Abstract: - Autonomic Computing is relatively new concept in the field of computing sciences. Autonomic computing brings different areas of computing science together with the idea of enabling computing systems with self-adaptive and self-managed features. With the addition of new elements in the system, the level of maintenance in terms of both hardware and software will be increased. However in case of self-managed system, addition of autonomic element may increase the complexity but maintenance level will be reduced. This paper is an attempt to explore the inverse relationship between maintenance and complexity in case of autonomic system at the software level. It also proposes a framework to assess the maintenance level of autonomic system using fuzzy logic technique. One of the major factor effects on maintenance of the autonomic system is examined.

Key-Words: - autonomic computing, maintenance, autonomicity, fuzzy logic.

1 Introduction
With the fast development of Web technology, lead in the increase of the requirement of the distributed, interoperability, reusability and adaptability enabled products, to cope with new environmental changes that automatically reduce developing costs. This make the system more complex on the basis of heterogeneity of the distributed communication system, maintenance cost, interconnectivity among the software based applications and dynamism. The systems are also become less secure due to its popularity and great usability. Autonomic Computing is the solution to cope with this unmanageable complexity in the IT industry that has motivated researchers of this field to work upon it and investigate. The increase in the usability of the distributed network, acts as vulnerability for the attacks. There are many ways of attacking such kind of vulnerable areas. So there is a need of making the system self-healing and self-protecting. To make a system universal acceptable, there is also a need to make the system self-adaptive, self-optimized and self-configured. All these motives would be grouped together in the one unit called Autonomic Computing. But the addition of new autonomic element will affect the level of maintenance in two different aspects, hardware and software. The autonomic system consists of three levels architecture. The first level is the autonomic agent, at the second level, sensors and effectors which work as a connector between autonomic agent and the managed element which is present at the third level.

Fig. 1 Autonomic System Level

The aim of autonomic system is to handle the management activities. This can be achieved by enabling the system to automatically perform according to the environmental changes for the optimization. This help in reducing the human intervention during the maintenance process which involve most of the system development cost and effort. The maintenance for distributed system is
complex in their working and is not reliable. The autonomic computing system is one approach which will increase the complexity but reduce the maintenance of the autonomic system which is reliable and effective. But with this there is a need to assess the increment in the level of complexity due to involvement of autonomic agent within the system.

The paper is organized in the five sections. The concept of Autonomic Computing provided in the 1st section. The 2nd section includes overview of autonomic system continued with its proposed architecture and its self-control loop. Related work mentioned in the 3rd section. 4th section will continue with description of the proposed work that will be shown using fuzzy logic rules. 5th section includes conclusion and related future work in the field of AC.

2 Autonomic Systems: Architecture Overview

The software development involves few steps and the procedure is called as SDLC. The design phase focuses on functionalities implementation. The system designing includes several assets and components. Some systems are developed to handle the runtime failures and errors. However, autonomic system designing activities are different because its functionalities are different. The architecture for autonomic system includes component at the functionality level but it does not includes application specified functionalities.[17][18]

The autonomic system involves functionalities similar to non-autonomic system but these functionalities are self adaptive by the system. These four features are club together and named as self management that continuously monitor system activities, and check component updates, install new features, reconfigure the system settings if required, and perform testing to ensure that the system is working accurately. When the system identifies problems, it activates automatic problem-determination algorithms to resolve the errors [12].

2.1 Architecture of Autonomic System

The autonomicity concept is not yet implemented as the whole in the distributed network but its characteristics are well defined and some are implemented in IBM development of DB2 like software [6].

So its architecture on the basis of its working is defined by IBM and other IT companies. The IBM has given some policies of Autonomic System as the working principle of AS. These policies are:

- system must know itself
- system must configure and reconfigure according to environmental changes
- system must optimize its resource allocation
- system must be able in self-protection
- system must be compatible with changes and different system standards

The four well defined characteristics make the Autonomic Computing enabled system. An Autonomic System consists of two main entities or you can say AS work in two parts:

- Autonomic Agent or Manager
- Managed Element

Autonomic systems consist of one or more managed elements and each managed element coupled with a autonomic manager that controls all the activities. The working of the Autonomic System includes Monitor, Knowledge, Analyze, Plan, and Execute [3].
These forms a MAPE loop with knowledge as add on factor which help in taking decision on the basis of predefined knowledge and history data[4]. There are sensors and effectors which connect the Autonomic Agent (AA) with the managed element. The autonomic system works in self-control loop within a system and solve the undesirable situation in the system [5]. The self-control of the loop passes in such a way:

- The AA monitored the Managed element 24*7
- The information is gathered by autonomic agent or manager using the sensors.
- The gathered information is analyze and planned accordingly
- After that AA take the major steps during the execution of the problem
- Finally AA provides the solution back to the managed element by the effectors.

In a distributed system, these entities will work as subunit of each system which is arranged in network and solves the problems. The information system formed after introducing the autonomic concept will manage the content dynamically using links which is formed by query process. The architecture of the autonomic system is lightweight and will be suitable for internet based application, software, mobile and sensitized environment. The linking method is reliable and can be used with the knowledge based procedure. The overall structure can be used as a part of different techniques like neural network or globally it can be implemented in the distributed system. [9]

3 Literature Review

Due to increase in software complexity in the IT sector, IBM released a manifesto in 2001. The complexity was due to the weigh in billions of lines of code that require skilled software developers and expertise to reconfigure and maintain the software programs for better results. The document also highlighted the increase in new level of complexity due to extend of several heterogeneous corporate-wide computing system beyond company boundaries. The systems’ complexity appears to be unmanageable within the limits of human capabilities. As system becomes more diverse and interconnecting, developers are less capable to rectify problem based components and leave such problems to be handled at runtime. This makes the system complex and massive. And there will be no way to handle the system timely by providing better responses to the rapid changes and conflicts.

The only solution remaining is the autonomic computation—systems that automatically manage themselves under high-level policies and objectives. The Paul Horn, the senior vice president of IBM research, introduced the autonomic idea in the computation world at Harvard University in a March 2001. P. Horn deliberately gave this autonomic term with the context of human body system. The nervous system of human configure heart-bit rate and body temperature with changes, heal the cuts and wound, optimize the body’s requirement by providing the other way resources, and also protect the body from infection. The same idea introduced in the Autonomic computing that involve less human intervention. To make it real and develop it the higher level will take a long time, worldwide effort by the developers and researchers.

For the first time, Jeffrey O Kephart et al. [1] has discussed all the main features or property of AS in detail with a proposed architecture of AS. There are many challenges in front of the IT industry which comes out by comparing the current computation and handling techniques with the new concept of computation that is Autonomic Computing. Autonomic Computing involved MAPE-K loop which monitor, diagnose and then rectify the problems using pre-determined knowledge data. Over the time, human will only take administrative level decisions, which the system can perform automatically. Ultimately, end users and system administrators will get the benefits of autonomic computing system. The journey toward completely autonomic computing system will take many years. Many applications have been developed under the autonomic system’s field by different IT companies.

McCann et al. [4] in their research discovered two approaches to compare different autonomic systems. They listed some metrics for the autonomic systems for the evaluation and comparison. Some metrics are: Quality of Service, degree of autonomy, adaptivity, response time, robustness and stabilization. The two approaches for autonomic systems are: tightly coupled autonomic system and decoupled autonomic system. These approaches have common concepts. Again both approaches need two kinds of elements: i) for the target system functionality implementation, and ii) to add solution for self-management in the system. These elements describe 2-level architecture: inter-relationship and
intra-element relationship. The first level deals with the relationships among the elements in a particular manner. The level describes the global aspects of the autonomicity, such as overall configuration [4]. Infrastructure elements provide documentation service; monitor the system and aggregation of valid information, interconnectivity and negotiation.

Martin R. et al. [22] discussed the requirement for measuring the complexity of the program code. They compared their proposed metrics with other metrics and concluded that the most notable complexity was the Cyclomatic measure.

Gill and Kemerer [7] done research in this field and they highlighted the requirement of software based metrics in relation with complexity for the assessment of maintenance of the software. Cyclomatic Complexity based metrics, in which linear independent program code is measure from the program control graph. Weyuker [8] research work again described a set of characteristics of software complexity and they provided an evaluation and comparison among four complexity as Cyclomatic complexity, Halstead’s programming effort, LOC, and information flow complexity.

4 PROPOSED WORK
Every sector like industries, government, commerce all depend on the information system which refers to the operation, structure and high level policies and objectives fall under strategy of an organization. Information system management will be done in both the direction; one is within the system and among the different systems. Complexity introduces due to both the factors and it affects productivity, efficiency and competitiveness. As a result development increases and increases the maintenance requirement. [14]

According to ISO/IEC 9126 quality model, maintenance is evaluated by calculating the analyzability, changeability, stability, testability and maintenance compliances degree. But in case of autonomic system; the maintenance activities are quite different from non-autonomic system. Therefore, maintenance assessment based factors would be different so there is a need to identify those factors based on working of the autonomic system.

This paper provides a framework for estimating maintenance of the autonomic agent. The definition of the framework is centered on the software attributes that are explored in empirical software engineering, such as interaction complexity, interface complexity, autonomic features and function point. The model includes some major and minor factors that give overall maintenance. The present paper considers complexity as the major factor for assessing maintenance. The framework defines how to measure complexity of the autonomic agent using proposed fuzzy inferences. The model also assists software engineers in the interpretation of the data gathered from the measurement process.

4.1 Maintenance Assessment Model (MAM) Overview:
The maintenance is the most crucial part of software quality assessment and it also involves 40 to 45 percent software development cost. So the assessment of maintenance degree of any autonomic system become very important part for deciding whether any particular software could be maintain under provided cost or not. It also helps in deciding its application area [11].

The Autonomic System is designed to resolve the IT based products complexity and their management manually by implementing the CHOP properties. To identify the autonomicity level, there is a need to identify the level of CHOP properties means how much a particular system is self-configured, self-healed, self-optimized and self-protected. Question arises here is: How you can examine CHOP properties level in a software product???

The problem rotates to find the answer but there are no such factors defined that give the CHOP level. To determine the autonomicity level, there is a need to check the system and within system, again check the capabilities of the system means self-features. There is no defined way to assess applied level of self-features. With the definition details and from the studies, CHOP features also follow some other factors which help in determining the autonomicity level through maintenance level. More the maintenance required less would be the autonomicity level or vice-versa. The maintenance
can be assessed on the basis of some major and minor factors.

With the following definition of the CHOP properties,

**Self-Configuring** means enabling the system to be automatically detects the updates for the configuration with the environmental changes. This definition indicates complexity and reusability factor of the autonomic agent.

**Self-Healing** means identify the problem within the system, diagnosing according to the high level objectives and reacting to problem. This definition indicates performance factor of the autonomic agent.

**Self-Optimizing** means efficiently increase allocation of the system resources and their utilization based on the different users requirement. This definition indicates reusability and performance factor of the autonomic agent.

**Self-Protecting** means protect system from failures causes and malware attacks and automatically but within the system policies and high level objectives. This definition indicates the security and performance factor of the autonomic agent.

There can be more factors that follow CHOP factors but broadly four major factors are taken at the second level of the MAM model. These four major factors (MAJ-FAC) which further divided into some minor factors (MIN-FAC). The addition of overall evaluation values will give the total maintenance and indirectly tells the autonomicity level. The four major factors are:

1. **Complexity**: complexity is the estimation of developer efforts that is needed during the development of the software, maintenance and execution of the code. [4]

2. **Reusability**: It is the estimation done on the product assets to check at what degree a component can be again used which finally reduces the development cost of the software. [10] [13].

3. **Performance**: It is the degree of process results that is calculated by processing speed, response time, efficiency, through put and resource consumption.

4. **Security**: Security means preventing the unauthorized access on data by applying security policies and firewalls.

Other than these four factors, there are minor factors that are based on the maintenance like portability, efficiency, reliability, accuracy etc. All these minor factors are classified under four major factors.

Complexity, reusability, and maintainability are interrelated and have a direct dependency among each other. If complexity factor increases then system software reusability will decreases and system maintenance will increases. Complexity increases with the increase in the dependency factor within or among the functional modules increases. On the other hand, if reusability and maintainability increase, the usage of the system will increase.[21]

In this paper, one of the major factors, complexity under which minor factor evaluation would be done using fuzzy logic rule formulation. Complexity is defined at the granularity level. The maintenance assessment for the autonomic agent will define the effort and cost required for its management. Here complexity is taken in two direction, feature based and code based. Code based means the programmer level effort would be identified. How complexity would be assess by the management team to maintain after deployment of the autonomic agent.

Code based complexity can be estimated using interaction complexity and interface complexity. Interaction complexity means complexity due to the sharing of the one asset properties by other assets. Interface complexity means when there is interdependency among the functionality within a code module. The relation between interaction and interface complexity is indirect that is, more the interaction, less would be the interface complexity. The third minor factor is the function point per autonomic feature [20].

![Fig. 4 Maintenance Assessment Model](image)

Autonomic feature other than CHOP are self-awareness, self-anticipatory. Function point per
autonomic feature means for one particular autonomic feature how many function points work. The inverse relationship is expressed using fuzzy logic and the overall effect can also be expressed using fuzzy logic. Again the complexity can be expected to be high if there are more autonomic features means if system is self-aware and self-healing properties then it requires less management whether it is complex than the non-autonomic system. Therefore the relation between complexity and no. of autonomic attributes will be direct. There is a need to define some rules for their presentation using fuzzy logic. As the architecture of the autonomic system defines the 3-level design in which autonomic agent is the top most level that control the whole system using its self-control loop based design. This self control loop autonomic agent is connected with the sensors and effectors which further connected with managed element. The autonomic agent has the CHOP features. For the implementation of these CHOP features, autonomic agent uses the self control loop that include Monitor, Analyze, Plan, Execute with some Knowledge. All these work in a connected loop, means there involve some connectivity among all these property based hardware. So the three-levels of the autonomic system are connected in tightly coupled way. First level involves interaction within itself and also connected with sensors and effectors. All these connectivity/interaction will increase the complexity level. But in our work, only software based complexity is taken. This can be represented by fuzzy logic rules.

This is the workflow diagram. The complete model implementation is shown using the diagram.

The MAM model works in the bottom-up manner. First all the minor factors will be evaluated for each major factor. Make the output for each major factors and result will help in estimating the major factors outputs. Using this result, CHOP level for particular autonomic application will be evaluated. This will together provide the maintenance level for the respective autonomic application.

4.2 Fuzzy Logic: In 1965, Lotfi Zadeh brought a new concept called “Fuzzy sets” [16]. He further extended the work on possibility theory which deal using membership functions and it is beneficial for managing vague and doubtful information. He converts this possibility theory into a formal system of mathematical logic, and introduced a new concept using natural language terms. This new logic for representing and manipulating fuzzy terms was called fuzzy logic. Fuzzy logic is a multivalve superset of Boolean (conventional) logic that deals with degrees of membership and degrees of truth. Fuzzy logic uses the logical values between 0 (completely false) and 1 (completely true). This mathematical tool helps in managing the uncertain and doubtful data.

In fuzzy logic, the membership functions (mf) have qualitative values like very low, low, medium, high, very high. Both the input and output to the mf have the values in a particular range between [0, 1]. In the proposed approach, the interaction complexity has the range [0, 1] which is divided into low, medium, high values. Similarly, Interface complexity and third input variable function point per autonomic features values is divided into low, medium, high range of [0,1]. Also, to represent weights, fuzziness triangular mf is defined. The triangular membership function has values (minimum, peak, maximum) which are defined as min ≤ peak ≤ max. The fuzzy system works in three major phases as follows [15].

- Input
- Fuzzy Rules
- Output
- Fuzzification
- De-fuzzification
- Fuzzy Inference Engine

Fig. 6 Fuzzy Rules

4.2.1 Rules Based Approach to Evaluate Maintenance Level

Estimating maintenance from complexity is not the new concept but for the autonomic system, maintenance is inversely proportional to the
complexity. Fuzzy concept applied on MAM Model help in identifying the maintenance. As the overall maintenance is evaluated by final evaluation of all the four major factors but one angle of this estimation is described by this paper. The membership functions are named as low (mf1), medium (mf2), high (mf3). The total combinations are shown in the table as follow:

Algorithm for estimating the complexity based on Rule Model. The following are the steps.

1. Perform domain analysis on Complexity of AA.
2. Identify factors due to which complexity of an AA affected.
3. Use the Dataset of Autonomic Application to evaluate the complexity based factors.

Divide the [0, 1] range into three membership function of the each factors. Define the membership functions for output which is the System Complexity. Design the fuzzy inference rules based on mf.

Provide the values to the fuzzy inference engine from the Dataset to check whether the result of most of the autonomic applications lie in which membership function of the output. Evaluate the final complexity category and relate it with the maintenance.

As maintenance and complexity has the inverse relation so final conclusion involves that

- When complexity is low, required maintenance is similar like the non-autonomic system.
- When complexity is medium, required maintenance will be medium which in between the autonomic system and non-autonomic system.
- When complexity is high, the maintenance required will again be low or medium because system automatically managed and maintain its system changes.

Dataset: For estimating complexity of the autonomic agent, various autonomic applications data have been collected from different industrial projects. The dataset includes autonomic application but not fully autonomic. Some of the autonomic features are enabled but not all. So result will be according to the applications.

The approach for estimating autonomic system complexity has been performed using MATLAB fuzzy tool. Some rules are as follows.

If (interaction complexity is low (0.12–0.42)) and (interface complexity is low (0.04–0.34)) and (function point per autonomic features is low (0–0.48)) then (complexity is low (0–0.48)).

If (interaction complexity is medium (0.12–0.42)) and (interface complexity is low (0.04–0.34)) and (function point per autonomic features is low (0–0.48)) then (complexity is low (0–0.48)).

If (interaction complexity is high (0.12–0.42)) and (interface complexity is low (0.32–0.49)) and (function point per autonomic features is low (0–0.48)) then (complexity is low (0–0.48)).
**Inputs:** The inputs values may differ based on the information available about the systems. Fuzzy rules are designed on the basis of the dataset based results and the knowledge.

**Outputs:** The output will obtain according to the rules formed but the rules follow some knowledge based logics. The system, input and output details are here:

The input to the triangular mf function is (min value, peak, max value) for the inputs and output. Peak is the median value between minimum and maximum. The result from our analysis comes out to be in the range Low (0-0.48) approx 0.36. This means that Maintenance level would be quiet at the similar level as it is for the non-autonomic system. The reason for this is as Autonomic Computing concept is the new and no such high level development has been done. Some universities and industries have developed few applications in this field but those applications are not fully autonomic so there complexity level and maintenance level assessment take some more fully developed autonomic application to estimate the accurate level. IBM in their research explain that the identification of fully autonomic system can be examine using two aspects,

1. **Functionality:** means how less is manual intervention (low, medium, high) in case of error conditions when the system is self-configure, self-optimize, self-healing.
2. **ROM (Recovery Oriented Measurement):** includes Availability, Maintainability, and Scalability.

The definitions of these two aspects indicate that if there is any way of identifying the CHOP features level then functionality level can be determine for the autonomic level estimation [2]. Similarly availability, maintainability and scalability level identification helps in deciding the fully autonomic system. The functionalities are implemented at the coding level. With the implementation of more and more autonomic features, code level complexity increases and result would be less human intervention and less management by developers.

For the development of the low cost system, need to implement the optimization of the components of the particular system which in result develop effective and efficient systems. Reliability has a direct impact on performance and requires continuous reliability feedback. Availability need to be implemented at every stage with the increase in the system complexity. [19]

The availability of the autonomic features at desired point of time will tell high recovery oriented measurement. Maintainability of the autonomic system will be high as system is made capable of handling all the undesirable situations. Scalability is little different attribute which depends on code reusability, understandability, portability etc. MAM model also include the estimation of the scalability level. If these aspects are evaluated then autonomicity level also be determine.

**5. Conclusion and Future Work**

Software maintenance for the autonomic system is different from the non-autonomic system. The system complexity is increasing with the increase in the software development in the IT world. Autonomic concept relatively new and its success required skilled IT professors and developers to achieve the autonomic computing goal. Still maintenance is the crucial part of the SDLC which is required for kind of software and applications. Autonomic system involves different maintenance activities. This model is not perfect but it can help developers to use their efforts in the right direction which in the end give better results and success. Assessment of maintenance level by examine the previous developed autonomic applications will help the developers.

The MAM model helps in enlighten the path of maintenance assessment and also help in achieving the autonomicity level. Future work includes estimating the remaining part of the MAM model based on some more dataset information and study.

**References**


Appendix

Fig. 4 Maintenance Assessment Model

Fig. 5 Workflow Diagram
Fig. 7 Proposed Fuzzy rule System

If (interaction complexity is low(0.12–0.42)) and (interface complexity is low(0.04–0.34)) and (function point per autonomic feature is low (0–0.48)) then (complexity is low (0–0.48))

If (interaction complexity is low(0.12–0.42)) and (interface complexity is low(0.04–0.34)) and (function point per autonomic feature is medium (0.46–0.68)) then (complexity is low (0–0.48))

If (interaction complexity is low(0.12–0.42)) and (interface complexity is low(0.04–0.34)) and (function point per autonomic feature is high (0.66–0.78)) then (complexity is very low (0–0.48))

Up to 27 rules

If (interaction complexity is high(0.56–0.75)) and (interface complexity is high(0.47–0.75)) and (function point per autonomic feature is high (0.66–0.78)) then (complexity is high (0.66–1))

Fig. 8 Fuzzy Rule