International Journal of Computer Trends and Technology
ISSN: 2231-2803 / https://doi.org/10.14445/22312803/IJCTT-V68I3P101

Volume 68 Issue 3, 1-6, March 2020
© 2020 Seventh Sense Research Group®

Original Article

Autonomic Computing Architecture by Self-defined URI

Nadir K. Salih[1], Abdel-hafiz A. Khoudour[2], Mawahib S. Adam[3], Samar M. Hassen[4]

[1]*College of Engineering, University of Buraimi, Oman*
[2]*Faculty of Postgraduate, University of Science and technology, Sudan*
[3,4] *College of Computer Science and Information Technology, Jazan University, Saudi Arabia*

Received Date: 25 January 2020
Revised Date: 07 March 2020
Accepted Date: 11 March 2020

Abstract – *The problem arises when publishing information on the Web by semantic web technologies, represented in clash names, to avoid that and to realize the self-management, we applied the self-defined URIs, which identify a uniform way to retrieve information about the resource being identified by the Uniform Resource Identifier (URIs), which help to obtain a good result used to adding self-management capabilities to a system. To achieve the goal, five patient records with coronary artery disease collected from the Statistic unit in Ahmed Gasim Hospital/ Khartoum contains 10 attributes based on the causes of the disease. We have applied for that open-source application (Protégé) to solve the problem statement and to obtain a good result that reduced errors costs and improved productivity with the possibility of editing reports and querying data from anywhere.*

Keywords *– Autonomic Architecture, Self-defined, Uniform resources identifier (URIS), Ontology Web Language (OWL).*

I. INTRODUCTION

On March 8, 2001, Paul Horn presented the importance of these systems by introducing ACSs (Autonomic Computing Systems) to the National Academy of Engineering at Harvard University [1][13]. It is possible to find some aspects of autonomic computing already in today's software products. For instance, Windows XP optimizes its user interface (UI) by creating a list of the most often-used programs in the start menu. Thus, it is self-configuring in that it adapts the UI to the behaviour of the user. It can also download and install new critical updates without user intervention, sometimes without restarting the system [14]. Therefore, it also exhibits basic self-healing properties. Dynamic host communication protocol (DHCP) and domain name service (DNS) services allow devices to self-configure to access a TCP/IP network [2].

Autonomic computing is a self-adaptive system [3][15]. Compose from autonomic elements, and are capable of managing their behaviours and their relationships with other systems/applications in accordance with high-level policies, and exhibit eight defining characteristics: The four primary aspects are (self-configuring, Self-optimizing, self-protecting, self-healing). Self-configuring: Systems adapt automatically to dynamically changing environments, such as plug and play devices, configuration setup wizards, and wireless server management. Self-healing: Systems discover, diagnose, and react to disruptions. They must be able to recover from a failed component by first detecting and isolating the failed component, taking it offline, fixing or isolating the failed component, and reintroducing the fixed or replacement component into service without any apparent application disruption. Self-optimizing: Systems monitor and tune resources automatically, requiring hardware and software systems to maximize efficient resource utilization to meet end-user needs without human intervention.
Self-protecting: Systems anticipate, detect, identify, and protect themselves from attacks from anywhere against unauthorized resource access, to detect intrusions and report and prevent these activities as they occur, and to provide backup and recovery capabilities that are as secure as the original resource management systems [4][16].

Self-defined URISA standard mechanism for identifying resources on the Web by adding URI (Uniform Resource Identifier) to resources it fits well into the Semantic Web for the following two main reasons:

- It provides a mechanism to uniquely identify a given resource.
- It specifies a uniform way to retrieve machine-readable descriptions about the resource being identified by the URI.

Another benefit of using URIs to represent subject and object resources is related to their global uniqueness. In this research, http://csrtahmedgasim.com/ only will create any new URI that guarantees the global uniqueness of URIs and certainly prevents name clashes. There are two different types of URI we can use to identify a given resource, namely hash URI and slash URI. A slash URI is simply a normal URI that we are all familiar with, for example:http://csrtahmedgasim.com//Patient/Man. A hash URI that we use and consists of the following components: normal URI + # + fragment-identifier-or http:// csrtahmedgasim. com/# Patient / Man [5].

Autonomic elements (AEs) are the basic building blocks of autonomic systems, and their interactions produce self-managing behaviour [1][17]. We can consider AEs as software agents and ACSs as multi-agent systems. Each AE has two parts: Managed Element(ME) and Autonomic Manager (AM).ACSs are established from Managed Elements (MEs), whose behaviours are controlled by Autonomic Managers (AMs). AMs execute according to the administrative policies and implement self-managing. The Managed Element is a component of the system. It can be hardware, application software, or an entire system. The Sensors retrieve information about the current state of the ME and then compare it with expectations that are held in the knowledge base by the AE, and the Effectors execute the required action. Sensors and effectors are linked together and create a control loop [18].Autonomic Managers (AMs): Is the second part of an AE uses a manageability interface to monitor and control the ME. It has four parts: monitor, analyze, plan, and execute. The monitor part provides mechanisms to collect information from a ME monitor and manage it. Monitored data is analyzed. It helps the AM to predict future states. The plan uses policy information (adds, modifies, and deletes) and analyzes data to achieve goals. Finally, the execute part controls the execution of a plan and dispatches recommended actions into the ME. AMs can change resource allocation to optimize performance according to the policies [19].Policies: a set of administrator ideas and are stored as knowledge to guide AM. These four parts provide control loop functionality. The external behaviour of AEs relates to relationships among them. AMs can be linked together via an autonomic signal channel Knowledge: the Standard data shared among the monitor, analyze, plan, and execute functions of an autonomic manager, such as topology information, historical logs, metrics, symptoms, and policies [20]. It is an implementation of a registry, dictionary, database, or another repository that provides access to knowledge according to the interfaces prescribed by the architecture. The knowledge used by an autonomic manager is obtained in one of three ways: The knowledge is passed to the autonomic manager. A policy consists of a set of behavioural constraints or preferences that influence the decisions made by an autonomic manager [21]. The knowledge is retrieved from an external knowledge source. An autonomic manager might obtain symptom definitions or resource-specific historical knowledge in this manner. A knowledge source could store symptoms that could be used by an autonomic manager; a log file may contain a detailed history in the form of entries that signify events that have occurred in a component or system [22].

The autonomic manager itself creates the knowledge [6].

Resource Description Framework (RDF) Recommend by World Wide Web Consortium (W3C) in 2004 for the semantic Web, is a framework for representing information in the Web[7], contains 3 triples, a subject can be a (URIs or a blank node), object (URIs, a literal or a blank node) and predicate(URIs) [8][23].

Ontology Web Language (OWL) is to combine different class entities and/or property entities to create new class entities and property entities. The thing is the class of all individuals and is a superclass of all OWL classes.

Nothing: is the class that has no instances and a subclass of all OWL classes. Class: A class defines a group of individuals that belong together because they share some properties and can be organized in a specialization hierarchy using a subclass of. Sub Class Of Class hierarchies may be created by making one or more statements that a class is a subclass of another class.

A. Property

Properties used to state relationships between individuals (Object property) or from individuals to data values (Data property). A Domain of property limits the individuals to which the property can be applied. If a property relates an individual to another individual, and the property has a class as one of its domains, then the individual must belong to the class. The range of property limits the individuals that the property may have as its value. If a property relates an individual to another individual, and the property has a class as its range, then the other individual must belong to the range class. Individual: are instances of classes and properties that may be used to relate one individual to another [8].An axiom: Is a basic statement that OWL ontology has. It represents a basic piece of knowledge. For example, a statement like "Man" is a sub-class of the Patient class" is an axiom. The axiomatic semantic can use to automate reasoning with RDF and RDF Schema.

B. Entities

These are the atomic constituents of axioms. Taxonomy: is the science of classification. Thesaurus: An extension to taxonomy [5].SPARQL to pull data from a growing collection of public and private data, but to query data conforming to the RDF data model [9], and consists of three parts: The pattern matching part, which includes several interesting features of pattern matching of graphs, like optional parts, a union of patterns, nesting, filtering (or restricting) values of possible matching, and the possibility of choosing the data source to be matched by a pattern. The solution modifiers, which once the output of the pattern is ready (in the form of a table of values of variables), allows to modify these values applying classical operators like projection, distinct, order, limit, and offset. Finally, the output of a SPARQL query can be of different types:

Yes/no queries, selections of values of the variables, which match the patterns, construction of new triples from these values, and descriptions about resources queries[10].In mid-

October 2001, IBM released a manifesto observing that the main obstacle to further progress in the information technology industry is a looming software complexity crisis. The term autonomic computing is emblematic of a vast and somewhat tangled hierarchy of natural self-governing systems, many of which consist of myriad interacting, self-governing components that in turn comprise large numbers of interacting, autonomous, self-governing components at the next level down[24][25]. It will be profitable to seek inspiration in the self-governance of social and economic systems as well as purely biological ones. An autonomic element will typically consist of one or more managed elements, coupled with a single autonomic manager that controls and represents them. The autonomic manager will relieve the human responsibility of directly managing the managed element [26][29].IBM frequently cites four aspects of self-management. Early autonomic systems may treat these aspects as distinct, with different product teams creating solutions that address each one separately [30]. Ultimately, these aspects will be emergent properties of general architecture, and distinctions will blur into a more general notion of self-maintenance [27][31]. System administrators and end-users will take the benefits of autonomic computing for granted. Self-managing systems and devices will seem completely natural and unremarkable, as will automated software and middleware upgrades. Autonomic computing is a Grand Challenge that reaches far beyond a single organization; its realization will take a concerted, long-term, worldwide effort by researchers in a diversity of fields [2][28]. The goal of an autonomic computing architecture is to reduce intervention and carry out administrative functions according to predefined policies. A study shows that 25 to 50 percent of IT resources are spent on problem determination, and almost half of the total budget is spent on preventing and recovering systems from crashes. All these issues have motivated researchers to investigate a new idea to deal with the management of complexity in the IT industry, and self-management systems have been introduced [1].

In this paper, we describe the implementation of self-managing systems through the framework build of semantic web technology elements and can be organized as follows: first, in section1, we have given an introduction and a relevant background about autonomic computing. Section 2 includes the major work-related directly to the context of the study. In section3, we have described the architectural module of autonomic computing and how to realize self-management. Section4 describes how to realize the self-defined URIS. Then, all these sections are followed by Result, evaluation, conclusion and further work.

II. RELATED WORK

A solution of spending long hours for short distances in traffic represented by Aithal K and Dr Shantharam Nayak in [11], implemented by automating the traffic signals using the concept of "Autonomic Computing"-self-management, which helps in effectively managing the traffic density in the cities thus allowing smooth traffic flow. Through using three types of Autonomic Computing characteristics (self-configuring, self-optimization and self-healing), which are used as the traffic signal characteristics. Therefore, to achieve all these using a sensor and a sensor node are connected with automatic Wireless Sensor Networks. Its advantages such as help managing the traffic density effectively and allowing smooth traffic flow, but its disadvantages such as only the vehicles covered by the sensors, leaving other space free, and the increasing fault, confusing the state, protection issue and vice versa. Described an architecture of stable autonomic systems, consist of main characteristics of autonomic systems (Self-healing, Self-optimizing, Self-protecting and Self-configuring) which they are to be achieved. A generic architecture of the autonomic approach recommended contains two entities. Autonomic Element, which is considered the heart and comprises managed resources and distributes services to humans or other autonomic elements, also contains sensors, effectors, and five-component (Monitor, Analyzer, Plan, Execute, and shared knowledge) And Autonomic Manager. Its basic objectives are to preserve correct software architecture and consists of an adaptable and malleable infrastructure for all basic components of any self-managed system. Also contains autonomic control loops, which consist of collecting, analyzing, deciding, and acting, that to domineer the flow of work done between sub constituents of autonomic elements and its task. Finally, some open problems pervading for designing autonomic system architecture, such as lack of open standards and rules, but there are still many challenges that need to be solved because systems become autonomous by nature, and most of the existing systems provide solutions for specific problems, but not unified refinement for full adaptation to a random environment.

Kramer and Magee in [13] focus on the use of an architectural approach to self-management to provide systems that are scalable, support dynamic composition, rigorous analysis, flexible and robust in the presence of change, and the objective is to minimize the degree of explicit management, a three-layer reference model proposed as a following: The goal management layer, which needs to encompass both application and system goals concerned with self-management, its challenge, is to achieve goal specification both, comprehensible by human users and machine-readable and the solution by designing a set of plans offline, either by construction, or verification process to satisfy system constraints for a range of possible system states. The change layer, responsible for executing changes in response either to changes in the state reported from the lower layer to goal changes, its challenges deal with distribution and decentralization in addition to finding change management algorithms that can tolerate inconsistency and which eventually terminate in a system that satisfies constraints. The component control layer, consisting of a setoff interconnected components, may be collocated and/or distributed over a network of communicating computer nodes. Its main challenge is to provide change management, which reconfigures the software components. To ensure application consistency and avoid undesirable transient behaviour, the solutions have focused on dealing as far as is possible with planning by designing a set of plans offline that can be shown either by construction or by a verification process to satisfy system constraints for a range of possible system states. Describe an architectural approach by creating two prototype autonomic systems that explore the use of autonomic systems for data centre management and resource allocation. For creating autonomic elements (self-managing components), using required behaviour, which must be self-management, capable of establishing, maintaining relationships with them and manage its behaviour and relationships to meet its obligations, focusing on policy-based self-management, relationships and integrity. To achieve the goals, they need to accomplish two fundamental goals. First, it must describe the external interfaces and behaviours required to make an individual component autonomic. Second, it must describe how to compose systems out of these autonomic components in such a way that the system as a whole is self-managing. An autonomic system requires a collection of autonomic elements that implement the desired function, system-level behaviours, and design patterns of self-management.

III. SELF-MANAGEMENT ARCHITECTURE MODEL

The main architecture of self-management represents two principle parts: Managed element involve (RDF, OWL, SPARQL), and AM involved of (Monitor, Analyze, Plan, and Execute).

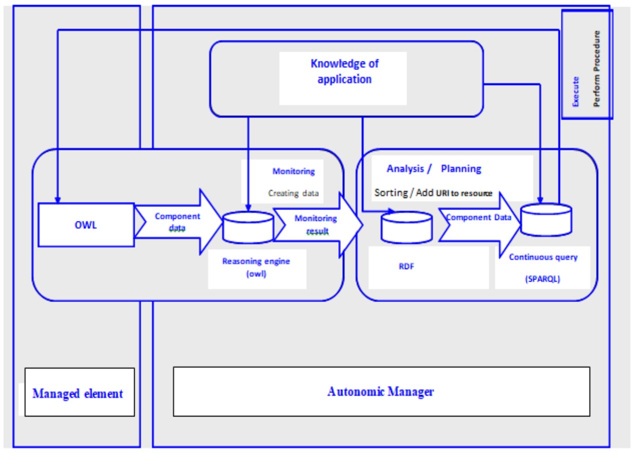

Fig. 1 Autonomic computing architecture

IV. SELF-DEFINED URIS

The illustrated architecture explains our new approach to realizing the objective, using steps and lifecycle of autonomic computing systems.

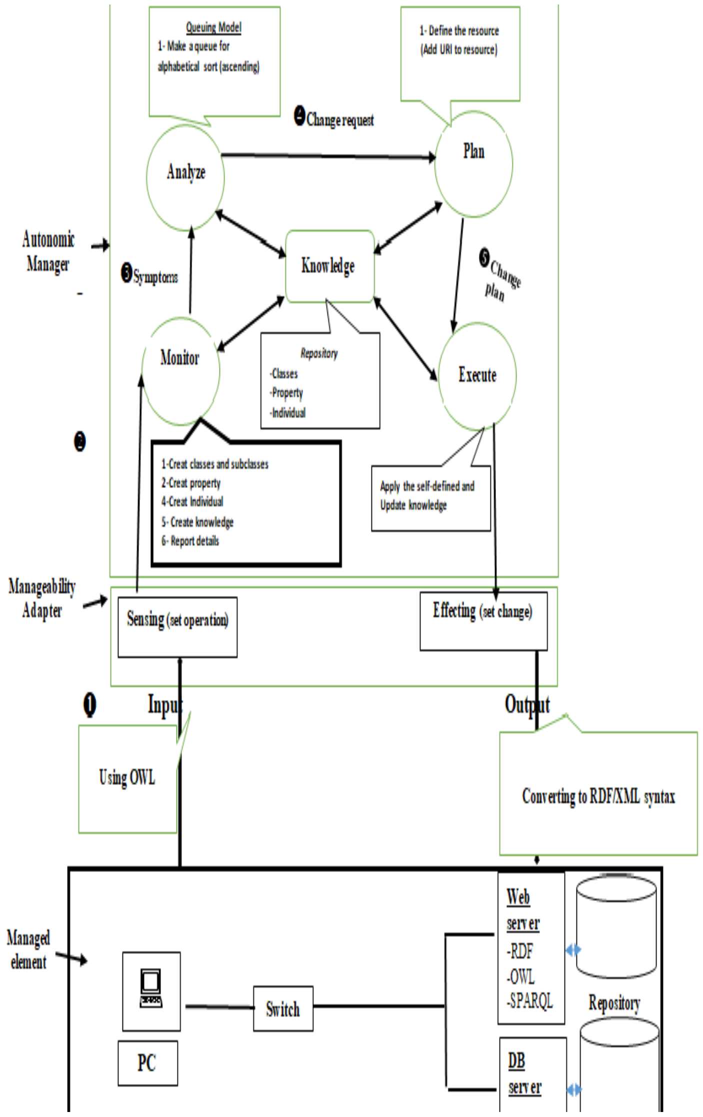

Fig. 2 Self-defined implementation architecture

Firstly, data is collected from managed element (OWL) via the sensor interface, which allows AM to monitor the managed element (Creating classes, property, Individual…etc.), then, the Monitor part use a mechanism to collect data aggregate them, and send a detailed report. After collecting all relevant contexts, Analyze process is performed by making a queueing model for sorting data (Ascending order). And Plan adapts and applies the system management behaviour by adding the URI to resources. In addition, execute control the execution of a plan with considerations for dynamic update through the effectors, which carry out changes to the managed element (output).

V. SELF-DEFINED LAYER(DEPLOYMENT)

The AC architecture is essentially a two-layered architecture consisting of a hierarchy of:

- Managed elements
- Autonomic Manager

This hierarchy, where the higher level is the Autonomic Manager and the lower level is the Managed element or resources.

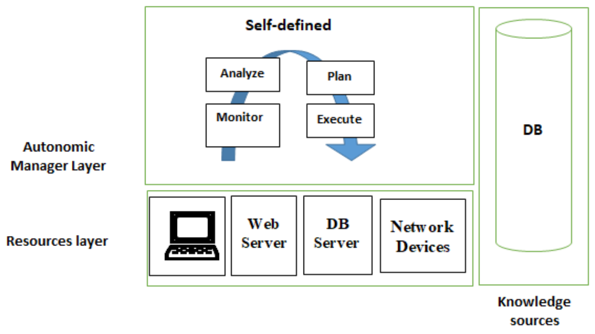

Fig. 3 layer deployment

VI. IMPLEMENTATION

Figure 5 shows the applicability of self-defined by providing Protégé application.

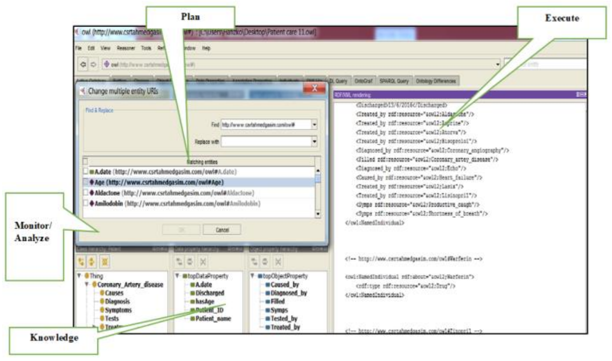

Fig. 4 self-defined interface

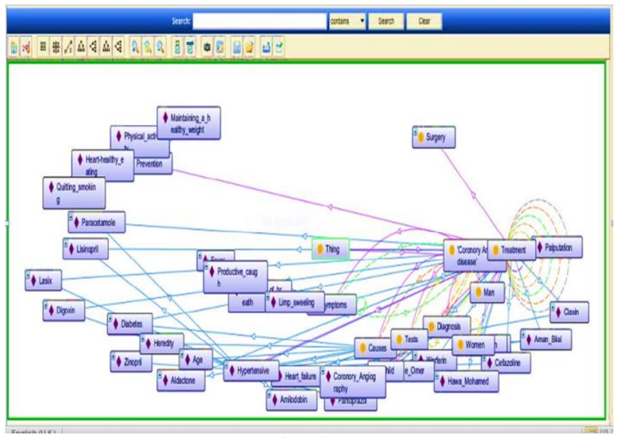

Fig. 5 Self-defined elements graph

This graph contains classes, subclasses and individuals, which is the output of the implementation (execute), and in terms of the autonomic system, describe the self-defined elements and their robustness.

VII. RESULT AND DISCUSSION

In this part, we describe the use of General techniques for autonomic systems to satisfy the objective. And what is realized by self-defined we have taken, in executing, the important evaluation results of the working demonstrate as the following:

The degree of autonomic is the Managed level. The system presents flexibility and elements robustness while changing. In addition, reusing the knowledge-based is possible.

VIII. CONCLUSION AND FUTURE WORK

This paper presented autonomic computing architecture by self-defined, a faithful implementation applied to use the semantic web technologies features to make the health care system self-defined, which could result from reducing the human involvement, and we summarize our contribution as a follow:

- Improve the hospital's ability to use patient data for generating new knowledge and future patient care through outcomes (Reports).
- Build a repository that collects and stores various data to help doctors, users, and patients to provide ad hoc queries to published data on the Web from anywhere.
- Help the clinical research centre to collect data.

This work can be followed further in a number of directions:

- Using semantic web rule language to provide the ability to write rules expressed in terms of OWL concepts.
- Design a self-defined application based on the architecture.
- Development of a system with object-oriented methodology.

REFERENCES

[1] Mohammad Reza Nami and Mohsen Sharifi. A Survey of Autonomic Computing Systems, (2006) 1-10
[2] Jeffrey O.Kephart and David M. Chess. The Vision of Autonomic Computing. Computer, (2003) 1-10
[3] Julie A. McCann, MarkusHuebscher. Evaluation issues in Autonomic Computing, (2004) 1-12
[4] Manish Parashar and Salim Hariri. Autonomic Computing: An Overview, (2005) 1-13.
[5] Liyangyu. A Developer's Guide to the Semantic Web, (2011) 1-621.
[6] IBM Autonomic computing white paper.An architectural blueprint for autonomic computing, (2005) 1-34.
[7] Eric Miller.An introduction to the Resource Description Framework, (1998) 1-5.
[8] OWL Web Ontology Language Overview, http://www.w3.org/TR/2004/REC-owl-features-20040210/<[06/04/2005].
[9] Bob Ducharme. Learning SPARQL-second edition, (2013) 1-386.
[10] Ian Horrocks. Ontologies and the Semantic Web, (2008) 1-10.
[11] UllasAithal K and Dr ShantharamNayak.Smart Traffic Signals using Autonomic Computing, 1 (2014) 1-3.
[12] Payal Mittal, AbhishekSinghal and AbhayBansal. A Study on Architecture of Autonomic Computing-Self Managed Systems, 92 (6) (2014) 1-4, 2014.
[13] Jeff Kramer and Jeff Magee.Self-Managed Systems: an Architectural Challenge, (2014) 1-1, 2014.

[14] Steve R. White, James E. Hanson, Ian Whalley, David M. Chess, and Jeffrey Kephart. An Architectural Approach to Autonomic Computing, (2014) 1-8.

[15] Nadir K Salih, TianyiZang. Variable service process for SaaS Application. Research Journal of Applied Sciences, Engineering and Technology. 4(22) (2012) 4787-4790

[16] Nadir K Salih, TianyiZang, Mingrui Sun. Multi-database in the healthcare network. International Journal of Computer Science Issues, 8(6) (2011) 210-214.

[17] Nadir K Salih, TianyiZang, G.K. Viju, A Mohamed. Autonomic management for the multi-agent system. IJCSI, 8(5) (2011)338-341.

[18] Nadir K Salih, TianyiZang. Need of Autonomic Management SaaS Application. International Journal of Computer Science Issues, (2016).

[19] Nadir K Salih, TianyiZang. Survey and comparison for Open and closed sources in Cloud Computing. International Journal of Computer Science Issues, 9(3) (2012) 118-123.

[20] Eman.M-Fageer, Nadir K.Salih. Self-configuring Booking SaaS Application. Red Sea University Journal of Basic and Applied Science. 2(3) (2017).

[21] Amin, Fatima M H, Nadir K.Salih. New Model to Achieve Software Quality Assurance in E-Learning Application. (IJCSI); Mahebourg , 14(3) (2017) 65-69.

[22] Eshtiag A AbdElrhman, Nadir K Salih. Modelling Variation in SaaS Application. (IJCSI).15(3) (2018) 22-30.

[23] SalihNK, H.Elbashier , ZangT,Eshtiag A AbdElrhman. Self-Diagnosis of Diabetes Using CBR Algorithm. Journal of Computer Science & Systems Biology. 11(3) (2018) 235-239.

[24] Amin, Fatima M H, Nadir K.Salih. Implementing the System, Instructor and Student Model to Achieve Required Software Quality Assurance. Research Journal of Applied Sciences, Engineering and Technology; (2019) 30-42.

[25] Nadir K Salih, TianyiZang. Variable service process by feature meta-model for SaaS Application. IEEE International Conference in Green and Ubiquitous Technology, IEEE, (2012) 102 – 105.

[26] Nadir K Salih, TianyiZang. Autonomic and cloud computing: Management Services for Healthcare. IEEE International Symposium on Industrial Electronics and Applications (2012).

[27] Nadir K Salih, TianyiZang. Modelling and Self-Configuring SaaS Application. International conference on software engineering research and practice (SERP14), held on July 21-24 Las Vegas, USA., (2014)

[28] Nadir K Salih, TianyiZang. Autonomic Management for Applicability and Performance in SaaS Model. International conference on parallel and distributed processing techniques and applications (PDPTA'14), held on July Las Vegas, USA., (2014).

[29] Nadir K Salih, TianyiZang. Self-management SaaS Application by CBR Algorithm. International conference on parallel and distributed processing techniques and applications (PDPTA'17), held on July 21-24 Las Vegas, USA, (2017).

[30] Nadir K Salih, TianyiZang. Implementation of Autonomic Management SaaSSystem .conference on software engineering research and practice, held on July 21-24 Las Vegas, USA, (2017).

[31] GK Viju, Nadir K Salih, TianyiZang. A novel approach to iris recognition for personal authentication. International Conference of Computer Applications and Industrial Electronics (ICCAIE), IEEE, (2011) 350-354.

[32] G.K.Viju, Nadir K.Salih. A secure multicast protocol for ownership rights. International Conference of Computing and Information Technology (ICCIT), (2012) 788-793.

[33] Sheima S. El-hwaij, Nadir K.Salih. Autonomic management by self-optimization for WEINMANN. IEEE, International Conference on communication, Control, Computing and Electronics Engineering (ICCCCEE), (2017).

[34] S.Shanmugapriya, Dr. K. Alagarsamy, A.Saranya. Android Platform for the Mobile Application Security System, SSRG International Journal of Mobile Computing & Application, 5(1) (2018).

[35] Marjan Farsi, Analyzing Tagging Behavior in Clustering Similar Web Resources through Interactive Visual Demonstration, SSRG International Journal of Computer Science and Engineering, 1(10) (2014).