The Journal of Supercomputing (2021) 77:388–417
https://doi.org/10.1007/s11227-020-03268-0

A review on architecture and models for autonomic software systems

Pooja Dehraj[1] · **Arun Sharma**[1]

Published online: 13 April 2020

Abstract
Autonomic computing was the term coined by IBM in 2001. The term autonomic computing was used to define the self-adaptable nature of the human body. According to IBM, the same self-adaptable feature was the need to be incorporated in the software systems. Autonomic computing is the combination of few self-capabilities such as self-configuration, self-healing, self-optimization, self-protection, self-awareness, etc. So, autonomic computing approach was then used to develop autonomic software systems. This approach makes the computing systems self-adaptable and self-decision-making support systems for various activities. It also helps to reduce the human intervention in the software management process. Though, the implementation of autonomic self-capabilities may increase the software complexity, which further requires human intervention for the software maintenance-related specific tasks. Still, IT industries are approaching to develop autonomic features in their existing architecture or developing new self-adaptable software systems. Autonomic computing has its importance for providing a bridge for handling and managing the run-time computation-based issues/exceptions of the software. So, the discussion of this solution has become a necessity for making the vision of autonomic decision making more clear and understandable for researchers and developers for the improvement in an autonomic area. The paper provides an insight vision of the autonomic decision-making concept and its importance for the various purposes such as intrusion detection, cloud-based data security, wireless sensor network, Internet of Things, Big Data and many other areas where management cannot be handled by a human in real time. To assess the degree of autonomic feature, there is another term used which is known as autonomicity. The paper also discusses some solutions suggested and implemented by different researchers during their studies for estimating the system's autonomicity level. These solutions will help in comparing different autonomic applications based on the autonomic features implemented in each application. This paper is an attempt to provide better understandability in the autonomic computational field.

Extended author information available on the last page of the article

Keywords Autonomic computing · Self-configuration · Self-healing · Self-optimization · Self-protection · System autonomicity level

1 Introduction

The systems which are capable of doing computation themselves with less human intervention are considered as autonomic systems. Autonomic computing is a combination of few self-abilities such as self-configuration, self-healing, self-optimization, self-protection, self-adaptability, self-awareness, self-openness, etc. Out of all, only four abilities are considered as major ones. The main four abilities are self-configuration, self-healing, self-optimization and self-protection. These are abbreviated as CHOP. The human body is a perfect example of self-adaption and self-management. IBM's motive was to bring solutions for handling software complexity. IBM proposed an idea of autonomic computing and its high-level policies with reference architecture for the autonomic system. These policies are summarized below:

- The system must understand its system activities and must perform intelligent response and do optimized resource allocation.
- The system must be compatible and reconfigure according to environmental changes and different system standards.
- The system must be able to protect itself from external damages.

While Horn [1] only gave the concept of autonomic computing, it was further elaborated by Kephart and Chess [2] in their paper. The authors have explained the essence of autonomic systems (AS), i.e., self- management/ self-capable behavior. This was the first time that anyone had attempted to correlate the autonomic nervous system properties with few attributes desired in a computer. After a detailed discussion on CHOP properties, the authors suggested the probable architecture of any general AS. According to them, architecture should be a collection of interactive autonomic elements (AE). Each element has an autonomic manager and the managed element. The manager itself has a MAPE-K architecture, i.e., Monitor, Analyze, Plan, Execute with the help of Knowledge database [3–7]. The architecture with the control loop will be discussed further in a literature survey in detail.

Again in 2004, DARPA self-regenerative program was followed with the aim of providing critical situation-enabled system all the time. For this purpose, four aspects are decided for such systems [8].

- Make system resistant to attack by providing a large number of versions with similar functionality but different implementation.
- Enable pushing of binary code of random size onto the system's stack that makes an attack difficult. Use a trusted model that keeps the system away from damaged resources.
- For accountability of authorized, malicious client's updates, use an architecture which is enabled with intrusion tolerance and also scalable in wide area.

- Identification of malicious operator should be done that helps in prevention of attack on a military system

The developers were successful in the implementation of the specialized computing system. But the system that handles the emerging requirements dynamically is still a challenging issue. Distributed systems have evolved to provide solutions to the problems in an isolated way such as availability, security, efficiency, reliability, automation etc. But emerging issues based on application and system management still require specific solutions that work at the run-time correctly. The solutions may be interactions, algorithms and behaviors. These solutions depend on context and dynamic state-based specification. The final goal of autonomic computing is to resolve these issues of the applications and system which are based on high-level policies and context.

From 2001, the autonomic slope started taking contributions from a different level of IT industries. Autonomic computing has its own importance for providing a bridge for handling and managing the increasing complexity of IT industry so the discussion of this solution has become a necessity for making it understandable for other researchers and developers for enhancement in the autonomic area. This paper is an attempt to provide better understandability in autonomic computation field.

The paper is divided into the following sections. Introduction section has already provided an evolution of the autonomic system. The detail of the autonomic computing is provided in Sect. 2. After doing a deep survey in this area, the literature review is categorized based on areas where autonomic computing has its current and future impact, which is shown in Sect. 3. Section 4 discusses the areas where autonomic computing has been explored and its scope for development. Finally, Sect. 5 includes a conclusion in brief.

2 Autonomic computing

As the autonomic computation concept is strongly linked with the human nervous system which controls minor to major activities of the body in situations such as unconsciousness, fear, anger, unhealthiness, injury, environmental weather conditions, anxiety, excitement, etc. Though implementation of fully autonomic applications, software or devices is yet not achieved. The reason may be complexities in implementing the autonomic concept at the core level. No doubt, the intervention of human in handling complex issues in systems will still be required at some level. Then, why there is a need for developing autonomic concept? The reason is to reduce maintenance cost, which includes 40–50% of the total software development cost. This is a reason which facilitates the IT industries to develop such artificially intelligent systems which will ultimately decrease the maintenance cost. If the time shifts to the digital systems, then it is obvious that the complexity to handle such a system will also increase [9].

The maintenance of the distributed system is quite complex at run time and is not reliable. The autonomic computing system is one approach which may reduce the complexity of such a system, and also it is a reliable and effective solution.

Autonomic computing methodology helps different IT vendors to discuss, compare and contrast different algorithms to design self-managing system. IBM [2] provided a referenced architecture of the autonomic system with a self-control loop or MAPE-K loop shown in Fig. 1, which works as a central processing unit for *autonomic management engine* (AME) [10, 11].

The author in this paper coined a new term for MAPE-K loop as self-control loop because it is controlled by the system itself. The entities' names are also replaced with the new terms such as detect, think, decide and process. The aim of the autonomic computing system is to reduce the management of the network and increase the performance by optimizing the developer's talent in designing the higher-level products and policies. This may be achieved via distributing the overall management work on the autonomic manager, which will initiate self-control loop and work according to the changes in the system's environment. The autonomic system is controlled by the loop which is designed to handle the self-abilities of the autonomic system. There are two types of system: open and closed system. In the case of open-loop system, the output has no effect on input but closed-loop system output affects the input. The closed system senses the changes in the system. Similarly, the autonomic system also detects the changes and diagnoses accordingly. The autonomic computing technique has its importance only because of its self-adapting and self-managing abilities. The loop handles all the self-functionalities. So considering all the requirements of the autonomic computing, the loop has been designed in such a way that each component uses the functionality and performs accordingly.

The autonomic computing loop compares with the human autonomic nervous system (ANS) due to the similarity in their working. Nervous system monitors body activities and regulates body temperature, blood pressure, heart bit rate, blood circulation and many more [12]. In the same way, self-control loop also monitors all the system's activity, detects the problem, thinks about a solution based on predefined knowledge, finally decides the plan and executes the plan back to the managed element. The loop continues to monitor the activities after executing the plan. If again problem occurs, then it starts analysis and performs it accordingly. There may be more than one self-control loop in the autonomic system that performs all the functions. IBM has categorized these loops into four categories based on the properties of the autonomic system [10]. These categories are a self-configuration loop, self-healing loop, self-optimization loop and self-protection loop. IBM chose only these four properties as the core features of the autonomic system. According to IBM,

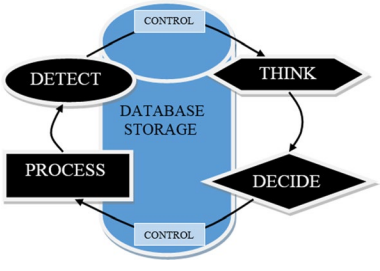


Fig. 1 Self-control loop

CHOP incorporates all the remaining features like anticipatory in the self-healing feature; self-awareness includes in self-configuration and self-optimization.

The reference architecture for the autonomic system was suggested by IBM. Human body has a nervous system which is connected with the brain, and brain controls the whole body using neurons connectivity. Similarly, the autonomic system has an autonomic agent which works like a brain. Sensors and effectors work as the connector between autonomic agent and managed element. Figure 2 shows the architecture;

Using sensors, the autonomic agent gathers the status of the managed element during continuous monitoring and sends it to the detector. When the detector identifies any exception in the environment of a managed element, the detector reports the exception to the self-control loop. The self-control loop starts analysis using control signal. For this analysis, the analyzer uses database knowledge for taking intelligent decision based on prestorage exception results, which are processed back to the managed element, and if the decision is new for the knowledge database, then updates will perform onto the database.

2.1 AC state of the art: from past to present

Before going into detail of the different research aspects of the autonomic computing, a state of the art of the autonomic computing is provided in brief starting from the function-oriented approach in the year 1960. A road map for the same is shown in Fig. 3.

The need for autonomic computing was explained by [13]. The authors provided an enough reason for the increasing complexity in software systems and need for autonomic computing era. In their work, the authors provided a graph to show the downtime: Average Hourly Impact shown in Fig. 4. The data are taken from IT Performance Engineering and Measurements Strategies: Quantifying Performance Loss, Meta Group, Standard, CT (October 2000).

The data showed an economic impact of the system failure and downtime. The reasons for such outages are categorized into four basic data center operations: database, network, system and application. Under these categories, the causes are user error, lack of automatic processes, log file full, application error, software error from third party, insufficient bandwidth, applications exceptions, etc. After year 2003, a

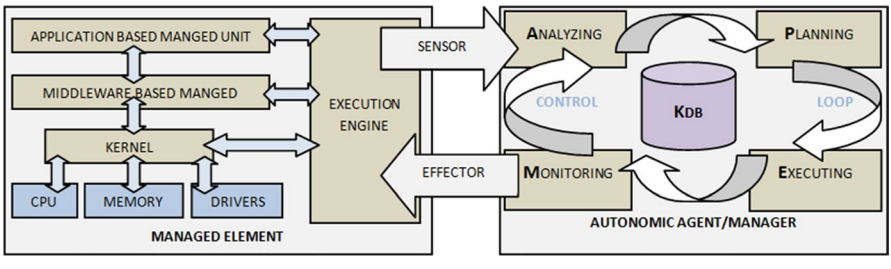


Fig. 2 Autonomic referenced architecture

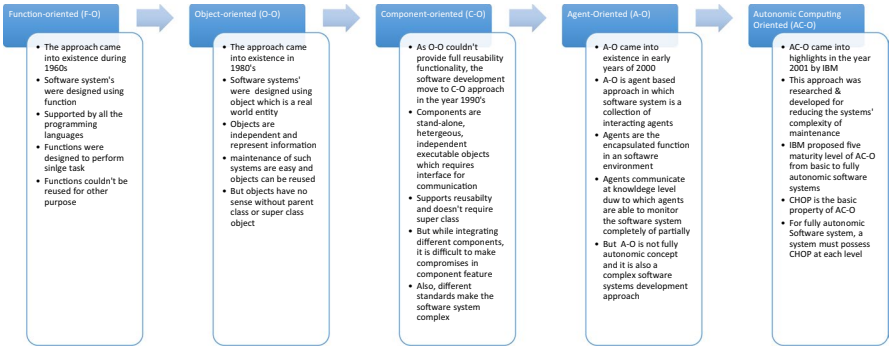


Fig. 3 Evolution of autonomic computing approach

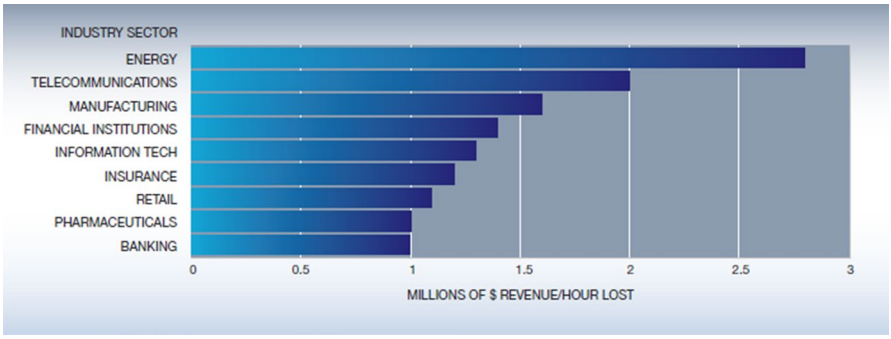
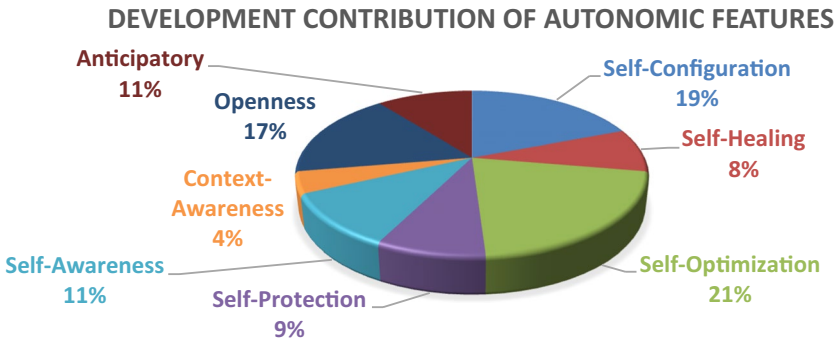


Fig. 4 Data from IT performance engineering and measurement strategies: Quantifying Performance Loss, Meta Group, Stamford, CT (October 2000)

number of researchers have performed their respective conceptual research in this area to implement autonomic computing approach in software system. Using the reference architecture proposed by IBM shown in Fig. 2, different autonomic software research projects in both academics and industries were developed. The research projects were developed using a few autonomic properties. The contribution of autonomic properties in research projects is shown below using a pie chart. A detailed discussion was provided by Salehie and Tahvildari [14]. The list of academics and industrial research projects is given in Table 1.

Table 1 Autonomic computing-based research projects

Autonomic computing-based research projects	
Industry-oriented	Academic-oriented
SMART, IBM	Software Rejuvenation, Duke University
Oceano, IBM	eBiquity, University of Baltimore County
Optimal Grid, IBM	Autonomia, University of Arizona
AutoAdmin, Microsoft	Recovery-Oriented Computing, UC Berkeley/Stanford
N1, Sun	AntHill, University of Bologna
The Adaptive Enterprise, HP	OceanStore, UC Berkeley



As autonomic computing is a part of artificial intelligence (AI), AC approach became the part of every AI-based development. Currently, autonomic computing concept has become the need of various popular developing research areas of IT industry such as cloud computing, big data, Internet of Things, green computing and grid computing. The basic reason behind this is to handle the complexity in the software architecture and its workflow. Autonomic computing also helps in providing run-time solution for a distributed network-based software systems. Cloud computing and IoT both involve distributed network, and such systems cannot be handled with the human intervention at run time. The state of the art of autonomic computing in the current research area is discussed in below points.

In Cloud Computing: Autonomic computing is one of the parameters which led to the growth of cloud computing. Service-oriented architecture, utility computing and hardware virtualization are the other parameters. The two basic benefits of cloud computing are the easy access to software resources and pay as per usage. With the development in cloud computing, the complexity of the cloud architecture and its management has been increased. It was believed that cloud automation is the solution for handling cloud computing complex infrastructure [15]. The cloud automation idea has been executed by setting business policies. The autonomic computing has helped in managing cloud resource optimization,

high data security assurance and effective cost management of cloud infrastructure. The IT industry has initiated the autonomic computing-based configuration policies for the system for the automatic system governance. This helped to rely on the cloud infrastructure for monitoring, executing necessary changes back to the systems. It has been claimed that autonomic computing is the future of cloud computing.

Benefits of cloud automation [15]:

- Usage: Automation helps in rescheduling the shutdown processes or manages the long running processes.
- Availability: Automated backup for data storage helps in data migration to another region
- Cost: According to the need, the IT industries will automate the purchase task. It also automates the workload movement between cloud providers.
- Performance: For nonhorizontal scaling of workload management, the automated cloud increases the machine type.
- Security: For those who conform to established business policies, the companies ensure automatic network change or endpoint security from the system.

In Internet of Things (IoT): Internet of Things ecosystem involves Internet protocol used for connecting devices, network, configuration and control function. Internet protocol is used for the deployment of IoT ecosystems on different technological domains. But deployment process involves large number of devices on multiple technologies which increases infeasibility of manual maintenance and management of such ecosystem. It was suggested that intelligent and autonomic setup helps in maintaining IoT ecosystem. Autonomic computing is the technique which minimizes the human intervention in the management of the IoT ecosystem.

In Fig. 5, IoT components and autonomic components interact with each other. Similar to other autonomic systems, in IoT ecosystem, there are no defined autonomic manager and managed elements. As per the versatility of Internet of Things, one of the IoT components will be assigned as an autonomic manager of its lower-level components. The lower-level component is then considered as a managed element. Autonomic computing technique in IoT performs the dynamic decision making. The decision is taken for the access management, device management, network management, configuration control function execution and identity management. It reduces the human intervention, manual management of IoT ecosystem which ultimately helps in providing effective and efficient response to the IoT ecosystem's users.

Benefits of IoT Ecosystem with Autonomic Computing:

- Self-configuration based: High-level polices help in automating the device configuration and adjusting the network setup.
- Self-optimization based: It improves the performance and efficiency of IoT ecosystem.
- Self-healing based: It automatically detects, analyzes, diagnoses and repairs the hardware and software issues.

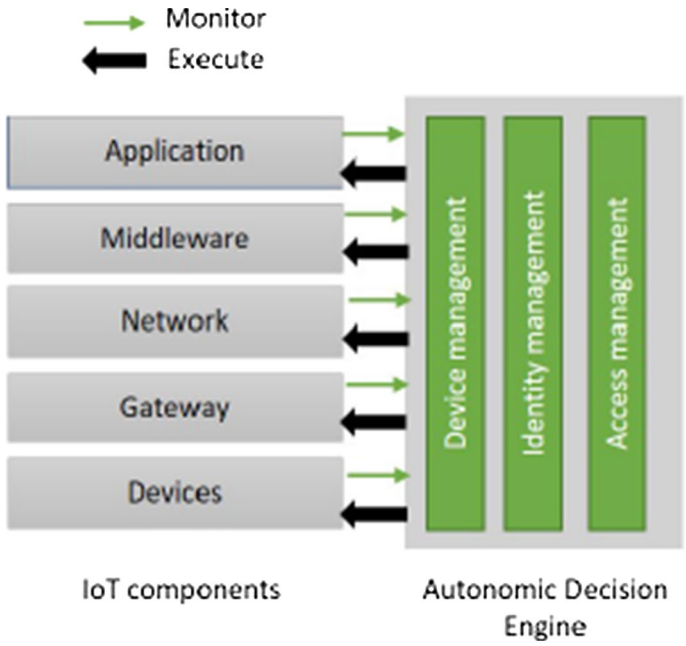


Fig. 5 Autonomic component system interaction for management using an autonomic scheme [16]

- Self-protection based: It protects the IoT ecosystem from malicious problems and sends earlier messages to protect the system from failures.
- Self-security based: Without halting the IoT ecosystem, the self-security feature of autonomic computing helps in initiating high-level policy based corrective and protective actions onto the IoT environment.
- Self-adaption based: This feature helps in continuous monitoring of the IoT ecosystems and taking dynamic decision to adapt to environmental changes.

In Big Data: Big data is another popular research area where autonomic computing technique has contributed to the development and management of numerous tasks. Intrusion detection is one of the most important processes that need to be available for data security and privacy. Autonomic computing-based intrusion detection is now provided in big data. Autonomic intrusion response system (AIRS) is an approach which was proposed by [17] to reduce the chances of intrusion in the distributed network system to secure data privacy and security. The approach is based on self-healing characteristic of autonomic computing. The authors implemented the autonomic computing with big data approach to demonstrate significant improvement. Their proposal is used to process data from system to detect the large amount of anomalies and attacks (Fig. 6).

The above architecture is completely based on MAPE-K loop of the autonomic system which was proposed by IBM. The MAPE-K loop activates the analysis of the large amount of data to understand the utilization of distributed software resources to take intelligent decision to meet expected utilization. This process continues to

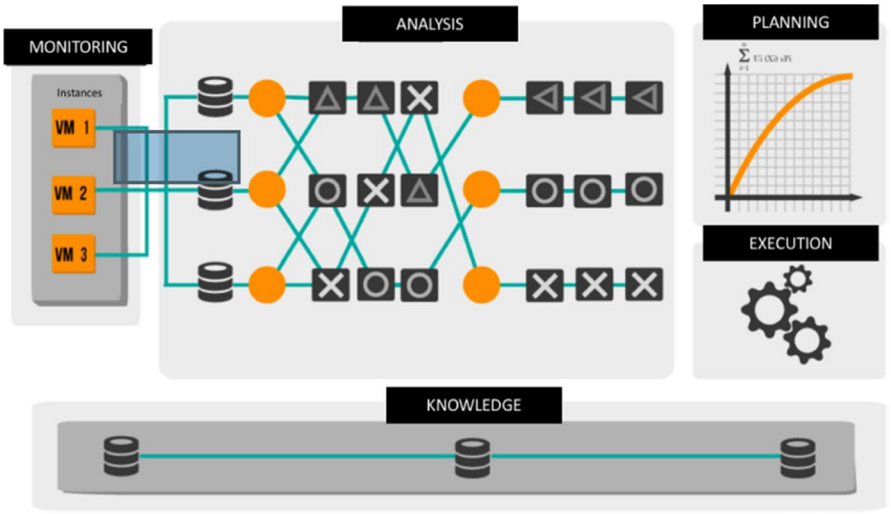


Fig. 6 Architecture of AIRS [17]

recognize and detect attack condition by comparing knowledge data stored from previous suspicious cases. To make knowledge database, the system collects log files, network traffic, sensor data. Using all the data, the MAPE-K loop takes intelligent decision to ensure intrusion detection process.

Benefits of Autonomic Computing in Big Data:

- It efficiently analyzes the network traffic of distributed system to process large amount data for optimized resource utilization.
- It processes the intrusion detection-based response system to prevent data privacy and ensure data security.
- Map-Reduce-based data processing policy helps in handling large data using parallel process.

3 Literature review: four aspects-oriented

Though, autonomic is relatively a developing paradigm in computing arena, a lot of researches in the form of proposing new architecture, metrics, development process, etc., have been done. The present section conducts an exhaustive and detailed review on various architectures, autonomic applications, metrics and software development process. The review consists of papers covering major publisher's literature work. The paper provides a brief detail with highlights of the previous research work.

Autonomic computing-based papers discussed autonomic reference architecture and its features. Many research works also discussed its challenges. This paper is an attempt to refine their work and brief the research work categorically so that it will be easy to learn a particular research work objectives and the work done by the researchers for autonomic computing. Initially, category-based research work is

discussed briefly and then a combined analysis of each category is provided after each category-based research work.

3.1 Architecture-based

The autonomic concept is purely inspired by the human structure and its intelligent function. After IBM, other researchers have also proposed autonomic computation architecture similar to IBM's reference architecture. Khalid et al. [18] in 2009 performed a survey on available autonomic frameworks and classified them as given below:

- Biologically inspired architecture: The concept of autonomic computing is evolved from the human body which is capable of handling internal and external activities by itself. Autonomic concept also requires the same functionality. Human body has a brain that controls each part and its activities. Similarly, the autonomic architecture also requires a central control unit like the brain which handles the system's activities.
- Architecture for large-scale distributed system: For a distributed system, IBM and Microsoft have developed applications at large scale. SMART and AutoAdmin are few of them. Both the applications work for controlling the administrator-level functionality automatically for large-scale systems.
- Agent-oriented computing paradigm architecture: In the case of autonomic concept, the architecture works in an agent-oriented manner. The autonomic manager acts like an agent which controls unwanted activities of the managed source. The agent communicates with other units of the system to handle the activities at run time.
- Component-based architecture: Component-based architecture also plays an important role in grid computing. It provides self-configuration functionality in the grid architecture-based applications and software.
- Technique-based architecture (TBA): This kind of architecture is designed on the basis of the requirements. Soft computing technique such as artificial intelligence uses TBA. TBA allows systems to learn, examine, plan, replan and then execute the plan to achieve autonomic goals.
- Service-oriented architecture: Few systems that need monitoring and analysis also require service-oriented architecture to enable the systems to handle management-based requirements automatically. They are reactive in nature.
- Nonautonomic system architecture: Without developing a fully autonomic system, the developers are using a concept of injecting some of the autonomic requirements in the existing system. This will be helpful for the legacy system. Monitoring and healing are the features that may be implemented in nonautonomic systems. This will reduce the development cost, time and also reform the system in the working conditions again.

After this, IBM in 2005 presented an autonomic computing-based white paper explaining the base architecture layer of autonomic computing with some layers

which are organized based on the IT processes such as Incident, Change and Problem Management. These managements may lead to the fulfillment of the autonomic capabilities. The following are the layers of the autonomic architecture:

Managed resource: Managed resource or managed element is a part of the autonomic architecture on which the autonomic computing-based control loop works.

Touch point: For the connectivity between MAPE-K loop and managed resource, sensors and effectors are required which take the input from the managed resource and feed the output back to the managed resource, respectively.

Autonomic manager: Autonomic manager is a kind of handler which works using MAPE-K loop to handle the internal and external activities of the managed resource.

Orchestrating autonomic manager: For a single managed resource, one autonomic manager is sufficient. For more than one managed resource, there is a need for many autonomic managers, and this will form an orchestrating autonomic manager.

Manual manager: Manual manager is quite different from the autonomic manager in terms of its management task. In a manual manager, the management is done by an administrator by setting high-level policies and rules.

In 2005, NASA initiated an autonomous technique-based project named as ANTS (Autonomous NanoTechnology Swarm). This project involves worker craft and messenger craft. The worker craft works in a group with the coordinating ruler to find information for evaluating the asteroids that issue instructions. The messenger craft sends the gathered information to control room on the earth. NASA has already implemented the concept of autonomic computing in DS1 and Mars Pathfinder mission [19]. They want to develop such a system which can make a decision in a critical situation in real time without sending the signals. Huebscher and McCann [20] conducted a study of some of the autonomic developments which included SAS, ASADA, SPS and ANTS along with autonomic computing.

Garlan and Schmerl [21] suggested an adaption model for the autonomic system. The model was tested by changing code in the managed unit. This model works as an architecture model for resolving issues. The explanation of this model is provided in their paper at an abstract level. The implementation procedure is shown in their paperwork. Their work laid down the development of self-healing feature externally in the system.

Wang [22] in their work proposed a rule-based model for autonomic self-adaptive systems. They had given a few reasons for using this rule-based approach:

- Due to the centralization of the rules-based model, a fine-grain level of consistency may be achieved.
- Also, the centralization of rules helps in understanding rules more clearly, and it will reduce redundancy from the rules.

For the administrator, the model-based rules can be stored externally for easy modification by the admin and their execution may be done using if–then conditioning loop action. The authors work shows in their paper using XML representation of the rules which may be reversed into executable code during run time. This approach is quite effective for the self-adaptive system.

Kumar and Sharma [23] proposed a vulnerability detection model using autonomic computing technique to reduce vulnerability rate in software. In their work, the model detects the vulnerability and then takes the appropriate intelligent action using AC for mitigation. Their approach works as post-processor means; first, the system is exposed to vulnerable situations and then takes the action.

In the research work, Pena et al. [24] have shown a model-driven approach (MDA) to model, deploy and manage a self-policy-based system during run-time process. The authors have provided numbers of deployment solution for self-management such as structural organization model (SOM), model of reusable autonomous and autonomic features (M-RAAF), platform service model (PSM) and acquaintance organization model (AOM), in a single model-driven architecture. All these models will run as per requirements defined for each model. To change the policies at run time, the models will perform its task as defined in Fig. 7 and steps are:

- M-RAAF improves the reusable property as the repository is going to be managed for this.
- AOM is used for the role organization based on the interaction existing between the agents.
- SOM is used to structure the agents into hierarchical structure to show social structure of the agent role.
- PSM is the last model used to deploy policy onto a system during run time.

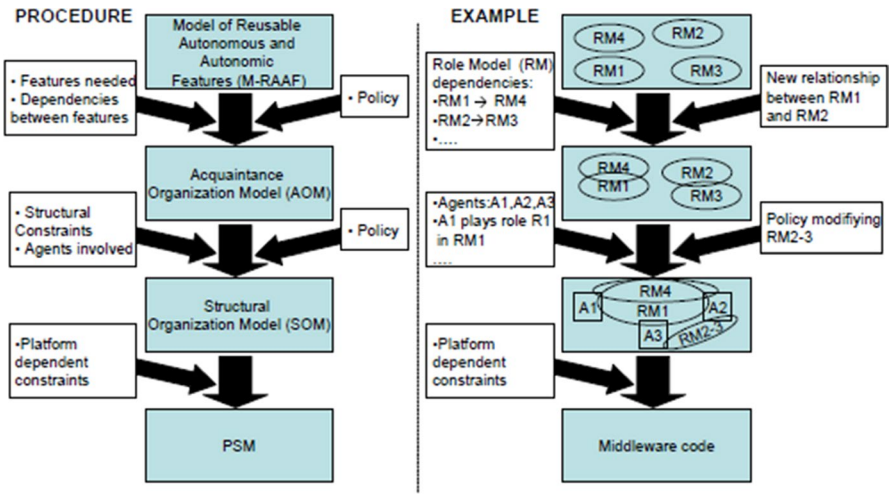


Fig. 7 Summary of MDA approach [24]

The deployment of policies in a system will start with a transformation of one-stage model to another next-stage model; for example, a transformation from M-RAAF to AOM, AOM to SOM and SOM to PSM will be done for applying policies during running process. It will add new functionalities as per the requirements.

Analysis of the architecture-based related work: From the above research work, it is concluded that the researchers mainly discussed the architecture-level difficulties, capabilities, development-based requirements and the frameworks on which autonomic system may be designed. Few researchers have actually tried to design and develop autonomic architecture such as adaptation model by [21], self-adaptive model by [22]. Few of them have only suggested an architecture-based design such as vulnerability detection model proposed by [23]. Besides this, the vulnerability model only works for detection and mitigation of vulnerability. The model is only implemented with self-healing feature. The model will not handle any self-configuration-, self-optimization- and self-protection-based issues. Implementation of other autonomic features will increase the architectural complexity. The model-driven approach defined by [24] is another architectural-based proposal. The architecture shown in Fig. 7 defines four different models which are combined in a single MDA. Deployment of different policies for four different models leads to the complex architecture structure. Its type of architecture-based software will also affect the performance and efficiency of the software system. Security- and vulnerability-based self-policy for such systems will be difficult to implement.

This kind of model which is specific to any particular autonomic feature will not work to handle different types of run-time issues. Also, higher level requirements are tried to be fulfilled in the proposed models but lower level user's requirements such as automatic recovery of deleted files, utility-based activation or deactivation of particular services must also be fulfilled automatically.

3.2 Metrics-based

Salehie and Tahvildari [14] discussed the complexity which is growing exponentially with the development and categorized it broadly into three categories:

1. Business domain complexity (BDC): complexity due to business processes, resources (hardware or software) and organizational setup which is measured in terms of size and cost, probability of fault occurrence and time.
2. System development complexity (SDC): complexity estimated or faced during the development of software, applications and system comes under SDC. During the development of each phase, the ease of developing items tells about the level of complexity of that item. It is also estimated in terms of size and cost.
3. System management complexity (SMC): It includes system management process complexity, security management process complexity and recovery measures-based complexity and others based on the maintenance of the system.

Based on their studies, the authors did a major identification of the functionalities and also found a relationship between CHOP and identified complexities.

- Self-configuration has a relation or impact on functionality, maintainability and portability.
- Self-healing has a relation with survivability, availability, reliability and maintainability factors.
- Self-optimization has a relation with performance.
- Self-protection has a relation with maintainability and reliability of the system.

Parashar and Hariri [12] explained the origin of autonomic computing (AC) using the parameters that govern the maintenance using Ashby's ultra-stable system for the human brain. Based on this, they identified two properties which may be taken as the identification marks between the autonomic system and the human body. The properties are:

- After adaption to the internal or external changes, the system must survive.
- The system must create a new equilibrium state when the system is thrown out of its optimal state of equilibrium.

The authors further divided the autonomic architecture workflow into two loops based on known and unknown environmental behavior:

- Local loop: When the system knows its environmental changes, the local loop handles the changes using its already stored knowledge database. Every parameter has its range or limits. If the parameters exceed their limits, then local loop control passes on to the global loop after updating the changes done during the local loop control.
- Global loop: When the control passes to the global loop, its role started. The global loop works only for handling the unknown environmental changes which are not handled by a local loop. It monitors and analyzes the state of environmental changes using some parameters such as performance, security, configuration and protection. Global loop updates the knowledge database based on the changes for further easy adaptability and better response. If the parameters exceed their limits, the loop automatically makes new adaption methods which will be added to the existing database.

The authors further divided the autonomic computation-based future challenges into four categories:

1. Conceptual challenges: for implementing autonomic concept during the software development process, there is a need for designing statistical models, relational models and abstraction models for developing relationships among elements.
2. Architectural challenges: implementation of autonomic computation technique including autonomic elements and interaction among the elements during its architectural designing based on local and global behavior.
3. Middleware challenges: there is a need to provide core-level services for autonomic behaviors realization but environment changes are still uncertain like identification of every aspect, verification, security and privacy, trust-level issues, etc.
4. Application challenges: implementation of autonomic behavior through programming, designing frameworks and middleware services.

Some of the researchers have given attention to the problem of evaluating autonomic computing systems. In one such earliest attempt, Huebscher et al. [11] in their paper after briefly introducing CHOP properties indicated that there is no set definition of autonomic systems. They focused on needs for such systems which are driven by increasing the cost and complexity of today's IT infrastructure. Authors have then discussed various software architectures for AC, which is mainly categorized into three categories: multiagent systems, architecture design-based autonomic system (AS) and hot-swapping components.

The important metrics that may be used for evaluation of AS given by authors are quality of service, cost, granularity, robustness, degree of autonomy, adaptivity, time to adapt and reaction time, sensitivity and stabilization.

Authors argued that AS is much more complex than the traditional systems and hence area of benchmarking and metrics derivation is an interesting one. The authors identified the metrics for comparing heterogeneous autonomic system (Fig. 8).

To identify heterogeneous autonomic systems, the two approaches were used: tightly coupled autonomic system and decoupled autonomic system. These approaches have common concepts. Again both approaches need two kinds of elements: (1) for the target system functionality implementation and (2) to add a solution for self-management in the system. These elements describe two-level architecture: interrelationship and intraelement relationship. The first level deals with the relationships among the elements in a particular manner. The second level describes the global aspects of the autonomicity, such as overall configuration [11]. Infrastructure elements provide documentation service: monitor the system and aggregation of valid information, interconnectivity and negotiation.

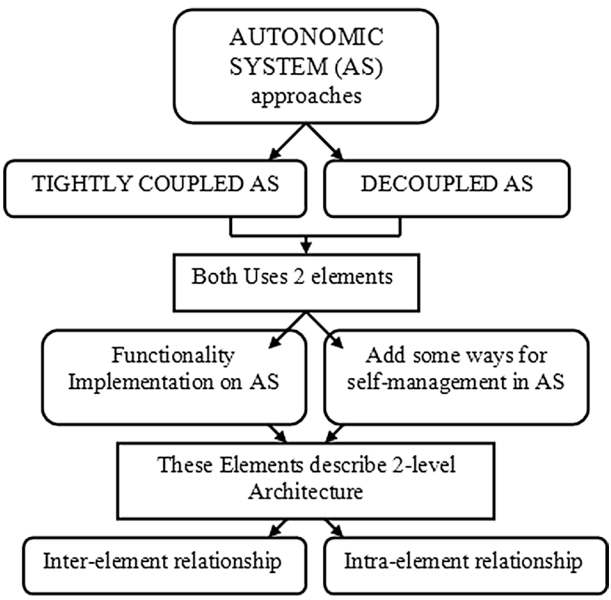


Fig. 8 Approach to identify autonomic system [11]

Brown et al. [25] implemented a practical benchmark for self-healing property of AS. The process involved introducing disturbances in System Under Test (SUT) subjected to performance workload just like any real e-commerce application. The healing capacity of SUT is determined by finding how the system heals and by measuring healing effectiveness level, i.e.,

- How effectively the system heals itself?
- How autonomic that healing is?

For the metric, a 90-point question survey was used which follows five-point classification set by the IBM Autonomic Computing Maturity Model. Authors have also addressed the various issues that present a challenge to benchmarking area:

1. Quantifying autonomic maturity directly.
2. Quantifying healing effectiveness to capture broader areas.
3. Accounting for incomplete healing.
4. Accounting for resources used.
5. Unified metrics in the end for all properties.

One good attempt at assessing the quality of AS through autonomicity is done in [26]. The authors have proposed an approach for measuring the level of autonomicity (LoA) as this will give autonomic functionality level in a system. They have presented some functionality, and these functionalities depend on some metrics. All the functionalities add up to give LoA. The functionalities considered are CHOP properties, and each one of them has a metric based on which it is measured. The functionalities considered are CHOP properties, and each one of them has a metric based on which it is measured:

Self-configuration depends on interoperability, calculated as

$$I = \sum_{1}^{i} \frac{n_{iactual}}{n_{iexpected}}$$

where n is the number of self-functionality and I is the actual number of self-functionality presented in the system divided by expected number of self-functionality.

Self-optimization depends on stability metric, calculated by variables such as current load distribution, CPU utilization, etc.

Self-healing depends on reaction time T, calculated as

$$T = t_{\mathrm{b}} - t_{\mathrm{a}}$$

where t_{b} is time to work out new configuration and be ready to adapt and t_{a} is time at which the change occurs in environment.

Self-protection depends on ability to detect any repeat events E calculated as

$$E = True \forall_{ij} \mathrm{if} \{p_{ij}\} t_1 \cap \{p_{ij}\} t_2 = \varnothing$$

where p_{ij} is the log of all identified trends and corresponding problems at different time intervals t_1, t_2.

For n numbers of self-functionalities, mathematically there will be $\sum_{i=0}^{n} C_r$ possible evaluating combination for LoA. The mathematical calculation is done by implementing the algorithm in C# program. IBM has defined five maturity levels for the system software [27–29].

Level 1: Basic—all the management will be done manually
Level 2: Managed—Out of four self-controlling functions, analyzing and planning would be done manually
Level 3: Predictive—only implementation-based decision would be taken manually by the developer's team
Level 4: Adaptive—administrator-level work will be to set high-level policies and generate automatic plans
Level 5: Autonomic—self-managed system policy

To find a fully autonomic level of any software, IBM also proposed two aspects for evaluating the LoA of any autonomic system [29]:

- Functionality: It means how less manual intervention (low, medium and high) is required in the case of error conditions when the system activates self-configuration, self-optimization and self-healing.
- ROM (recovery-oriented measurement): It includes availability, maintainability and scalability level in the system.

To estimate LoA of an autonomic system, the authors [30] proposed maintenance assessment model (MAM) using which application complexity level (ACL) was evaluated for some autonomic computing based on live projects using fuzzy approach. The authors again applied neuro-fuzzy approach to improving the previous results but the approach applied only one of the major factors which highly contributes to the ACL, i.e., complexity [31, 32].

Many companies have developed autonomic applications and implemented some of the autonomic features. For the recovery-oriented purpose, Patterson et al. [33] mentioned in their work about recovery-oriented computing (ROC) application. The recovery-oriented programming addresses software, hardware and human failures by providing an effective mechanism for detecting and recovering them. The construction and implementation of ROC system were out of the discussion from their paper but it laid down a discussion basis where a researcher could think of efficient architecture and programming model that may solve ROC and provide the maximum availability for computer systems.

Analysis of the metrics-based related work: Many research works have been done for proposing metrics-based analysis of autonomic systems with their challenges. Also identification of some quality factors has been done based on CHOP factors. The overall quality and autonomicity may be estimated using proposed approaches but practically, these proposed approaches have not been implemented on any

autonomic application. The two aspects defined by IBM in [29] have not been used for measuring autonomicity of the any developed autonomic application by the IBM itself. The metrics proposed by [26] have also not been implemented for any autonomic applications. The reason may be due to lack of autonomic features in developed autonomic applications such as recovery-oriented computing application which only works for system recovery-level computing. IBM defined five maturity levels to define autonomicity level in each phase. But fully autonomic software systems have not been developed till now. The developed software systems generally focused on particular autonomic features, e.g., Gryphon, SMART-DB2, Storage Tank, etc., which are also specific to few autonomic features only. So, the review work will provide a brief about all the autonomic features and its autonomicity level. All the developed autonomic applications are not fully autonomic in nature.

3.3 Applications-based

After IBM, other companies also worked for implementing autonomic computation technique in their applications. There are many projects under IBM where autonomic properties are either integrated with existing software or being developed as new software to work with existing systems. Some of their initiatives are listed Table 2.

Table 2 Autonomic applications

Software	Functionality
Gryphon: Pub/Sub (Middleware)	Distributes large volumes of data/content in real time to thousands of clients distributed throughout a large "public" network, such as a wide-area extranet or intranet that is too large or complex to be centrally administered to support specific applications [34]
HWLM: Heterogeneous workload management (Total System)	Allows installations to define business objectives for a clustered environment. This business policy is expressed in terms that relate to business goals and importance, rather than the internal controls used by the operating system [35]
LEO: DB2′s learning optimizer	Comprehensive way to repair incorrect statistics and cardinality estimates from a query execution plan (QEP). By monitoring previously executed queries, LEO compares the optimizer's estimates with actual ones at each step in a QEP and computes adjustments to cost estimates and statistics that may be used during future query optimizations [36]
SMART: Self-managing and resource tuning DB2	Designed to reduce the human intervention needed to run and maintain a database [37]
Storage tank	Heterogeneous file sharing, policy-based file and storage management, high performance and scalability. This technology is currently used in IBM's Tivoli's Storage Manager Product [38]
UFiler	Facilitates access and sharing of files that may be geographically distributed over an entire enterprise or the Internet. It allows access to files anytime [29] and anywhere, and files are protected through fine-grained access-control lists [5]

Analysis of the application-based related work: Application-level studies from different literature explained the implementation of autonomic features at some level. All the autonomic applications are not fully autonomic. Few of these works at the administrative level and some of these are developed for middleware usage. IBM has designed and implemented a few applications for autonomic workflow. The applications are only designed and developed for specific autonomic functionality. For example, learning optimizer (LEO) is used for the query optimizer. Oracle and Microsoft have also designed their own query optimizer such as oracle query optimizer (OQO) and SQL query optimizer (SQO). The comparison among such autonomic functionalities in different software applications has been performed [39–43] on different parameters such as configuration manager, index reorganization, performance monitor, query selection, consistency checking, storage management, serviceability utility, maintenance plan, integrity management, database tuning utility, incremental restore, database recovery. But autonomicity-based quality parameter has not been used to compare such optimizers empirically. The authors [44] have performed autonomicity-based empirical analysis of above-mentioned query optimizers using a hybrid approach of fuzzy analytical hierarchy process and a newly proposed evaluation approach. But the results have not been validated and verified. The lack of validation and verification is because the research has not been carried out for such analysis. Metrics can be used for comparing different applications and software systems for measuring the performance. Performance is one of the major quality factors used to examine the utility of particular software. Similarly, different comparisons can be done after discovering the specific autonomic-based quality metrics. There is a need to implement lower-level users-based requirements in the autonomic software and applications so that autonomic concept can be work in every environment after designing on each level of the system's architecture.

3.4 Software development-based

Pfannemüller et al. [45] integrated a dynamic software product line-based context-aware feature into the database knowledge component of adaptation logic for improving analysis and planning of reconfiguration-based activities. The purpose of the adaptation logic component is to reduce the use of case-based specification in system software.

Sharma et al. [46, 47] in their paper discussed the paradigm shift in software development due to autonomic computing concept. In their paper, the authors presented a generic architecture of AS and its life cycle. Both these aspects are crucial while considering the assessment of such systems. In the architecture, the authors defined the interactions between the system and its external environment through three system components:

- Negotiation: It is the link between the system and the environment. After negotiating requested services, it communicates its own requirements to other AS in touch.

- Execution: It is solely responsible for executing the policies resulting in a particular behavior of a system.
- Observation: This component records all the changes and stores them in the knowledge base.

The autonomic systems' life cycle has four stages starting with developing the AS (design and implementation), followed by testing and verification phase; then the full working stage (installation, configuration, optimization, upgradation, monitoring, problem determination and recovery); the last stage is the end of life cycle of AS (replacement or uninstall).

Shuaib et al. [48] identified the main characteristics that relate to the quality of AS from ISO 9126:1998 standards. The authors briefly elaborated these attributes as:

- Functionality: suitability and interoperability.
- Efficiency: response time, processing time and throughput.
- Portability: adaptability, installability, coexistence and replaceability.

The system which is added with autonomic capabilities will become complex. The complexity will also be increased with the development of the Internet of Things (IoT) in autonomic application [49]. IoT itself has complexity to develop because it involves not only computers and phones (or mobiles) but also the home systems, e.g., household hardware which required automatic responses using Internet service. This will definitely increase software and hardware complexity. When such advanced ideas are combined, then handling system's failures requires a higher level of recovery-oriented measures (ROM). This kind of recovery measures must work dynamically with less human intervention.

Availability of system and system's resources is an important aspect in the field of computer network and distributed systems. The IT industries have made many attempts to provide on-time services but due to unidentified loophole in security and privacy ways, there is always a chance of failure and unavailability of resources.

Oreizy et al. [50] presented their efforts for self-adaptive systems, which is one of the self-managed properties. Self-adaption means the system automatically adapts itself according to whether the changes are internal or external. For this, the authors provided with an Adaption Management and Evolution Management process for the system which may be developed with a self-adaptive feature. The process involved two steps: Adaption Management and Evolution Management.

The first step involves monitoring, analysis, planning and then deploying the changes which are examined during the analysis phase. This step makes the actual changes to be updated within the knowledge database to make the system adaptable for similar changes in future. The second step, Evolution Management, handles the consistency, integrity during the implementation and architectural modeling.

As the autonomic concept is influenced by the human body; it is the best example of self-adaptable systems. Based on this, Wang and Suda [51] proposed a biological network-based architecture for constructing self-adaptive systems. They related the computation in a system with the human biological system, e.g., bee colony. In

continuation, they mentioned the architecture, principles and platform for biological network. The authors used a case study of Aphid system which may be available, adaptable and scalable. It is also a Web content application. Using the Aphid system, the authors tried to validate all the aspects that are scalability, availability and adaptability.

Sterritt and Hinchey [52] discussed the autonomic vision and need of creating a self-managed system to handle increasing complexity during and after the development of the software system. The cost used for such complex systems is also a concern for the fulfillment of tomorrow's needs. In their work, the authors properly explained the growing needs of autonomic systems, related research work, technologies developed and used during its growth. The current state of autonomic research was explained using some real-time examples like NASA sensor network applications.

For the same discussion, Solomon et al. [53] also mentioned the growing requirements of self-computation-based system. Autonomic computation provides a base for self-optimization approach, i.e., how to optimize resource utilization. Autonomic computation is similar to real-time systems, and for this, it requires information for specific parameters using which decision making becomes an easy task. Thus, mathematical characterization of the validation model is quite crucial. The authors introduced an adaptable technique for the autonomic computing concept. The identification of adaptable technique is based on the pseudo-random arrival rate which is injected in the autonomic system that works as disturbances in the autonomic systems. Based on the disturbances occurring in the systems, the observations will be collected which will help in analyzing throughput rate, response rate, CPU load, etc. Adaptable technique determines the sampling rate, and to extend the Kalman filtration process, recursive parameter estimation technique (RPE) is used. As a result, an adaptable system will be designed on which control strategy relies upon. The work-based experiments and results are shown in their paper.

An autonomic system not only reduces the maintenance but also helps in managing high-level objectives using policies and rules designed by developers for runtime management at the administrator level. In their work, Portela and Perdomo [54] presented a survey on autonomic computing and highlighted one of the autonomic factors that may detect and correct the unwanted exceptions-based failure. This property is self-healing which has a great effect on managing internal harms of the system. This property may be implemented using artificial intelligence.

Kurian and Chelliah [55] observed that though the vision of autonomic computing (AC) is highly ambitious, an objective analysis of autonomic computing and its growth in the last decade throws more incisive and decisive insights on its birth deformities and growth pains. Predominantly software-based solutions are being preferred to make IT infrastructures and platforms adaptive and autonomic in their offerings, outputs and outlooks. However, the autonomic journey has not been as promising as originally envisaged by industry leaders and luminaries, and several reasons are being quoted by professionals and pundits for that gap. Precisely speaking, there is a kind of slackness in articulating its unique characteristics, and the enormous potentials in business and IT acceleration. There are not many real-world applications to popularize the autonomic concept among the development

community. Though some inroads have been made into infrastructure areas like networking, load balancing etc., and very few attempts have been exercised in application areas such as enterprise relationship planning (ERP), software configuration management (SCM) or customer relationship Management (CRM). In this paper, they would like to dig and dive deeper to extract and explain where the pioneering and path-breaking autonomic computing stands and explain varied opportunities and possibilities, which insists hot pursuit of the autonomic idea. A simplistic architecture for deployment of autonomic business applications is introduced, and a sample implementation in an existing CRM system is described. This should form the basis of a new start and ubiquitous application of AC concepts for business applications.

De Nicola et al. [56] proposed a very new approach using a language-based approach to develop an autonomic system. Their work presented the details of the Service Component Ensemble Language (SCEL), its specific design principles, syntax and semantic operations. This will help in defining dialect with a sample example. The SCEL approach helps in designing the autonomic system for various domains. With the language specification approach, the authors tried to bring different programming abstraction in the form of aggregation, their behaviors and knowledge data together according to a specific policy.

Bees swarming technique presents self-optimization and self-adaption behavior. Nhane and Song [57] used this technique for the autonomic system. The authors explained the details about the bees swarming technique by exploring the environment in which bees live and how they get divided into different groups to fulfill their desired target task. The authors also suggested the improvement from autonomic manager to bees autonomic manager (BAM) to make this approach generalized for the self-optimization attribute in the autonomic system. BAM follows the bees algorithm to identify different roles and their allocation to the different groups of resources to optimize the complete function and also suggests an Adapt Case Modeling Language for the autonomic system. Using a case study of a computer network, the authors depicted the usefulness of BAM.

Schneider et al. [58] used a self-healing property of autonomic computation with the restricted Boltzmann machine (RBM) for proposing a self-healing ability for their work. RBM is a generative stochastic artificial neural network (ANN) that can be made capable of learning probability distribution (PD) over a set of inputs. They also suggested that there is no surety of a system to find an appropriate solution for the particular exception. The authors used their approach and combined it with RBM to find out specific problems within the system and by the system only not by the administrator. The RBM helps in identifying the reasons for the exceptions occurring in the system, and it also uses a learning algorithm to identify and predict the effect of exceptions using the past history stored results. The systems that use RBM will validate the effects and will maintain a separate list of valid data only based on positive results of the system. In the case of negative response, its effects are marked as invalid along with their confidence value. The list of potential exceptions or faults along with their confidence values is stored in a sorted form. This sorted list helps in prioritizing self-healing strategies. In their work, the authors also compared the artificial neural network technique (ANN) with hidden Markov models (HMMs) based on their performance metrics and respective advantages.

Analysis of the software development-based related work: The development of autonomic system will introduce complexity in the overall system's architecture and workflow which further leads to management task. The above research work suggested that the development of autonomic system may be managed using agile modeling process, RBM, bees swarming, ANN, BAM and many others. Agile modeling follows modification easily during any phase of software development life cycle (SDLC). Bees swarming helps in implementing self-optimized feature, RBM is combined with ANN for implementing self-healing feature and self-adaptive feature may be implemented using biological network-based bee colony technique. There is lot of work done for the software development-based approach, and more research is required to do highly autonomic software systems.

4 Challenges towards autonomic computing

Technology-based development not only made the working on system efficient but has also enhanced the complexity in developing and handling systems for the developers and testers. The manpower used for management purpose may be reduced using the autonomous capable systems. But this is not an easy task and it may take many years of hard work to reach such level of technology with many challenges. This period of autonomous development has already been started since 2001 after IBM suggested this solution. From the year 2001, many researchers have highlighted the challenges and tried to resolve them. As a result, autonomic applications have been developed and only a few autonomic features were implemented but fully developed autonomic application or software is still need long-time discussion, finding and implementation of solutions. Such a solution initially required the identification of challenges for autonomic computing, and based on these challenges, some authors' works have been discussed under this paper. During studies, the authors have also identified some future challenges and designed a framework of challenges at the granular level under the coarse-level aspects of challenges.

With the improvement in the development sector of IT, the requirements of autonomicity in the products also increased to ensure the better quality. For this, the autonomic-based challenges should be ensured in some areas such as networking, management, modeling process, security and privacy [59, 60]. Refer to Fig. 9.

The framework presents the coarse to fine granular level of the autonomic-based challenges [62–64]. All the fine-level aspects require the autonomic concept for the better quality of service (QoS) [65]. Few challenges such as cloud computing, Internet of Things (IoT) and Big Data are discussed in the paper below.

Dehraj et al. [66] mentioned that the assessment of quality for autonomic systems needs a few different parameters as self-capable features are implemented in such systems. For this purpose, the authors gave two parameters (trustworthiness and autonomicity) that need to be considered for estimating the quality of autonomic systems. These two factors can be assured in the systems if the development process incorporates such factors-based test cases and modeling techniques so that better quality product/ systems will be achieved.

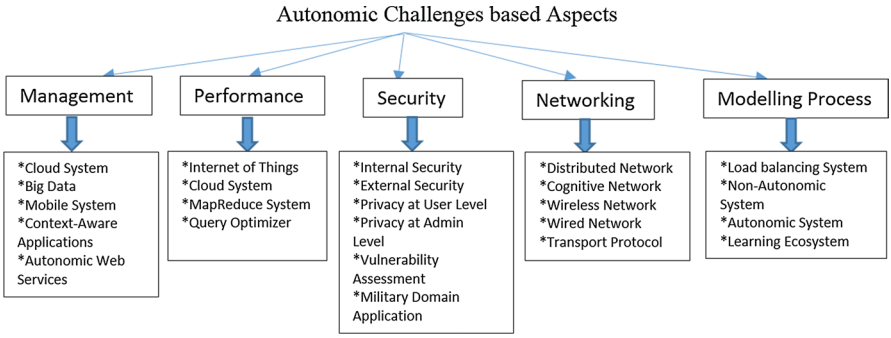


Fig. 9 Coarse- to fine-level autonomic computing-based aspects [61]

Due to increased complexity and size, a cloud may have become a big challenge in current computing scenario. Still, security management for the cloud is a critical issue. Smith et al. [67] tried to solve the security issue in a cloud by implementing an exception using the anomaly detection framework. The author firstly discussed a few data analysis techniques and deviation detection-based techniques in the system. In their proposed approach, the data transformation technique will be used for analyzing and handling the diversity in the data. This results in reducing the dataset by finding the outliers from the diversity data using a data transformation step. After that, the author applied the feature selection step to select those data values which shows some relationships and dependencies on other values. At this stage, the multidimensionality of the datasets will be reduced to some extent. This will help in better analysis of the data values. The whole procedure helps in identifying the abnormal behavior of the outlier values automatically. Wu et al. [68] worked in the same direction and designed an intrusion detection model for the autonomic systems. They used the auction method for detecting the intrusion at the agent coordination layer which lies between managed resource and autonomic manager. Using sensors and effectors, the method works efficiently to detect environmental changes. From the identified results, the auction method will use its technique and allocate resources to achieve high optimization. It will improve intrusion detection accuracy. Resource management is another challenge that needs to be handled in cloud computing. Autonomic provisioning in resource management tasks helps to improve QoS. Using the self-optimization feature of AC, resources may be best utilized. Singh et al. [69] provided a detailed description of the importance of resource management in the cloud using autonomic computing. The work concluded that automation in cloud computing helps in selecting resource utilization and resource scheduling algorithm at run time for a particular load-balancing task. Quality may be improved in terms of energy, cost and time. The authors also highlighted some limitation related to resource management.

- It is difficult to identify the best suitable resource distribution because it requires finding reasons for the workload.

- Different parameters need to be examined for different problems to estimate quality.
- Automatic workload execution should be done to avoid resource under-loading and overloading.

Big data is again a high-priority area where autonomic computing may play its role in improving analysis of the data. The handling of large data has become a cumbersome task for the developers. Big data autonomic handler (BGAH) approach may be implemented using autonomic concept. Berekmeri et al. [70] worked in the same context and merged autonomic with MapReduce. For improving performance in big data, the author proposed an algorithm for Big Data MapReduce. The author used the online feeding process to the MapReduce function. The author uses two constraints for online feeding: relaxed performance with two feedback control mechanisms which are used to minimize configuring of less number of clusters. The second constraint is strict performances which is used to feed-forward and results in suppressing the effect of a large workload. SCADA stands for supervisory control and Data Acquisition (SCADA). SCADA systems are used to control complex systems such as power generation, manufacturing plants and transportation networks. Now SCADA has been adopting trends such as virtualization, wireless communications, analytics and big data. Implementation of autonomic features in SCADA helps to manage the system itself. For this purpose, Nazir et al. [71] proposed architecture for SCADA security along with autonomic features. This type of architecture provides high availability of resources and reduces software and hardware failure.

Internet of Things (IoT) is another application area where researchers are finding solutions for better utility functions at home level. IoT is also facing the problem of complexity management due to which the developers are now combining IoT with a cloud so that the management will be done at a modular level. The interconnection of a large number of systems and resources makes the IoT network quite complex. Also, IoT suffers from the slow run-time execution environment. When IoT is combined with cloud, it will improve execution time and response faster. One of the major advantages of cloud computing is to store a large amount of data and also high computation power. With these advantages, there is also a problem of integrating final-user context-awareness. Golchay et al. [72] worked for this purpose by providing a gateway using smartphones between cloud and IoT. The phone acts as an autonomic enabled entity which reconfigures accordingly environmental changes, e.g., battery usage, when it is low. This complete network will work to monitor and provide response dynamically by selecting a suitable solution which acts as a self-adaption mechanism. Using the cloud, resource management may be done better but due to less context-awareness in the cloud, it becomes difficult for the users to communicate. The author uses the functionality of autonomic computing and applied some of them in their work. They applied awareness property for monitoring the activity using listener or event notifier. The listener will send notification step by step with the change. They also applied coordination mechanism and deployment which requires adaptive functionality for call redirection and migration using smartphones mobility. For this adaptive property, they used disrupted tolerant network

(DTN) for on-demand deployment services like opportunistic and spontaneous deployment services.

5 Conclusion

From the year 1960 to 2001, the self-management technique was not so much required in the IT industry but after 2001, the requirement of autonomic computing has raised many advantages which have overcome the weaknesses such as complexity management, security and privacy management and resource management. Autonomic computing makes the system self-dependent. From all the previous works discussed in this paper, the majority of the researchers have focused only on highlighting autonomic concept, its reference architecture, policies and MAPE-K loop. They have also discussed the importance of autonomic computing in a different domain where AC can bring cost-effective and quality results. From the above studies, the authors have identified a different vision for autonomic computing: the vision of autonomicity estimation and how the requirement of autonomous features changes with the change in the internal or external environment of the system during run time. According to the authors, with the shift of system development paradigm from nonautonomous to autonomous, the developers have also tried to develop a self-adaptable system with the change in the management requirements. These systems are not considered as fully autonomic as they are not capable of handling high-level management task automatically. These types of systems were made partially self-adjustable and self-recoverable. They were useful in handling issues at the lower level. But security- and privacy-based resources require automatic management and recovery plans. They need to ensure with an autonomic enabled system at their usage level. The need for autonomic computing varies with the domain and the level. If any application or software is designed with autonomic abilities based on its usage and domain, then their management graph will show improvement. Also, the testing of such systems will be easier and cost-effective. Our future work will be to understand the proposed solutions provided by researchers in both academic and industrial levels and will try to design a general architecture which will be cost- and quality-effective in all the SDLC phases and also for testing.

References

1. Horn P (2001) Autonomic computing: IBM's perspective on the state of information. IBM
2. Kephart JO, Chess DM (2003) The vision of autonomic computing. IEEE Comput 36(1):41–50
3. SAS home page (2015). https://www.darpa.mil/ato/programs/suosas.htm. Accessed 9 May 2019
4. Cobleigh JM, Osterweil LJ, Wise A, Lerner BS (2002) Containment units: a hierarchically composable architecture for adaptive systems. SIGSOFT Softw Eng Notes 27(6):159–165
5. Garlan D, Schmerl B, Chang J (2001) Using gauges for architecture-based monitoring and adaptation. In: Working Conference on Complex and Dynamic Systems Architecture, Brisbane, Australia
6. Kaiser G, Gross P, Kc G, Parekh J, Valletto G (2002) An approach to autonomizing legacy systems. In: Proceedings of the Workshop on Self-Healing, Adaptive and Self-MANaged Systems
7. Wolf AL, Heimbigner D, Bend JK (2000) Don't break: using reconfiguration to achieve survivability. In: Proceedings of the 3rd Information Survivability Workshop

8. Badger L (2004) Self-regenerative systems (SRS) program. Abstract.www.tolerantsystems.org/. Accessed 13 May 2019
9. Nami MR, Sharifi M (2007) Autonomic computing: a new approach. In: First Asia International Conference on Modelling & Simulation (AMS'07). IEEE, pp 352–357
10. IBM Corporation (2005) An architectural blueprint for autonomic computing, 3rd edn
11. McCann JA, Huebscher MC (2004) Evaluation issues in autonomic computing. In: Grid and Cooperative Computing Workshops. Springer, pp 597–608
12. Hariri SA (2005) Autonomic computing: an overview. Unconventional programming paradigms. Springer, Berlin, pp 257–269
13. Ganek AG, Corbi TA (2003) The dawning of the autonomic computing era. IBM Syst J 42(1):5–18
14. Salehie M, Tahvildari L (2005) Autonomic computing: emerging trends and open problems. ACM SIGSOFT Softw Eng Notes 30(4):1–7
15. Cloud Computing: Why the Future of Cloud lies in autonomics. https://www.comparethecloud.net/articles/why-the-future-of-cloud-lies-in-autonomics/. Accessed on March 2020
16. TahirM, Ashraf QM, Dabbagh M (2019) Towards enabling autonomic computing in IoT ecosystem. In: 2019 IEEE International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing, International Conference on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech). IEEE, pp 646–651
17. Vieira K, Koch FL, Sobral JBM, Westphall CB, de Souza Leão JL (2019) Autonomic Intrusion detection and response using big data. IEEE Syst J. https://doi.org/10.1109/JSYST.2019.2945555
18. Khalid A, Haye MA, Khan MJ, Shamail S (2009) Survey of frameworks, architectures and techniques in autonomic computing. In: Fifth International Conference on IEEE Explore
19. Muscettola N, Nayak PP, Pell B, Williams BC (1998) Remote agent: to boldly go where no AI system has gone before. Artif Intell 103(1–2):5–47
20. Huebscher MC, McCann JA (2008) A survey of autonomic computing—degrees, models, and applications. ACM Comput Surv 40(3):7
21. Garlan D, Cheng SW, Huang AC, Bradley S, Steenkiste P (2004) Rainbow: architecture-based self-adaptation with reusable infrastructure. IEEE Comput 37(10):46–54
22. Wang Q (2005) Towards a rule model for self-adaptive software. ACM SIGSOFT Softw Eng Notes 30(1):1–5
23. Kumar M, Sharma A (2017) An integrated framework for software vulnerability detection, analysis and mitigation: an autonomic system. Sādhanā 42(9):1481–1493
24. Pena J et al (2006) A model-driven architecture approach for modeling, specifying and deploying policies in autonomous and autonomic systems. In: 2006 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing. IEEE, pp 19–30
25. Brown B, Redlin C (2005) Measuring the effectiveness of self-healing autonomic systems. In: Second International Conference on Autonomic Computing (ICAC'05)
26. Eze T, Anthony R, Soper A, Walshaw C (2012) A generic approach towards measuring level of autonomicity in adaptive systems. Int J Adv Intell Syst 5(3&4):553–566
27. AutonomicComputingToolkit (2015). https://www.ibm.com/developerworks/autonomic/books/fpu1mst.htm. Accessed 1 June 2019
28. About IBM Autonomic Computing (2015). https://www-03.ibm.com/autonomic/about_get_model.html. Accessed 1 June 2019
29. www.research.ibm.com/autonomic/academic/research.html (2015). Accessed 1 June 2019
30. Sharma A, Dehraj P (2015) Complexity based maintenance assessment for autonomic agent. In: WSEAS- Conference, Rome, Italy, pp 7–9
31. Sharma A, Dehraj P (2015) Complexity assessment for autonomic system using neuro-fuzzy approach. In: CSI- Conference. Springer, Delhi
32. Kumari N, Sunita S (2013) Comparison of ANNs, fuzzy logic and neuro-fuzzy integrated approach for diagnosis of coronary heart disease: a survey. IJCSMC 2(6):216–224
33. Patterson D, Brown A, Broadwell P, Candea G, Chen M, Cutler J, Enriquez P, Fox A, Kiciman E, Merzbacher M, Oppenheimer D (2002) Recovery-oriented computing (ROC): motivation, definition, techniques, and case studies. Technical report UCB//CSD-02–1175, UC Berkeley Computer Science, pp 1–25.
34. Astley M, Bhola S, Saccone R (1997) The Gryphon project. IBM. www.research.ibm.com/distributedmessaging/gryphon.html. Accessed 2015

35. Zhang R (2007) Autonomic performance recuperation for service-oriented systems. In: IEEE International Conference on Services Computing
36. Stillger M, Lohman GM, Markl V (2001) LEO-DB2's learning optimizer. VLDB 1:19–28
37. Lohman GM, Lightstone SS (2002) SMART: making DB2 (more) autonomic. In: 28th International Conference on Very Large Data Bases
38. Menon J, Pease DA, Reese R, Duyanovich L, Hillsberg B (2003) IBM storage tank—a heterogeneous scalable SAN file system. IBM Syst J 42(2):250
39. JangraA, Bishla D, Bhatia K, Priyanka P (2010) Functionality and security analysis of ORACLE, IBM-DB2 & SQL server. Glob J Comput Sci Technol 10(7)
40. Mateen A, Raza B, Hussain T, Awais MM (2008) Autonomic computing in SQL server. In: Seventh IEEE/ACIS International Conference on Computer and Information Science (ICIS 2008). IEEE, pp 113–118
41. RazaB, Mateen A, Sher M, Awais MM, Hussain T (2010) Autonomic view of query optimizers in database management systems. In: 2010 Eighth ACIS International Conference on Software Engineering Research, Management and Applications (SERA). IEEE, pp 3–8
42. Oracle Corporation: Technical Comparison of Oracle database Vs IBM DB2 UDB: Focus on performance, Oracle White Paper. https://www.oracle.com/technetwork/database/database10g/twp-perf-oracledb10gr2vsibmdb2udb-159510.pdf. Accessed 15 Sept 2019
43. Herodotou H, Babu S (2010) Xplus: a SQL-tuning-aware query optimizer. Proc VLDB Endow 3(1–2):1149–1160
44. DehrajP Sharma A (2019) An empirical assessment of autonomicity for autonomic query optimizers using F-AHP approach. Appl Soft Comput J 90:106137
45. Pfannemüller M, Krupitzer C, Weckesser M, Becker C (2017) A dynamic software product line approach for adaptation planning in autonomic computing systems. In: 2017 IEEE International Conference on Autonomic Computing (ICAC). IEEE, pp 247–254
46. Sharma A, Chauhan S, Grover P (2011) Autonomic computing: paradigm shift for software development. CSI Commun 35
47. Chauhan S, Sharma A, Grover P (2013) Developing self managing software systems using agile modeling. ACM SIGSOFT Softw Eng Notes 38(6):1–3
48. Shuaib H, Anthony R, Pelc M (2011) A framework for certifying autonomic computing systems. In: The Seventh International Conference on Autonomic and Autonomous Systems
49. Holler J, Tsiatsis V, Mulligan C, Karnouskos S, Boyle D (2014) From machine-to-machine to the Internet of Things: introduction to a new age of intelligence. Academic Press, New York
50. Oreizy P, Gorlick MM, Taylor RN, Heimhigner D, Johnson G, Medvidovic N, Wolf AL (1999) An architecture-based approach to self-adaptive software. IEEE Intell Syst Appl 14(3):54–62
51. Wang M, Suda T (2001) The bio-networking architecture: a biologically inspired approach to the design of scalable, adaptive, and survivable/available network applications. In: Proceedings. 2001 Symposium on Applications and the Internet. IEEE
52. Sterritt R, Hinchey M (2005) Tutorial proposal: autonomic computing in real-time systems. www.artes.uu.se/events/summer05/Hinchey_tutorial.pdf
53. Solomon B, Ionescu D, Litoiu M, Iszlai G, Prostean O (2010) Measurements and identification of autonomic computing processes. In: 2010 IEEE International Conference Computational Intelligence for Measurement Systems and Applications (CIMSA), pp 72–77
54. Portela AER, Perdomo JG (2011) Survey: termites system with self-healing based on autonomic computing. In 2011 6th Colombian Computing Congress (CCC), pp 1–6
55. Kurian D, Chelliah PR (2012) An autonomic computing architecture for business applications. In IEEE, Information and Communication Technologies (WICT), 2012 World Congress, pp 442–447
56. De Nicola R, Ferrari G, Loreti M, Pugliese R (2013) A language-based approach to autonomic computing. Formal methods for components and objects. Springer, Berlin
57. Nhane ALO, Song MAJ (2014) Self-optimization in autonomic computing systems based on the methodology of bees swarm intelligence. The Steering Committee of the World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)
58. Schneider C, Barker A, Dobson S (2015) Autonomous fault detection in self-healing systems using restricted boltzmann machines. In: IEEE Conference. arXiv preprint arXiv:1501.01501
59. Manzalini A, Deussen PH, Nechifor S, Mamei M, Minerva R, Moiso C, Zambonelli F (2010) Self-optimized cognitive network of networks. Comput J 54(2):189–196

60. Raza B, Mateen A, Sher M, Awais MM, Hussain T (2010) Autonomic view of query optimizers in database management systems. In: 2010 Eighth ACIS International Conference on Software Engineering Research, Management and Applications (SERA). IEEE, pp 3–8
61. Sharma A, Dehraj P (2016) Towards autonomicity: from man to machine and its challenges. In: CTICON- Conference, Delhi, India, May 27–28
62. Al-Oqily I, Alzboon M, Al-Shemery H, Alsarhan A (2013) Towards autonomic overlay self-load balancing. In: 2013 10th International Multi-Conference on Systems, Signals & Devices (SSD). IEEE, pp 1–6
63. Atif Y, Badr Y, Maamar Z (2010) Towards a new-digital learning ecosystem based on autonomic Web services. In: 2010 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST). IEEE
64. Alaya MB, Monteil T (2012) Frameself: a generic context-aware autonomic framework for self-management of distributed systems. In: 2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). IEEE
65. Exposito E, Chassot C, Diaz M (2010) New generation of transport protocols for autonomous systems. In: 2010 IEEE GLOBECOM Workshops (GC Wkshps). IEEE
66. Dehraj P, Sharma A, Grover PS (2018) Incorporating autonomicity and trustworthiness aspects for assessing software quality. IJET 7(1.1):421–425
67. Smith D, Guan Q, Fu S (2010) An anomaly detection framework for autonomic management of compute cloud systems. In: 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops (COMPSACW). IEEE
68. Wu Q, Zhu L, Cao J, Zheng R (2012) Proactive intrusion detection model based on autonomic computing. In: International Conference on Automatic Control and Artificial Intelligence (ACAI 2012), IET, pp 1601–1604
69. Singh S, Chana I, Singh M (2017) The journey of QoS-aware autonomic cloud computing. IT Prof 19(2):42–49
70. Berekmeri M, Serrano D, Bouchenak S, Marchand N, Robu B (2016) Feedback autonomic provisioning for guaranteeing performance in mapreduce systems. IEEE Trans Cloud Comput 6(4):1004–1016
71. Nazir S, Patel S, Patel D (2017) Autonomic computing meets SCADA security. In: Proceedings of 2017 IEEE 16th International Conference on Cognitive Informatics and Cognitive Computing, ICCI* CC 2017. London South Bank University, pp 498–502
72. Golchay R, Mouël FL, Frénot S, Ponge J (2011) Towards bridging IOT and cloud services: proposing smartphones as mobile and autonomic service gateways. arXiv preprint arXiv:1107.4786

Affiliations

Pooja Dehraj[1] · Arun Sharma[1]

✉ Pooja Dehraj
poojadehraj2000@gmail.com

Arun Sharma
arunsharma@igdtuw.ac.in

[1] Indira Gandhi Delhi Technical University for Women, Delhi, India