A

Seminar-I Report

on

AUTONOMIC COMPUTING (MODELS AND APPLICATIONS)

Submitted in Partial Fulfillment of
the Requirements for the Third Year
of

Bachelor of Engineering

in

Computer Engineering

to

North Maharashtra University, Jalgaon

Submitted by

Ankita Kishor Wani

Under the Guidance of

Mr.Sandip S.Patil

DEPARTMENT OF COMPUTER ENGINEERING
SSBT's COLLEGE OF ENGINEERING AND TECHNOLOGY,
BAMBHORI, JALGAON - 425 001 (MS)
2015 - 2016

SSBT's COLLEGE OF ENGINEERING AND TECHNOLOGY, BAMBHORI, JALGAON - 425 001 (MS)

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the seminar-i entitled *Autonomic Computing (Models and Applications)*, **submitted by**

Ankita Kishor Wani

in partial fulfillment of the Third Year of *Bachelor of Engineering* **in** *Computer Engineering* **has been satisfactorily carried out under my guidance as per the requirement of North Maharashtra University, Jalgaon.**

Date: April 4, 2016
Place: Jalgaon

Mr.Sandip S.Patil
Guide

Prof. Dr. Girish K. Patnaik
Head

Prof. Dr. K. S. Wani
Principal

Acknowledgements

I would like to express my deep gratitude and sincere thanks to all who help me to complete this Seminar work successfully. I would like to thank to my college Director Prof. Dr. Sanjay P. Shekhawat. My sincere thanks to Principal Prof. Dr. K. S. Wani, SSBT COET for having provided me facilities to complete my Seminar work. My deep gratitude goes to Prof. Dr. G. K. Patnaik, head of the department, for granting me opportunity to conduct this Seminar work. I am also sincerely thankful to Mr.Sandip S.Patil, Seminar guide, for his valuable suggestions and guidance at the time of need. I am sincerely thankful to Mr. Akash Waghmare, Incharge of Seminar and Great thanks to my friends, my Seminar associates and all those who helped directly or indirectly for completion of this Seminar. Last but not the least thankful to my Parents.

Ankita Kishor Wani

Contents

List of Figures

Abstract

Autonomic computing, a new deployment technology introduced by IBM a decade ago, to manage the ever increasing complexity of IT systems, has become a part of many large scale deployments today. A lot of inroads have been made by autonomic computing in the areas of networking, data centers, storage, and database management. But few attempts have been exercised in business applications such as ERP, SCM or CRM, and Online Retail.The overarching goal of autonomic computing is to realize computer and software systems and applications that can manage themselves in accordance with high-level guidance from humans. Meeting the grand challenges of autonomic computing requires scientific and technological advances in a wide variety of fields, as well as new software and system architectures that support the effective integration of the constituent technologies.

Chapter 1

Overview

Enterprises strive to meet their current challenges, they require an IT infrastructure that supports their business goals. An IT infrastructure that enables business to be more responsive, variable, focused, and resilient.

Autonomic systems are such systems that are self-configuring, self-healing, selfprotecting and self-optimizing. They are Intelligent open systems that manage complexity, know themselves, continuously tune themselves, adapt to unpredictable conditions, prevent and recover from failures and provide a safe environment. They let enterprises focus on business, not on IT infrastructure.

Section 1.1 describes some Overview related to Autonomic Computing and Section 1.2 describes its different Concepts and Last section describes summery of this chapter.

1.1 Autonomic Computing's Overview

In a business environment, the main objective of analytic is to improve a business outcome. These outcomes can include:

1.1.1 Definition

The term autonomic comes from an analogy to the autonomic central nervous system in the human body, which adjusts to many situations automatically without any external help. We walk up a flight of stairs and our heart rate increases. If it is hot, we perspire. If it is cold, we shiver. We do not tell ourselves to do these things, they just happen. Similarly, the way to handle the problem of managing a complex IT infrastructure is to create computer systems and software that can respond to changes in the IT (and ultimately, the business) environment, so the systems can adapt, heal and protect themselves.

The IBM Autonomic Computing Initiative (first proposed in 2001) is an industry leading effort focused on managing complexity. Autonomic Computing is IBMs term for an approach and blueprint including a set of products, tools and services that add self-managing capabilities to Information Technology systems. Its goal is to shift the burden of support tasks such as configuration, maintenance, and fault management from people to technology.

Autonomic computing systems consist of four attributes. As illustrated in the following 4-quadrant chart, they are:

- Self-configuring (able to adapt to changes in the system)

  Can dynamically adapt to changing environments -Self-configuring components adapt dynamically to changes in the environment, using policies provided by the IT professional. Such changes could include the deployment of new components or the removal of existing ones, or dramatic changes in the system characteristics. Dynamic adaptation helps ensure continuous strength and productivity of the IT infrastructure, resulting in business growth and flexibility.

- Self-healing (able to recover from detected errors)

  It can discover, diagnose and react to disruptions Self-healing components can detect system malfunctions and initiate policy-based corrective action without disrupting the IT environment. Corrective action could involve a product altering its own state or effecting changes in other components in the environment. The IT system as a whole becomes more resilient because day-to-day operations are less likely to fail.

- Self-optimizing (able to improve use of resources)

  It can monitor and tune resources automatically Self-optimizing components can tune themselves to meet end-user or business needs. The tuning actions could mean reallocating resourcesuch as in response to dynamically changing workloadsto improve overall utilization, or ensuring that particular business transactions can be completed in a timely fashion. Self-optimization helps provide a high standard of service for both the systems end users and a businesss customers.

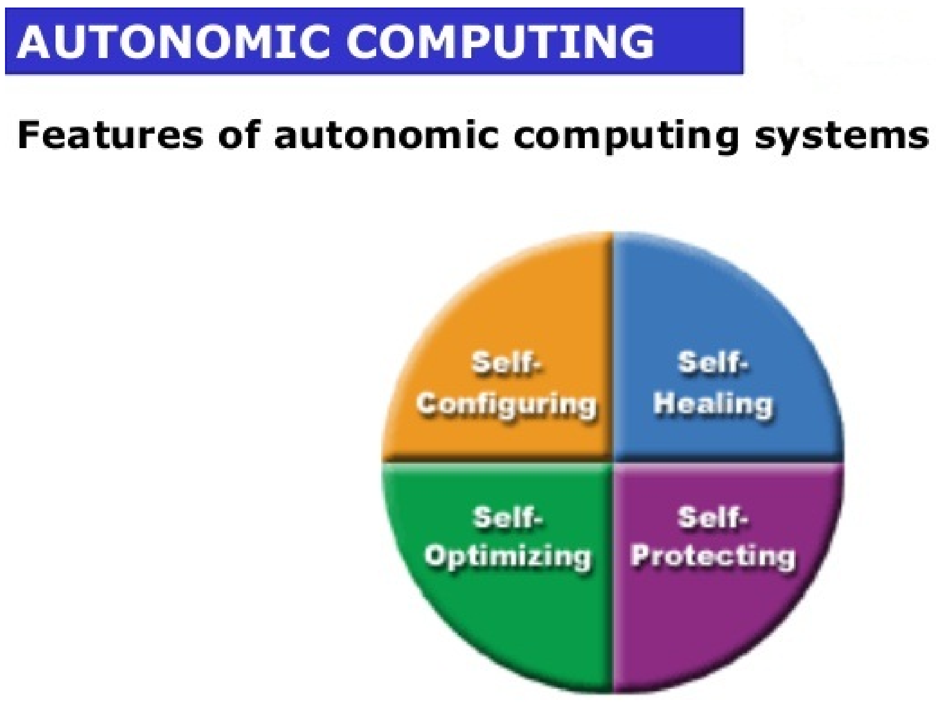

Figure 1.1: Features of Autonomic Computing

- **Self-protecting (able to anticipate and cure intrusions)**

  It can anticipate, detect, identify and protect against threats from anywhere Self-protecting components can detect hostile behaviors as they occur and take corrective actions to make themselves less vulnerable. The hostile behaviors can include unauthorized access and use, virus infection and proliferation, and denial-of-service attacks. Self-protecting capabilities allow businesses to consistently enforce security and privacy policies.

1.2 Autonomic Computing Concepts

In an autonomic environment, components work together, communicating with each other and with high-level management tools. They can manage or control themselves and each other. Components can manage themselves to some extent, but from an overall system standpoint, some decisions need to be made by higher level components that can make the appropriate trade-offs based on policies that are in place. The following figure represents the control loop that

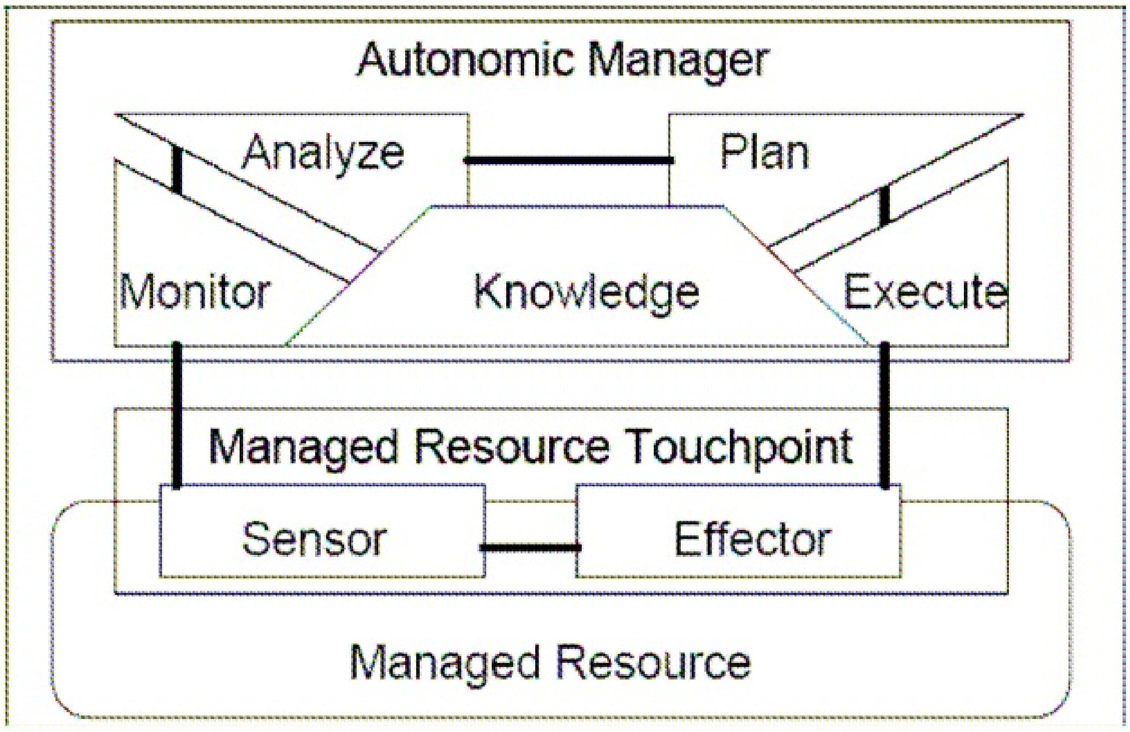

Figure 1.2: Concept of Autonomic Computing

is the core of the autonomic architecture. The autonomic manager implements autonomic control loops by dividing them into four parts: monitor, analyze, plan, and execute. The control loop carries out tasks as efficiently as possible based on high-level policies.

- **Monitor: Through information received from sensors, the resource monitors the environment for specific, predefined conditions. These conditions dont have to be errors; they can be a certain load level or type of request.**
- **Analyze: Once the condition is detected, what does it mean? The resource must analyze the information to determine whether action should be taken.**
- **Plan: If action must be taken, what action? The resource might simply notify the administrator, or it might take more extensive action, such as provisioning another hard drive.**
- **Execute: It is this part of the control loop that sends the instruction to the effector, which actually affects or carries out the planned actions. The managed resources are controlled system components that can range from single resources such as a server, database server, or router to collections**

of resources like server pools, clusters, or business applications. All of the actions in the autonomic control loop either make use of or supplement the knowledge base for the resource. For example, the knowledge base helps the analysis phase of the control loop to understand the information its getting from the monitor phase. It also provides the plan phase with information that helps it select the action to be performed.

1.3 Summary

In this chapter, an overview of An Autonomic Computing is described.Description regarding various concepts of autonomic computing is also given.Later chapter will give the Introduction of Autonomic Computing system in brief.

Chapter 2

Introduction

The increasing scale complexity, heterogeneity and dynamism of networks, systems and applications have made our computational and information infrastructure brittle, unmanageable and insecure. This has necessitated the investigation of an alternate paradigm for system and application design, which is based on strategies used by biological systems to deal with similar challenges a vision that has been referred to as autonomic computing.

Autonomic computing refers to the self-managing characteristics of distributed computing resources, adapting to unpredictable changes while hiding intrinsic complexity to operators and users. Started by IBM in 2001, this initiative ultimately aims to develop computer systems capable of self-management, to overcome the rapidly growing complexity of computing systems management, and to reduce the barrier that complexity poses to further growth.

Section 2.1 contains the introduction of An Autonomic Computing, 2.2 contains the properties of autonomic computing, section 2.3 describes characteristics of autonomic computing and Section 2.4 contains summary of this chapter.

2.1 Introduction

Computing systems have reached a level of complexity where the human effort required to get the systems up and running and keeping them operational is getting out of hand. A similar problem was experienced in the 1920s in telephony. At that time, human operators were required to work the manual switchboards, and because usage of the telephone increased rapidly, there were serious concerns that there would not be enough trained operators to work the switchboards [Mainsah 2002]. Fortunately, automatic branch exchanges were introduced to eliminate the need for human intervention.

Autonomic computing seeks to improve computing systems with a similar aim of decreasing human involvement. The term autonomic comes from biology. In the human body, the autonomic nervous system takes care of unconscious reflexes, i.e. bodily functions that do not require our attention, e.g. bodily adjustments such as the size of the pupil, the digestive functions of the stomach and intestines, the rate and depth of respiration and dilatation or constriction of the blood vessels. Without the autonomic nervous system, we would be constantly busy consciously adapting our body to its needs and to the environment. Before an autonomic computing there were many problems which are described below.

2.1.1 Problem of growing complexity

Currently, this volume and complexity is managed by highly skilled humans; but the demand for skilled IT personnel is already outstripping supply, with labour costs exceeding equipment costs by a ratio of up to 18:1. Computing systems have brought great benefits of speed and automation but there is now an overwhelming economic need to automate their maintenance.

In a 2003 IEEE Computer Magazine article, Kephart and Chess warn that the dream of inter connectivity of computing systems and devices could become the nightmare of pervasive computing in which architects are unable to anticipate, design and maintain the complexity of interactions. They state the essence of autonomic computing is system self-management, freeing administrators from low-level task management while delivering better system behavior.

General problem of modern distributed computing systems is that their complexity, and in particular the complexity of their management, is becoming a significant limiting factor in their further development. Large companies and institutions are employing large-scale computer networks for communication and computation. The distributed applications running on these computer networks are diverse and deal with many tasks, ranging from internal control processes to presenting web content to customer support.

Additionally, mobile computing is pervading these networks at an increasing speed: employees need to communicate with their companies while they are not in their office. They do so by using laptops, personal digital assistants, or mobile phones with diverse forms of wireless technologies to access their companies data. This creates an enormous complexity in the overall computer network which is hard to control manually by human operators. Manual control is time-consuming, expensive, and error-prone. The manual effort needed to control a growing networked computer-system tends to increase very quickly.

80 percent of such problems in infrastructure happen at the client specific application and database layer. Most 'autonomic' service providers guarantee only up to the basic plumbing layer (power, hardware, operating system, network and basic database parameters).

2.1.2 The Autonomic Human Nervous System

The human nervous system is, to the best of our knowledge, the most sophisticated example of autonomic behavior existing in nature today. It is the bodys master controller that monitors changes inside and outside the body, integrates sensory inputs, and effects appropriate response. In conjunction with the endocrine system, the nervous system is able to constantly regulate and maintain homeostasis. A homeostatic system (e.g., a large organization, an industrial firm, a cell) is an open system that maintains its structure and functions by means of a multiplicity of dynamic equilibriums that are rigorously controlled by interdependent regulation mechanisms. Such a system reacts to every change in the environment, or to every random disturbance, through a series of modifications that are equal in size and opposite in direction to those that created the disturbance.

The manifestation of the phenomenon of homeostasis is widespread in the human system. As an example, consider the mechanisms that maintain the concentration of glucose in the blood within limits - if the concentration should fall below about 0.06 percent, the tissues will be starved of their chief source of energy; if the concentration should rise above about 0.18 percent, other undesirable effects will occur. If the blood-glucose concentration falls below about 0.07 percent, the adrenal glands secrete adrenaline, which causes the liver to turn its of glycogen into glucose. This passes into the blood and the blood-glucose concentration drop is opposed. Further, a falling blood-glucose also stimulates appetite causing food intake, which after digestion provides glucose.

2.2 Autonomic Computing

The autonomic computing A computing systems that can manage themselves given high-level objectives from administrators. When IBMs senior vice president of research, Paul Horn, introduced this idea to the National Academy of Engineers at Harvard University in a March 2001 keynote address, he deliberately chose a term with a biological connotation. The autonomic nervous system governs our heart rate and body temperature, thus freeing our conscious brain from the burden of dealing with these and many other low-level, yet vital, functions The term autonomic computing is emblematic of a vast and somewhat tangled hierarchy of natural self-governing systems, many of which consist of myriad interacting, self-governing components that in turn comprise large numbers of interacting, autonomous, self-governing components at the next level down.

The enormous range in scale, starting with molecular machines within cells and extending to human markets, societies, and the entire world socioeconomic, mirrors that of computing systems, which run from individual devices to the entire Internet. Thus, we believe it will be profitable to seek inspiration in the self-governance of social and economic systems as well as purely biological ones. Clearly then, autonomic computing is a grand challenge that reaches far beyond a single organization. Its realization will take a concerted, long-term, worldwide effort by researchers in a diversity of fields. A necessary first step is to examine this vision: what autonomic computing systems might look like, how they might function, and what obstacles researchers will face in designing them and understanding their behavior.

In a self-managing autonomic system, the human operator takes on a new role: instead of controlling the system directly, he or she defines general policies and rules that guide the self-management process. For this process, IBM defined the following four types of property referred to as self-star (also called self-*, self-x, or auto-*) properties.

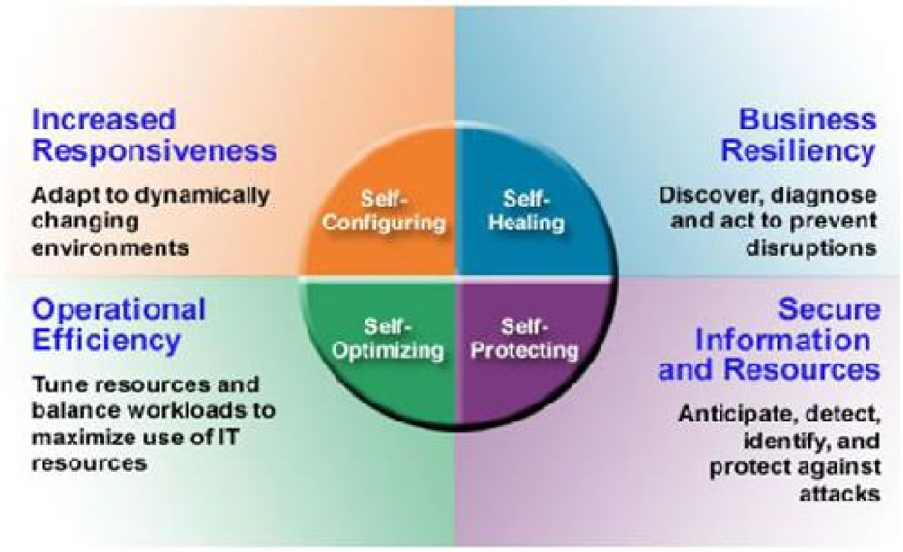
Increased Responsiveness
Adapt to dynamically changing environments
Self-Configuring
Self-Healing
Business Resiliency
Discover, diagnose and act to prevent disruptions
Operational Efficiency
Tune resources and balance workloads to maximize use of IT resources
Self-Optimizing
Self-Protecting
Secure Information and Resources
Anticipate, detect, identify, and protect against attacks

Figure 2.1: Properties of Autonomic Computing

2.2.1 Properties of Autonomic Computing

1. **Self-configuration:**
   Automatic configuration of components.

2. **Self-healing:**
   Automatic discovery, and correction of faults.

3. **Self-optimization:**
   Automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements;

4. **Self-protection:**
   Proactive identification and protection from arbitrary attacks.

Others such as Poslad and Nami and Bertel have expanded on the set of self-star as follows:

- **Self-regulation:**
  A system that operates to maintain some parameter, e.g., Quality of service, within a reset range without external control.

- **Self-learning:**
  Systems use machine learning techniques such as unsupervised learning which does not require external control.

- Self-awareness (also called Self-inspection and Self-decision):
  System must know itself. It must know the extent of its own resources and the resources it links to. A system must be aware of its internal components and external links in order to control and manage them.

- Self-organization:
  System structure driven by physics-type models without explicit pressure or involvement from outside the system.

- Self-management (also called self-governance):
  A system that manages itself without external intervention. What is being managed can vary dependent on the system and application. Self - management also refers to a set of self-star processes such as autonomic computing rather than a single self-star process.

- Self-creation (also called Self-assembly, Self replicating):
  System driven by ecological and social type models without explicit pressure or involvement from outside the system. A systems members are self-motivated and self-driven, generating complexity and order in a creative response to a continuously changing strategic demand.

- Self-description (also called self-explanation or Self-representation):
  A system explains itself.

2.3 Characteristics of Autonomic Computing

Even though the purpose and thus the behaviour of autonomic systems vary from system to system, every autonomic system should be able to exhibit a minimum set of properties to achieve its purpose:

- Automatic:
  This essentially means being able to self control its internal functions and operations. As such, an autonomic system must be self-contained and able to start-up and operate without any manual intervention or external help. Again, the knowledge required to bootstrap the system (Know-how) must be inherent to the system.

- Adaptive:
  An autonomic system must be able to change its operation (i.e., its configuration, state and functions). This will allow the system to cope with

temporal and spatial changes in its operational context either long term (environment customisation/ optimisation) or short term (exceptional conditions such as malicious attacks, faults, etc.).

- Aware:
  An autonomic system must be able to monitor (sense) its operational context as well as its internal state in order to be able to assess if its current operation serves its purpose. Awareness will control adaptation of its operational behaviour in response to context or state changes.

2.4 Summary

In this chapter,Introduction of Autonomic Computing is given and some properties and characteristics of autonomic computing are specified. Next chapter will take overlook of History of Autonomic Computing.

Chapter 3

History

In 2001, IBM had internally initiated and investigated the ground-breaking autonomic idea. Through its seminal paper in 2003 [2], IBM had articulated its vision, design principles, development methodologies, and prospects to the outside world. The basic concepts of the MAPE-loop (Monitor-Analyze-Plan-Execute) as fundamental characteristics of an autonomic component and the major self-CHOP (self-configuration, -healing, -optimisation, -protection) characteristics were defined. All those were already described.

Section 3.1 described Release History of Autonomic Computing. Last section gives summary of the Chapter.

3.1 Evolution of Autonomic Computing

In 2001, IBM suggested the concept of autonomic computing. In their manifesto [Horn 2001], complex computing systems are compared to the human body, which is a complex system, but has an autonomic nervous system that takes care of most bodily functions, thus removing from our consciousness the task of coordinating all our bodily functions. IBM suggested that complex computing systems should also have autonomic properties, i.e. should be able to independently take care of the regular maintenance and optimization tasks, thus reducing the workload on the system administrators. IBM also distilled the four properties of a self-managing (i.e. autonomic) system: self-configuration, self-optimization, self-healing and selfprotecting.

The DARPA Self-Regenerative Systems programme started in 2004 is a project that aims to develop technology for building military computing systems that provide critical functionality at all times, in spite of damage caused by unintentional errors or attacks [Badger 2004]. There are four key aspects to this project.

First, software is made resistant to error and attacks by generating a large number of versions that have similar functional behaviour, but sufficiently different implementation, such that any attack will not be able to affect a substantial fraction of the versions of the program.

Second, modifications to the binary code can be performed, such as pushing randomly sized blocks onto the memory stack, that make it harder for attackers to exploit vulnerabilities, such as specific branching address locations, as the vulnerability location changes. Furthermore, a trust model is used to steer the computation away from resources likely to cause damage. Whenever a resource is used, the trust model is updated based on outcome. Third, a scalable wide-area intrusion-tolerant replication architecture is being worked on, which should provide accountability for authorised but malicious client updates. Fourth, technologies are being developed that supposedly allow a system to estimate the likelihood that a military system operator (an insider) is malicious and prevent it from initiating an attack on the system.

Finally, we would like to mention an interesting project that started at NASA in 2005, the Autonomous NanoTechnology Swarm (ANTS). As an exemplary mission, they plan to launch into an asteroid belt a swarm of 1000 small spacecraft (so called pico-class spacecraft) from a stationary factory ship in order to explore the asteroid belt in detail. Because as much as 6070

to be lost as they enter the asteroid belt, the surviving craft must work together.

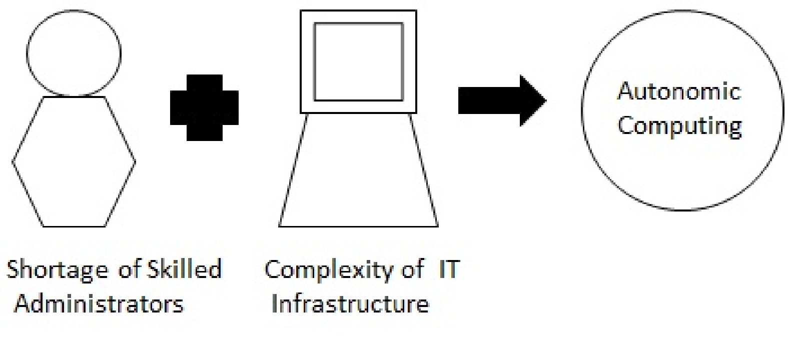
Shortage of Skilled Administrators
Complexity of IT Infrastructure
Autonomic Computing

Figure 3.1: Need of Autonomic Computing

3.2 Need for Industry

As IT systems become more and more complex, there is a need being felt for powerful and cutting-edge technologies and solutions in order to radically minimize and moderate the heterogeneity and multiplicity-induced IT complexity. Professors and professionals have been cooperating to cultivate and inculcate the innovation spirit among students and scholars to bring forth a bevy of competent solutions for the ills afflicting autonomic computing.

3.3 Summary

This chapter described History of An Autonomic Computing system, information about its evolution and next chapter will give you information about architecture of the system.

Chapter 4

Architecture Of Autonomic System

The architectural concepts presented in this blueprint define a common approach and terminology for describing self-managing autonomic computing systems. The autonomic computing architecture concepts provide a mechanism for discussing, comparing and contrasting the approaches that different vendors use to deliver self-managing capabilities in an IT system.

Section 4.1 describes some of the architecural concepts. Section 4.2 gives information about MAPE-K control loop of the autonomic computing system and Last section gives summary of Chapter.

4.1 Autonomic computing system

This blueprint organizes an autonomic computing system into the layers and parts . These parts are connected using enterprise service bus patterns that allow the components to collaborate using standard mechanisms such as Web services. The enterprise service bus integrates the various blueprint building blocks, which include:

- Touch points for managed resources
- Knowledge sources
- Autonomic managers
- Manual managers

The lowest layer contains the system components, or managed resources, that make up the IT infrastructure. These managed resources can be any type of resource (hardware or software) and may have embedded self-managing attributes. The next layer incorporates consistent, standard manageability interfaces for accessing and controlling the managed resources. These standard interfaces are delivered through a touchpoint. Layers three and four automate some portion of the IT process using an autonomic manager.

1. Touch points for managed resources

   A touch point is an autonomic computing system building block that implements sensor and effector behavior for one or more of a managed resources manageability mechanisms. It also provides a standard manageability interface. Deployed managed resources are accessed and controlled through these manageability interfaces. Manageability interfaces employ mechanisms such as log files, events, commands, application programming interfaces (APIs) and configuration files. These mechanisms provide various ways to gather details about and change the behavior of the managed resources. In the context of this blueprint, the mechanisms used to gather details are aggregated into a sensor for the managed resource and the mechanisms used to change the behavior of the managed resources are aggregated into an effector for the resource.

2. Managed resource

   A managed resource is a hardware or software component that can be managed. A managed resource could be a server, storage unit, database, application server, service, application or other entity. As shown in Figure 2, a managed resource might contain its own embedded self-management control loop, in addition to other autonomic managers (described later in this chapter) that might be packaged with a managed resource.

   Intelligent control loops can be embedded in the run-time environment of a managed resource. These embedded control loops are one way to offer self-managing autonomic capability. The details of these embedded control loops may or may not be externally visible. The control loop might be deeply embedded in a resource so that it is not visible through the manageability interface.

3. Manual Managers

   A manual manager provides a common system management interface for the IT professional using an integrated solutions console. Self-managing autonomic systems can use common console technology to create a consistent human-facing interface for the autonomic managers of IT infrastructure components. As indicated earlier, autonomic capabilities in computer systems perform tasks that IT professionals choose to delegate to the technology,

according to policies. In some cases, an administrator might choose for certain tasks to involve human intervention, and the human interaction with the system can be enhanced using a common console framework, based on industry standards, that promotes consistent presentation to IT professionals. The primary goal of a common console is to provide a single platform that can host all the administrative console functions in server, software and storage products to allow users to manage solutions rather than managing individual components or products.

The customer value of an integrated solutions console includes reduced cost of ownership attributable to more efficient administrationand shorter learning curves as new products and solutions are added to the autonomic system environment. The shorter learning curve is achieved by using standards and a Web-based presentation style. By delivering a consistent presentation format and behavior for administrative functions across diverse products, the common console creates a familiar user interface, reducing the need for staff to learn a different interface each time a new product is introduced.

These resource scopes define a set of decision-making contexts that are used to classify the purpose and role of a control loop within the autonomic computing architecture.

4.2 MAPE-K Model

To achieve autonomic computing, IBM has suggested a reference model for autonomic control loops [IBM 2003], which is sometimes called the MAPE-K (Monitor, Analyse, Plan, Execute, Knowledge) loop. This model is being used more and more to communicate the architectural aspects of autonomic systems. Likewise it is a clear way to identify and classify much of the work that is being carried out in the field. Therefore this section introduces the MAPE-K loop in

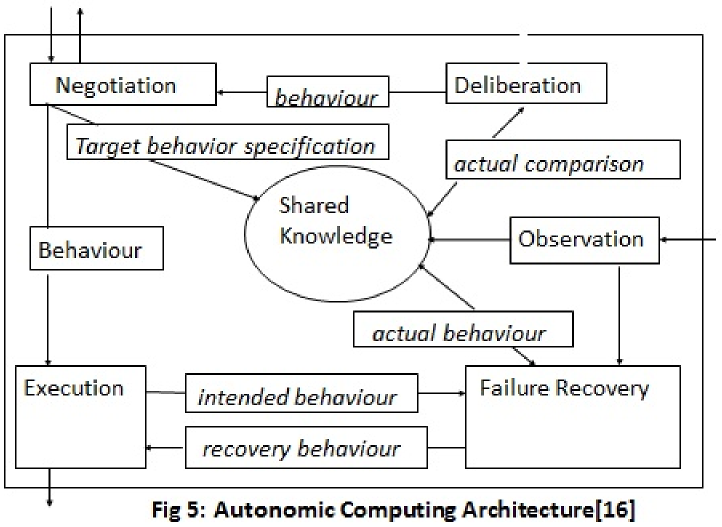

Figure 4.1: Autonomic Computing's Architecture

more detail and then taking each of its components in turn, describes the work that focuses its research on that component. It has following parts:

- **Monitoring**
- **Analyse**
- **Planning**
- **Execution**
- **Knowledge**

In the MAPE-K autonomic loop, the managed element represents any software or hardware resource that is given autonomic behaviour by coupling it with an autonomic manager. Thus, the managed element can for example be a web server or database, a specific software component in an application (e.g. the query optimiser in a database), the operating system, a cluster of machines in a grid environment, a stack of hard drives, a wired or wireless network, a CPU, a printer, etc.

Sensors, often called probes or gauges, collect information about the managed element. For a web-server, that could include the response time to client

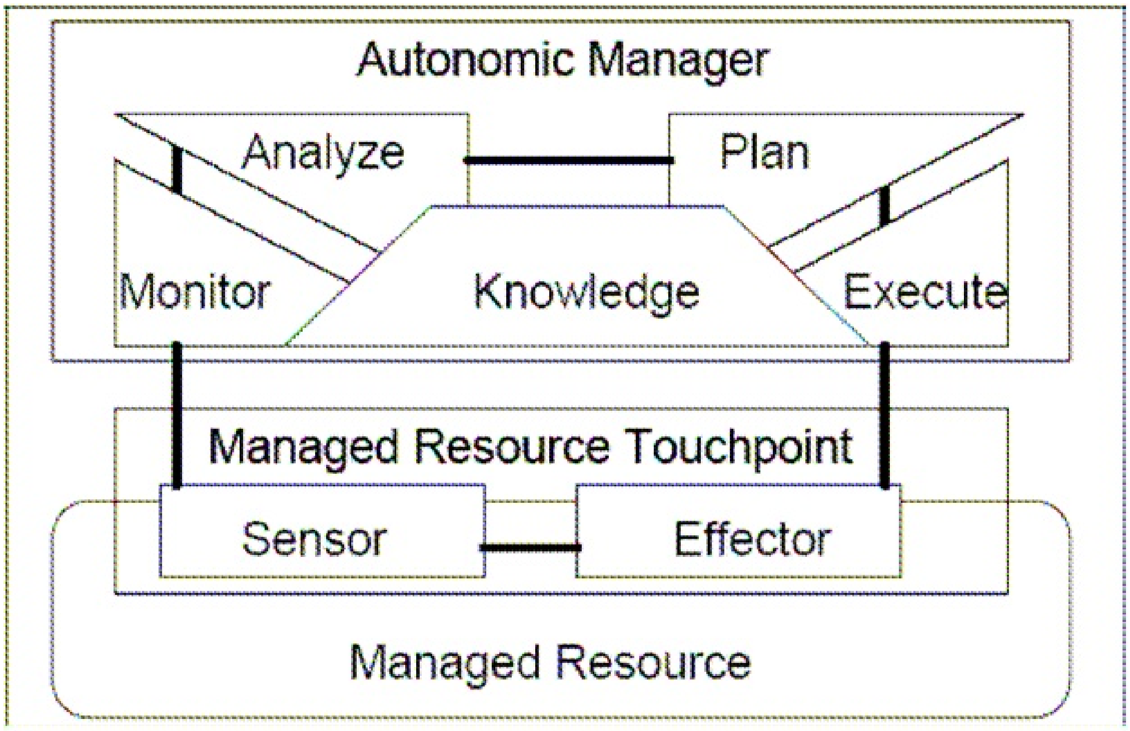

Figure 4.2: Common Managed Resources

requests, network and disk usage, CPU and memory utilisation. A considerable amount ofresearch is involved in monitoring servers [Roblee et al. 2005; Strickland et al. 2005; Xu et al. 2005; Sterritt et al. 2005; Diao et al. 2005].

4.2.1 Autonomic Manager

The data collected by the sensors allows the autonomic manager to monitor the managed element and execute changes through effectors. The autonomic manager is a software component that ideally can be configured by human administrators using high-level goals and uses the monitored data from sensors and internal knowledge of the system to plan and execute, based on these high-level goals, the low-level actions that are necessary to achieve these goals. The internal knowledge of the system is often an architectural model of the managed element (see Section 4.4.2). The goals are usually expressed using event-condition-action (ECA) policies, goal policies or utility function policies [Kephart and Walsh 2004]. ECA policies take the form when event occurs and condition holds, then execute action, e.g. when 95 percent of web servers response time exceeds 2s and there are available resources, then increase number of active web servers.

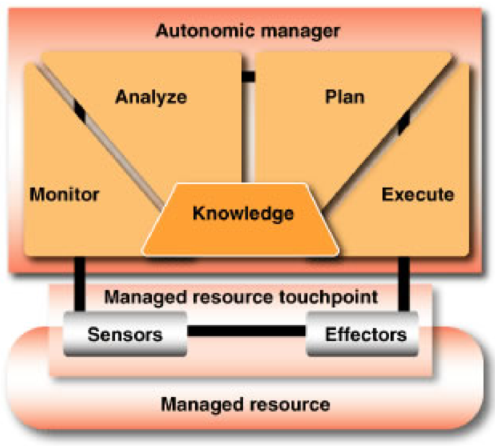

Figure 4.3: MAPE-K Control Loop

The autonomic manager uses internal rules (i.e. knowledge) to add or remove resources as necessary to achieve the desirable state. Goal policies require planning on the part of autonomic manager and are thus more resource-intensive than ECA policies. However, they still suffer from the problem that all states are classified as either desirable or undesirable. Thus when a desirable state cannot be reached, the system does not know which among the undesirable states is least bad. Utility functions solve this problem by defining a quantitative level of desirability to each state. A utility function takes as input a number of parameters and outputs the desirability of this state. Thus, continuing our example, the utility function could take as input the response time for web and application servers and return the utility of each combination of web and application server response times. The major problem with utility functions is that they can be extremely hard to define, as every aspect that influences the decision by the utility function must be quantified.

- Monitoring

  Monitoring involves capturing properties of the environment (either physical or virtual, e.g. a network) that are of significance to the self-X properties of the system. The software or hardware components used to perform monitoring are called sensors. For instance, network latency and bandwidth measure the performance of web servers, while database indexing and query optimisation affect the response time of a DBMS, which can be monitored. The Autonomic Manager requires appropriate monitored data to recognise failure or sub-optimal performance of the Autonomic Element, and effect appropriate changes. The types of monitored properties, and the sensors used, will often be application-specific, just as effectors used to execute changes to the Managed Element are also application-specific.

Passive monitoring. Passive monitoring of a computer system can be easily done under Linux. For example, the top command returns information about CPU utilisation by each process. Another common command that returns memory and cpu utilisation statistics is vm stat. Furthermore, Linux provides the /proc directory, which is a pseudo-file system (this directory does not store actual files on a hard disk) containing runtime system information, e.g. system memory, CPU information, per-process information (such as memory utilisation), devices mounted, hardware configuration, etc. Similar passive monitoring tools exist for most operating systems, e.g. Windows 2000 XP.

Active monitoring. Active monitoring means engineering the software at some level, e.g. modifying and adding code to the implementation of the application or the operating system, to capture function or system calls. This can often be to some extent automated. For instance, ProbeMeister5 can insert probes into the compiled Java byte code.

- Analyse

In the broadest sense, planning involves taking into account the monitoring data from the sensors to produce a series of changes to be effected on the managed element. In the simplest case, we could define event-condition-action (ECA) rules that directly produce adaptation plans from specific event combinations, as already mentioned in Section 4.1. Examples of such policy languages and applications in autonomic computing include [Damianou et al. 2000; Lymberopoulos et al. 2003; Lobo et al. 1999; Agrawal et al. 2005; Batra et al. 2002; Lutfiyya et al. 2001; Ponnappan et al. 2002].

However, applying this approach in a stateless manneri.e. where the autonomic manager keeps no information on state of the managed element, and relies solely on the current sensor data to decide whether to effect an adaptation planis very limited. Indeed, it is far better for the autonomic manager to keep information on the state of the managed element that can be updated progressively through sensor data and reasoned about.

Taking further the issue of state information that the autonomic manager should keep about the managed element, much research has examined a more complete approach to managing a systemcalled the architectural model-driven approachin which some form of model of the entire managed system is created by the autonomic manager, usually called an architectural model, that reflects the managed systems behaviour, its requirements and possibly also its goal. The model may also represent some aspect of the operating environment in which the managed elements are deployed, where operating environment can be understood as any observable property (by the sensors) that can impact its execution, e.g. end-user input, hardware devices, network connection properties.

- Knowledge

  The division between planning and the knowledge used to effect adaptation in the autonomic MAPE-K loop is quite fuzzy, as one would expect. The knowledge in an autonomic system can come from sources as diverse as the human expert (in static policy based systems [Bougaev 2005] to logs that accumulated data from probes charting the day-to-day operation of a system to observe its behaviour, which is used to train predictive models [Manoel et al. 2005; Shivam et al. 2006].

4.3 Summary

This chapter described Architecure of Autonomic Computing System and MAPEK contorl loop of the system.the next chapter decribes some Advantages and challenges of the proposed system .

Chapter 5

Advantages and Disadvantages

Cloud computing delivers infrastructure, platform, and software (applications) as subscription-based services in a pay-as-you-go model. In industry, these services are referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as Service (SaaS), respectively. Cloud infrastructure offering reliable, secure, and cost-efficient services is a Challenging task. It requires co-optimization at multiple layers (infrastructure, platform, and application) exhibiting autonomic properties, Challenges. Autonomic systems exhibit the ability of self-monitoring, Self-repairing and self-optimizing by constantly sensing themselves and tuning their performance. Such autonomic features are also exhibited by market economy, where resources/services are priced so as to maintain equilibrium in the supply and demand. Clouds constitute an interesting Venue to explore the use of autonomic features, because of their dynamism, large scale, and complexity.

Section 5.1 contains Advantages of Autonomic computing over Cloud computing. Section 5.2 contains disadvantages or challenges that have to be faced by autonomic computing. Last section of this chapter contains summary of chapter.

5.1 Advantages

Autonomic computing has numerous benefits to offer. As fewer system or network errors will occur due to self healing. Long-term benefits will allow individuals, organizations and businesses to collaborate on complex problem solving.

Like other computing system it has also many advantages . It has many characteristics as well as self-protecting, self-healing ,self-optimizing,self-managing

properties which make it powerful in its own way.Along with these features it has few more advantages as listed below:

1. **Immediate benefits include reduced dependence on human intervention to maintain complex systems accompanied by a substantial decrease in costs.**
2. **Simplified user experience through a more responsive and real-time system.**
3. **Full use of idle processing power- including home PC's through networked system.**
4. **Seamless access to multiple file types as Open standards will allow users to pull data from all potential sources by re-formatting on the fly.**
5. **Deeper and more accurate returns due to natural language queries.**
6. **Optimized usage across both hardware and software because of scaled power.**
7. **Storage and costs.**
8. **Stability along with high availability and high security system.**

5.2 Disadvantages

Autonomic Computing faces numerous technology- and marketplace-related obstacles to widespread adoption. One of the autonomic computing key challenges is to develop approaches for different types of systems that run on various hardware and software platforms which frequently include heterogeneous components.

Another hurdle is to determine which existing techniques can yield approaches that will consistently and automatically work without user intervention. Further, there also exists a risk that technologies that monitor, manage, and heal systems on their own could also harm systems for example, by introducing bugs, deleting important system settings, or removing executables. some challenges of the autonomic system are:

1. **Conceptual Challenges**

Conceptual research issues and challenges include (1) defining appropriate abstractions and models for specifying, understanding, controlling, and implementing autonomic behaviors; (2) adapting classical models and theories for machine learning, optimization and control to dynamic and multi agent system; (3) providing effective models for negotiation that autonomic elements can use to establish multilateral relationships among themselves; and (4) designing statistical models of large networked systems that will let autonomic elements or systems detect or predict overall problems from a stream of sensor data from individual devices.

2. Architectural Challenges

   Autonomic applications and systems will be constructed from autonomic elements that manage their internal behavior and their relationships with other autonomic elements in accordance with policies that humans or other elements have established. As a result, system/application level self-managing behaviors will arise from the self-managing behaviors of constituent autonomic elements and their interactions. System and software architectures in which local as well as global autonomic behaviors can be specified, implemented and controlled in a robust and predictable manner remains a key research challenge.

3. Middle-ware Challenges

   The primary middle-ware level research challenge is providing the core services required to realize autonomic behaviors in a robust, reliable and scalable manner, in spite of the dynamism and uncertainty of the system and the application. These include discovery, messaging, security, privacy, trust, etc. Autonomic systems/applications will require autonomic elements to identify themselves, discover and verify the identities of other entities of interest, dynamically establish relationships with these entities, and to interact in a secure manner. Further the middle-ware itself should be secure, reliable and robust against new and insidious forms of attack that use self-management based on high-level policies to their own advantage.

4. Application Challenges

The key challenges at the application level is the formulation and development of systems and applications that are capable of managing (i.e., configuring, adapting, optimizing, protecting, healing) themselves. This includes programming models, frameworks and middle-ware services that support the definition of autonomic elements, the development of autonomic applications as the dynamic and opportunistic composition of these autonomic elements, and the policy, content and context driven definition, execution and management of these applications.

5.3 Summary

This chapter contains the information about the Advantages and Disadvantages of An Autonomic Computing system. Next chapter will describe its various applications in detail.

Chapter 6

Applications

Autonomic managers could be introduced into existing applications, which could control activities of individual modules. It is assumed that the developer of original ERP is not involved in these introductions and therefore, we are dependent on direct database manipulations.

Section 6.1 Describes different uses of autonomic computing in CRM module.Section 6.2 gives the Applications in ERP modules.Last section gives summary of this chapter.

6.1 Autonomic Computing for CRM-Customer Relationship Model

At the outset, CRM is a typical monolithic application where the introduction of autonomous components might not seem appropriate. In most cases, customers using traditional CRM systems slowly start sinking in the huge volume of data produced by these systems. The introduction of autonomic components into this monolithic architecture brings better manageability and quicker approach to their business goals. The following sample autonomic computing managers will help the users of CRM systems manage the complexities that are increasing day by day: Autonomic Computing Architecture.

- Lead Generation Manager

  A primary function of any CRM is the management of leads, including all details and communication. The Lead Generation Manager is a self-optimizing autonomic component, which goes into a loop.

  The administrator specifies in the policy manager that the lead generation should be optimized for a utility function, for example, to maximize the revenue generated by the leads. The optimization could have be carried out based on the number of leads generated, proposals sent, and so on. The budget for lead generation activities is one of the parameters for the policy. The plan component allocates the budget to various lead generation activities a) online: such as search engines and web-site banners and b) off line: such as exhibitions and print advertisements. The execution takes place automatically, as in the case of many online promotions or manually, as in the case of exhibitions. The revenue generation results are assimilated with pointers to the lead generation activity. This information is analyzed to find out the most effective lead generation media and the budget is reallocated to each of these media based on rules specified in the policy. A simple greedy algorithm might suffice in most of the cases.

- Pricing Manager

  A pricing manager could optimize pricing policies across various pricing schemes. The pricing manager may deploy various pricing schemes from season to season and measure the effectiveness of each in terms of revenues, profits, and so on. Based on suggested algorithms, efficient schemes can be selected.

- Remainder Manager

  Customer relationships are maintained by effective reminders of various activities throughout the life-cycle like callback after initial contact, reminder after proposal, activation after a period of inactivity and so on. The optimal results of the reminders differ for each customer segment and product, based on elapsed time, type of reminder, and so on. For example, a customer, who is interested in a buying children health drink might revisit the idea either after 1 week (if she has not yet bought it) or after 6 weeks (to buy again, after finishing the competitor product she bought). The Reminder Manager activates various reminders for products, measures the effectiveness, and optimizes the reminder process for the given customer segment and product.

6.2 Autonomic Computing for ERP(Enterprise Resource Planing)

Enterprise Resource Planning systems today encompass multiple modules taking care of planning, tracking, and controlling multiple resources and activities in an enterprises. The systems have become so complex, that each enterprise employs a team of engineers to ensure the proper running of the system. This results in huge expenses every year. Often, ERP becomes so complex that it is a mere collection of various modules and interactions between these modules are not regulated. These modules could be production planning, purchasing, inventory, finance, HR, and even extended modules like CRM and so on.

Some of the typical managers, which could be introduced in an existing ERP system are:

1. Inventory Manager:

   Inventory management is a very important section in ERP, which has great implications on the financial performance of the company. Most of the ERP systems implement simple parameter watches minimum stock quantity, maximum stock quantity, minimum order quantity and so on. The autonomic Inventory management module collects data from multiple modules such as production, sales, and purchase, executes daily purchase parameters, and measures the effects. The main utility function is the inventory turnover ratio. Sub-parameters such as average inventory cost, and average wait time might also influence the purchase pattern.

2. Vendor Manager:

   Long term Vendor Management with selection and rating is very important to the supply side of any organization. Normally, companies keep two or more vendors for each product and divide the purchases, in a manner to optimize the supply, and at the same time keep all of them interested in supplying.

   The vendors are rated on various feedback parameters such as pricing, delivery times, quality, and so on. Each time a purchase has to be made, the purchase quantities are allocated based on the ratings. Similar managers can be deployed in multiple areas of ERP such as production, finances, HR, CRM, and SCM. The autonomic integration framework ensures the addition of features in a smooth manner without affecting the existing functionality.

6.3 Summary

This chapter contains the information about the Applications of the system in CRM and ERP and its various applications in detail. Later chapter contains the Conclusion.

Conclusion And Future Scope

In this seminar I have described Autonomic Computing as my Seminar topic and made detail explanation on each sub points of it.

Study of existing implementations has revealed that the deployment of autonomic computing techniques has made some progress in networking, load balancing, power management, and so on, but few inroads have been made into business applications. We have introduced a simplified architecture in this paper, keeping in mind the possibilities of wide spread acceptance from multiple technology corners, and the core trends like policies, knowledge segregation, services, and cloud. The proposed implementation of autonomic agents in the areas of ERP and CRM shows that the technology is well suited to implement complex and sophisticated systems of the future.

As the business requirements and software technologies keep changing and evolving all the way, standards will keep coping with and driving these changes. Its determinate that standards are required to play a more effective role in business development and market place. Continuous investigation and enhancement of the defined standards is inevitable. There is ongoing enhancement in the various fields of autonomic computing.

Bibliography

[1] www.ibm.com/comparative analysis/advantages.html

[2] www.ibm.com/architectural bluprint of autonomic computing/autonomic computing.pdf

[3] www.wikipedia.com/autonomic computing.html/definition

[4] www.auto.com/autonomic computing/autonomic upp viewer.pdf

[5] applications/business applications of autonomic computing.pdf

[6] ephart, J.O.; Chess, D.M. (2003), The vision of autonomic computing

[7] . Diao, L. J. Hellerstein, S. Parekh and Griffith, "A Control Theory Foundation for Self-Managing Computing Systems," IEEE Journal on Selected Areas in Communication. X. Dong, S. Hariri and L. Xue, "AUTONOMIA: An Autonomic Computing Environment," 2003.

[8] ww.autonomic computing.com/image

[9] ami, M.R.; Bertels, K. (2007). A survey of autonomic computing systems": 2630. line feed character in /—conference= at position 18 (help)

[10] utonomic computing self protection the vision of autonomic computing

[11] . Calinescu, "Challenges and Best Practices in Policy-Based Autonomic Architectures," in Third IEEE International Symposium on Dependable, Autonomic and Secure Computing, 2007. IBM Corporation. An architectural blueprint for autonomic computing.