Chapter 2

Background and Literature Survey

2.1 Introduction

Self-managing computing systems have the ability to manage themselves and dynamically adopt to change in accordance with business policies and goals. Based on the situations observed in the IT environment, Self-managing-computing system can execute some management activities, so that IT professional are freed from maintenance [15]and focus more on high value tasks by making technology work smarter. It means that we are letting the computing systems and infrastructures to manage themselves. In order to create this kind of environment, the administrator must write business policies and goals and letting the computing systems and infrastructure to configure, heal, optimize and protect itself, using those policies from malicious activities. Many research organizations are spending in million dollars to design and create systems which can be self-managing. Many experimental results have been produced and following section summarizes the work carried out by various research organizations.
2.2 Early initiation of Self-Management Technology

Soon, after the invention of Internet, one of the first projects, related to self management was initiated by DARPA [2] for military application in 1997. This project was named as Situational Awareness System (SAS). The aim of this project was to create personal communication and location devices for soldiers on the battle field. Soldiers could enter status reports like discovery of enemy tanks, on their personal devices, and this information is automatically broadcast to all other soldiers, which could then call up the latest status report to enter into an enemy area.

Important Observations:

1. The personal devices have to be able to communicate with each other in difficult environmental conditions when enemy is jamming the equipment in operation. This was addressed by multi-hop routing which is a form of decentralized peer-to-peer mobile adaptive routing algorithm.

2. Also, at the same time device must minimize enemy interception. This was addressed by enabling the device to transmit in wide range frequencies (20-2500Mhz) with bandwidth ranging from 10bps to 40Mbps.

Another DARPA project [2] related to self-management is the DASADA project which is implemented in the year 2000. The objective of this project was to develop a technology that might enable critical systems to meet high-assurance, dependability and adaptability requirements. In this model, the components are constructed, customized and evaluated before and after system-assembly to ensure that they can operate together with the rest of the system and its current context. This project pioneered the architecture driven approach to self-management capabilities based on monitoring the system by IBM organization.

In the field of autonomic computing, NASA has used autonomic behavior in the year 2000, in its Data Space 1 (DS1) mission and Mars pathfinder that have autonomic behavior. Also, NASA has developed an interesting project in the year 2005 called Autonomous Nano Technology Swarm (ANTS) which has an architecture that consists of reconfigurable components that form structure for deep space and planetary exploration, inspired by insect colonies.

Kephart J. O et al., IBM Thomas J Watson research center, have presented the framework [6] for developing self managing system which was first proposed by Paul Horn, a senior vice
president of IBM’s research in March 2001. They compared the autonomic-system term with biological connotation namely autonomic nervous system that monitors our heart-rate and body temperature thus freeing our conscious brain from the burden of dealing with these situations and many other vital functions. IBM always cites four aspects of Self-management capabilities that include Self-configuration, Self-healing, Self-optimization and Self-protection. Early, these capabilities are treated as distinct with different product teams creating solutions that address each one separately. The journey toward fully autonomic computing has taken many years with several important and valuable milestones. At first, automatic function will merely collect and aggregate information to support decisions made by human administrators. Later they serve as advisors, suggest possible course of action for administrator to consider.

The authors [14] have discussed the introduction to motivation and concept of autonomic Computing and challenges. They also have focused some research work that has taken place using IBM’s reference Model in Web server applications.

![Autonomic Reference Model](image)

**Figure 2.1 Autonomic Reference Model- MAPE-K Autonomic Control Loop.**

In 2001, IBM suggested the concept of Autonomic computing where the complex computing systems must be able to independently take care of regular maintenance and optimization of tasks to reduce the workload on the system administrators. To implement Self-management capabilities into the system, IBM has introduced a reference model for autonomic control loop which is called as MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) and is depicted in Figure 2.1. This reference model used to communicate with architectural aspects of autonomic system and provides a way to identify and classify the work which is being carried out in the field of Autonomic-computing.
The MAPE-K loop is a framework for implementing the autonomic computing which is inspired by the agent model [16] proposed by Bigus J et al. Here, intelligent agent perceives its environment through sensors and uses this to determine the actions to be executed on the environment. The managed resource which is a web server, or database or any software component of an application is given an autonomic behavior by combining with resources like for web server. The information could be response time to client requests, network and disk usage, CPU and memory utilization. The Effectors are the means to carry out the changes to the managed resources which mainly could be modifying mainly the configuration parameters in a web-server. The MAPE-K framework consists of autonomic-manager which is a software component configured by the administrators using the high level policies and the monitored data from the sensors. It also uses the internal knowledge of the system to plan and execute the actions based on the high level policies.

2.3 Implementation of the MAPE-K control loop Architecture

The following section describes the work and its contribution to the autonomic-computing area by configuring the components of MAPE-K architecture as suggested by IBM.

2.3.1 Autonomic Computing Tool Kit.

Melcher B et al., have described the prototype implementation of the MAPE-K loop [17], developed by IBM as a part of “developersworks Autonomic Computing Tool kit”. This is called as autonomic management engine that provides a practical framework and reference implementation for incorporating autonomic capabilities into a software system. Though, it is not a complete implementation of autonomic manager, it gives a foundation on which an autonomic manager can be built. It is implemented using the JAVA language and can communicate with other applications through XML messages. The authors used this framework to create network-service-configuration, where network devices are modeled as managed-resources and dedicated servers or gateways plays the role of autonomic manager.
Important Observations:

1. It provides a basic foundation on which an autonomic manager can be built.
2. It is applicable to all areas where the autonomic manager can be implemented at the software application level.
3. The mechanism in the tool-kit that supports global communication between managed resources and Autonomic manager has not been explored.

2.3.2 ABLE – Multi Agent based Autonomic Toolkit.

Bigus, J. P. et al., have also presented the toolkit for building the Autonomic systems. It is basically called as Agent Building and Learning Environment (ABLE) [18]. It is software architecture and framework composed of component theory, development tooling and agent platform for building the autonomous-intelligent-agents and multi agent systems. The authors demonstrated the two application case-studies using multiple agents and diagnostic application. The ABLE toolkit consists of ABLE agent framework, ABLE Component library, ABLE development tools and ABLE agent platform which are combined together to enable deployment of multi-agent autonomous applications.

a. ABLE agent framework

The ABLE is a JAVA based framework for developing and deploying intelligent agent applications. It allows algorithms to be packaged as a Java beans and deployed as autonomous agents. Figure 2.2, shows the ABLE framework comprising of set of Java interfaces and basic Classes.

AbleBeans- These are standard JavaBeans Components used in the ABLE framework. It defines a set of common attributes like name, comment, state and standard processing methods such as init(), reset(), process() and quit() allowing JavaBeans to be connected to form AbleAgents.

AbleEvents – It is a class used to implement any agent communication-design-pattern that provides the means to send data between agents to request actions to be performed by other agents, or request transactions with results returned either to the original request or to some other agent.
AbleObject – This class provides a base implementation of the AbleBean interface, defining the standard behavior for all the AbleBeans provided with ABLE tool-kit. It extends Java UnicastRemoteObject and implements the AbleEventListener, AbleBean and AbleEventQueueProcessor remote interfaces. It has an instance of AbleEventQueue to handle autonomous timer facility as well as asynchronous event processing functions for the beans.

AbleAgent- These are Ablebeans and are also the containers for other Ablebeans. It has its own thread for processing events asynchronously. It provides a useful abstraction for packaging the set of AbleBeans wired together to perform a specific function.

b. ABLE Component Library
In an ABLE system this is the fundamental component library of AbleBeans. These include data-access and filtering-beans, machine learning algorithms, machine reasoning and inference engines and high level data mining agents. It also contains a set of data-type classes, defining Boolean, categorical, Discrete, Numeric and string literals, variables and fields.

c. ABLE Development Tools
It provides an Able-agent editor which is Swing based interactive development and test environment using which agents cab be loaded, edited and saved in external files. The
Agent editor can graphically create Abel agents by using the library of core AbleBeans and Able-agent as building blocks. Dataflow, events and property connections can be added using the GUI environment. The Agents can be hand-coded and tested using the editor.

d. **ABLE Agent Platform**

It is a distributed agent platform supporting agents on multiple systems that communicates using Java remote method invocation. It also provides a set of services for Able Agents that form multi-agent systems. These services include standard agent life-cycle-transitions, directory facilitator and communication functions.

e. **ABLE Application Designs**

ABLE toolkit is quite flexible and can be used to add intelligence to applications. The following section describes three application case-studies to illustrate how the ABLE toolkit can be used to quickly develop autonomic solutions.

### 2.3.2.1 Example on ABLE framework

The following section presents three application case studies to demonstrate the use of ABLE components for quick development of autonomic solutions. Also, it presents the various ways in which ABLE framework, component library, development tools and agent platform can be combined to enable development of multi-agent applications.

1. **System administration using ABLE**

![System Administration Diagram](image)

Figure 2.3 System Administrations Using ABLE  
Source: J. P. Bigus, [18]
A basic System administration multi agent system was developed for IBM e-SERVER i-series system using ABLE technology which is shown in the figure 2.3. The objective was to provide an overall view of system health using multiple agents to monitor CPU utilization, the workload as indicated by the number of jobs running on the server, current disk utilization and the expected disk utilization. The monitor agents are autonomous and use built-in ABLE timer-event-processing to monitor the associated system resources at selected time intervals. When any of the agents detects a significant situation, it sends an event that is processed by the SysAdmin agent. After receiving the event it is then processed by an internal AbleRuleSet agent that invokes other agents to get additional information. The Task agents do not take any actions. The actions are executed by SysadminActions agent by using AbelRule agent which either prompts the user to approve the actions or automatically takes remedial actions such as killing a runaway job or deleting a system object. A rudimentary SystemAdmin client GUI shown in figure 2.3 communicates with the SysAdmin Agent using RMI connectivity that is built into the ABLE agent framework. The SysAdmin agent also sends a report of actions taken to the client for display to the user. The client can also send these findings to list of e-mail addresses so that users can keep track of system management agent actions.

**Important Observations:**

The SysAdmin agents were developed using Java language into the ABLE tool kit. The ABLE agent-editor and ABLE rule set editors are used to develop and test the agents along with their associated rule sets defining their behavior. The application that runs on a single client system can be built using the distributed RMI capability and can monitor multiple server systems.

2. **A diagnostic application using ABLE :**

This demonstrates the second application that uses ABLE technology in IBM products in the area of server diagnostics. The I-Series electronics support team is constructing a set of ABLE agents to perform data collection, problem determination and problem source identification tasks. When the customer call comes into to the support-center, the customer-support representative asks a series of questions describing the situation, symptoms and other relevant information. Later one or more ABLE agents are dispatched to the customer i-Series machine.

**Important Observation:** Multi-agent-approach can be used to develop Diagnostic agents incrementally to resolve the most common and difficult problems first and over time, can be combined to cover more and more of the problem space. ABLE Rule language can also be used to define diagnostic reasoning for additional flexibility.
3. Auto-tune Agent for Apache web servers using ABLE:

Auto-tune agent was developed to address the basic operation namely, closed-control-loop, of autonomic system, where the status of target system is monitored, compared to the desired goal state, and then adjusted to the required goal state. Based on ABLE Auto-tune agent mechanism, a multi-agent feedback control system was developed for automatically tuning the Apache Web server parameters. Basically, the Apache tuning work is performed by the system administrator and the objective of tuning is to maintain the system CPU and memory at the desired level to avoid overload or reserve resources for other applications. The tuning work must be done frequently because the relationships between CPU, Memory utilization level and the available tuning parameters are affected by the work-load and the work load can vary over time. To automate Apache server tuning process, three Auto-tune agents were developed for automatic feedback control system and deployed for tuning the Apache web server parameters.

Important Observations:

1. Series of agents were designed that play a role of intelligent nodes in an autonomic computing system.
2. These intelligent agents manage storage devices, operating system, network resources etc.,
3. At some point in the network there is a requirement of relatively simple agents dominated by reflexive behavior whereas at higher level there is requirement of complex reactive behavior, learning and adaptation.
4. The paper describes the functionality provided in the ABLE tool-kit and demonstrates the utility in the real-time applications.
5. ABLE agents are applied for multiple problems in system management, event processing, performance of the system and system health monitoring.
6. The possibility of evolving new technique to build a truly autonomic computing system has not been studied.
7. A new mechanism to expand the depth and breadth of intelligent behaviors of the agents as well as the entire autonomic computing system needs further investigation.

2.3.3 Kinesthetic eXtreme Autonomic loop

In continuation to the implementation of a autonomic solutions, the complete autonomic loop called Kinesthetic eXtreme [19] have been developed. The work was motivated by the problem of adding autonomic properties to the existing system which were not designed with
autonomous properties by adding monitoring sensors, API and monitoring functionalities. It mainly focuses on the collection and processing of monitoring data from the existing systems and execution of adaptation and repairs rather than algorithms and policies for adaptation planning. This concept was demonstrated on two case-studies namely, instant-messaging service, and Adaptive-streaming of multimedia data-failure-detection and recovery plus load balancing in a geographical Information System called GeoWorlds.

2.3.4 StarMX framework for Self managed Java Based Systems

Asadollahi R. et al., have proposed a StarMX, an open source project [20] which aims to address the simplification of development of self managing systems by reducing the design and implementation effort. Also, it supports autonomic behavior by providing runtime services by incorporating the JMX technology and policy engine.

![StarMax framework Diagram](Image)

Figure 2.4 StarMax framework
Source: Asadollahi R [20]

The Figure 2.4 shows the high level static view of StarMax framework. It consists of two main components namely execution engine and service layer.

- **Execution Engine** – This component automates the self-managing operations. It enables the autonomic manager and executes the management-logic defined by the application developer to adapt the system with its current situations using service provided by the service layer. There are two key components for the execution-
engine in the framework architecture. These include Process and Execution chain. The process is a building block for the autonomic manager that represents a single function or a multiple functions of an Autonomic manager as suggested by IBM’s MAPE-K loop. The execution chains allow defines all MAPE loop entities in a single process that is formed by chaining the processes that acts as autonomic manager. There is an activation mechanism which when activated; the processes in the chain are executed sequentially. The process is defined either by policy language or by java code org.starms.core.process interface.

b. **Service Layer** - This component in the framework provides several services that are used to enable the behavior of the execution-engine. Also, the framework provides them as APIs. The service layer includes the following services.

1. **Lookup:** It provides mechanism to access the anchor objects for the processes and responsible for preparing and finding the anchor object instance.

2. **Proxy Generation:** It creates a proxy-object dynamically during the lookup-time.

3. **Activation Mechanism:** It defines the techniques for triggering the execution chains either using the time-based method or event-based method.

4. **Caching:** It improves the performance of the look up service by holding the previously accessed anchor-objects and is able to detect registration and deregistration to invalidate the cache at appropriate time.

5. **Memory Scopes**: it provides repository facilities used by the processes to store data for the future usages or to exchange data with other processes. This service enables data-communication among autonomic manages.

6. **Data gathering**: It collects statistical data about the execution of each process or execution chain and this is made available to processes to adjust the behavior.

7. **Logging**: This provides the logging facility to record the events into a log-file which can be analyzed by the administrator.

The StarMx is demonstrated on TPC-W, a web based application developed for an online bookstore system that focuses on how to keep the system available and maintain its operating performance when the number of users increases. i.e., the system is enabled with self-optimizing capability to tune itself dynamically.
Important Observations:

1. StarMx is a generic configurable management framework employed to facilitate in-cooperating Self-CHOP properties in computing system and provide runtime services for self-managing activities on Java-based applications.
2. It provides rich set of features for developing and operating self-managing applications.
3. It provides a flexible approach for creating the management closed-loop that supports IBM’s MAPE-K framework, which is the key part of autonomic system.
4. StarMx is based on standards and well-established principles and has no dependency on application characteristics.
5. StarMX is designed for Java-based systems that incorporates Java Management Extensions (JMX) technology and is capable of integrating with various policy/rule engines.
6. The capabilities of StarMx framework need to be further enhanced under complex scenarios.
7. A new technique that has the ability to manage the StarMx framework and its properties dynamically needs to be investigated.
8. A new framework that supports the Web Services Distributed Management (WSDM) needs to be explored.

2.4 Problem Determination using Self-Managing Autonomic Technology

The problem determination scenario is a process in the autonomic computing that represents simple self-managing system [21] that uses an intelligent control loop to collect system information, from which autonomic manager analyzes and plans the appropriate response actions and subsequently makes necessary adjustments to resolve the problem. The following section describes the complexity involved in the problem determination scenario.
Figure 2.5 presents today’s Complex IT infrastructure and computing environment which is thought of as collection of resources such as hardware, software, data-management technology, networking technology and technology services combined together to perform a specific set of functions for business applications. IT professionals in the industry are performing their key tasks and processes to create powerful computing systems to make individuals and business more productive. As the software systems and technology become more complex, it becomes more difficult for the IT professionals to manage the software components and hence they fail to focus on the services that they provide to customers, suppliers, employees, and business partners. It means that the growing complexity of the IT infrastructure limits the benefit of information technology to end-customers.

The software product business has taken noticeable growth in the last few years. This is due to reasons like- acceleration in software business activity, improvement in the talent and support ecosystem, innovations in technology, delivery business-models and increased adaptation of IT by enterprises. Figure 2.6, shows the expected export-growth of Indian software product that would be increased to INR 34,400 billion, over the years 2011-2016 with a growth rate of 12% y-o-y.
Figure 2.6 Software product growths
Source: Research report by CII and Boston consulting

Figure 2.7, shows the general computing environment that supports today’s business environment. It is comprised of diverse set of software products which are running on certain hardware platforms. These software products which support various applications have their own log-file for recording the working footprints of a product occur at various instances of time. Identifying the anomalies in the software product is called problem-determination. Once the problem is determined, system administrator uses some tools and technologies to correct the errors for the smooth running of computing system.

Figure 2.7 General Computing environment and problem determination scenario
Source: IBM’s white paper [21]
In the problem-determination scenario, every software product has its own log-file format and all files are belonging to different vendors which are heterogeneous in nature. This heterogeneity in the log-file format makes the administrator's job more difficult in writing management-tools that enhance the complexity issues of problem determination. This is a major obstacle in developing the autonomic computing system that identifies symptoms and initiates the corrective measure to fix the problem.

As can be seen from figure 2.8 and 2.9, the two log-file formats are diverse in nature, though they belong to same category of application. In order to take the benefits of autonomic computing and to address the heterogeneity of log-files, IBM has introduced the concept of writing a code called adapter for every product to convert its log event recorded in the log-file to Common Base event (CBE) logs, so that the software product can effectively be monitored by the autonomic manager for runtime critical errors.

<table>
<thead>
<tr>
<th>Software: Microsoft Internet Information Services 5.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Version: 1.0</td>
</tr>
<tr>
<td>#Date: 2010-07-07 10:49:14</td>
</tr>
<tr>
<td>10:49:14 127.0.0.1 - GET /iisstart.asp 302 19453</td>
</tr>
<tr>
<td>10:49:14 127.0.0.1 - GET /localstart.asp 401 156</td>
</tr>
<tr>
<td>10:49:19 127.0.0.1 SHARMA-9690DD22\anjana GET /localstart.asp 200 3078</td>
</tr>
<tr>
<td>10:49:22 127.0.0.1 - GET /ishelp/Default.htm 200 219</td>
</tr>
<tr>
<td>10:49:22 127.0.0.1 - GET /warning.gif 200 203</td>
</tr>
<tr>
<td>10:49:22 127.0.0.1 - GET /web.gif 200 156</td>
</tr>
<tr>
<td>10:49:22 127.0.0.1 - GET /help.gif 200 47</td>
</tr>
<tr>
<td>10:49:22 127.0.0.1 - GET /mmc.gif 200 47</td>
</tr>
<tr>
<td>10:49:22 127.0.0.1 - GET /winXP.gif 200 109</td>
</tr>
<tr>
<td>10:49:22 127.0.0.1 - GET /print.gif 200 125</td>
</tr>
</tbody>
</table>

Figure 2.8 IIS log file instances
Figure 2.9 Apache log file instances

Figure 2.10 shows the how the complexity issue in problem-determination can be addressed during the process of autonomic computing.

In this environment, the Resource manager consists of various components such as applications, database, application-server, servers, storage device, and networks. These components are to be monitored as a part of troubleshooting process when a problem occurs.
Every component has its individual log-file in its own format in various locations. As discussed earlier, the IBM autonomic computing architecture includes adapters to translate log events into the standard format called Common Base event (CBE) [22] [23]. The contents of CBE Logs are then used by the Autonomic manager to look for problems and take necessary actions if needed. The figure 2.10, also illustrates an autonomic manager engine that automates this process.

As a part autonomic computing implementation IBM has developed the Log and Trace analyzer which enables the reading of logs in the Common Base format, correlating the logs based on different criteria (for example, time-based or field-based (such as URLs)) and viewing the correlated log records. Autonomic manager has a source of knowledge in the form of symptom service to take the best possible action when it discovers a problem. The symptom service includes a symptom-database that contains information about how to detect patterns that indicate problems, how to diagnose a specific problem that has occurred, and how to resolve that specific problem. The symptom database also contains a standardized set of interfaces and data formats that facilitate the determination of actionable causes from problem data.

Topol B, et al. [24], have discussed the first attempt made by IBM and SunMicrosystems, in developing the self-healing system. Basically, Self-healing is the process in Autonomic Computing that releases people from discovering, recovering and failures. Self-healing systems are capable of healing themselves at runtime in response to changing environments by prompt execution of remedial actions. In recent years, many studies have been attempted by IBM and SUNMicrosystems to develop self-healing systems The Self-healing features of IBM and Sun Microsystems are as follows:

2.4.1 Self healing features of IBM’s autonomic Computing, Tools and technologies.

As a part of Self healing system IBM has proposed an Adaptive Service Framework (ASF) which is applied to CISCO in the form of Self-adaptive-behavior. The ASF works as follows:

1. Monitoring Component of the framework monitors the log-events of the managed resources and passes to Adapter.
2. Adapter translates the generated log-events of the component into a Common Base event (CBE) which is the standard representation of a log-file.
3. Autonomic Manager analyzes the CBE logs to fetch the Symptom and retrieves the healing method by means of the symptom rule and policy engine. The healing method is
applied to the applicable component and the feedback through resource manager the healing by itself is enabled.

4. If there is critical problem encountered by the component and cannot be solved easily, the autonomic manager sends message to the support service provider/ vendor for requesting a solution.

**Important Observations:**

1. The possibility of evolving new technology to develop an efficient system-management and error-recovery whenever critical error occurs during runtime environment has not been attempted.
2. A new mechanism for every component in the Adaptive Service Framework that eliminates the use of many Adapters to translate the log-events to CBE logs has not been addressed.
3. A new technique for Adaptive Service Framework that reduces high healing-time and during emergency Situation the immediate action execution has not been investigated.

As the work of autonomic computing progresses the IBM has developed the autonomic computing toolkit which is a collection of tools and technologies, scenarios and documentation. The components of the toolkit are used by the users to learn the autonomic behavior of their products and systems. It also facilitates the development of autonomic manager for problem determination, enabling the applications and other resources to be managed so that self-configuring, self-healing, self-optimizing and self-protection can be incorporated. These components include Autonomic management engine, Resource Model builder, common-base-event, Generic log adapter, Log-trace-Analyzer and Symptom database.

**2.4.2 Autonomic management engine**

This component of IBM toolkit is used develop and adaptive level of self-management, autonomic maturity level 4. It is an example of implementation of autonomic manager, i.e. consists of four parts of MAPE-K (monitor, analyze, plan and execute) loop reference model presented in figure 1. It can be used as a stand-alone management engine or it can be embedded into an application. The stand-alone method uses SARA (Simple Agent Reference Application) which provides an environment to run resource models. In case of embedded method, the application is responsible for loading the resource models, initializing the instances of resource models and providing event-links and action plug-ins as required.
2.4.3 Resource Model Builder:

This component defines and provides logic to implement the phases of MAPE-K loop and performs the self-management and monitoring capabilities as developed by the IBM Tivoli Resource Model builder. Basically, it contains metrics, events, thresholds and parameters that are used to determine critical errors of IT resources along with specifications for corrective actions in case of error conditions. The resource model retrieves the information about the monitored resources through interface and acts as a bridge between sensors and effectors in the autonomic MAPE-K architecture.

2.4.4 Generic Log Adapter (GLA):

This component is used to translate product-log-messages into a Common Base event format so that the product can become a managed resource. Basically it is a technology to help products to adapt to the autonomic reference architecture without requiring the product to change creation of their log-file. Figure 2.11, describes the conversion process of events from the application log to the Common base event. During the process of conversion the GLA uses the information defined in the adapter configuration file which contains series of components that describe the conversion rules for the associated log-file.

![Diagram of GLA to generate Common Base event](image)

2.4.5 Log Trace Analyzer (LTA):

This component is used as managed level 2 of autonomic manager, it is responsible for viewing, analysis and correlation of log-files. It contains a log analysis engine and correlation engine. The log analysis engine takes an event recorded in the common base event, matches this event based on predefined rules against the symptom database and returns the solutions and directives that match the symptom.
2.4.6 Symptom data Base or Symptom Catalog:

Symptom data base is a mechanism used by the IBM LTA for autonomic computing that helps IT administrator and support professionals for easy problem determination. It is a collection of XML files that contains records of unwanted happenings and problem indications that could occur in the operation of the software or hardware infrastructure. For every symptom it also contains the cause of the problem and recommended solution for the problem. Normally, the symptom database is created by the development team from the scratch using unstructured data generated by the test and support team.

The authors Chilukuri S. K et al., India software Lab, IBM software group [25], have proposed a tool and technique to build the symptom database that facilitates the conversion of problem data to a symptom database, later used by LTA.

In the software development process the IT companies accumulate huge amount of data that contains defect records and resolutions, queries or responses to the customer, into database as per their need. This data is classified as problem data. The information in the problem data used for proactive problem diagnosis and resolution. The database which is called problem-repositories contains huge amount of content, collected over a period of time by product management team. This makes lookup operation difficult and time consuming for the IT staff. Also, problem-data lacks structured information to be effectively used by the IT management tools. In order to simplify and automate the process of lookup the IBM Autonomic toolkit provides a tool called LTA that collects events occurring on a product from various sources like log-data, capture packets over network etc. The LTA has an analysis feature to correlate the collected events against symptom database and recommends the solution.

The symptom database needs to be developed from the information collected during the development of the product which requires lot of time and effort. Also, to enable the reuse of problem repositories the authors have proposed an experimental tool called Symptom Database Builder (SDB) to translate the existing problem repository to a common symptom format. Although, the authors have presented technology to build the symptom database, the complex nature and need for a huge knowledge-base to operate autonomic tools.

Important Observation:

A possibility of evolving a new mechanism to maintain and represent the symptom rules in human readable form in the symptom database has not been investigated for easy problem determination.
2.4.7 Self-healing capabilities of Autonomic Computing:

Let us consider a typical application as shown in figure 2.12, which uses a database for data storage. Both the database and application run on independent servers. The application accesses the database through a Java Database Connectivity (JDBC) connection. The case study demonstrates implementation of self-managed autonomic technology and tools to monitor the IT environment and problem correction actions when autonomic managers detect a failure in the application component.

Using IBM Autonomic computing toolkit the components like AME, resource model GLA are developed and deployed on both the application and Database node [26]. Figure 2.12 shows the Autonomic application scenario environment with all the components deployed. The following sequence of steps may be executed to demonstrate the automation of self-healing capability into an IT environment.

1. All the components of application are up and running on their respective nodes.
2. An error such as terminating the database instance execution so that the data insertion is disabled for the application is introduced in the IT environment.
3. The application records the log event into its log file reporting a connection failure to the database.
4. A resource model deployed in the application node and running on the autonomic manager detects symptom or problem reported by the application. As an initial step to recover the error, the resource model signals the application to put data insertion on hold. The application then keeps the task on hold and waits until the problem is resolved.
5. IBM DB2 logs an event in its log file stating that the database instance is down.
6. The GLA which monitors the IBM DB2 log-file converts the log events into Common Base events and stores them into a CBE DB2 log file.
7. A resource model deployed in the database node and running on the autonomic manager monitors the CBE DB2 log-file and analyses the event report, database instance down and takes corrective action which is to restart the database instance.
8. Once the action i.e., restarting the database instance takes place, the application resumes its normal work.

35
The survey data shown in figure 9 there is a tremendous growth in product development, hence the IBM’s approach of writing separate code called Adapter that converts heterogeneous log instances into a standard format called CBE is tedious that results into the following research gaps.

**Important Observations:**

1. A simple tool to manage heterogeneity in log-file format of the software product and alleviate the need for writing the adapter for every product has not been addressed.

2. A new framework to implement autonomic solutions that supports an easier platform for administrator to maintain and write the symptom rules for monitoring the applications and also associated actions that are needed to be executed when that rule is breached is not been investigated.

**Figure 2.12 Autonomic Self-healing Environments**

*Source: M. Edson [15]*
2.5 Self healing features of Sun Microsystems autonomic Computing

As a part Self-healing system, the SUN Microsystems has proposed Predictive self-healing in Solaris™ 10 Operating System (OS) that introduces a new architecture for building and deploying systems and services capable of Predictive Self-Healing [27]. This technology enables Sun™ systems to accurately predict component failures and mitigate many serious problems before they actually occur. Solaris Fault Manager and Solaris Service Manager are the two main components of Predictive Self-Healing. Solaris Fault Manager receives data relating to hardware and software errors and automatically diagnoses the underlying problem. Once diagnosed, Solaris Fault Manager automatically responds by off lining faulty components. Solaris Service Manager makes services, rather than processes, into first-class citizens, permitting automatic self-healing. Base Solaris services have service descriptions which include full dependency information for start, stop, and restart; applications can easily be converted to run under Solaris Service Manager

Important Observations:

1. It simplifies the task of composing, configuring and deploying the solutions.
2. It maximizes the service and system availability by automatically diagnosing and recovering from faults.
3. It guides system administrators through tasks that require human intervention and provides links for continuous updates of repair procedures and documentation.
5. Simplified administration model for managing services, reducing cost of ownership.
6. Fast and easy repair of problems with links to knowledge articles.
7. Scalable architecture can be rapidly upgraded and adapted to new problems without requiring downtime.

A few studies focused autonomic capability in terms of Self-healing. In continuation to the work the authors, Yoo G, et al., [28] have proposed Hybrid prediction Model that predicts the proper algorithms namely ID3 algorithm, Fuzzy Inference Algorithm, Fuzzy Neural network algorithm, and Bayesian Network, depending on system-situations and historic data, fuzzy inference and inverse inference for improving reliability in self healing system.
Important Observations:

1. The predictive model provides more effective and more rapid monitoring mechanism, as it predicts the four algorithms namely, ID3 algorithm, Fuzzy Inference Algorithm, Fuzzy Neural network algorithm, and Bayesian Network, for self-healing in an autonomic system.
2. The prediction of algorithm is based on system request and situation.
3. A new context-aware-prediction model to improve monitoring ability has not been investigated.
4. A possibility of evolving new technique for Self-growing Prediction model which models the fuzzy neural network that has learning and context-aware-ability has not been investigated.

2.6 Autonomic computing functionalities for Wireless Sensor Networks

Currently, many Wireless Sensor Networks have self-configuration features at protocol and hardware level [29], [30] which are used to detect anomalies in network and adjust performance for certain tasks. Wireless sensor networks (WSN’s) are becoming popular in military and civilian applications such as surveillance, monitoring, disaster recovery, home automation and many others. Almost any sensor network application requires some form of self-configuration and autonomic functionality.

One of the components of wireless sensor node is the power source which can be a battery. The wireless sensor node is a microelectronic device which can only be equipped with a limited power source, over the remote inaccessible place with less human control and existence, power source play critical role in the survival of sensor nodes. Power source should be intelligently divided over sensing, computation and communication stages as per requirement. Sensors can hibernate when inactive. Lot of current researchers are focusing on power-aware protocols and algorithms for wireless sensor-networks. Following section gives an insight into the work carried out in this direction.

The reducing Power Consumption for Mobile Multimedia Handsets concept has been discussed in the paper [31]. Also they have proposed three techniques called as wake-up-techniques which reduce the battery power consumption in a mobile multimedia handset. Using these approaches the system is switched into the sleep-mode when the memory queue for the packet arrivals is empty. Various wake-up mechanisms are considered based on these concepts. First one among the 3 approaches is the threshold approach mechanism. Here, depending on the
number of packets which have arrived in the memory queue, if it is above a threshold, the system is switched on else it remains in the sleep-mode. The second approach known as vacation approach switches on the system, on completion of the vacation time. The third approach called the hybrid approach combines the above two approaches. In this approach the system is turned on when either the length of the memory queue goes above the threshold level or at the end of vacation time. A comparative study of these approaches sheds light on some of the important system performance parameters. For example, it is seen that the threshold approach comes with reduction in the switch-on rate of the system whereas the vacation approach has the lowest mean-packet waiting time.

**Important Observations:**

1. In order to maintain lower value for both the system switch-on rate and the mean packet waiting time, the hybrid approach needs to be selected.
2. It is indicated that there is a possibility to obtain a set of threshold values in order to determine a small switch-on rate and the probability for packet dropping in case of threshold approach while such a possibility does not exist for vacation-approach to obtain the range of vacation-time.

The authors Q. Tie et al. have focused on optimizing the performance of a large-scale wireless sensor networks for improved transmission QoS [32], when the hardware consumption is limited. This paper proposes a novel evaluation scheme based on packet buffer-capacity of nodes using queuing network-model. Here the packet buffer-capacity parameter of the queue is analyzed for each node type when it is in the best working condition. This method expands the queuing network model into the equivalent queuing-network model by adding holding nodes to the existing network for evaluating the congestion condition within the queuing network, and to obtain effective arrival and transmission rates. This work establishes an M/M/1/N type open queuing network model with holding nodes for WSNs and includes designing of approximate iterative algorithms to calculate arrival rates when the system reaches a steady-state. Experimental results indicate that the model is consistent with the real world data.

**Important Observations:**

1. The scheme discusses modeling for only a single-server model in WSN and proposes a method for calculating the packet buffer-capacity size of nodes.
2. Recent research-focus is on the convergence of multiple processor nodes that can be used for M/M/m/N queues, which are also multi-server queues.
3. In case of large-scale WSNs, to explore the possibility of evolving the new technique to prioritize the clusters so that there is an effective improvement in the performance of WSNs.

The authors Xu. Y et al., have carried out an analysis on the fundamental lower bound [33] for node buffer size in intermittently connected wireless network. Due to some external constraints there is a possibility of occurrence of node inactivity which is the main cause for intermittent connectivity of the network. It is found that in a static random network, each node keeps a constant message generation rate. In such networks, the buffer occupation in each node does not approach zero instead of having infinite network capacity and node processing speed. An in-detail analysis has been done on buffer occupation when the channel capacity is infinite, and the results can be viewed as a lower bound for networks with finite channel capacity. The analysis shows that when the probability of node inactivity is below the critical value, the state of the network is supercritical and the fundamental achievable lower bound of node buffer size is $\Theta(1)$, i.e. the minimum node-buffer-size requirements are asymptotically independent of the size of the network and when the probability of node inactivity is greater than the critical value, the network state is subcritical, and the achievable lower bound on node-buffer-size shoots up as the network expands, with the order of $\Theta(\sqrt{n})$.

The authors of the paper [34] have focused on the aspect of peak power consumption by the hardware components. This affects the power supply, packaging and the cooling requirements of the systems hardware equipments. Higher peak power consumption by the hardware leads to bulky and expensive systems. If ever the components and systems actually require peak power, then it becomes necessary to put a limit on power consumption to a less-than-peak power budget. This, in turn will lead to intelligent provisioning of the power supplies, packaging and cooling infrastructures of the hardware components. This paper deals with the study of dynamic approaches to limit the power consumption by the main memories. It proposes 4 techniques namely Knapsack, LRU-Greedy, LRU-Smooth, and LRU-Ordered in which the power-states of the memory devices are adjusted as a function of load on the memory subsystem. The simulations carried out from 3 benchmark applications prove that these techniques are consistent in limiting the power to a pre-established budget accompanied by very low performance degradation.
Important Observations:

1. The simulation results indicate that the performance of limiting power using these techniques is same as that of the energy conservation approach used in state-of-the-art techniques exclusively designed for performance-aware energy management.

2. A new algorithm to address the issues related to selection of ideal power-budget in case of different scenarios and studying the effect of greater concurrency in memory accesses in the context of chip multiprocessors has not been developed.

In continuation to the work related to power consumption of wireless sensor nodes the authors Han-Lin Li, et al. [35] have discussed Energy-Aware Flash Memory Management in Virtual Memory System which revisits the design of virtual memory system using flash memory for many portable devices due to its improvements in storage capacity, reliability and lower power consumption along with its limitations. This paper in particular, concentrates on the energy efficient aspect as power is the first-order design consideration for embedded systems. Frequent write operations into the flash memory lead to frequent garbage collection thus incurring significant energy overhead. This is due to the write-once feature of the flash memory. In order to address this issue of increased energy consumption and to prevent excess energy lost, the authors have proposed 3 methods to reduce the number of writes occurring to the flash memory. They are HotCache scheme, Sub-paging technique and Duplication-aware garbage collection method. In the HotCache scheme, an SRAM cache is introduced in order to buffer frequent writes. In the sub-paging technique, the pages are partitioned into subunits and only dirty pages are written into the flash memory when page-fault occurs. The duplication-aware garbage collection method makes use of data-redundancy that exists between the flash memory and the main memory to bring reduction in the writes which occur due to garbage collection.

Important Observations:

1. Intra-page locality, a type of data locality, is an inherent feature of the flash memory and is responsible for data allocation. This property of the flash memory should be carefully preserved while data is written from the storage buffer to flash memory. Destruction of this property leads to increase in the energy consumption by the flash memory.

2. Experiments have been carried out using the 3 techniques and the results show an average energy reduction of 42.2% using the combination of the 3 techniques.

3. A new technique that addresses the reduction of power consumption of memory of a sensor node that is not performing any useful task has not been addressed.
2.7 Autonomic Computing by other organizations

With growing complexity in the software and hardware and the limited number of IT professional in the company, IBM has the biggest strength in the field of Autonomic computing. Although not much progress is achieved in other organizations in this field, the following section describes the contributions of other organizations in the field of autonomic-computing.


   The Hewlett-Packard’s [8] approach to Autonomic computing has produced an Adaptive Enterprise strategy that provides tools, services, and products that deliver IT service levels that match the flow of real-time business activities or changes whenever needed. It mainly focuses on aligning IT resource allocation with changing business needs with as aim to create service-oriented architecture.

2. Microsoft –Dynamic Systems Initiative

   The Autonomic Computing initiative of Microsoft has produced a self-managing Dynamic Systems Initiative (DSI) [36] to help IT teams capture and use knowledge (relevant to distributed systems) to design more manageable systems and automate ongoing operations, resulting in reduced costs.

2.8 Shortcomings and Research Gaps identified:

1. To explore the possibility of evolving the new mechanism to maintain the symptom rules in symptom database (not in XML files) that helps system administrator for easy problem determination and taking necessary action which supports self-healing capability into computing system.

2. To develop a simple tool to manage heterogeneity in log-file format of the software product and alleviate the need for writing the adapter for every product.

3. To develop a new framework to implement autonomic solutions that support an easier platform for administrator to maintain and build the symptom rules for monitoring the applications and also associated actions that are needed to be executed when that rule is breached.

4. To develop a new technique that addresses the reduction of power consumption of memory of a sensor node that is not performing any useful task.
Summary

The autonomic manager consists of monitoring, analysis, plan, and execute components. The Autonomic Computing Toolkit is relatively a simple *black-box* implementation of these components called Autonomic Management Engine (AME). It is primarily intended to provide an implementation that can be used to test the interfaces and other components of the autonomic environment; AME monitors system resources using resource models, sends aggregated events, and performs corrective actions for problems. AME constantly monitors the system looking for events to handle. AME is available in the Autonomic Computing Toolkit in the AME bundle. Several shortcomings were identified in the IBM’s Autonomic computing approach and few issues were addressed as part of the recent research work.