Chapter-1

Introduction and Statement of the problem

1.1 Introduction

Ceaseless expansion and non-stop improvement in the field of automation of software and hardware applications are pretty obvious because of the tremendous advancement in computing systems, communication technologies and software. These changes are too conspicuous in managing the software in corporate-sector which is precisely termed as Autonomic-Computing [1] However, as the scale and complexity of these systems and applications grow, their development, configuration and management-challenges are beginning to break the current paradigms and overcome the capabilities of the existing tools and techniques. A general problem of modern distributed computing systems is that their complexity and in particular the complexity of their management [2], [3], is proving to be a major limiting factor in their further development.

As networks and distributed systems grow and change, system-deployment failures, hardware and software, and human errors are inevitable, collectively these added to the challenges in handling the system administration in tune with the terrific pace of changes taking place, which hampers effective system administration. In a situation like this Human intervention is in dispensable to assure the enhanced the performance and capacity of the components in an IT system. This escalates the overall costs-even as technology component costs decrease. It is also as observed fact that due to complexity in the computing system,
network-management-issues and communication-protocols are reaching such a level that it is proving to be beyond human control. This necessitates the development of network-protocols that supports features like adaptive, robust, and scalable with fully distributed self organizing architectures.

Currently, these voluminous and complex computing systems [4], [5] and their applications are managed and controlled manually by the highly skilled IT personnel; but the demand for skilled IT personnel is falling short of supply, where labor-costs exceed equipment costs. Besides this, manual control of these complex computing systems is undoubtedly hard, time consuming, expensive and above all prone to error.

A potential solution to this could be to develop networked-computing-systems that could manage themselves without direct human intervention. As the concept Self-management is gaining momentum one can seek inspiration from the autonomic function of the human central nervous system, where autonomic controls use motor-neurons to send indirect messages to organs at the subconscious plane. These messages regulate temperature, breathing, and heart-rate without any conscious thought. Taking inspiration from Autonomic nervous system, IBM laid foundation for autonomic-systems as an ideal substitute to human being to manage the modern computing systems. Autonomic computing helps to address the prevailing complexity by using technology to manage technology. It works independently on predefined policies and rules without human- interaction. Further, it manages and configures itself on its own based on general policies and knowledge that is gained over the time.

By enabling computer systems to have self-configuring, self-healing, self-optimizing and self-protecting features [6], [7] Autonomic computing is expected to have many benefits such as reduced operating costs, lower failure-rates, more security, and the ability to have systems that can respond promptly to the needs of the business within the market in which they operate.

1.2 Self-Management Attributes of system Components

System Components from hardware such as storage units, desktop computers and servers to software such as operating systems, middleware and business applications in Autonomic computing environment can include embedded control-loop functionality which is divided into four broad embedded control-loop categories. These categories are considered to be the attributes of the system-components and coined by Self-CHOP [8] These four attributes are depicted in Figure 1.1 and defined as follows:
1. **Self-Configuring**—Can dynamically adapt to changing Environments.
It is the ability of the system to automatically adapt to dynamically changing environment using the policies provided by IT professionals. Such changes may include the deployment of new components or removal of the existing ones. Dynamic Adaptation helps to increase the strength and productivity of the IT infrastructure, resulting in the business-growth and flexibility as well.

![Figure 1.1 Four Attributes of Autonomic Computing: Source R. Telford [9]](image)

2. **Self-Healing**—Can discover, diagnose and react to disruptions.
It is the ability of the system to detect the malfunction and accordingly initiate policy-based corrective actions without disrupting the IT environment. Corrective-actions may involve a product altering its own state or effective changes in other components in the environments. This helps IT system as a whole to become more resilient.

3. **Self-Optimization**—Can monitor and tune resources automatically.
It is the ability of the system to monitor and tune resources automatically to the business-needs. Tuning action may include reallocating resources to improve overall utilization and ensuring the transactions to be completed in a timely fashion. This provides high standard of service for both the system’s end-user and the customers.

4. **Self-Protection**—Can anticipate, detect, identify and protest against threats.
It is the ability of the system to anticipate, detect hostile behaviors as they occur and take corrective-actions to render itself less vulnerable. The hostile behavior may include unauthorized access and use, Virus infection and denial of service-attacks. This provides businesses to consistently enforce both security and privacy.
1.3 Autonomic Computing Architecture Concepts.

Figure 1.2 shows the Autonomic Computing Architecture that defines a common approach and terminology for describing the autonomic Computing system [10]. It also provides a mechanism for discussing, comparing and contrasting the approaches that different vendors use to deliver self-managing capabilities into an IT system.

1.3.1 Autonomic Computing System

An Autonomic Computing System has the properties of Self-CHOP that senses its operating environment, models its behavior suitable to that environment and takes actions to change its behavior. It is organized into layers and parts as shown in Figure 2. These parts are connected by using distributed infrastructure that allows the components to collaborate using standard mechanism such as Web-services that integrates various blue print components, which include: Managed resources, Touchpoints, Autonomic Managers and an integrated solution console.

1.3.2 Managed Resources

Today, most of the IT infrastructure is a collection of complex heterogeneous IT system that consists of managed resources as the run time entities at its operational level. These include a server, storage unit, database, application server etc. As shown in figure 2. a managed resource
may contain its own embedded self-management control loop that actually offers self-
management capabilities

1.3.3 Touchpoints

A Touchpoint is a building block of an Autonomic Computing system that acts as interface to
an instance of managed-resources such as server or an operating system. It implements sensor
and effector behavior for one or more managed-resources. It also provides manageability
interfaces through which managed resources are accessed and controlled. Manageability Interfaces employ mechanism such as log-files, events, commands, Application Programming interfaces (API’s) and configuration files. These mechanisms provide various ways to collect
details and change the behavior of managed resources. The mechanism used to gather details are aggregated into a sensor for the managed-resources and the mechanisms used to change the behavior of the managed-resources are aggregated into effectors for the resources.

1.3.4 Architectural Details

This section provides the additional details about the Autonomic architectural [12] which
includes examples of core capabilities of Autonomic Computing associated with the architectural elements discussed in the previous section.

**Autonomic Manager:** The Autonomic manager is a core component of the Autonomic control loop. For a system-component to be self-managing, it must have the following functionalities:

- An automated method to collect the details from the system:
- An automated method to analyze those details to determine if something needs to change
- An automated method to create a plan, or sequence of actions, that specifies the necessary changes.
- An automated method to execute those plans.

When all these functions are automated an intelligent control-loop is formed. The Figure 1.3 dissects the loop into four parts that share knowledge. IBM Autonomic Computing Architecture defines an abstract information framework for self-managing IT systems. In the information framework, an autonomic system is a collection of autonomic elements. Each autonomic element consists of an autonomic manager and the managed resource. The communication between the Autonomic manager and the Managed Resource is done through the interfaces, which exposes two types of components, sensors and effectors.
The sensors are used by the Autonomic Manager [13] to obtain the internal state of the Managed Resources, and the effectors are used by the Autonomic Manager to change the behavior of the Managed Resources. The Autonomic Manager enables self-management of the resource using a “monitoring, analysis, planning, and execution” control loop, with supporting knowledge of the computing environment, management policies, and some other related considerations.

The autonomic computing information model only provides the conceptual guidance on designing self-managed systems; in practice, the information model needs to be mapped to an implementable management and control architecture for Autonomic Networks. Specifically, measurement techniques, rule engines, planning methodologies, dynamic resource allocation techniques, security and management schemes need to be developed for autonomic elements, and a scalable management platform is required to coordinate the autonomic elements into a self-managing system.

1. The monitor-function collects the details like topological information and configuration property settings, from the managed resources using touchpoints, and organizes them into symptoms that need to be analyzed. This data includes information about managed resource configuration, status, offered capacity and throughput. The monitor function aggregates, correlates and filters these details until it determines a symptom that needs to be analyzed. For example it aggregates and correlates the content of log files for multiple resources to determine a symptom with respect to a particular log-entry and is passed to
Analyze function. For successful operation, Autonomic Manager has the ability to rapidly organize and make sense of this data.

2. The Analyze-function provides the mechanisms to observe and analyze the situations to determine if some change needs to be made and passes a change request to Plan function. It is responsible for determining, if the Autonomic manager can abide by the established policy, now and in future. In many cases it models complex behavior using prediction techniques that allow the Autonomic manager to learn about the IT environment and help predict future behavior. In addition to that it must be able to perform complex data analysis and reasoning on the symptoms provided by the Monitor-function.

3. The Plan-function creates or selects a procedure to extract a desired alteration in the managed resource. It uses policy-information to guide its works and passes the appropriate change plan, which represents a desired set of changes for the managed resources to the Execute function.

4. The Execute function provides the mechanism to schedule and perform the necessary changes to the system. Once an Autonomic manager has generated a change-plan that corresponds to a change-request, some action needs to be taken to modify the state of one or more managed resources. The Execute-function is responsible for carrying out the procedure that was generated by the plan-function through a series of actions. Using the Touchpoints effectors interface these actions are performed for managed resources.

These four functional parts (Monitor, Analyze, Plan and Execute), collaborate and communicate with one another and exchange appropriate knowledge and data as shown in figure 1.3. The shared knowledge includes historical logs, symptoms and policies. The Autonomic manager gathers this knowledge in one of the three ways:

1. **By effectors:** The policy-knowledge obtained by effectors consists of set of behavioral constraints or preferences that influence the decisions made by an Autonomic manager.

2. **By External Information Service:** The Policy-acknowledge retrieved from an external information service consists of resource specific historical knowledge as log-file in the form of entries that signify events that have occurred in the system.

3. **By own creation:** The policy-acknowledge is created by the autonomic manager using the monitor part based on the information collected through sensors. The monitor part may also create knowledge based on the recent activities by logging the notifications that get from managed resources.
### Table 1.1 Knowledge Examples in problem determination

<table>
<thead>
<tr>
<th>S. No</th>
<th>Type of Knowledge</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Solution Topology Knowledge</td>
<td>Captures knowledge about the components and their construction and configuration for a solution. The Plan function of Autonomic manager can use this knowledge for installation, configuration and planning.</td>
</tr>
<tr>
<td>2</td>
<td>Policy Knowledge</td>
<td>Policy knowledge that is consulted to determine whether or not changes need to be made in the system. The Autonomic system requires uniform method for defining the policies that govern the decision making. Also the policies can be shared across the Autonomic Manager to manage entire system.</td>
</tr>
<tr>
<td>3</td>
<td>Problem Determination Knowledge</td>
<td>Problem Determination Knowledge includes monitored data, symptoms and decision trees. As the systems response to actions taken to correct problems, learned knowledge can be collected within the Autonomic Manager. Autonomic Computing system uses uniform method to represent problem determination knowledge such as Common Base Event.</td>
</tr>
</tbody>
</table>

The Autonomic Computing identifies the various types of system knowledge that includes solution topology knowledge, policy knowledge and problem determination knowledge scenarios. Table 1.1 summarizes various types of knowledge that may be present in the Autonomic System. Common syntax and semantics must be used to express each knowledge type.
1.4 Levels of Autonomic Maturity

Implementation of Self-Managing capabilities into an IT environment is quite an evolutionary process. Basically, it is done by each organization through the adoption of Autonomic Computing technologies, supporting-processes and skills. IT infrastructures can be classified into five levels of Autonomic-Maturity that shows how the autonomic capabilities, supporting processes and skills in the systems are evolving. Figure 1.4 shows the description of each level.

Level 1: Basic

The starting point where most systems are today, this level represents manual computing in which all system elements are installed and managed as separate entities. These environments require extensive, highly skilled IT staff who must aggregate and analyze multiple sources of system-generated-data and manage the IT environment from a broad spectrum of individual consoles with multiple interfaces. Highly skilled staff sets up, monitors, and eventually replaces system elements.

Level 2: Managed

Customers achieve this level through consolidation of data and actions from disparate systems onto fewer consoles, using management tools such as those offered by the IBM Tivoli® portfolio. The IT staff continues to analyze this data and initiate management actions; benefits include greater system awareness and improved productivity.

Level 3: Predictive

At this level, the system monitors and correlates data to recognize patterns and recommends actions that are approved and initiated by the IT staff. At the predictive level, management-integration across multiple components begins to manifest. With the implementation of predictive capabilities, benefits include the possibility to reduce dependency on deep skills.

Level 4: Adaptive

At the adaptive level, not only does the system monitor, correlate and develop action plans, but also takes actions according to the established policies. This level allows the staff to manage performance against service level objectives. This assists an organization in establishing a balance between human and system interactions and helps the IT infrastructure to be more responsive to changing business conditions and improves resiliency.
Level 5: Autonomic

At this final level, the components of the infrastructure are well integrated and dynamically manage themselves according to business rules and policies. The autonomic level allows staff to focus on enabling business requirements. Business policy becomes the primary driver behind IT management and the business benefits from the improved business agility and resiliency.

1.5 Challenges in Autonomic Computing

In order to implement an effective and comprehensive Autonomous Computing environment, the researchers must address the key challenges [14] with varying complexity. These key challenges are described as follows:

- Development and implementation must occur at the level of global enterprise IT systems. This requires each individual component in an autonomic system to function properly and for the components to communicate and work efficiently with each other in line with agreed open standards.
- New Algorithms are required for new features including the ability to remember previous occurrences, to take previous system experiences and sort out the information into good and bad and then to use that information to improve the rules for making decisions on future actions.
- Ability to handle and function in an increasingly heterogeneous environment
- IT management must also be prepared for change. i.e., the organizations must be convinced about the benefits of adopting Autonomic Computing. They have to be
convinced so that such systems will not deprive them of control; but instead help simplify systems-management and thus provide them with more time to deal with other more important issues such as formulating IT policies.

- IT personnel may also be expected to resist the implementation of autonomic systems as the system would threaten their job-security
- The cost in terms of time and money may be so high that the organization will not be willing to take the risk.
- System Identity: Before a system can transact with other system, how, should we design our own systems to define the extent of its own boundary in dynamic environment?
- Interface design: In a heterogeneous environment How to build consistent interfaces and points of control?
- Translating business policy to IT policy: The end result needs to be transported to the user. How to create human Interfaces that remove complexity and allow users to interact naturally with IT systems?
- Adaptive Algorithms: New methods will be needed to equip our systems to deal with changing environment and transactions, How to create Adaptive algorithms to take previous system-experience and use that information to improve the rules.
- Improving Network monitoring functions: Need to protect security, detect potential threats and achieve a level of decision making that allows for the redirection of key activities.

1.6 Statement of the problem

The problem is to “enhance the Self-managing capabilities for software systems by introducing Self-healing and Self-configuration features in the system-architecture”. The problem is subdivided into following sub problems:

1. To explore a new mechanism to express the symptom rules and associated actions for Symptom-catalog, in human readable form to facilitate better problem determination task to administrator that supports self-healing capability in computing system

2. To create a generic framework which allows every product to retain its development architecture in heterogeneous environment and yet be able to cooperate with other products in IT enabled systems, like avoidance of the additional programming effort required to write a separate code called Adapter which converts log-file entries of monitoring application into a standard format called Common Base Event (CBE).
3. To explore the creation of language features for runtime administration and management of symptom-rules with associated actions for any application/product to facilitate better problem determination task to administrator that supports self-healing.

4. To Explore the Self-configuration capability into Wireless Sensor Network for efficient utilization of memory power consumption of sensor node.

1.7 Organization of the Thesis

The chapters in the thesis are organized as follows:

Chapter-1 Introduction and statement of problem, presents motivational material and attributes of self management capabilities, Autonomic Computing architectural details and key challenging issues in Autonomic Computing.

Chapter-2: Literature Survey presents the early initiation of Self-management activity along with IBM’s reference called MAPE-K architecture and its implementation. It also gives an insight into core autonomic computing problem determination technology and self-healing features of IBM Corporations, SUN Microsystems and other organizations.

Chapter-3: A Simplified approach to the design of Symptom-catalog for self-healing systems provides a new mechanism to express the symptom rules and associated actions for Symptom-catalog, in human readable form to facilitate better problem determination task to administrator that supports self-healing capability in computing system. This includes variable record length field format for symptom-catalog to represent symptoms organized as hash table along with possible actions to be taken.

Chapter-4: This chapter presents Know-How mechanism to manage heterogeneity in log file format of the software product and alleviate the need for writing the adapter for every product. This enhances the system performance.

Chapter-5: This presents user friendly language processor and editor to write complex policies and attaching associated actions for effective monitoring of the applications/products. This includes generation of dependent and non dependent nested rules and dependency of rules in case of rule deletion and attaching more action-plug-ins to ensure more flexible system.

Chapter – 6: In Wireless Sensor Network it proposes a new mechanism to use resource aware management policies that consists of processor and memory etc., based on the requirement the components are activated to overcome energy constraint. This includes modified memory architecture that has Self configuration capability to manage the buffer for efficient utilization of memory power consumption.