ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 6, Issue 2, pp: (47-53), Month: April - June 2018, Available at: www.researchpublish.com

AUTONOMIC COMPUTING

[1]Prof. Jincy C Mathew, [2]Sona Bai B, [3]Srieeharini B N

[1, 2, 3] Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Department of Master of Computer Applications, NHCE, Bangalore, India Country

Abstract: **Autonomic computing is a new technology to solve crisis of software complexity which means to hide the complexity of system management from human users by means of technologies managing technologies, to establish guidable, state-aware, and self-adaptive computer systems. According to the analysis of domestic and foreign representative literature, this paper makes an overview on concept of autonomic computing and presents autonomic computing properties and sampling applications. Then the important theories and technologies for autonomic computing system models are given. Finally, the research problems and future directions of autonomic computing are discussed.**

Keywords: **Autonomic computing, technology, computer systems.**

I. INTRODUCTION

Autonomic computing technology is an emerging research topic to solve the software complexity crisis. It is inspired by the function of the human nervous system and is aimed at designing and building systems that are self-managing. Its thoughts is to make systems manage their own under the guidance of management strategies developed by IT managers, through the technologies managing technologies to hide the system complexity.[16]

As a result, the system can realize self-configuration, self-optimization, self-healing and self-protection. According to the self-adjustment mechanism of autonomic nervous system and the existing theory and technology (including service-oriented computing, adaptive control theory, optimization theory, policy-based management [5], multi-agent technology, etc.), autonomic computing is to build autonomic computing system. Therefore, it makes the information system achieve self-management in holistic. [7]

“Autonomic Computing” is a new vision of computing initiated by IBM. This new paradigm shifts the fundamental definition of the technology age from one of computing, to one defined by data. Access to data from multiple, distributed sources, in addition to traditional centralized storage devices will allow users to transparently access information when and where they need it. At the same time, this new view of computing will necessitate changing the industry's focus on processing speed and storage to one of developing distributed networks that are largely self-managing, self-diagnostic, and transparent to the user.[1]

II. MODULE SPECIFICATION

A. Key Elements of Autonomic Computing:

The elements of autonomic computing can be summarized in to 8 key points.

Knows Itself

An autonomic computing system needs to "know itself" - its components must also possess a system identity. Since a "system" can exist at many levels, an autonomic system will need detailed knowledge of its components, current status, ultimate capacity, and all connections to other systems to govern itself. It will need to know the extent of its "owned" resources, those it can borrow or lend, and those that can be shared or should be isolated.

Configure Itself

An autonomic computing system must configure and reconfigure itself under varying (and in the future, even unpredictable) conditions. System configuration or "setup" must occur automatically, as well as dynamic adjustments to that configuration to best handle changing environments

Optimises Itself

An autonomic computing system never settles for the status quo - it always looks for ways to optimize its workings. It will monitor its constituent parts and fine-tune workflow to achieve predetermined system goals.[9]

Heal Itself

An autonomic computing system must perform something akin to healing - it must be able to recover from routine and extraordinary events that might cause some of its parts.

Protect Itself

A virtual world is no less dangerous than the physical one, so an autonomic computing system must be an expert in self-protection. It must detect, identify and protect itself against various types of attacks to maintain overall system security and integrity

Adapt Itself

An autonomic computing system must know its environment and the context surrounding its activity, and act accordingly. It will find and generate rules for how best to interact with neighbouring systems. It will tap available resources, even negotiate the use by other systems of its underutilized elements, changing both itself and its environment in the process -- in a word, adapting.[14]

Open Itself

An autonomic computing system cannot exist in a hermetic environment. While independent in its ability to manage itself, it must function in a heterogeneous world and implement open standards -- in other words, an autonomic computing system cannot, by definition, be a proprietary solution.

Hide Itself

An autonomic computing system will anticipate the optimized resources needed while keeping its complexity hidden. It must marshal I/T resources to shrink the gap between the business or personal goals of the user, and the I/T implementation necessary to achieve those goals -- without involving the user in that implementation

B. Autonomic Computing Architecture:

The autonomic computing architecture concepts provide a mechanism discussing, comparing and contrasting the approaches different vendors use to deliver self-managing attributes in an autonomic computing system. The autonomic computing architecture starts from the premise that implementing self-managing attributes involves an[10] intelligent control loop. This loop collects information from the system make decisions and then adjusts the system as necessary. An intelligent control loop can enable the system to do such things as:

- Self-configure, by installing software when it detects that software is missing
- Self-heal, by restarting a failed element
- Self-optimize, by adjusting the current workload when it observes an increase in capacity
- Self-protect, by taking resources offline if it detects an intrusion attempt.

A. Control loops

A control loop can be provided by a resource provider, which embeds a loop in the runtime environment for a particular resource. In this case, the control loop is configured through the manageability interface provided for that resource (for example, a hard drive).In some cases, the control loop may be hard-wired or hard coded so it is not visible through the manageability interface.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 6, Issue 2, pp: (47-53), Month: April - June 2018, Available at: www.researchpublish.com

Autonomic systems will be interactive collections of *autonomic elements*—individual system constituents that contain resources and deliver services to humans and other autonomic elements. , An autonomic element will typically consist of one or more managed elements coupled with a single autonomic manager that controls and represents them.

In an autonomic environment, autonomic elements work together, communicating with each other and with high-level management tools. They regulate themselves and, sometimes, each other. They can proactively manage the system, while hiding the inherent complexity of these activities from end users and IT professionals. Another aspect of the autonomic computing architecture is shown in the diagram below. This portion of the architecture details the functions that can be provided for the control loops. The architecture organizes the control loops into two major elements —a managed element and an autonomic manager. A managed element is what the autonomic manager is controlling. An autonomic manager is a component that implements a control loop.

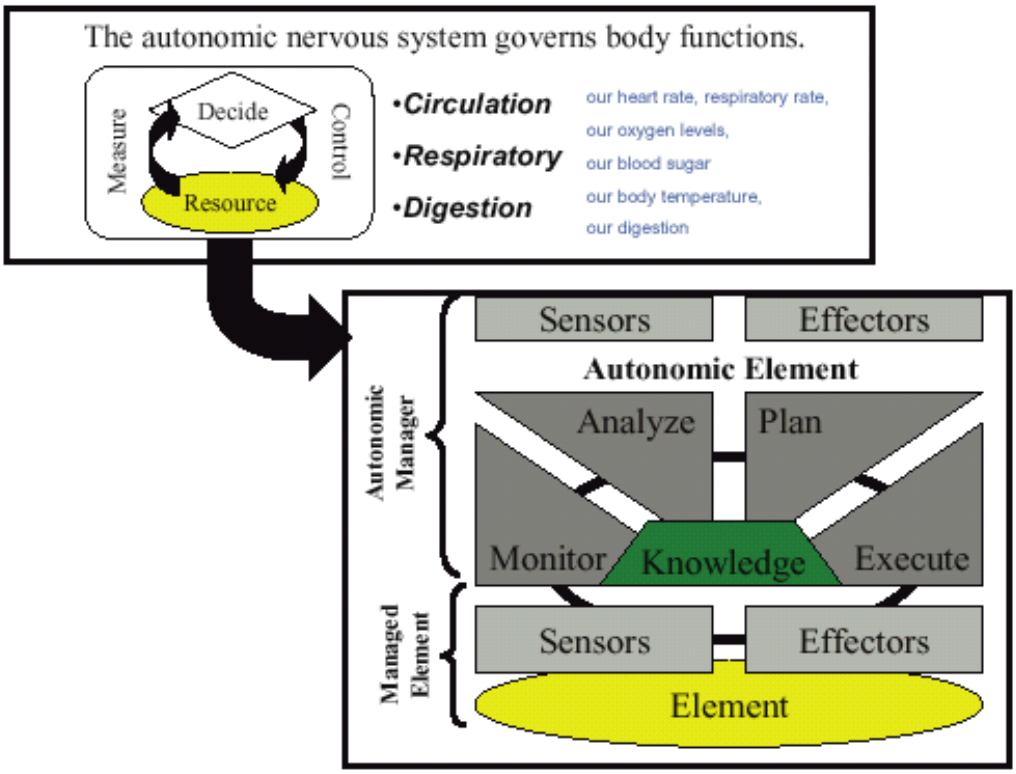

Figure 1: In an autonomic computing architecture, control loops facilitate system management.

B. Managed Elements

The managed element is a controlled system component. The managed element will essentially be equivalent to what is found in ordinary non autonomic systems, although it can be adapted to enable the autonomic manager to monitor and control it. The managed element could be a hardware resource, such as storage, CPU, or a printer, or a software resource, such as a database, a directory service, or a large legacy system. At the highest level, the managed element . could be an e utility, an application service, or even an individual business .The managed element is controlled through its sensors and effectors:

- The sensors provide mechanisms to collect information about the state and state transition of an element. To implement the sensors, you can either use a set of "get "operations to retrieve information about the current state, or a set of management events (unsolicited, asynchronous messages or notifications)that flow when the state of the element changes in a significant way.
- The effectors are mechanisms that change the state (configuration) of an element. In other words, the effectors are a collection of "set "commands or application programming interfaces (APIs)that change the configuration of the managed resource in some important way.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 6, Issue 2, pp: (47-53), Month: April - June 2018, Available at: www.researchpublish.com

The combination of sensors and effectors form the manageability interface that is available to an autonomic manager. As shown in the figure above, by the black lines connecting the elements on the sensors and effectors sides of the diagram, the architecture encourages the idea that sensors and effectors are linked together. For example, a configuration change that occurs through effectors should be reflected as a configuration change notification through the sensor interface.

C. Autonomic manager

The autonomic manager is a component that implements the control loop. The autonomic manager distinguishes the autonomic element from its non autonomic counterpart. By monitoring the managed element and its external environment, and constructing and executing plans based on an analysis of this information, the autonomic manager will relieve humans of the responsibility of directly managing the managed element.

The architecture dissects the loop into four parts that share knowledge:

- The monitor part provides the mechanisms that collect, aggregate, filter, manage and report details (metrics and topologies) collected from an element.
- The analyze part provides the mechanisms to correlate and model complex situations (time-series forecasting and queuing models, for example). These mechanisms allow the autonomic manager to learn about the IT environment and help predict future situations.
- The plan part provides the mechanisms to structure the action needed to achieve goals and objectives. The planning mechanism uses policy information to guide its work.
- The execute part provides the mechanisms that control the execution of a plan with considerations for on-the-fly updates.

The following diagram provides a more detailed view of these four parts by highlighting some of the functions each part uses.

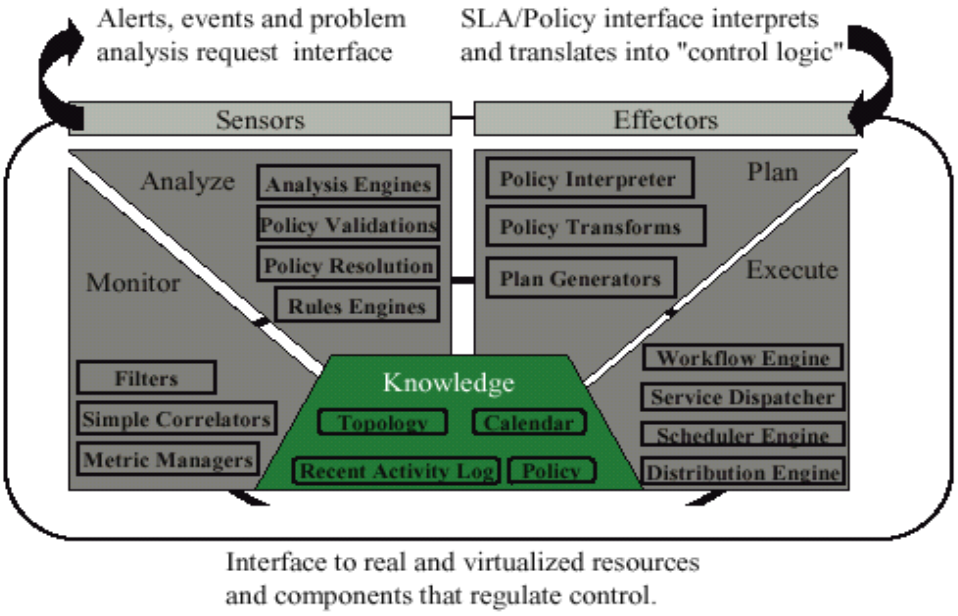

Figure 2

The functional details of an autonomic manager:

The four parts work together to provide the control loop functionality. The diagram shows a structural arrangement of the parts —not a control flow. The bold line that connects the four parts should be thought of as a common messaging bus rather than a strict control flow. In other words, there can be situations where the plan part may ask the monitor part to collect more or less information. There could also be situations where the monitor part may trigger the plan part to create a new plan. The four parts colloborate using asynchronous communication techniques, like a messaging bus.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 6, Issue 2, pp: (47-53), Month: April - June 2018, Available at: www.researchpublish.com

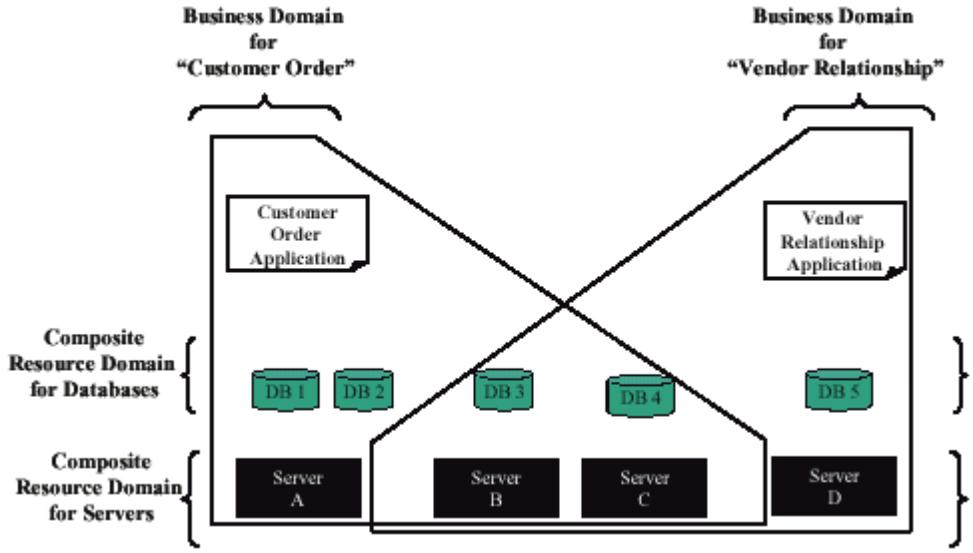

Figure 3: IT systems can share resources to increase efficiency

Now, let us apply the autonomic computing architecture to this example, to see how the autonomic managers would be used. The following diagram illustrates some of the autonomic managers that either directly or indirectly manage DB 3 and some of the interaction between these autonomic managers. There are six autonomic managers in this illustration: one for each of the management domains, one embedded in the DB 3 resource and one dedicated to the specific database resource. Since the decision-making contexts for these autonomic managers are. [17]

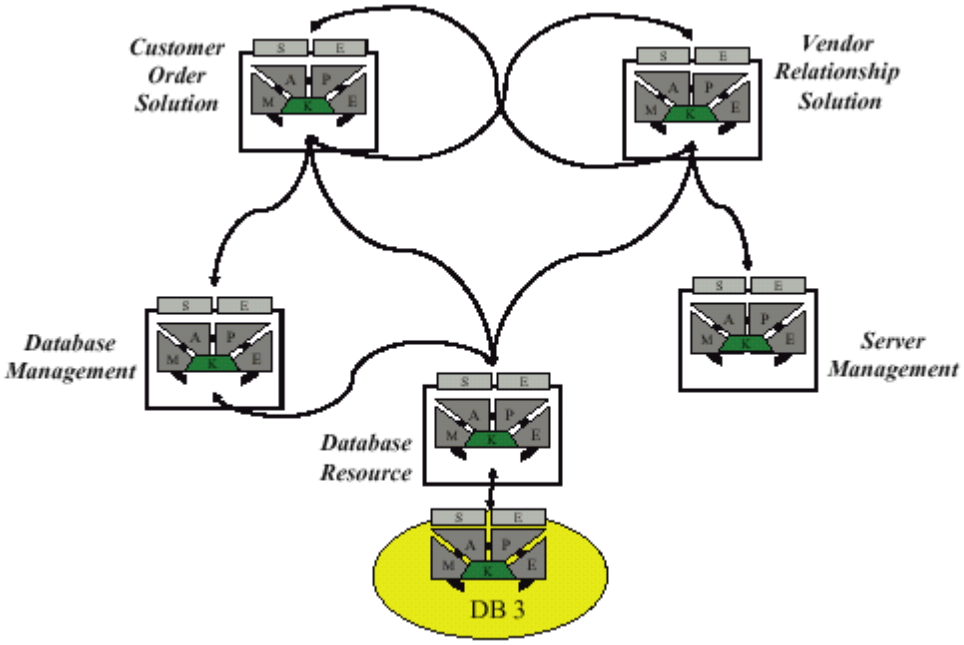

Figure 4: Six autonomic managers directly and indirectly manage the DB3 resource

D. Self-managing systems change the IT business

The mechanics and details of IT processes, such as change management and problem management, are different, but it is possible to categorize these into four common functions —collect the details, analyze the details, create a plan of action and execute the plan. These four functions correspond to the monitor, analyze, plan and execute parts of the architecture. The analyze and plan mechanisms are the essence of an autonomic computing system, because they encode the "know how " to help reduce the skill and time required of the IT professional. Fully autonomic computing is likely to evolve as

designers gradually add increasingly sophisticated autonomic managers to existing managed elements. Ultimately, the distinction between the autonomic manager and the managed element may become merely conceptual rather than architectural, or it may melt away—leaving fully integrated, autonomic elements with well-defined behaviours and interfaces, but also with few constraints on their internal structure. Each autonomic element will be responsible for managing its own internal state and behaviour and for managing its interactions with an environment that consists largely of signals and messages from other elements and the external world. An element's internal behaviour and its relationships with other elements will be driven by goals that its designer has embedded in it, by other elements that have authority over it, or by subcontracts to peer elements with its tacit or explicit consent.

III. CONCLUSION

Is it possible to meet the grand challenge of autonomic computing without magic and without fully solving the AI problem? It is possible, but it will take time and patience. Long before we solve many of the more challenging problems, less automated realizations of autonomic systems will be extremely valuable, and their value will increase substantially as autonomic computing technology improves and earns greater trust and acceptance. A vision this large requires that we pool expertise in many areas of computer science as well as in disciplines that lie far beyond computing traditional boundaries.

We must look to scientists studying nonlinear dynamics and complexity for new theories of emergent phenomena and robustness. We must look to economists and e-commerce researchers for ideas and technologies about negotiation and supply webs. We must look to psychologists and human factors researchers for new goal-definition and visualization paradigms and for ways to help humans build trust in autonomic systems. We must look to the legal profession, since many of the same issues that arise in the context of e-commerce will be important in autonomic systems that span organizational or national boundaries. Bridging the language and cultural divides among the many disciplines needed for this endeavour and harnessing the diversity to yield successful and perhaps universal approaches to autonomic computing will perhaps be the greatest challenge. It will be interesting to see what new cross-disciplines develop as we begin to work together to solve these fundamental problems.

No company is going to be able to control all the parts of an autonomic system. It's going to have to be an open source system because there are so m any parts. These parts will provide plenty of opportunity for competition.

REFERENCES

[1] J. Kephart, D. Chess, "The Vision of Autonomic Computing", IEEE Computer Society, vol. 36, pp. 41-50, Jan. 2003.

[2] P Horn, "Autonomic computing: IBM's perspective on the state of information technology", IBM Corporation, 2001.

[3] "SMART (Self Managing and Resource Tuning)", IBM Research, 2003.

[4] "An architectural blueprint for autonomic computing", IBM, 2003.

[5] Dearle Alan, Graham N.C. Kirby, Andrew J. McCarthy, "A Framework for Constraint-Based Deployment and Autonomic Management of Distributed Applications", IEEE Computer Society, pp. 300-301, May. 2004.

[6] Aniruddha Bohra, Iulian Neamtiu, Pascal Gallard, "Remote Repair of Operating System State Using Backdoors", IEEE Computer Society, pp. 256-263, May.2004.

[7] Andrew C. Huang, Armando Fox, "Cheap Recovery: A key to self-managing state", ACM Transactions on Storage, vol. 1, pp. 38-70, Feb. 2005.

[8] Guangzhi Qu, Salim Hariri, Santosh Jangiti, "Online Monitoring and Analysis for Self-Protection against Network Attacks", IEEE Computer Society, 2004.

[9] S. M. Sadjadi, P. K. McKinley, "Transparent Self-Optimization in Existing CORBA Applications", IEEE Computer Society, pp. 88-95, 2004.

[10] Beishui Liao, Yuan Yao, Ji Gao, "Conceptual Model and Realization Methods of Autonomic Computing", Journal of Software, vol. 19, pp. 779-802, Apr. 2008.

[11] JO Kephart, R Das, "Achieving self-management via utility functions", IEEE Internet Computing, vol. 11, pp. 40-47, 2007.

[12] R Sterritt, "Autonomic computing: The natural fusion of soft computing and hard computing", IEEE Computer Society Press, pp. 4754-4759, 2003.

[13] C Franke, W Theilmann, Y Zhang, R STERRITT, "Towards the autonomic business grid", IEEE Computer Society, pp. 107-112, 2007.

[14] Y S Dai, "Autonomic computing and reliability improvement", IEEE, pp. 204-206, 2005.

[15] J. P. Bigus, "ABLE: A toolkit for building multi-agent autonomic systems", IBM Systems Journal, vol. 41, pp. 350-359, Sep. 2002.

[16] baolei Ge, Shuyi Wang, "Research on Autonomic Computing System", Dalian University of Technology, 2008.

[17] J. Jann, L. A. Browning, R. S. Burugula, "Dynamic Reconfiguration: Basic Building Blocks for Autonomic Computing on IBM pSeries Servers", IBM Systems Journal, vol. 42, pp. 29-37, 2003.

[18] Sam Lightstone, "Foundations of Autonomic Computing Development", Proceedings of Engineering of Autonomic and Autonomous Systems(EASe '07), pp. 163-171, Mar. 2007.