International Journal of Engineering & Technology, 7 (1.1) (2018) 421-425

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Incorporating autonomicity and trustworthiness aspects for assessing software quality

Pooja Dehraj [1] *, Arun Sharma [1], P. S. Grover [1]

[1] *Indira Gandhi Delhi Technical University for Women, Delhi, India*
**Corresponding author E-mail: poojadehraj2000@gmail.com*

Abstract

Autonomic computing covers few self-abilities like self-configuration, self-healing, self-optimization, self-protection, self-adaptability, self-awareness, self-openness etc. in software systems. These self-abilities will lead towards lowering the overall maintenance cost of the software because of minimum level of human intervention. The term Autonomicity refers to the level of autonomic (self) features implemented in the system. The International software quality standard ISO 9126 is now replaced by new software product quality standard ISO/IEC 25010:2011 which defines the framework/model to specify and evaluate the quality of software as a product. However, this does not take into account the self-* features (autonomic aspects) and trust factor of modern day software systems. The present paper proposes here that autonomic characteristics of any system must be considered while assessing the quality of any software product. This autonomic-oriented quality model may be used to assess the software quality in a number of domains. Therefore, a new enhanced software quality model is proposed which considers autonomicity and trustworthiness as a factor of quality.

Keywords: *Autonomicity; Software Quality Model; ISO 9126; Trustworthiness.*

1. Introduction

Modern day computer and software systems are distributed and highly complex. Managing and using them are really challenging and extremely difficult task. IBM started exploring and developing systems, which are self-managing and have the capability of self-correcting, self- configuring, self-healing, and self-monitoring [self-* features]. All these capabilities are incorporated into self-management and these systems can be referred as self-managed systems. Self-configuration feature enables to detect run time environmental changes and provides intelligent configuration based solutions for further execution on the managed element without high level human intervention when the managed element is no longer activated within the permitted limits. Self-correcting capability allows the system to handle the occurrence of defects and errors. Self-healing systems are able to handle internal unwanted anomaly by resetting configuration level policies to correct anomaly. For this process, the autonomic agent will search for the solutions that can be implemented on the managed element to resolve particular anomaly using internet service, knowledge database or high level policies.

On the other hand, self-monitoring feature performs continuous monitoring of the managed element to identify changes in the systems first. When autonomic agent detects any unexpected changes in the system then self-capabilities of the system will work on it to resolve those changes by performing their respective task. The goal of this Intelligent System Development (ISD) is to initiate development of self-management enabled computer systems, to handle the rapidly growing system's complexity, maintain it in an expected limit bound and to manage their functions. Such systems have been characterized as Autonomic Systems. Present paper proposes to include Trustworthiness and Autonomicity as one of the quality factor for autonomic computing based software products which may be assessed as the capability of implementing self-features to the quality model ISO 9126 along with other existing characteristics and sub-characteristics.

2. Autonomic computing

Autonomic Computing (AC), proposed by IBM in 2001[1] [2], refers to the self-managing attributes of distributed computing systems, by adapting unpredictable changes and hiding intrinsic complexity from users. This will help in maintaining some level of system's complexity without the need of manual management. IBM proposed the base for this idea and defined eight high level policies and reference architecture. These high level policies are handled by the administrator as the management responsibilities are shifted to system. These policies are summarized in few policies mentioned below:

- system must understand its system activities and must respond accordingly and do optimized resource allocation
- system must be compatible and reconfigure according to environmental changes and different system standards
- system must be able to self-protect

Modern day large software systems, such as DB2, Windows 7/10 and many other big software systems have autonomic attributes, though to a limited extent. However, lot of efforts and resources are being spent to include more autonomic features in the new software products and systems, such as Cloud-based Systems [3]. Clouds are large-scale, complex and heterogeneous distributed systems. Management of cloud resources is a difficult task. There is a need to automate and integrate intelligent strategy for resource provisioning to offer secure, reliable, and cost-efficient services [4]. Thus, it would be worth to include autonomic feature, while assessing and determining the quality of the software product.

This feature, when implemented in applications, will prove to be a better index of quality of the software product. Due to increasing requirement of developing self-managed applications, incorporation of the attribute Autonomicity into the quality model will bring more objectivity in terms of performance, maintenance, reusability and similar attributes for assessing software quality. To ensure quality along with autonomicity, trustworthiness factor has been incorporated.

3. Related work

The autonomic concept is purely inspired by human structure and its intelligent function. After IBM, other researchers have also proposed autonomic computation architecture similar to IBM's reference architecture. Khalid, et al. [5] in 2009 performed a survey on available autonomic frameworks and classified them into various categories like biologically inspired architecture, agent oriented, component oriented, technique based, model oriented, service oriented architectures and others.

Nami et al. [6] described architecture of autonomic element and extended their work to derive a generic architecture which tells how the autonomic element interacts with each other. Authors raised the issues like robustness, trust, and interactions among autonomic agents along with how to transfer the management system knowledge from humans to system? After this, IBM again in 2005 presented an autonomic computing based white paper explaining the base architecture layer of autonomic computing with some layers which are organized based on the IT processes such as Incident, Change and Problem Management. These managements may lead to fulfillment of the autonomic capabilities.

According to ANSI standard [4], "Software quality is the totality of features and characteristics of a product or a service that bears on its ability to satisfy the given needs". Any product that conforms to industry defined set of parameters and has all the features that enhance its performance is said to be of high quality. As of now, there is no parameter in the definition of ISO/IEC 25010:2011 Model, wherein the autonomic features of a product are considered or evaluated; though huge efforts are being spent in incorporating the autonomic features in the software product. There are continuous efforts by researches and software industry to improve the quantitative measurement methods and models.

4. ISO/IEC 25010 model

Software quality model is a framework uses different quality parameters to estimate quality of a software product. These quality models based metrics may be used in many contexts such as, during the software/application development process, for controlling the development time and cost or when identifying the most suitable commercial components/products, and so on. These quality characteristics against which stated quality requirements can be compared for completeness. The International Organization for Standardization (ISO) has defined a set of software quality. First standard was ISO/IEC 9126 and it has been replaced by ISO/IEC 25010 [7].

The ISO/IEC 25010:2011 [7] is a modified quality assessment standard in which autonomic capabilities is added as a new feature of the computer systems based software. As per this standard, quality attributes and the sub-attributes are explained below.

1) Functional Suitability refers to the capability of the product or system that meets stated and implied needs when used under specified conditions. Its sub-characteristics are completeness, correctness, and appropriateness.
2) Performance Efficiency may be defined as how efficient the developed software product or system is with regard to the amount of resources used under specified conditions. It is sub-divided as resource utilization, capacity and time behavior.
3) Compatibility expresses the degree to which a software product or system's component can share information with other components to perform their required functions considering configuration level policies and privileges. It is further categories into: co-existence, and interoperability.
4) Stability defines the degree to which a system resist to environmental changes of the system's component and helps in achieving goals with complete user's satisfaction in terms of efficiency, effectiveness in a specified context. The further characteristics of reliability is divided into: accessibility, operability. recognizability, user error protection, learnability, user interface aesthetics.
5) Reliability measures the probability degree to which a system or component performs its specified functions for a specified time duration. It is evaluated using the available required input states. For this purpose, it is divided into: recoverability, maturity, availability and software faults tolerance
6) Security deals with the appropriate authorization and authentication of the specified product, so that user of system have 'controlled' access, appropriate to the level and type of authorization. Further subdivisions of security are: integrity, confidentiality, computer programs or data non-repudiation and authenticity.
7) Maintainability is the ability to correct or modify or update the software component or the product as per user functional and non-functional requirements. It is further sub-divided into: testability, modularity, analyzability, reusability and modifiability.

Portability is a measure of system's efficiency and its effectiveness using which the system configuration level policies can be transferred from one hardware or software product to another operational environment of other system. Based on that, it is categories into: replaceability. Adaptability and installability.

According to ANSI standard [8], "Software quality is the totality of features and characteristics of a product or a service that bears on its ability to satisfy the given needs". Any product that conforms to industry defined set of parameters and has all the features that enhance its performance is said to be of high quality. As of now, there is no parameter in the definition of ISO/IEC quality model, wherein the autonomic features of a product are considered or evaluated; though huge efforts are being spent in incorporating the autonomic features in the software product. There are continuous efforts by researches and software industry to improve the quantitative measurement methods and models.

5. Proposed enhancement in quality model

In view of the rapid and host of new developments, ISO 9126 should also be modified accordingly to include the autonomic computing aspect. Any autonomic system (AS) architecture includes the elements to be managed and the manager, which provides self - * functionality i.e. self-configuration, self-healing, self-optimization and self-protection [9].

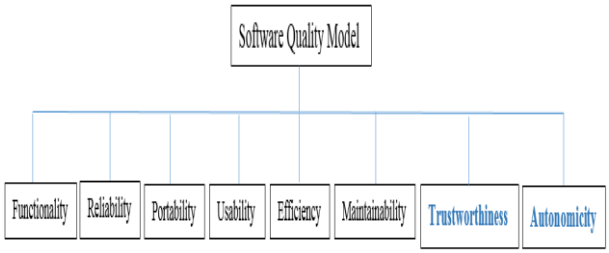

Fig. 1: Enhanced Software Quality Model.

We propose that the model ISO/IEC 9126 should also incorporate the autonomicity and trustworthiness attributes as quality attribute to its existing attributes. We refer to this as modified 'Enhanced Software Quality Model'. This model has a broader range and sub-characteristics, which define the software quality properties

more realistically and accurately. All the other attributes are same as the ISO/IEC 9126 quality model. After the addition of this attribute, the enhanced quality model will appear as in figure (Fig. 1).

Autonomicity

The aim of autonomic computing system is to reduce the management of the network and increases the performance by optimizing the developer's talent on designing the higher level products and policies. This may be achieved via distributing the overall management work on the autonomic manager which will initiate MAPE-K loop and work according to the changes in the system's environment. The autonomic system is controlled by the loop which is designed to handle the self-abilities of the autonomic system. There are two types of system- Open and Closed system. In case of Open loop system, the output has no effect on input but closed loop system output affects the input. Closed system senses the changes in the system. Similarly, autonomic system also detects the changes and diagnoses accordingly. The autonomic computing technique has its importance only because of its self-adapting and self-managing abilities. IBM provided a referenced architecture of the autonomic system with a self-control loop shown in figure 2 which works as central processing unit for Autonomic Management Engine (AME) [9] [12]. Autonomic Management Engine is kind of component that monitors the managed element and decide what will be necessary steps should be applied onto the managed element to resolve the problem identified within the system [13] [14].

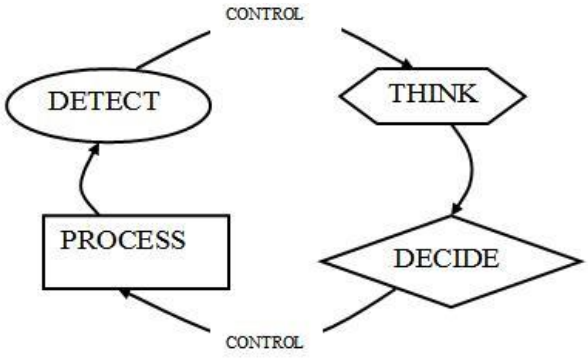

Fig. 2: Self-Control Loop.

There are many challenges in front of the IT industry which comes out by comparing the current computation and handling techniques with the new concept of computation that is Autonomic Computing [13]. The loop handles all the self-functionalities. So considering all the requirements of the autonomic computing, the loop has been designed in such a way that each component uses the functionality and performs accordingly. The attribute Autonomicity, or autonomic capability may be defined as the degree to which the system incorporates the self-*features. One good attempt at assessing the quality of AS through Autonomicity is done in [15]. The authors have proposed an approach for measuring level of Autonomicity (LoA) as this will give autonomic functionality level in a system. They have presented some functionality and these functionalities depend on some metrics. All the functionalities add up to give LoA. In another paper [16], to estimate LoA of any autonomic system, the authors proposed Maintenance Assessment Model (MAM) using which Application Complexity Level (ACL) was evaluated for some autonomic computing based on live projects using fuzzy approach.

The ISO 9126 software quality model has been taken as a base to design enhanced software quality model. To evaluate the quality of autonomic manager, a trustworthy autonomicity characteristic has been considered as one of the factor. Trustworthiness can be taken as a sub-factor for assuring quality of the software product. It assures that the system will perform according to the requirements. For autonomic perspective, trustworthiness means autonomic capabilities will assure their correctness, completeness, function conformity, integrity, confidentiality, etc. So to evaluate overall quality of the autonomic computing enabled system, there is a need to consider autonomicity and trust also. Figure 2 describes hierarchical structure of the quality attributes.

It encompasses various sub-attributes, which may include (and more, depending on how self-* features have been incorporated):

- Computation Index: Includes the complexity of the managed element, autonomic manager, and their interface. It will also include the LOC or function point based size of the autonomic manager. It assures that complexity will be maintained under quality control.
- Completeness: It refers that the software system has been developed according to the specified requirements. Software is complete and trustworthy.
- Function Conformity: It means software function's formation and maintenance will work smoothly
- Failure Response Time: It is the time taken by the system to respond to particular changes occurred in the system. It may also include the time taken to start the adaptation after the change is identified.
- Failure Recovery rate: It may also be viewed as number of task completed related to self-* properties.
- Failure Tolerance Rate: It refers to the level of error prone situations that can be handling by the software itself and continue to perform system's activities.
- Failure Recovery Rate: It is the ability of the system to recover itself from the unexpected situations which have made some software failures in the system. Higher the recovery rate better is the system.
- Throughput Rate: It depends on output rate generated by the software system when particular input based condition gets executed. When time taken by the software to execute a condition and utilization of resources is high then throughput rate increases.
- Failure Analyzability: It is the ability of the software system to identify the failure situation before it happens. If a system is capable of identifying this situation then there will be reduction in failure rate.
- Security & Confidentiality Level: It means the system is protected and it will not be harmed by any malicious attacks. Confidentiality means the information is protected and will be shared between authorized users. Only authorized user can get access to the sensitive information and system is protected from any harmful intruder.
- Integrity: It is the ability of the software system to assure that the information provided by the system should be correct and trustworthy. It ensures no unauthorized access to the system.

Some more may be included, depending on the extent of the self-* features built into the software product and their characterization.

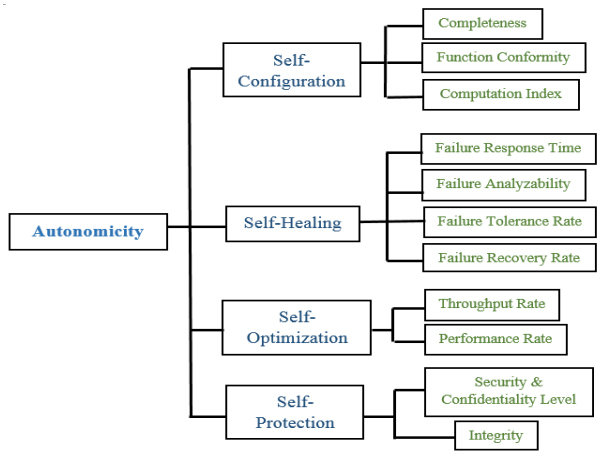

Fig. 3: Autonomicity Attributes for Quality.

There are few common attributes which may be fall under configuration, healing, protection and optimization. These factors are:

- Correctness: It means there is no vulnerability in the software system and it also assures no system failure.
- Agent Activation Time: It is the time taken by the autonomic manager to get activated for a particular event.
- Accuracy Rate: It means how accurate and optimized solution is provided by autonomic agent.

There are quite a number of software tools/packages that incorporate self-*features. As for example, software systems, such as IBM Tivoli, DB2 software [17], Windows 7/10, Cisco IOS Autonomic Networking System [6], Autonomic Job Scheduler, ERP packages and many other software systems have autonomic characteristics, though to a limited extent. Most of the products used for peer-to-peer systems are having autonomic features, like it can handle network configuration, manage node failures, and optimize some performance automatically without much intervention of human beings. Datacenter management software, which are evolving very fast now days due to voluminous and variety data, also include self-tuning capabilities. This enables systems to automatically adjust for varying workloads. Self-healing and self-organizing features are available in Map-Reduce. Similarly, peer to peer systems used for social networking, telephony (e.g. Skype, Face Time) and Facebook etc. also provide support for self-management features.

Lot of efforts and resources are being spent to include more and more autonomic features in the new software products and systems, such as Cloud-based Systems, self-organizing virtual private networks, self-cared IT systems, etc. [8] [9]. Thus, it would be worth to include autonomic features, while assessing and determining the quality of the software product. Academia and industry are putting in lot of efforts and resources to develop software architectures, frameworks and systems incorporating autonomic features into such large complex systems [4,10,11]. Incorporating autonomicity, as an attribute of software in the ISO standards, will lead to more objective and realistic description and estimation of quality of software products/systems.

Trustworthiness

It refers to the assurance of different quality parameters in the software product. It assures that software will work according to the expected functionalities based on which requirements have been provided in the requirement elicitation phase of the software development process. It assures that the software is correct, complete, and secure. Nazila et al [18] in their paper defines the importance of trustworthiness attribute in the Internet based software systems. They have provided a detailed description of all the trust based factors that can be included as a metrics for trust estimation in software systems. They used GQM technique to estimate trustworthiness in socio-technical systems.

To ensure trust in software, another author Rong Jiang [19] mentioned in his article about the trustworthy attributes that may be included in software architecture. The author uses the Grey Decision-making Method (GDMM) and Principle of Maximum Entropy (POME) for estimating the trustworthiness of software architecture and also proves the rationality and scientificity of the method. He also verifies the feasibility of his studies through case analysis. According to the Tim Boland et al [20], trustworthiness is a quantifiable attribute that can be measured using some parameters that support trustworthiness of the software. For this purpose, the authors proposed a framework composed of models with the goal to evaluate trustworthiness of the software. The authors use structured assurance case methodology to determine the trustworthiness of the systems. This kind of modelling approach consist of claims, subclaims, arguments and evidences related to all the factors that are consider to be trust estimating factors for a software. With this approach, the authors used a computation model for which they represented the trustworthy factors using the kiviat chart. The jagged line inside the circle shows the actual measure of the trustworthiness of each term, and outside circle shows the potential trustworthiness of each term. Using these parameters, the author evaluates the trustworthiness of the software.

Trustworthiness has not been included as a quality factor in any of the ISO/IEC quality model for quality estimation of software. Based on the literature work proposed by other researchers on trustworthiness, we have identified the following attributes for trustworthiness to assure quality of a complete software system. Figure 4 shows basic six attributes of the trustworthiness which include every aspects of the software workflow and also assure trust in the software.

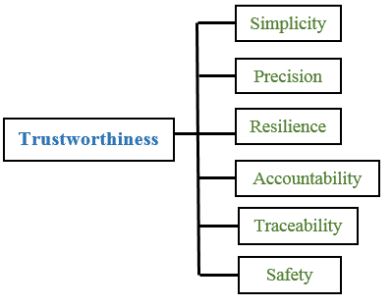

Fig. 4: Trustworthiness Attributes for Quality.

- Simplicity: It is the state of software which assures that product to being simple which is easy to understand and explain.
- Precision: It states that the information or solution provided by the software is exact and accurate.
- Accountability is a kind of assurance by the software that quality evaluation is being performed based on the performance and software behavior.
- Safety: It states that software that assures that the software is safe from any harm or danger.
- Resilience: The capacity of software to assure that system can recover quickly from any unwanted harm and toughness.
- Traceability: It is the ability to assure that the information will be verified from historical database, documented recorded easily if required.

Using these parameters, trustworthiness of autonomic and non autonomic computing based software systems may be evaluated that helps in estimating the quality of the software. Trustworthiness may also include few more parameters but we have identified these parameters based on our studies.

6. Conclusion

Continuous increase in complexity within multi-tier computing architectures remains an open and challenging problem in the software arena. Due to increase in complexity, the costs associated with operating and maintaining systems, also increases sharply. Though, the development of autonomic system will introduce complexity in the overall system which further leads to management task. Therefore, a proper balancing of the complexity of the system with the autonomicity factor will be a crucial aspect in developing a quality system. It becomes imperative and more realistic to include autonomicity and trustworthiness, along with other 6 attributes (see above), while assessing the quality of software systems. In the present paper, we have proposed the inclusion of autonomic feature to the existing software quality model ISO 9126. There is a need to explore various sub-characteristics of autonomicity further, so that appropriate evaluation criteria and metrics may be proposed, which will help in assessing the autonomicity level of the software product and finally its quality. We are working on these issues and results will be reported later.

References

[1] IBM, "An Architectural Blueprint for autonomic computing", *IBM White Paper*, Vol. 7, "(2005), pp.1-31.

[2] IBM Corporation "Autonomic Computing Toolkit." http://www.128.ibm.com/developerworks/autonomic /overview.html (2005).

[3] Mulcah JJ & Huang S, "Autonomic Software Systems", *IEEE International Conference on Software Maintenance and Evolution*, (2014), pp.1-4. https://doi.org/10.1109/ICSME.2014.92.

[4] Chauhan SK & Sharma A, "Runtime Decision Making for Developing Autonomic Systems", *International Journal of Computing, Intelligent and Communication Technologies*, (2013).

[5] Khalid A, Haye MA, Khan MJ & Shamail S, "Survey of Frameworks, Architectures and Techniques in Autonomic Computing", *IEEE Explore, Fifth International Conference*, (2009) https://doi.org/10.1109/ICAS.2009.38.

[6] Nami MR & SharifiM, "Autonomic Computing: a new approach" *First Asia International Conference on Modelling & Simulation, (*2007), pp.352-357 https://doi.org/10.1109/AMS.2007.20.

[7] ISO/IEC 25010:2011: Systems and Software Engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models

[8] http://www.ibm.com/developerworks/data/library/techarticle/dm0709saraswatipura/index.html

[9] Ranjan R, Buyya R & Parashar M, "Autonomic Cloud Computing: Technologies, Services, and Applications, Concurrency and Computation", *Practice and Experience*, Vol.24, No.9, (2012), pp.935-937. https://doi.org/10.1002/cpe.1865.

[10] Schneider C, Barker A & Dobson S, "A Survey of Self-healing System Frameworks", *Software: Practice and Experience*, Vol.45, No.10, (2014).

[11] Chauhan SK, Sharma A & Grover PS, "Proposed Runtime Decision Making framework for Autonomic Software Systems", *Proceedings of the 19th International Conference on Computers, Zakynthos Island, Greece*, (2015), pp.371-375

[12] Kiran & Chauhan S, "Software Metrics–A Tool for Measuring Complexity", *International Journal of Software and Web Services*, (2012).

[13] Chess DM, "The vision of autonomic computing", *IEEE Computer*, Vol.36, No.1, (2003), pp. 41-50. https://doi.org/10.1109/MC.2003.1160055.

[14] Shuaib H, Anthony R & Pelc M, "A framework for certifying autonomic computing systems", *The Seventh International Conference on Autonomic and Autonomous Systems*, (2011).

[15] Eze T, Anthony R, Soper A & Walshaw C, "A Generic Approach towards Measuring Level of Autonomicity in Adaptive Systems", *International Journal on Advances in Intelligent Systems*, Vol.5, No.3&4, (2012), pp.553-566

[16] Sharma A & Dehraj P, "Complexity based Maintenance Assessment for Autonomic Agent", *WSEAS-Conference, Rome, Italy*, (2015).

[17] American National Standard Institute (ANSI), ISO Glossary, http://www.standardsportal.org /usa_en/resources/glossary.aspx.

[18] Mohammadi NG, Paulus S, Bishr M, Metzger A, Könnecke H, Hartenstein S, Weyer T & Pohl K, "Trustworthiness attributes and metrics for engineering trusted internet-based software systems", *International Conference on Cloud Computing and Services Science*, (2013), pp.19-35.

[19] Jian RA, "trustworthiness evaluation method for software architectures based on the principle of maximum entropy (POME) and the Grey decision-making method (GDMM)", *Entropy*, Vol.16, No.9, (2014).

[20] Boland T, Cleraux C & Fong E, "Toward a preliminary framework for assessing the trustworthiness of software", *National Institute of Standards and Technology*, (2010). https://doi.org/10.6028/NIST.IR.7755.