Measurable Quality Characteristics of a Software System on Software Architecture Level

S.R. Khayami, A. Towhidi, K. Ziarati

***Abstract*—The main purpose of most software produces, into present a software system with high quality. Most of the experts believe that to achieve this subject, performing all of the stages of producing the software must be based on qualitative programming and performing prepuces. These points specially in producing the software architecture as one of the most important stages of this process must be considered more carefully. Regarding to different and various definitions for quality and architecture of software, this essay must at first investigate these two matters and there based on achieved results, determine the Evaluable qualitative characteristics at the architecture level.**

***Index Terms*—Evaluating the software architecture, Software architecture, Qualitative architecture factors, Qualitative characteristics of software.**

I. Introduction

Most of the experts in software developing believe that quality is nothing that can be added to software, but it is one that must be in it. It means that the production levels must be programmed and performed based on mentioned quality for final production. On the other hand, today all of the software industry developers are regarding a high quality production as their own main purpose. The experiences have showed that whenever it is necessary to design anything with high demotions and complexities, a general view which is called "architecture" is needed. The meaning of architecture is to explain general construction of a system as if the behavioral characteristics and defined in it. Architecture gives us an overall point of view of the whole system that is necessary for controlling and progressing. The main aim of this paper is to present the qualitative characteristics which at architecture level of a software, being investigable and evaluable. This evaluation is performed to (predicate) anticipate the qualitative characteristics of produced production based on relative architecture.

In this paper at first we pass away the contents of software quality. Then by the aim of making a better comprehension of architecture content, the relative definitions are investigated. At the end, regarding the mentioned matters in two previous parts, the evaluable qualitative characteristics are distinguished at the level of architecture.

R. Khayami is Ph.D. student, Departement of Computer Science and Engineering, Shiraz University, Shiraz, Iran (corresponding author phone: 0098-917-1004856; fax: 0098-711-6271747; e-mail: khayami@shirazu.ac.ir).

Dr. A. Towhidi is with the Departement of Computer Sci. and Eng., Shiraz University, Shiraz, Iran, (e-mail: towhidi@shirazu.ac.ir).

Dr. K. Ziarati is with the Departement of Computer Sci. and Eng., Shiraz University, Shiraz, Iran, (e-mail: Ziarati@shirazu.ac.ir).

II. The Quality of Software

Pressman in his book "software engineering" describes the quality of a software system as follows [1]: Conformity of software with operational and effective necessities which are clearly presented, and also respecting the production standards and software development which are clearly supported, and the existence of implied characteristics which are expected from all advanced professional software. So it is distinguished that we should search for the characteristics which exist in product software and has high concord with above divination. Usually the necessities of a software system are divided in to two groups: Functional requirements and non-functional requirements are those which software is designed for performing them and define the functional and executive purpose of that system. The non-functional requirements mostly focus on how a software system works and performs. Affairs such as efficiency, extendibility, maintenance, security and even the production expense are located in domain of non-functional requirements. Technically, these characteristics of a system are called "software qualitative characteristics".

To describe the qualitative characteristics usually the qualitative models are used. Many models have been suggested to describe the quality of software system, such as Mc Call [2], Boehm [3], FURPS [1], IEEE [4], and ISO [5]. These models have been presented as tree-construction of qualitative characteristics and their relationships.

The characteristics of the first level of these models are called quality factors. Characteristics such as: efficiency, reliability, maintainability, portability, usability, functionality are in most models from the implied point of view. The ISO/IEC qualitative model from the coverage point of view of qualitative characteristics has more expansion in comparison with other models and defines the quality more precise. The quality factors and sub-factors of this model are as follow:

Operation: suitability, interoperability accuracy and security.

Reliability: Maturity, fault endurance and recovery.

Usability: Understandability, learnability and operability.

Performance: efficiency or time behavior in performing the processes and best use from resource behavior.

Maintainability: analyzability, changeability, stability and testability.

Portability: adaptability, install ability, replace ability, interoperability.

III. Software Architecture

A. Definition of Software Architecture

The word "architecture" has a Latin root and it means "master in making". Architecture of a system shows the collection of technical maps of different aspects of that system. In fact it is a high definition of the system that presents the purposes and operation of system for designers and makers and also users and shows the conformity with customers' necessities. On the other hand, architecture shows a single definition of performing of software system. In general, the focus of architecture in one point is after the analysis and before designing, and through the analysis of analytical model into sub-systems and their mediators and finally determining the main parts and identifying the system processes are defined. In fact, architecture is high designation of mentioned software that above points have essential effects on its totality. The details which are just related to one part of the system are presented in low level designation. We can say any "system architecture" is a "design", but every "design" cannot be "system architecture". Architecture is the first stage of system alternation presented by stock holders as the operation of technical elements are defined in it. As it is showed in Fig. 1 an effective model represented by user sequence diagram of alternation process and then by refinement and making it exacter, this diagram is presented at the level of architecture system. Decisions made from the point of system and have a wide domain are called architecture decisions. But those decisions made in a limited stage and have a local view are not considered as architecture decisions. This grouping let us make a difference between designation and performing the details and architecture decisions. The architecture decisions define the constructional and important elements of system and also the observable characteristics and their relationships. For example, style selection, selecting the number of indicators, or the observable indicators from outside, qualitative considerable characteristics, each of them is an architecture decision. Architecture decisions happened or made at the first level of architecture [7].

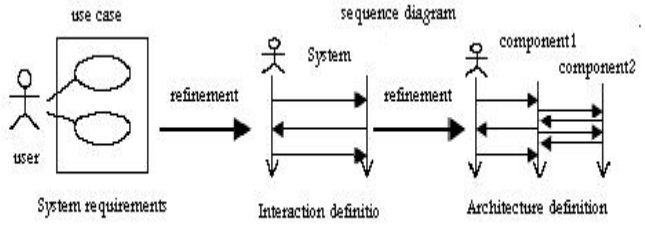

Fig. 1. Refinement of System Architecture

There is no exact definition for software architecture. In some references there are more than 100 definitions. The most famous definitions are as follow [8, 9, 10]:

- Software architecture is a collection of design architecture that has a specific frame. These elements are 3 groups: processing elements, data elements and connecting elements.
- Software architecture is a collection of indicators and connectors with the definitions of their interoperation.
- Software architecture for a program or for an accounting system is construction (s) of that system which includes of software indicators, external manifest characteristics of those indicators and their relationships.
- Architecture is a collection of important decisions about organization of a software system, selecting the constructional elements and their mediator, with behaviors of elements which by means of them can have some cooperating with other elements. Composition of constructional and behavioral elements in big and under-developing sub-systems and also a method to guide and organize them are presented in architecture.
- Essential organization of a system which includes of indicators, the relations of each of them with the other indicators and with environment and the principles of completion and designation.

$$Arch = Comp \cup Connect;$$
$$Comp = \{C_i\}; Connect = \{C_{ij}\}; i, j \in [1..n] \quad (1)$$

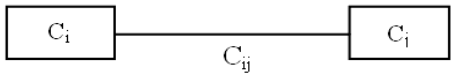

Fig. 2. Component and Connection in Software Architecture

As it is clear in these definitions, different words and expressions have been used. These definitions are showed in (1) by mathematical relations. In mentioned Fig. 2 C_i is the ith indicator C_{ij} as a connector between i and j; and n is the number of system indicators and Arch is the architecture which includes of the collection of elements and their relations. Inshore, software architecture means to present construction of the system which shows the software indicators and observable characteristics from the external side of these indicators and their relations [11]. In other word, software architecture is construction of indicators in one system, their internal relation and principles and guidance lines which make designing and evaluating a system possible.

B. Architecture Presentation

Software architecture is as a complex matter that cannot be explained in a simple one dimension frame. To make this subject clear, we can use the same or equivalent concept of it in the domain of building. There is not only one map of architecture but also many maps from different aspects for a building must be considered. For example, designation of rooms, designation of classification, electronically institution maps and building ventilation, pipe-laying and the security design of the building can be mentioned here. None of those above mentioned overviews is not by itself architecture, and all of them together make the concept "architecture". One orthopedist, nerve specialist, psychologist, gland specialist and other medical specializations, each of them identifies one aspect of complex construction of human body. For a complete explanation, although the explanation of each of

them is necessary but their entire set together makes a complete explanation. So, architecture is a collection of technical maps that each map includes of explaining of one specific aspect of the system. To explain the different aspects of architecture, a series of model is used and each model uses specific rules, signs, semantics and syntactic. In the method of architecture presenting, the most important matter is to consider the readers' point of views. A documentation which at the level of production is easy but unclear and hard for the readers is not used. About software architecture documentation, our readers are those who have gotten a profit from architecture and architecture tokes effectiveness of the readers that these groups are called software architecture stock holder. So the documentations must be produced in such ways that for all of the stock holders are understandable. In other word any stack holder depending on the level of his information should comprehend his own understandable information. To understand this matter that what kind of people are these stock holders and how they want to use the documentation helps you to organize and use these architectural documentations. Requirements engenders, designers, performing specialists, experimenters, quality guarantee group, project manage, customer and find user are the examples of architectural stuck holders. The stockholders are divided in to two groups: experienced and inexperienced. The experiences have showed that the inexperienced necessary information from the point content are like these who are experienced and have the some specialization but at the lower level and at the more primary level. One of the organization methods of representing the required concepts in frame architecture is 1 + 4 view [12]. This method defines many keys and essential views of a system in order to achieve a complete explanation of a system. These views are as follow:

Logical view: This view gives a logical presentation of groups and main sub-systems in under designing software system. This view connives at any performing or any physical details. At the next stages logical indicators are being changed to physical processes and hardware.

Process view: this view shows that how different processes interact with each other and how their relationship is.

Establishment view: this view shows any way to define real processes and explains how these processes are established on the physical hardware.

Performing view: This view shows that how real software performs and usually includes of contents such as: real resource code, code construction and construction of used libraries in system. Many efforts have been done in order to present graphical presentation of this view but none of them had a well-formed affection on it.

Applied cases view: This view includes of the body of applied case which must be defined for understanding to system behavior. Another point about how to present the considerable contents is software architecture. In order to do this fact, there are different ways. Engineering software diagrams like data flow diagram (DFD), Majors connection diagram, entity relationship diagram (ERD), are diagrams which are used to present some considerable views. One of the most common methods is to show different views by means of UML diagrams.

C. Uses and Reasons of Software Architecture Importance

For investigating the importance of software architecture from the technical point of view, we can give three reasons [13]:

1) Architecture as a means of relation between the system stock holders: Software architecture system can be a common view of all stuck holders. If they have the same idea about that software, it can be as a basis for reciprocal understanding, consultation and also a connecting factor among them. As it is said before, in fact architecture makes a common language between them which is understandable for everyone and helps to understand the system better.
2) Early decision making of designation: Software architecture includes of high level decisions and trade –off in designation that leads to produce a software system and define its characteristics. The studies show that the expense of correcting a discovered error a long the requirements recognition phase or in architecture phase is more less than correcting that error when the error is distinguished in experiment phase.
3) Possible re-usability in architecture: We said that the high level architecture defines the high level designation and indicator recognition and their construction in a system. So it is possible that in one similar system with those requirements and characteristics, the indicators can be again used and as you know, this matter has specific importance in production process and developing software systems.

IV. Investigable Quality of Software Architecture

In part one, the importance of quality and quality characteristics in software production has been mentioned. In part two, software architecture, as one of the most important software production stages was investigated and its different aspects have been discussed. This part tries to show the effects of these two contents on each other. In fact we want to show that how much quality of final production can measure in architecture, also could produced software quality characteristics be estimated by its architecture or not?, or are these considerable quality purposes in final system observable in architecture level or not?

The aim of this part is not to present the exact measurable standards in architecture or define the relative characteristics of any sub-factors. The aim of this part is to show investigate ability and evaluate ability of quality factors or sub-factors at the level of architecture. In fact we show that in order to evaluate the quality of production, can use the architecture. So it is showed that which collection of qualities if intends to be in final production must be investigated at the level of architecture. In this stage to investigate the software system quality we have used ISO/IEC model. This model not only covers the main common factors between the different models, but also considers its sub-factors, regarding many other factors and sub-factors of other models. In this way if we can show the evaluate ability of quality factors for this model at the architecture level, we can generate it for all of

the models too. So in continue investigation of factors and sub-factors of ISO/IEC model has been discussed.

A. The Analysis of Operation Factor at the Level of Architecture

The first sub-factor operation is suitability. This sub-factor measures the conformity of operation system with stock holders' requirements. So regarding these defined matters by system stack holders and their refinement and making this case more recognizable define the indicators and by investigating the characteristics of these indicators, evaluate the amount of behavioral conformity with required reactions. Next sub-factor is accuracy. This sub-factor shows that how the correct and exact result is made by the system. Although the exact measurement must be based on the affairs in code system, but investigating the general affair and activity at the level of architecture was possible and readable. Therefore by studying defined processes in architecture and following the affairs by indicators, the accuracy of these processes can be evaluated. The next sub-factor is interoperability with other systems. If a system has such a characteristic, at the level of architecture, the indicator(s) in order to give relating mediator role must be considered. The indicators which have middle ware role in connections system. The last sub-factor is security. Security means to control accessibility and to do the affairs in defined permitted domains. So the first matter is to recognize the volunteers identity who wants to do the affairs and then to control the accessibility level. So the indicator or the mechanisms in processes of system must be designed and exists in relative architecture. Then the existence or non-existence of relative elements are defined by investigating the indicators duty or their operations. Also the suggested mechanisms could be compared with the standard mechanisms for security.

B. The Analysis of Reliability Factor at the Level of Architecture

Three aspects of this factor are: maturity, Fault tolerance and Recovery. Although most of these sub-factors have been defined based on the events of run time, but we can anticipate the possibility of fault occurrence in run time of system by investigating the system indicators and their operations in different situations. Fault endurance emphasis ion this aspect of system that, the system is capable to continue it work in spite of the fault existence. This matter is defined at the level of architecture by investigating the existence or non-existence of exception handling methods or elements of redundancy. A system which has not considered any mechanism to do this fact, it is easily anticipated that how it encounters with faults of performance time. Recovery from fault manner, emphasizes on reversibility of data and performance to previous situation of fault appearance and also on amount of time and consumed resource. When such mechanisms are in a software system, surely in its architecture exist relative recognizable methods. In fact in order to provide this factor, there must be designed the components or mechanism for any aspect, and this matter is well observable at the architecture level.

C. The Analysis of Usable Factor at Architecture Level

Usability is defined by means of understandability, learning and using the system this faction emphasizes on one aspect of software which shows the comfort and simplicity of affair by means of user. User mediator architecture and divisions of service presentation to the user are the most important matters which are effective in mentioned sub-factors. User mediator architecture from the conformity point of view with user mental contents of operation, possibility of error correction and guide mechanisms are subjects that can provide the aims of this factor. Using the standard design user mediator methods can have a good emphasis on providing the aims of this factor. Investigating the existence of such indications in software architecture can be a good and suitable proof to anticipate the quality do final production.

D. The Analysis of Output Factor at Architecture Level

Sub-factors related to efficiency or time output of system in performing the processes and best use of resource system are two fields which recognize the output. Performing processes at minimum time with minimum resources are always the most important purposes of software designers. Times included in this matter are divided in to three groups: necessary time to show a reaction to an occurrence, processing length of time and decision making and finally time of reply arrival anticipating these times based on included indicators in any stage and their performance is possible. Also the best use of resources in performing the affairs is another point which should be considered in investigating the system replier situations. Doing unnecessary affairs, unnecessary use of resources like data redundancy and also improper delay (retardation) in releasing the resources are reducer samples of system output. So by investigating and controlling these points in system indicators, the output situation can be participated and the amount of this factor in proposed architecture can be defined.

E. The Analysis of Maintainability in Architecture Level

This factor has been expressed in many scientifically te4xts as the main important quality factor. In researches it is defined that more that 50% to 70% of life cycle expense of a software system belong to performed expressions on the first coy of system [14]. The importance of this factor is too much that some evaluating methods of soft ware architecture have just investigated this quality factor or sub-factors [15]. As mentioned before most techniques of software production by the aim of increasing the maintainability, has been developed. Relative sub-factors include of analyzability, changeability, amount of confirmation, and testability. Analyzability emphasizes on possible recognition of manners and deficiencies of a system. Also this factor tries to define those parts which must be corrected. Simplicity of performing the required reforms in a system is called change ability. The possibility of preventing the non-required affections in one part on other parts is defined by amount of system confirmation. Testability affects on amount of possibility of experiment and its operation test. To evaluate this diagnostic at architecture level, the system can be analyzed according to indicators characteristics and their relations with each other. The amount of modularity of system and the modules properties from the external coupling point view with other indicators and internal parts cohesion of any these system indicators can define the relative

sub-factors of this factor. If the system has been divided correctly to suitable modula, the system can be analyzed more easily, the changes are performed by specific affections, recoveries are simply performed in indicators and we can test any part in suitable form. So this factor can be evaluated by controlling the modularity level of the system.

F. The Analysis of Portability Factor at Architecture Level

In order to evaluate the portability factor of sub-factors adaptability, we should investigate the install ability and interoperability. To increase adaptability it is necessary to have layers or indicators in one system, to adapt the main core of system with different environments. This layer must provide relative messages of system with its affair environment in suitable form, or provide the possible recovery of this part without any affection on other parts. In order to install easily a not automatically, the indicators in architecture must be considered. And finally in order to interoperability, the system must do the required exchanges in affair environment with systems and associate the resources.

This property, according to using software systems and saving the time and firms' investment, is necessary. Because usually any system in one firm in different times and by different producers are performed , so non-existence of associability or non-existence of adaptability, their processes can make many problems along performing the mentioned firm's affairs. So we should consider for any mentioned sub-factor in this part, one component or mechanism. A suitable evaluation of portable quality factor can be obtained by investigating and analyzing the capacity of these components or mechanisms.

V. Conclusion

This essay follows two main purposes. By considering different definitions which exists for software architecture, the first purpose is connected to introduce and to clarity the relative contents in order to not only remove the ambiguity, but also to provide a suitable view of main contents. But the main purpose of this essay is to define software quality factors which are evaluable at architecture level. In this essay by analyzing the quality factors and sub-factors model (ISO/IEC) at architecture level, a method to evaluate them has been suggested. So it is showed that all quality properties of software final production are under the impression of decisions at software architecture level. The quality aim of final production is provided when at architecture level some suitable alternatives are considered. Based on final quality aims we can evaluate proposed architectures for a software system and recognize their strength and weakness. This matter provides not only a comparative aspect but also provides a method to evaluate independently architecture. With such evaluation, some parts of system which need to be revised and improved could be determined and give suitable suggestion to remove disadvantages of them.

References

[1] R. S. Pressman, *Software Engineering: A Practitioner approach*, McGraw-hill, 2000.
[2] J.P. Cavano and J.A. McCall, "A Framework for the Measurement of Software Quality", *Procs. ACM Software Quality Assurance Workshop*, 1978, pp.133-139.
[3] B.W. Boehm, J.R. Brown, H. Lipow, G.J. Macleod and M.J. Merrit, "Characteristics of Software Quality", Elsevier North-Holland, 1978.
[4] IEEE Std 1061-1992, "IEEE Standard for a Software Quality Metrics Methodology", IEEE, 1992.
[5] ISO/IEC 9126, "Information Technology-software Product Evaluation: Quality Characteristics and Guideline for Their Use", ISO/IEC, 1991.
[6] Wikipedia, the free encyclopedia, 2207.
[7] S. T. Albin, *The Art of Software Architecture: Design Methods and Techniques*, John Wiley & Sons, 2003.
[8] Software Engineering Institute (SEI), Carnegie Mellon University, www.sei.cmu.edu, 2007.
[9] R. Kazman, L. Bass and P. Clements, *Software Architecture in Practice 2Ed*, Addison Wesley, 2003.
[10] *IEEE Std 1471-2000*, "IEEE standard recommended practice for architecture description", IEEE, 2000.
[11] F. Losavio, L. Chirinos, A. Matteo, N. Levy and A. Ramdane-Cherif, "ISO quality standards for measuaring architectures", *The Journal of System and Software*, Elsevier, 2004.
[12] P.B. Kruchten, "The 4+1 view model of architecture", IEEE Software, 12(6), 1995, pp.:42_50.
[13] M. Klein, P. Clements and R. Kazman, *Evaluating Software Architectures: Methods and Case Studies*, Addison Wesley, 2002.