Software Architectural Style for Autonomic Cloud Computing

Zakarya A. Alzamil
Software Engineering Department
King Saud University
Riyadh, Saudi Arabia
e-mail: zakarya@ksu.edu.sa

***Abstract*— Most of the autonomic cloud computing architectures are either a domain specific architecture or focus on certain properties of autonomic computing. In addition, they do not concentrate on the core issues related to the design and architectural concerns with respect to autonomic cloud computing in which the cloud can manage itself. In this paper, we propose a generic software architectural style for autonomic cloud computing systems that is based on a simplified layered approach. The proposed architectural style consists of five layers in which the bottom layer consists of cloud hardware/software resources, the second layer consists of a virtual machine that provides flexibility to service providers to utilize cloud resources, the third layer consists of an autonomic manager that manages cloud services, the fourth layer consists of a cloud service provider which provides services to cloud clients, and finally, the fifth and top layer represents the client layer that enables users to utilize the provided cloud services. This architectural style is a flexible and expandable software architecture solution for autonomic cloud computing systems, in which the service providers in the cloud can integrate their services within the architecture of the cloud computing software system. Additionally, this architecture enables the software architects to design and model their cloud computing software system in a flexible way that will maximize the reuse of existing cloud software components within their software system.**

Keywords- autonomic cloud computing; cloud computing architecture; software architecture; software architectural style; cloud computing architectural style.

I. INTRODUCTION

Cloud computing is a computing model that aims to provide services over the Internet by providing shared computing resources that are accessible by cloud service providers, as well as cloud clients. Cloud computing is defined by the National Institute of Standards and Technology (NIST) as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1]. A given cloud computing implementation can be viewed as a collection of interconnected computers that are presented as unified computing resources that provide services based on a certain service level agreement. Cloud computing provides different services. The most common cloud computing services are three service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). In addition, cloud computing may be deployed based on four deployment models: private cloud, community cloud, public cloud, or hybrid cloud [1][2]. Cloud computing relies on sharing of resources, as well as adaptation to existing technologies and paradigms without the need to know such technologies and paradigms. In addition, cloud computing adopts concepts from Service-Oriented Architecture (SOA) that can help users to breakdown the business problems into services that can be integrated to provide a solution. Cloud computing is widely used as a Web service that provides services at minimal management. The advantage of cloud computing is the flexibility of offering and delivering shared resources. Typically, the cloud service is a subscription-based service in a pay-as-you-go model. Cloud computing is a complex, large scale distributed system whose management is crucial in order to offer services in a reliable and timely manner. This requires the automation and integration of cloud service provision and management in an autonomic computing manner.

The autonomic computing model is derived from the human body autonomic nervous system [3] in which the computing system is capable of managing itself and can dynamically adjust to changes in policies without human intervention. The main property of autonomic computing is the self-management, which consists of self-configuration, self-optimization, self-healing, and self-protection [15]. Self-configuration is the system's ability to dynamically configure itself according to high-level policies, with the rest of system adjusting itself automatically and seamlessly. Self-optimization is the system's ability to automatically optimize its usage of resources and improve its performance and efficiency. Self-healing is the system's ability to automatically detect, diagnose, and repair localized software and hardware problems. Self-protection is the system's ability to automatically defend itself from malicious attacks or cascading failures, as well as from end users who accidentally make software changes, e.g., deleting an important file [3].

Software architecture deals with the design and implementation of the high-level structure of the software. It is the result of assembling a certain number of architectural elements in some well-chosen form to satisfy the major functional and non-functional requirements of a system, such as reliability, scalability, portability, and availability [4]. Software development based on common architectural idioms has its focus shifted from lines-of-code to coarser-grained architectural elements (software components and connectors) and their overall interconnection structure [5]. In order to understand the architectural style, one should

Copyright (c) IARIA, 2019. ISBN: 978-1-61208-712-2

understand the concept of software architecture. There are several definitions of software architecture. Perry and Wolf [6] define software architecture in terms of building blocks that are concerned with the selection of architectural elements, their interactions, and the constraints on those elements and their interactions necessary to provide a framework in which to satisfy the requirements and serve as a basis for the design. ISO/IEC/IEEE 42010 Standard [7] defines software architecture as "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution". Bass et al. [8] define software architecture as the structure or structures of a system, which comprises software elements, the externally visible properties of those elements, and the relationships among them. These definitions identify the software architecture at the macro level as the software system's blueprint. The architectural style is determined by a set of element types, the topological layout of the elements indicating their interrelationships, a set of semantic constraints, and a set of interaction mechanisms that determine how the elements coordinate through the allowed topology [8]. Shaw and Clements [9] define the architectural style as a set of design rules that identify the kinds of components and connectors that may be used to compose a system or subsystem, together with local or global constraints on the way the composition is done. An architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constraints on how they can be combined. These can include topological constraints on architectural descriptions (e.g., no cycles) or some constraints on execution semantics [10].

In this paper, we propose an autonomic cloud computing architectural style for software systems that is based on a simplified layered approach. We have used the decision support system's architectural elements proposed in [11], as will be described in Section III, to support the self-management of autonomic cloud computing software systems. The proposed architectural style consists of five layers: cloud hardware/software resources layer, virtual machine layer, autonomic manager layer, cloud service providers layer, and client layer. This paper is organized as follows. In Section II, we describe the related works, and in Section III, we present our proposed approach. The conclusions are presented in Section IV.

II. RELATED WORK

Several studies have proposed architectural approaches for autonomic cloud computing. In [12], a software process based development approach for designing and building an autonomic cloud computing system is described. According to this approach, a sequence of software steps is followed for the complete design, such as control parameter identification, system model, system input identification, model identification, model update, system decision type, prediction creation, coordinator creation, data measurement, managed system control, and autonomic system control. A cluster of application servers running on top of a cloud is described as an application of autonomic management architecture to show how the development approach can be reconfigured for self-management and optimization for Web services.

A mechanism to implement autonomic cloud computing with the usage of information proxies is described in [13]. An information proxy provides useful information about a resource such as its state, works that need resources, overall resource utilization, etc. The proposed approach aims at improving the collaboration among peers in a large-scale network for the purpose of distributed resource scheduling. Results from the study showed that information proxies may improve the resource scheduling of large scale distributed systems. The information proxies help in building neighborhood nodes that contain information about the co-located nodes that share similar characteristics.

Artificial intelligence techniques such as multi agent and mobile computing are proposed in [14] for designing autonomic cloud computing. In this proposed approach, autonomous cloud agents are implemented with multi agent system which is capable of monitoring and correcting resource scheduling activities. The aim of this approach is to provide a monitoring system that facilitates autonomic clouds based on mobile agent computing. An agent enabled cloud consists of a mobile agent platform distributed on different virtual machines, and a software agent installed on the front-end to act as a proxy between the interface and agents.

An architectural blueprint for autonomic computing system is presented in [15]. The presented architecture constitutes layers that are connected using enterprise service bus patterns in which the layers collaborate using Web services. The basic building blocks of the layers include managed resources which contain system components such as hardware or software, knowledge sources such as interfaces for accessing and controlling the managed resources, autonomic managers that perform various self-management tasks to embody different intelligent control loops, and manual managers that provide a common system management interface for the informational technology professional using an integrated solutions console.

In [16], the authors explore the architectural features and requirements of cloud computing. General guidelines are presented to software architects and cloud developers for creating future architectures. The architectural requirements are classified according to the stakeholder of such software system such as cloud providers, the enterprises that use the cloud, and end-users.

A software defined cloud is proposed as an approach for automating the process of optimal cloud configuration [17]. Such optimization is obtained by extending the virtualization concept to all resources in a data center with emphasis on mobile cloud applications, in which a better quality of service can be obtained by easy reconfiguration and adaptation of physical resources in a cloud infrastructure.

In [18], a conceptual architecture of autonomic computing for cloud resources' management and provision that support SaaS applications is presented. The aim of such proposed model is to maximize efficiency and minimize the cost of services. In addition, the model aims at ensuring that

Copyright (c) IARIA, 2019. ISBN: 978-1-61208-712-2

the resource provisioning system is able to allocate resources only for requests from legitimate users.

An autonomic mobile cloud management framework is proposed in [21] for efficient service/resource management of mobile ad hoc cloud computing systems. The security and privacy of the proposed framework is investigated. The proposed framework uses mobile cloud application-enabling fabric to create and manage cloud applications in which a composition of autonomic cloud elements can be managed. Autonomic cloud elements can virtualize the physical resources, compose other elements, and communicate with other cloud elements using some common interface.

In [22], an elastic architecture is presented for autonomic cloud computing based on control loops and thresholds based rules. The experiment shows that cloud computing and autonomic computing may be leveraged together for elasticity provisioning. The proposed architecture enables the resources to be allocated and deallocated as needed, to adjust to the workload.

An autonomic Service Level Agreement (SLA) monitoring framework that is managed by trusted third party is proposed in [23]. The proposed framework uses calculation formulas to calculate the score of the cloud service providers and is composed of an SLA establishment module to support SLA generation and management, and a service monitoring module to monitor quality of service. The proposed framework is integrated into a real cloud based on the Apache CloudStack platform.

In [24], autonomic computing paradigm features have been used to Supervisory Control And Data Acquisition (SCADA) system's security by focusing on the self-protecting SCADA system. The proposed framework aims at leveraging autonomic computing elements to cope with cyber security threats and challenges to SCADA industrial applications. The hierarchical autonomic managers are incorporated within the framework to extract and refine inferences for decision making support.

Most of the aforementioned software architectures and frameworks are either a domain specific architecture or focus on certain properties of autonomic computing. We have observed that most of the existing studies of autonomic cloud computing did not concentrate on the core issues related to the design and architectural concerns with respect to autonomic cloud computing in which the cloud can manage itself. As stated earlier, cloud computing relies on sharing of resources, as well as adaption with existing technologies and paradigms without the need to know such technologies and paradigms which support independency of such cloud components. Therefore, we adopt a layered approach for our proposed architectural style to support independency among cloud components that support self-management in which each layer is independent from other layers. In addition, as discussed in the next Section, we have used the decision support system's architectural elements [11] that support autonomic manager to enhance the self-management of cloud resources. In the next section, we present our proposed architectural style for autonomic cloud computing software system.

III. AUTONOMIC CLOUD COMPUTING ARCHITECTURAL STYLE

The aim of the proposed software architecture is to propose a generic architectural style that serves as a software architecture foundation for autonomic cloud computing systems that are not limited to certain domain. As stated earlier, we have used the decision making subcomponents i.e., knowledge base, data mining/Online Analytical Processing (OLAP), and a judgmental heuristics of the decision support system approach that was described in [11] to propose an autonomic manager for cloud resources' self-management.

Cloud computing facilitates the accessibility to the shared computing resources by the cloud service providers. As a result, the software architectural style for such software system should be flexible and reusable to facilitate the interaction between the service providers and the computing shared resources. Therefore, the proposed architecture is based on a simplified layered approach, which supports flexibility and reusability of its components. Within the layered style, each layer is server to the layer above it and client to the layer below it.

Autonomic computing requires self-managing environments that, automatically, act and reflect the changes to cloud elements based on the observed changes, which can be achieved through employing an autonomic manager. The autonomic manager monitors and gathers required information from a system, analyzes collected information to detect whether it is necessary to take some action, creates a plan that describes the necessary changes, and executes the plan to implement these actions [19]. Monitoring cloud elements and/or services requires software or hardware sensors to capture the properties of such element or its related physical or virtual components within the environment, and an effector to adjust to the produced changes [20].

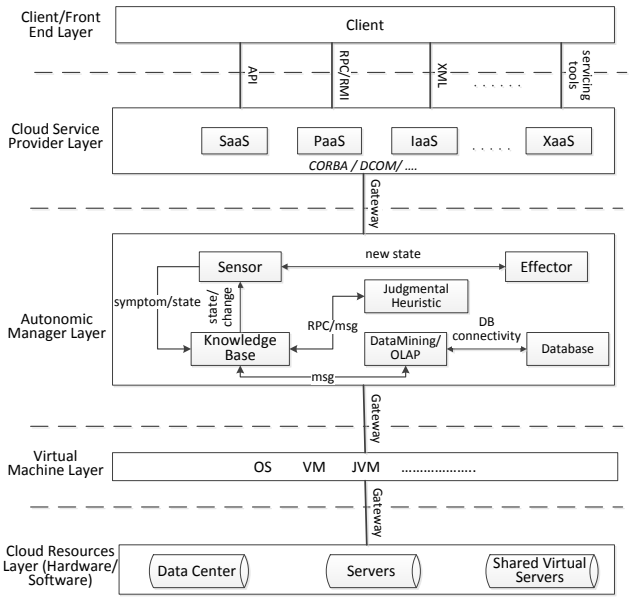

Figure 1. Autonomic cloud computing software architectural style

The proposed architectural style consists of five layers in which the bottom layer is the cloud hardware/software resources layer, the second layer is the virtual machine layer that provides flexibility to service providers to utilize cloud resources, the third layer is the autonomic manager layer which manages cloud services, the fourth layer is the cloud service provider layer that provides services to cloud clients to utilize, and the top layer is the client layer that enables the user to utilize the provided services. Figure 1 depicts the proposed software architectural style for autonomic cloud computing systems. In addition, the specification of the proposed architectural style is presented in Table I. In the following subsections, we briefly describe each layer of the proposed software architectural style for autonomic cloud computing starting from the bottom layer.

TABLE I. SPECIFICATION OF AUTONOMIC CLOUD COMPUTING ARCHITECTURAL STYLE

Item	Description
Element types	Standalone subsystems or components
Connectors	Typically procedure call Message passing
Topology layout	Hierarchical Multi-level client-server Each layer exposes an interface (API) to be used by above layers
Semantic constraints	Connectors are protocols of layer interaction Standardized layer interfaces to maintain layer independence
Interaction mechanisms	Each layer acts as a service provider to layers above and service consumer of layer below

A. Cloud resources layer

The cloud resources layer is the bottom layer that contains all hardware and software resources including the shared resources. It consists of data centers, servers, and other shared virtual resources. The cloud resources layer is the infrastructure of cloud computing system and it may include commercialized, as well as public domain and open source resources. This layer is interconnected with the virtual machine layer via a gateway, which can be defined as a proxy to maximize the independency among the different layers.

B. Virtual machine layer

The virtual machine layer contains the operating system or virtual machine that facilitates the environment to link cloud services to cloud resources. It operates as an interface between the cloud service providers and cloud resources to maximize the utilization of such resources by cloud services and, at the same time, to minimize the incompatibility among the Web services and the available cloud resources. This layer is connected to the layer above via a gateway which acts as a proxy between the two layers. It should be noticed that this layer may be skipped in the case where a service and the resource belong to the same platform and they have a well-defined connector. In such case, there is no need for a virtual machine to be in the middle.

C. Autonomic manager layer

This layer is the autonomic manager which is responsible for providing the self-management of cloud services. The autonomic manager is a configurable software and/or hardware component that consists of sensor, effector, and a decision making subcomponents i.e., knowledge base, data mining/OLAP, and judgmental heuristics. The autonomic manager monitors the managed resources and cloud services, in which the sensor collects data about cloud elements to monitor their states. When symptoms are discovered, the element state is identified and passed to the knowledge base to check whether an update of such state is available. The knowledge base looks for a fact or rule that is applicable for such element's state, in which a prediction of such state change is identified by the data mining or OLAP approach. OLAP is a business intelligence technique that helps in discovering some knowledge by extracting data from the database and viewing it from different points-of-view. The data mining explores data from the database and puts it into the knowledge base of the expert system to make knowledge-based reasoning for quantitative analysis to aid decision making. In other words, the data mining aims to discover new knowledge by extracting information from a database, analyzing it from different perspectives, and transforming it into an understandable structure of knowledge for further use. In some cases, there is a need for human intervention and/or interpretation to collect some information from human experts to identify the element's state change. In such cases, the system may use judgmental heuristics, which is a normative approach that aims to support the human in combing many factors into an optimal decision. Judgmental heuristics use a decision-analytic approach that applies the principles of decision theory and/or probability theory into the decision analysis. The normative system is based on graphical probabilistic models, i.e., probability distribution over model variables in terms of directed graph, also known as influence diagram. The database at this layer can be a traditional database, relational database, or multidimensional database. The database structure, e.g., the blackboard, as well as the components operating on it, are managed by a database management system (DBMS). In addition, such sub-system is controlled by the blackboard state. The autonomic manager identifies the element's new state, such as new configuration, new usage for an element, better optimization or utilization, fixing problem, or fixing security vulnerability. The new state and a request of change are passed from the sensor to the effector to execute such state change.

D. Cloud service provider layer

This layer consists of cloud services, such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). In addition, it may contain any other cloud services which we describe as "X as a

Copyright (c) IARIA, 2019. ISBN: 978-1-61208-712-2

Service (XaaS)". This layer provides the environment of such cloud services offered by the cloud service providers. This layer is connected to the layer above via different types of connectors such as Remote Procedure Calls (RPC), Remote Method Invocation (RMI), Application Programming Interface (API), or Extensible Markup Language (XML).

E. Client/Front end layer

This layer represents the cloud client or the front end user, which is the consumer of cloud services. This layer enables the client to request any available service using servicing tools that may utilize different technologies. Each service within this layer is defined using a specific connector, in which the client may utilize the Web services via the identified connector such as RPC, RMI, XML, API, or any other servicing tool connector.

IV. CONCLUSION

In this paper, we have introduced a software architectural style for autonomic cloud computing systems. The proposed architecture style is based on a simplified layered approach, and consists of five layers: a cloud hardware/software resources layer, a virtual machine layer, an autonomic manager layer, a cloud service provider layer, and a client layer. Within the layered style, each layer is a server to the layer above it, and a client to the layer below it.

The proposed software architectural style can accommodate most cloud computing software systems for different domains. In addition, this architectural style minimizes the dependency among its components which can enhance the reusability, integration with other software systems, and expandability. Such feature will enable software architects to design and model their cloud computing software system in a flexible way that will maximize the reuse of existing cloud software components within their software system.

The proposed architectural style is an abstract framework prototype for autonomic cloud computing software systems, and in order to understand its advantages and/or limitations, an experimental and investigation study is needed to judge the applicability of such framework on real autonomic cloud computing systems. We plan to conduct an experimental study using some commercial cloud software systems and perform a comparison study with the existing relevant architectural styles to better understand the advantages of such proposed software architecture.

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing", Special Publication 800-145, National Institute of Standards and Technology, U.S. Department of Commerce, 2011.

[2] C.S. Yoo, "Cloud computing: architectural and policy implications", Review of Industrial Organization, Vol. 38, No. 4, June 2011, pp. 405-421.

[3] J. O. Kephart and D. M. Chess, "The vision of autonomic computing", IEEE Computer, 36(1) , Jan. 2003, pp. 41-50.

[4] P. Kruchten, "Architectural blueprints - the "4+1" view model of software architecture", IEEE Software 12 (6), November 1995, pp. 42-50.

[5] N. Medvidovic and R. Taylor, "A classification and comparison framework for software architecture description languages", IEEE Transactions on Software Engineering, Vol. 26, No. 1, January 2000, pp. 70-93.

[6] D. Perry and A. Wolf, "Foundations for the study of software architecture", ACM SIGSOFT Software Engineering Notes, Vol. 17, No. 4, October 1992, pp. 40-52.

[7] ISO/IEC/IEEE 42010:2011(E), "Systems and software engineering-Architecture description", IEEE/ISO/IEC, First edition, December 2011.

[8] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, SEI series in Software Engineering, 2nd Edition, Addison-Wesley, 2003.

[9] M. Shaw and P. Clements, "A field guide to boxology: preliminary classification of architectural styles for software systems", IEEE Proceedings of the 21st Annual International Computer Software and Applications Conference, COMPSAC '97, 1997, pp. 6-13.

[10] D. Garlan and M. Shaw, "An introduction to software architecture", CMU Software Engineering Institute Technical Report, CMU-CS-94-166, January 1994.

[11] Z. Alzamil, "Software architectural style for decision support systems", Proceedings of the 11th International FLINS Conference on Decision Making and Soft Computing (FLINS2014), World Scientific Proceedings Series on Computer Engineering and Information Sciences, Vol. 9, August 2014, pp. 3-10.

[12] B. Solomon, D. Ionescu, M. Litoiu, and G. Iszlai, "Designing autonomic management systems for cloud computing", IEEE International Joint Conference on Computational Cybernetics and Technical Informatics (ICCC-CONTI) , 2010, pp. 631–636.

[13] D. C. Erdil, "Dependable autonomic cloud computing with information proxies",. IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW) , 2011, pp. 1518– 1524.

[14] A. Cuomo, M. Rak, S. Venticinque, and U. Villano, "Enhancing an autonomic cloud architecture with mobile agents", Euro-Par 2011, Parallel Processing Workshops, 2012, pp. 94–103.

[15] IBM, "An architectural blueprint for autonomic computing", white paper, 3rd Edition, june 2005, http://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf, [retrieved: September, 2018].

[16] B. Rimal, A. Jukan, D. Katsaros, and Y. Goeleven, "Architectural requirements for cloud computing systems: an enterprise cloud approach", Journal of Grid Computing, Vol. 9, 2011, pp. 3-26.

[17] R. Buyya, R.N. Calheiros, J. Son, A. Dastjerdi, and Y. Yoon, "Software-defined cloud computing: architectural elements and open challenges", 3rd International Conference on Advances in Computing, Communications and Informatics (ICACCI 2014), September 24-27, 2014.

[18] R. Buyya, R.N. Calheiros, and Li Xiaorong, "Autonomic cloud computing: open challenges and architectural elements," Third International Conference on Emerging Applications of Information Technology (EAIT), Nov. 30-Dec. 1 2012, pp. 3-10.

[19] M. Maurer, I. Breskovic, V. C. Emeakaroha, and I. Brandic, "Revealing the mape loop for the autonomic management of cloud infrastructures", IEEE Symposium on Computers and Communications (ISCC) , 2011, pp. 147–152.

[20] M. Huebscher and J. McCann, "A survey of autonomic computing - degrees, models and applications", ACM Computing Surveys, Vol. 40, No. 3, Article No. 7, August 2008, pp. 7-28.

[21] D. M. Shila, W. Shen, Y. Cheng, X. Tian, and X. Shen "AMCloud: Toward a secure autonomic mobile ad hoc cloud computing system", IEEE Wireless Communications, April 2017, pp. 74-81.

[22] E. F. Coutinho, P. A. Rego, D. G. Gomes, and J. Neuman de Souza "An architecture for providing elasticity based on autonomic computing

Copyright (c) IARIA, 2019. ISBN: 978-1-61208-712-2

concepts", Proceedings of the 31st Annual ACM Symposium on Applied Computing, 2016, pp. 412-419.

[23] A. Maarouf, Y. Mifrah, A. Marzouk, and A. Haqiq "An autonomic SLA monitoring framework managed by trusted third party in the cloud computing", International Journal of Cloud Applications and Computing, Volume 8, Issue 2, April-June 2018, pp. 66-95.

[24] S. Nazir, S. Patel, and D. Patel "Autonomic computing architecture for SCADA cyber security", International Journal of Cognitive Informatics and Natural Intelligence, Volume 11, Issue 4, October-December 2017, pp. 66-79.

Copyright (c) IARIA, 2019. ISBN: 978-1-61208-712-2