International Journal of Engineering Research and General Science Volume 3, Issue 2, March-April, 2015
ISSN 2091-2730

REVIEW ON AUTONOMIC COMPUTING: THE BRIEF INTRODUCTION

Geeta M. Rane
Student of Third Year of Computer Science and Engineering
Shri Sant Gadge Baba College of Engineering & Technology, Bhusawal,
North Maharashtra University, Jalgaon, Maharashtra, India.
E-mail- onlygeeta95@gmail.com

Abstract— Computing coordination is constantly growing in complexity. With that development, the challenge of operating and keep up them improved. In some situations, these schemes may occur in lenient and detached environments creating such operations even more challenging. To address the previous problems, the idea of autonomic computing invented. In the year of 2001, IBM introduced a new computing system i.e. an Autonomic Computing (AC) system. It will constantly check and elevate its status, and automatically regulate itself to altering surroundings. The AC system is a self-managing system. Self-management is attained through key aspects such as self-optimization, self-configuration, and self-protection and self-healing. This paper presents autonomic computing that emphases on the architecture of an AC system and the framework elements needed for constructing the AC system.

Keywords— Autonomic Computing, self-management, features, architecture of AC, framework element, building blocks, growing complexities, IT professionals.

I. INTRODUCTION

The advances in computing and communication technologies and software have resulted in a tremendous growth in computing systems and applications that impact all aspects of our life. As the scale and difficulty of these systems and applications raise, their development, configuration and management challenge are start to break current paradigms, completely covers the capabilities of existing tools and methodologies, and rapidly concentrate the systems and applications hard, incontrollable and insecure.

Autonomic computing is emerging as a significant new strategic and fundamental approach to the design of complex distributed computer systems. IBM introduced a new term Autonomic Computing (AC) in the year of 2001. It depicts the systems that are self-managing. The term Autonomy was inspired by the term autonomic nervous system that that controls vital body functions without an individual's knowledge. The Autonomic Computing systems involve of four key characteristics: Self-configuring, Self-optimization, Self-protection and Self-healing [1], [2], [3]. At 2005, IBM merged around 475 autonomic features into more than 75 products.

NASA is one of the leading organizations that build complex mission critical systems with autonomous behavior. To them, Autonomy provides great benefit; it helps developing spacecraft systems that can explore regions of space where traditional crafts cannot explore. Some of the successful systems with autonomic features developed by NASA are Deep Space 1, Earth Observing 1 and Mars Exploration Rovers [1].

1. Need of Autonomic Computing

In the developments of human and society automation has always been the base of progress. If human can handle one of his needs automatically, then he has free mind and resources to focus on another task. So step by step he can get ability to focus on more composite problem.

But the computing system verified that evolution via automation also produces complexity as a compulsory side effect. Follow the evolution of computers from single machines to modular system to personal computer networked with larger machines and an unmistakable pattern appears. Relate with previous machines unbelievable progress in almost every aspect if computing. Along with that growth has become increasingly refined architectures governed by software whose complexity now demands tens of millions of lines of code. In fact, the increasing difficulty of IT infrastructure threatens to challenge the very benefits IT goals to provide. Until now the computer systems trusted mainly on human interventions and administration to manage this difficulty. When considering about the current rates of growths, there will not be adequate skilled IT people to keep the world's computing systems running. Even in tentative financial times, still have great demand for trained IT workers.

Even if people could in some way come up with enough skilled people, the difficulty is growing beyond human ability to be able to it. As computing grows the overlapping connections, reliance and work together applications call for administrative decision making and retorts quicker than any human can deliver[2]. Recognizing root causes of failure becomes more problematic, while searching ways of growing system efficiency creates problem with more flexible than any human can hope to solve. Without new tactics, things will get worse. To solve problem people need computer systems with autonomic behavior.

2. Related Work

The author Mona A. Yahya, Manal A. Yahya, Dr. Ajantha Dahanayake Proposed a framework that explains to software analysts the main aspects to consider for accommodating software autonomy in the requirements engineering phase [1]. The paper proposed by the author Jeffrey O. Kephart, David M. Chess highlighted the architectural considerations of AC and different engineering and scientific challenges to the autonomic computing system [2]. The paper stressed three basic approaches to make a system Autonomic [6], the author Vítor E. Silva Souza Proposed a design for adaptive system using goals requirements. The author D. B. Abeywickrama, N. Bicocchi, F. Zambonelli proposed the goal modelling helps in identifying the functional and non-functional requirements of an adaptive system [7].

II. Features of Autonomic Computing

Autonomic computing systems combine four main features:

- Self-Configuration
- Self-Healing
- Self-Optimization
- Self-Protection

Fig 1 shows the features of autonomic computing

Fig: Features of Autonomic Computing

1. Self-configuration

The author Jeffrey O. Kephart, David M. Chess says that connecting, organizing, and integrating large, compound systems is challenging, time consuming, and mistake susceptible to even for experts[2]. With the capability to dynamically configure itself, an IT environment can adapt. Immediately with minimal intervention to the deployment of new components or changes in the IT environment.

2. Self-healing

Self-healing IT environments can detect problematic operations and then initiate corrective action without troublesome system applications. The Autonomic Computing systems will detect, diagnose, and repair localized problems resulting from bugs or failures in software and hardware, maybe through a regression tester [2]. Using information about the system configuration, a problem diagnosis component would analyze information from log files, possibly supplemented with data from additional monitors that it has requested. The system would then match the analysis against known software patches, install the appropriate patch, and retest.

3. Self-optimizing

Self-optimization refers to the ability of the IT environment to efficiently maximize resource provision and utilization to meet end users' needs with minimal intervention. In the near term, self-optimization mainly addresses the difficulties of managing system performance. In the extended span, self-optimizing components may learn from experience and automatically and proactively adjust themselves in the background of an overall business objective [2].

4. Self-protecting

A self-protecting environment allows authorized people to access the right data at the right time and can take proper actions automatically to make itself less susceptible to attacks on its run-time organization and business data. A self-protecting IT environment can detect hostile or invasive behavior as it occurs and take autonomous actions to make itself less vulnerable to unauthorized access and usage, computer program viruses, denial-of-service attacks, and common disappointments.

III. THE CONCEPTUAL ARCHITECTURE OF AUTONOMIC COMPUTING SYSTEMS

The building block of autonomic computing system is called an autonomic element. An autonomic element is a distinct system component that contains resources and delivers services to human and other autonomic elements. The general structure of an autonomic element is depicted in Figure [4]. The five structural blocks for an autonomic system are:

- Autonomic manager
- Knowledge source
- Touchpoint
- Manual manager
- Enterprise service bus

Fig: Architecture of Autonomic Computing

1. Autonomic Manager

An autonomic manager is an application that automates some management function and externalizes this function according to the behavior defined by management crossing point. The autonomic manager is a constituent that implements the control loop. For a system element to be self-managing, it must have an automated method to bring together the details it necessities from the system; to study those details to define if something needs to change; to create a plan, or order of actions, that states the vital changes; and to perform those actions. When these functions can be automated, a smart control loop is formed [4]. As shown in Figure, the architecture separates the loop into four parts that share knowledge. These four parts work together to provide the control loop functionality. The four parts communicate and work together with one another and exchange appropriate knowledge and data. Autonomic managers, in a manner related to touchpoints, provide sensor and effector manageability interfaces for other autonomic managers and manual managers to use.

Autonomic manager internal structure

a. Monitor

The monitor function gathers the details from the managed resources, via touchpoints, and associates them into indicators that can be analyzed. The details can consist of topology data, metrics, and arrangement property settings and so on. This data includes information around managed resource arrangement, rank, accessible capacity and throughput. Some of the data is stationary or

variations gradually, whereas further data is dynamic, varying constantly over and done with time. The monitor function aggregates, compares and filters these details up to it governs an indicator that needs to be analyzed [4].

Autonomic managers must gather and process large quantities of data from the touchpoints sensor interface of a managed resource. An autonomic manager's skill to promptly organize and make sense of this data is vital to its successful operation.

b. Analyze

The analyze function provides the mechanisms to observe and analyze conditions to determine if some change needs to be made. The analyze function is responsible for determining if the autonomic manager can tolerate by the recognized policy, now and in the future. In many cases, the analyze function models complex performance so it can hire predictions techniques such as time series forecasting and queuing models. These mechanisms permit the autonomic manager to learn about the IT environment and help forecast future behavior.

Autonomic managers must be able to perform complex data analysis and reasoning on the symptoms provided by the monitor function. If changes are required, the analyze function creates a change request and logically passes that change request to the plan function.

c. Plan

The plan function creates or selects a procedure to pass a desired modification in the managed resource. The plan functions can receipts on various forms, extending from a single command to a compound workflow. The plan function produces the suitable change plan, which signifies a desired set of changes for the succeeded resource, and understandably permits that change plan to the execute function [4].

d. Execute

The execute function offers the mechanism to plan and perform the necessary changes to the system. Once an autonomic manager has produced a change plan that agrees to a change request, some actions may need to be taken to change the state of one or more accomplished resources. The execute function of an autonomic manager is responsible for carrying out the procedure that was formed by the plan function of the autonomic manager through a sequence of actions. These actions are implemented using the touchpoint effector interface of a managed resource. Amount of the execution of the alteration plan could contain updating the knowledge that is used by the autonomic manager.

2. Knowledge Source

A knowledge source is an implementation of a record, vocabulary, database or other source that make available access to knowledge according to the interfaces arranged by the architecture. In an autonomic system, knowledge contains of certain types of data with architected syntax and semantics, such as signs, policies, change requests and change plans. This knowledge can be kept in a knowledge source so that it can be shared between autonomic managers. The knowledge kept in knowledge sources can be used to spread out the knowledge capabilities of an autonomic manager.

Data used by the autonomic manager's 4 functions (monitor, analyze, plan and execute) are kept as shared knowledge. The shared knowledge contains data such as topology information, old logs, metrics, signs and policies.

3. Touchpoints

A touchpoint is the part in a system that exposes the state and management operations for a resource in the system. An autonomic manager interconnects with a touchpoint through the manageability interface. A touchpoint is the application of the manageability interface for an exact manageable resource or a set of connected manageable resources.

Manageability Interface

The manageability interface for governing a manageable resource is organized into its sensor and effector interfaces. The manageability interface reduces the complexity by offering a standard interface to autonomic managers, rather than the different manageability interface mechanisms related with several kinds of manageable resources. A touchpoint equipment the sensor and effector behavior for definite manageable resource types by plotting the standard sensor and effector interfaces to one or more of the manageable resource's manageability interface mechanisms. The sensor and effector in the architecture are connected together.

A manageability capability refers to a logical collection of manageable resource state information and operations. Some examples of manageability capabilities are: identification, metrics, and configuration. For each manageability capability, the user of the manageability interface must be able to acquire and regulate state data through the manageability interface.

4. Manual Manager

A manual manager is an application of the user interface that allows an IT expert to execute some management function manually. The manual manager can work together with other autonomic managers at the similar level or organize autonomic managers and other IT professionals working at "lower" levels.

The manual manager building block is the architectural illustration of the human action and typically includes a human using an organization support. A manual manager can allow an IT professional to give management functions to autonomic managers.

5. Enterprise Service Bus

An enterprise service bus is an operation that supports in integrating other building blocks (for example, autonomic managers and touchpoints) by leading the connections among these building blocks. The enterprise service bus can be used to "connect" several autonomic computing building blocks.

IV. FRAMEWORK ELEMENTS

Following is a disintegration of the proposed framework by the author Mona A. Yahya et.al. Inputs to the framework are the Requirements Specification and System Goals as distinct by the stakeholders.

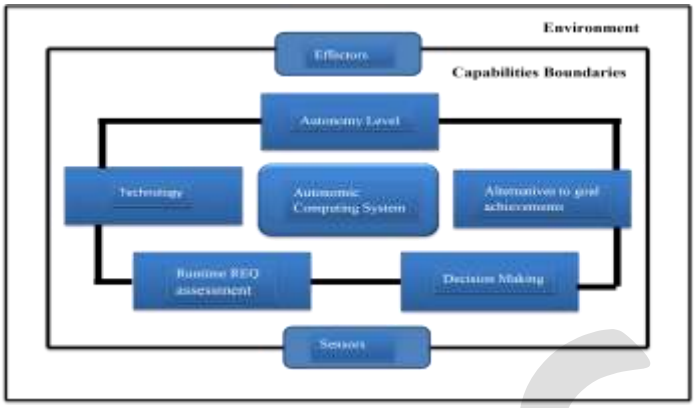

Fig: Elements Needed to Design Framework for Autonomic Computing

1. **System Environment:** It is important to examine the environment in which the system will alive. This will support in forecasting the probable changes and essential replies to be attained by the system.
2. **System Capability:** Capabilities mean what the system can do to relate with the environment in terms of physical connections and data interaction.
3. **Level of Autonomy:** autonomic computing is deliberated progressive in nature. There are five levels of autonomy: Basic, Managed, Predictive, Adaptive, and Autonomic. With the basic being the level in which most IT systems are in currently; and the Autonomic level being the uppermost level. Determining the level of autonomy shall allow the engineers make precise choices on how the system in hand will be developed.
4. **Choice of Technology:** A development technology might be verbalized by the system requestors, or the area in which the software will function. However, developing for autonomy needs a dissimilar development environment that allows runtime adaptation. The XML-Based Autonomic Computing Expression Language (ACEL) and Software Component Ensemble Language (SCEL) operating on a Java Runtime Environment are two examples of technology supporting autonomic computing.
5. **Runtime Requirement Assessment:** Autonomic computing necessitates a mechanism to monitor and evaluate the achievement or failure of completing the system goals at runtime. This mechanism could be based on the selected technology or developed separately. Earlier research has recommended a number of ways to this such as the definition of Awareness Requirements, and the idea of requirements reflection.
6. **Decision-Making:** Understanding the runtime performance of requirements supports in determining about the action to be taken by the system.
7. **Goal Achievement Alternatives:** Clarifying another way to accomplish goals is a vital part for developing autonomic requirements. The uncertainty related with the environment in which autonomic systems occur, may delay certain actions. However, that should not prevent the system from accomplishing the main goal [1].

ACKNOWLEDGMENT

I feel great pleasure in submitting this paper on "Autonomic Computing: The Brief Introduction". I wish to express true sense of gratitude towards my H.O.D., Prof. D. D. Patil. I also wish to thank my teacher Prof. L. D. Panjwani who at very discrete step in preparation of this paper contribute her valuable guidance and help to solve every trouble that arose. Also, most likely I would like to

express my sincere gratitude towards my family for always being there when I needed them the most. With all respect and gratitude, I would like to thank all the people, who helped me directly or indirectly, I owe my all success to them.

CONCLUSION

For decades, the change of technology and science has reflected the growth of difficulty in several computer environments. In the year of 2001, IBM introduced a new advanced computing 'The Autonomic Computing' system which is self-managed. Any autonomic computing system has to be consists of four key characteristics of self-managing properties that is self-protection, self-healing, self-configuration and self-optimization. The architecture of an Autonomic Computing system includes four building blocks i.e. autonomic manager, knowledge source, touchpoint, manual manager and enterprise service bus. The offered framework is based on previous research in this field; however with forthcoming research other aspects may appear and need the development of the framework.

There are several advantages of autonomic computing systems. As well as addressing complexity, autonomic computing suggests the promise of a lesser cost of ownership and a reduced maintenance problem as systems become self-managing. Simplified user knowledge through a more approachable, here and now system. One more advantage is that Autonomic Computing saves the cost scale to use.AC scales the power, packing and costs that enhance usage through both hardware and software. AC realizes the idea of enablement by shifting existing properties to higher-order business.

REFERENCES:

[1] Mona A. Yahya, Manal A. Yahya, Dr. Ajantha Dahanayake "Autonomic Computing: A Framework to Identify Autonomy Requirements Complex Adaptive Systems", Publication 3 Conference Organized by Missouri University of Science and Technology 2013- Baltimore.
[2] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," IEEE Computer Society, January 2003.
[3] Sterritt, Roy, M. Parashar, H. Tianfiels, and R. Unland, "A concise introduction to autonomic computing", Advanced Engineering Informatics, 2005.
[4] IBM White Paper. An architectural blueprint for autonomic computing. IBM; 2003.
[5] S. S. Laster and A. O. Olatunji, "Autonomic Computing: Towards a Self-Healing System," in American Society for Engineering Education, Illinois, Indiana, 2007.
[6] V. E. S. Souza, "A Requirements-Based Approach for the Design of Adaptive Systems," in ICSE, Zurich, Switzerland, 2012.
[7] D. B. Abeywickrama, N. Bicocchi and F. Zambonelli, "SOTA: Towards a General Model for Self-Adaptive Systems," in Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on, Toulouse, 2012.