions if needed. Figure 7b also illustrates an autonomic manager engine which automates the process that a specialist would use. For example, IBM has developed the Log and Trace Analyzer for Autonomic Computing, which enables the reading of logs in the common format, correlating the logs based on different criteria (for example, time-based or field-based (such as URLs)) and viewing the correlated log records [21]. To take the best possible action when it discovers a problem, the manager will rely upon other sources of knowledge. One such source might be a symptom service. The symptom service includes a symptom database that contains information about how to detect patterns that indicate problems, how to diagnose that a specific problem has occurred, and how to resolve that specific problem. The symptom database will include a standardized set of interfaces and data formats that facilitate the determination of actionable causes from problem data. In many instances, multiple symptom databases are possible and likely, all presented as part of a symptom service. The process of writing and populating knowledge bases, such as the symptom database, is made simpler by the existence of a common format for log data.

Symptoms will be more easily expressed using the common set of terminology and data, alleviating the need for symptoms to be coded using the nuances of how, for example, an individual product says that it has "stopped." Once the decisions are made about how to best resolve a problem, the autonomic manager may then query other managers, such as a policy engine, represented in the lower right corner of Figure 7b, to determine which corrective actions can be taken. The policy engine matches proposed solutions against rules and policies to help ensure that an action's possible effect on business-critical processes is appropriate to the overall situation. For example, the symptom database may report that two separate actions could be used to address a symptom. The first, which would fix the problem, is to reboot a system. The second may be a temporary solution, such as increasing swapper space. In this example, if there were a policy that stated a critical system was not to be rebooted during business hours, the policy service would instruct the autonomic manager to use the temporary solution; the autonomic manager, in turn, would then provide feedback to the resource managers, which would make the necessary changes.

Figure 7b [21]: Autonomic Computing Self-Healing System



The Benefits of Autonomic Computing

Autonomic computing will produce many benefits for individuals, organizations and business. However, there will be many short term benefits and long term benefits throughout the era of autonomic computing. Companies that function on the low levels of maturity will most likely be exposed to various short term benefits and a minimal of long term benefits. On the other hand, organizations that function on the advanced levels of maturity will be exposed to numerous short term and long term benefits. IBM created a listing of the short term and long term benefits stated below [23].

Short-term I/T related benefits:

- Simplified user experience through a more responsive, real-time system.
- Cost-savings - scale to use.
- Scaled power, storage and costs that optimize usage across both hardware and software.
- Full use of idle processing power, including home PC's, through networked system.
- Natural language queries allow deeper and more accurate returns.

- Seamless access to multiple file types. Open standards will allow users to pull data from all potential sources by re-formatting on the fly.
- Stability. High availability. High security system. Fewer system or network errors due to self-healing.

Long-term, Higher Order Benefits:

- Realize the vision of enablement by shifting available resources to higher-order business.
- Embedding autonomic capabilities in client or access devices, servers, storage systems, middleware, and the network itself. Constructing autonomic federated systems.
- Achieving end-to-end service level management
- Collaboration and global problem-solving. Distributed computing allows for more immediate sharing of information and processing power to use complex mathematics to solve problems.
- Massive simulation - weather, medical - complex calculations like protein folding, which require processors to run 24/7 for as long as a year at a time.

The Challenges of Autonomic Computing

In order for any new vision to become a reality especially in the scientific and technical environments, each individual, organization, and businesses are going to have to merge their brilliant minds together and institute a sense or purpose and urgency. Even though IBM is determined to successfully overcome the challenges, one company cannot accomplish this vision single-handedly which is part of the challenge of autonomic computing. Another challenge is creating the open standards and new technologies needed for systems to interact effectively, to enact pre-determined business policies more effectively, and to be able to protect themselves and heal themselves with a minimal dependence on human interaction. According to IBM, this broader system view has many other challenges such as [23]:

On a conceptual level, the way we define and design computing systems will need to change:

- The computing paradigm will change from one based on computational power to one driven by data.
- The way we measure computing performance will change from processor speed to the immediacy of the response.
- Individual computers will become less important than more granular and dispersed computing attributes.
- The economics of computing will evolve to better reflect actual usage - what IBM calls e-sourcing.

Based on new autonomic computing parameters the functionality of individual components will change and may include:

- Scalable storage and processing power to accommodate the shifting needs of individual and multiple autonomic systems.
- Transparency in routing and formatting data to variable devices

- Evolving chip development to better leverage memory
- Improving network-monitoring functions to protect security detect potential threats and achieve a level of decision-making that allows for the redirection of key activities or data.
- Smarter microprocessors that can detect errors and anticipate failures

Current University Research in Autonomic Computing [27]

Self-configuring:
- Georgia Tech -self-configuring storage system
- George Mason University -self adjustment of computer system configuration parameters under dynamic loading
- University of Southern California at Los Angeles -network reconfiguration for applications optimization
- Stanford University-stability aspects of system reconfiguration

Self-healing:
- University of Maryland -remote repair of OS faults
- Brown University –transaction models for automated failure recovery in software
- University of Minnesota –agent based network monitoring systems for faults and security
- Duke University –proactive fault management at the applications level (cleaning & restart)
- University of California at Berkeley -OceanStore, Recovery-oriented computing (ROC)

Self-optimizing:
- Michigan State University –adaptive middleware to support CORBA-based applications self-optimization, self-optimizing wireless network applications
- University of Michigan Ann Arbor -self optimization of QoS in eCommerce systems
- Vanderbilt University –self optimization of QoS in eCommerce systems
- York University (UK) -optimization of database access

Self-protecting:
- Texas A&M University-Cooperating Security Manager,
- Purdue -Intrusion Detection Using Autonomous Agents, COAST

Conclusion

For decades, the advancement of technology and science has mirrored the increase of complexity in many computer environments. However, recent studies have shown that there is an unbalance of growth between the advancements of science and technology with that of complexity. as the scale and complexity of these systems and applications grow, their development, configuration and management challenges are beginning to break current paradigms, overwhelm the capabilities of existing tools and methodologies, and rapidly render the system and applications, brittle, unmanageable, and insecure. These new finding have researchers and scientists in an uproar. As a result, researchers were faced with the task of finding an alternative approach to complexity.

In 2001, IBM developed a means of overcoming the unbalance of growth between complexity and technology. IBM's solution for reducing the total cost of ownership and coping with the rapidly growing complexity of integrating and managing today's computer-based systems was called autonomic computing. Autonomic computing is a computer environment that can detect and adjust

its system automatically to heal itself without the assistance of any human interaction. The name autonomic insinuates that it is a metaphor based on biology. The biological term is called the autonomic nervous system which is the body's master controller that monitors changes inside and outside the body integrates sensory inputs, and effects appropriate response. It consists of sensory neurons and motor neurons and subdivided into two divisions, parasympathetic (rest and digest) and sympathetic (fight and flight) nervous systems. In order for any computing system to be autonomic, it has to consists of eight key characteristics and possess at least one of the four fundamental elements of self-managing properties outlined in this paper. The building block of an autonomic computing system is called an autonomic element that consists of managed elements and an autonomic manager.

Autonomic computers that self heal themselves basically endure a process in order to maintain satisfactory quality of service of the principle system during runtime in the presences of any fault. The process is a closed loop cycle that consists of a monitoring cycle, an error detection and diagnosis cycle, analysis and a selection of a repair operation cycle, and an execute repair operation cycle. Each self-healing system process has a fault model in terms of what faults they are expected to be able to self-heal because with out a fault model there is no way to assess whether a system actually can heal itself in situation of interest.

There are many advantages of autonomic computing systems. As well as addressing complexity, autonomic computing offers the promise of a lower cost of ownership and a reduced maintenance burden as systems become self-managing. Additionally, section 5 outlines more short term and long term advantages of autonomic computing.

Nonetheless, there are some limitations also to this vision. The challenges of re-engineering today's systems of systems away from the complexity dilemma toward tomorrow's persuasive and ubiquitous computations and communications will require unifying standards, new economic models and trust of the users, as well as innovations to address the hard technical issues.

IBM's vision of autonomic computing is much like a journey than a destination. The journey to achieve the ultimate vision of autonomic systems has just begun. This new era of computing is greater than any single IT company. Many universities are exploring various aspects of autonomic computing such as self-configuring, self-healing, self-optimizing, self-protecting, grid computing, and routing. Increasing constraints on resources and greater focus on the cost of operations, has led NASA and others to utilize autonomy. Achieving overall autonomic behaviors remains an open and significant challenge, which will be accomplished through a combination of process changes, skills evolution, new technologies and architecture, and open industry standards.

References

[1] Eriksson, Joakim, N. Finne, and S. Janson, "Towards Self-Managing", ERCIM News, No 58, 2004.

[2] Dudley, Gary, N. Joshi, D. Ogle, B. Subramanian, and B. Topol, "Autonomic Self-Healing Systems in a Cross-Product IT Environment", Proceeding of the IEEE international conference on Autonomic Computing (ICAC '04), 2004.

[3]HP Invent, "The Self-managing system", www.hp.com/go/nonstop,2005.

[4] Tosi, Davide, "Research perspective in self-healing systems", 2004.

[5] IBM White Paper, “An architectural blueprint for autonomic computing”, IBM; 2003.

[6] Horn, Paul, “Autonomic computing: IBM’s perspective on the state of information technology”, In IBM Research, October 2001.
http://www.research.ibm.com/autonomic/); 2001.

[7] IBM Corporation, “An architectural blueprint for autonomic computing”, April 2003.

[8] Garlan, David and B. Schmerl, “Model-based adaptation for self-healing systems”, ACM, 2002.

[9] Sterritt, Roy and M. Hinchey, “Autonomic computing-panacea or poppycock?”, EASe Birds of a Feather Session, 2005.

[10] Sterritt, Roy and M. Hinchey, “Tutorial proposal: autonomic computing in real-time systems”, 2003.

[11] J.O. Kephart and D.M. Chess, “The Vision of Autonomic Computing”, In: IEEE Computer 36 (2003), No 1, pp. 41-50, doi:10.1109/MC.2003.1160055.

[12] Hawthorne, Matthew and D. Perry, “Architectural styles for adaptable self-healing dependable systems”, ACM, 2005.

[13] Tesauro,Gerald, D. Chess, W. Walsh, R. Das,A. Segal, I. Whalley, J. Kephart, and S. White, “ A multi-agent systems approach to autonomic computing”, AAMAS, 2004.

[14] Abdullah, Gani and G. Manson, “Towards a Self-Healing Networking Controlling Access to Network Applications”, Informing Science, 2003.

15] Parashar M and Hariri S, “Autonomic computing: an overview”, 2004. J.-P. Banaˆtre (ed). LNCS, Springer Verlag, 2005; 3566: 247–259.

[16] Bantz, D.F., C. Bisdikian, D. Challener, J.P. Karidis, S. Mastrianni, A. Mohindra, D.G. Shea, and M. Vanover, “Autonomic personal computing”, IBM System Journal, 1(42), 2003.

[17] Koelher, Jana, C. Giblin, D. Gantenbein, and R. Hauser, “Research report on autonomic computing architectures”, IBM Research, 2003.

[18] Ganek AG and Corbi TA, “The dawning of the autonomic computing era”, IBM System Journal 2003; 42(1).

[19] Russell LW, Morgan SP, Chron EG, “Clockwork: a new movement in autonomic systems”, IBM System Journal 2003; 42(1):77–84.

[20]Philip Koopman, “Elements of the Self-Healing System Problem Space”, ICSE 2003 WADS.

[21] IBM Corporation, “Automating problem determination: a first step towards self-healing computer systems”, www.research.ibm.com/autonomic, 2003.

[22] Sterritt, Roy, M. Parashar, H. Tianfiels, and R. Unland, “A concise introduction to autonomic computing”, Advanced Engineering Informatics, 2005.

[23] IBM Corporation, “Autonomic computing - a manifesto”, www.research.ibm.com/autonomic, 2001.

[24] Sterritt R and Bustard D, “Autonomic computing—a means of achieving dependability?”, Proceedings of IEEE international conference on the engineering of computer based system (ECBS’03), Huntsville, Alabama, USA; April 7–11 2003.

[25] Sterritt R, “Autonomic computing. Innovations”, System Software Engineering 2005; 1(1).DOI: 101007/s11334-101005-0001-5.

[26] The Peripheral Nervous System. http://www.users.rcn.com/jkimball.ma.ultranet/BiologyPages/P/PNS.html. 2003.

[27] Zhou, Michelle, “A survey of autonomic computing”, Artificial Intelligence Seminar, 2005.

[28] Rohr, Matthias, “Self-healing component-based enterprise information (software) systems”, TrustSoft Seminar, 2005.