Volume 1 No. 1, APRIL 2011
ARPN Journal of Systems and Software
©2010-11 AJSS Journal. All rights reserved
http://www.scientific-journals.org

Software Architecture Performance Quality Model: Qualitative Approach

Ishaya Gambo[1], Abimbola Soriyan[1] and Philip Achimugu[2]
[1]Computer Science and Engineering Department, Obafemi Awolowo University, Ile-Ife, Nigeria
[2]Computer and Information Science Department, Lead City University, Ibadan, Nigeria

ABSTRACT

As a communication vehicle among stakeholders, software architecture gives the entire view of the system's major components, the behaviour of those components as visible to the rest of the system, and the ways in which these components interact and coordinate to achieve the system's goal. The architecture determines the non-functional attributes of software systems that are built into quality models. In this paper we consider the performance attribute of a system. Most performance quality models have been developed and proved quantitatively. This paper approaches performance issues qualitatively using a proposed developed performance quality model called Software Architecture Scenario-Based Performance Quality Model (SASPUM). The validity of the model characterized into three categories: *stimuli*, *architectural decisions*, and *responses*, can be tested on any existing software architecture using the performance Assessment of Software Architecture (PASA) and the Architecture Trade-off Analysis Method (ATAM).

Keywords: *Software Architecture, Quality Model, SASPUM, PASA, ATAM*

1. BACKGROUND

Quality issues are paramount to all stakeholders involve with the use of information system in any given and established organization. This has been the on-going research efforts for many in the software engineering community. Basically Hoyer et al.,(2001) gave two major camps when discussing the meaning and definition of (software) quality. According to their work, quality means conformance to specification and also meeting customers' needs. Putting it in a more comprehensive and technical way, we have lots of quality strategies proposed by Crosby (1979), Deming (1988), Feigenbaum (1983), Ishikawa (1985), Juran (1988), Shewhart, (1931) and models by Jim McCall *et al.* (1977), Boehm *et al.* (1978), (Grady, 1992), Dromey, (1996), ISO 9000 in ISO, (1997), ISO/IEC 9126 (2001), ISO/IEC 15504 (SPICE6), etc. All the quality models seem to be a good reference tool for defining the quality of a software product. In fact, in the computerized society we are in, these software product qualities are increasingly important.

However, it is obvious to see that most organizations have deeply relied on software to perform their vital functions in terms of operations and otherwise. This has made the software engineering community to pay closer attention to developing software systems with the right quality. The quality of software will affect the overall performance of the system, which includes the delivery of services, maintenance of system etc. The main quality criteria for software- systems is their "fit for purpose", in both functional and non-functional terms.

Chung provides an extensive listing and description of quality attributes in *Non-Functional Requirements in Software Engineering* (Chung, 2000). According to Chastek (2003), "quality attributes arise in the business case, market analysis, and requirements, and can refer to the product being developed or to the development of that product". Such quality attributes tend to be global, referring to the entire product or the development of that product.

Moreover, performance is an important quality attribute of software systems, which can be determined and ensured by the architecture. Performance failures result in damaged customer relations, lost productivity for users, lost revenue, cost overruns due to tuning or redesign, and missed market windows. Clements in his opinion on performance pointed out that "Performance is largely a function of the frequency and nature of inter-component communication, in addition to the performance characteristics of the components themselves, and hence can be predicted by studying the architecture of a system" (Clements, 1996). This further supports Perry and Wolf (1992), Garlan and Shaw (1996), and Clements and Northrup (1996) that says "there is growing recognition of the role of architecture in determining the quality of a software system." More so, Clements and Northrup (1996) opined that "Whether or not a system will be able to exhibit its desired (or required) quality attributes is largely determined by the time the architecture is chosen."

Furthermore, the role software architecture plays in the overall system quality has been established (Clements et al., 2002). If the architecture is not right, the system will not meet its requirements. In the software engineering community, software architecture has been identified as an increasingly important part of software development and quality control and management need to be carried out throughout the whole development process to ensure implementation of required quality characteristics. Several quality model like the McCall's, Boehm's, FURPS/FURPS+, Dromey's and ISO/IEC 9126 have been shown in literatures as useful tools for quality requirement engineering as well as for quality evaluation, since they define how quality can be measured and specified. In software performance engineering (SPE) processes, quantitative approach to constructing software systems to meet performance objectives have been systematically employed and deployed. Since most performance models are predicted quantitatively, this paper employs the qualitative approach to performance quality model.

2. SOFTWARE QUALITY MODEL

Quality models for software architecture are taxonomies of quality attributes, commonly used to specify and evaluate non-functional requirements. Most models e.g. Cavano *et al.*, (1978) and ISO/IEC (1991) offer a two-level approach, distinguishing externally observable and internally measurable attributes, yielding stakeholder-specific composite quality criteria. Significant work has been performed to determine which internal attributes influence which external ones and most models stick to a two-level hierarchy.

Albin, (2003) opined that “a quality model is a taxonomy of quality attributes and their relationships.” A quality attribute is a specific characterization or property of a process or product, which can be measured or observed. Quality attributes encompass the traditionally called “ilities”. A quality attribute may have several metrics, each of which define a measurement or scale (quantitative or qualitative) and a method or technique to observe or measure the attribute. Metrics can be internal or external: internal metrics are applied during construction to the executable system or to its source code (e.g. subsystem performance or code complexity), whereas external metrics are applied to an executing software product (e.g. functionality, reliability and performance).

In most quality treatments, it is mentioned that some quality attributes (like performance and reliability) have quantitative metrics that can be measured by executing the system, and other quality attributes have qualitative values that can be observed by executing the system (e.g. testing for usability) or through non-operational scenarios (e.g. identifying required steps to introduce a new function into the system). There are thus two dimensions of interest: the nature of the measurement (quantitative vs. qualitative) and the time of measurement (executing a system vs. analyzing a description). Clearly, all four possible combinations are in use:

a. Executing/Quantitative: performance and stability.
b. Executing/Qualitative: usability.
c. Analyzing/Quantitative: modifiability
d. Analyzing/Qualitative: modifiability and some kinds of performance.

In some quality models, external attributes are correlated with top-level “characteristics” and internal ones with subordinate “sub-characteristics”. Some quality attributes are influenced by the system architecture, and thus can be (partially) evaluated from the architecture descriptions (Albin, 2003); in particular, functionality, performance, modifiability and reliability.

Software quality models have been a research matter since the 70’s which are shown by the penned literature at that time (William *et al.*, 1977), (Boehm *et al.*, 1976). In their paper Boehm *et al.*, (a1976) the authors give an overview of quality attributes and create a quality model which mainly corresponds with the ISO 9126-1 quality model (ISO/IEC, 1998) which standardizes a software product's quality.

In this paper an improved software architecture-based quality model is developed to ensure that a software product corresponds to the demanded software. The existing models are based on the decomposition of the overall quality into abstract quality characteristics which are necessary for meeting the stakeholders' requirements. Each of these characteristics can be further refined into one or more sub-characteristics which also can be refined again. This refinement process results in a tree of quality characteristics whose leaves are concrete quantifiable quality indicators called metrics.

However, there exists a scenario-based approach to the analysis of software architecture by Kazman, et al., (1997) and Kazman, et al., (1996). In their approach, various stakeholders, which include users, system administrators, and maintainers in the system, were considered. They also developed usage scenarios from their various points of view. These scenarios are expressed informally as one sentence descriptions. They typically capture uses of the system that are related to quality attributes, such as ease of modification. Applying this approach to analyze an architecture will make the evaluation to be on how much work is required to satisfy the scenarios. Kazman, et al., approach focuses on early evaluation of software architectures to reveal problems at a point in the software development process where they can be most easily and economically corrected. In this paper, the qualitative approach is also scenario-based, and can used to analyze an existing architecture at the later stage, not necessarily at the early stage. The approach in this paper is based on the identified architectural quality attribute ‘performance’ that can be used along side with ATAM for qualitative reasoning and analysis given the different views an architecture and architectural patterns, tactics and quality sensitive scenarios. With the model, the right questions about an architecture can be asked by telling us what artifacts (such as specific design decisions also known as architectural tactics and patterns, which are the tools used) to examine in achieving the quality attribute (performance) with the scenarios generated.

3. RESEARCH DESIGN

Barbacci, *et al.*, (1995) in their work opined that “a variety of qualitative and quantitative techniques are used for analyzing specific quality attributes.” These techniques have evolved in separate communities, each with its own vernacular and point of view and have typically been performed in isolation. Qualitative approach or method is conducted in a natural setting and involves a process of building a complex and holistic picture of the phenomenon of interest. This approach examines the process of assigning meanings. It is based on interpretation and constructivism, taking into account that there exist multiple realities and multiple truths based on the construction of a social reality that is constantly changing. Therefore, the investigator and the object of study are interactively intertwined in such a way that discoveries are created mutually within the context of the situation that molds the investigation (Denzin *et al.*, 2002), (Guba *et al.*, 1994). Also qualitative research methods mainly analyze visual and textual data in such a way that the sample is restricted to just a few or even only one example. Hence, this type of method allows the complexity of the problem to be confronted, keeping in mind that results are not the objective. Rather, the goal is to be able to generate new theorems or improve existing ones.

More so, the qualitative approach involves the use of case studies where a single entity or phenomenon (‘the case’) bounded by time and activity (e.g., a program, event, institution, or social group) and collects detailed information through a variety of data collection procedures over a sustained period of time are explore. The case study is a descriptive record of an individual's experiences and/or behaviors kept by an outside observer. This approach is usually used for the investigation of operational issues and social phenomena, or in other words, situations in which people are involved and different kinds of processes take place. The main characteristic of this qualitative approach is that, it is aimed at constructing a theoretical framework that emerges from the analysis of the data gathered. This gives us the advantage of explaining the results in a coherent manner.

However, the qualitative approach used in this paper employed an interpretative approach that can be use to generate qualitative questions using scenarios. This can be applied on an existing software architecture with performance quality identified, where the Kruchten’s 4+1 view model can also be used as the architectural view and design pattern to breakdown into structural parts the existing architecture in order to provide the needed architectural quality for evaluation. The model was developed in Unified Modeling Language (UML) using AgroUML as a Computer Aided Systems Engineering tool. The model is used to ask the right questions about the architecture, that is, what artifacts (such as specific design decisions also known as tactics and patterns) to examined with the questions, and then measure the results gotten from those questions. A scenario-based approach will be used to generate these questions. The effectiveness of the model was tested and validated using the performance Assessment of Software Architecture (PASA) and Software Architecture and Architecture Trade-off Analysis Method (ATAM).

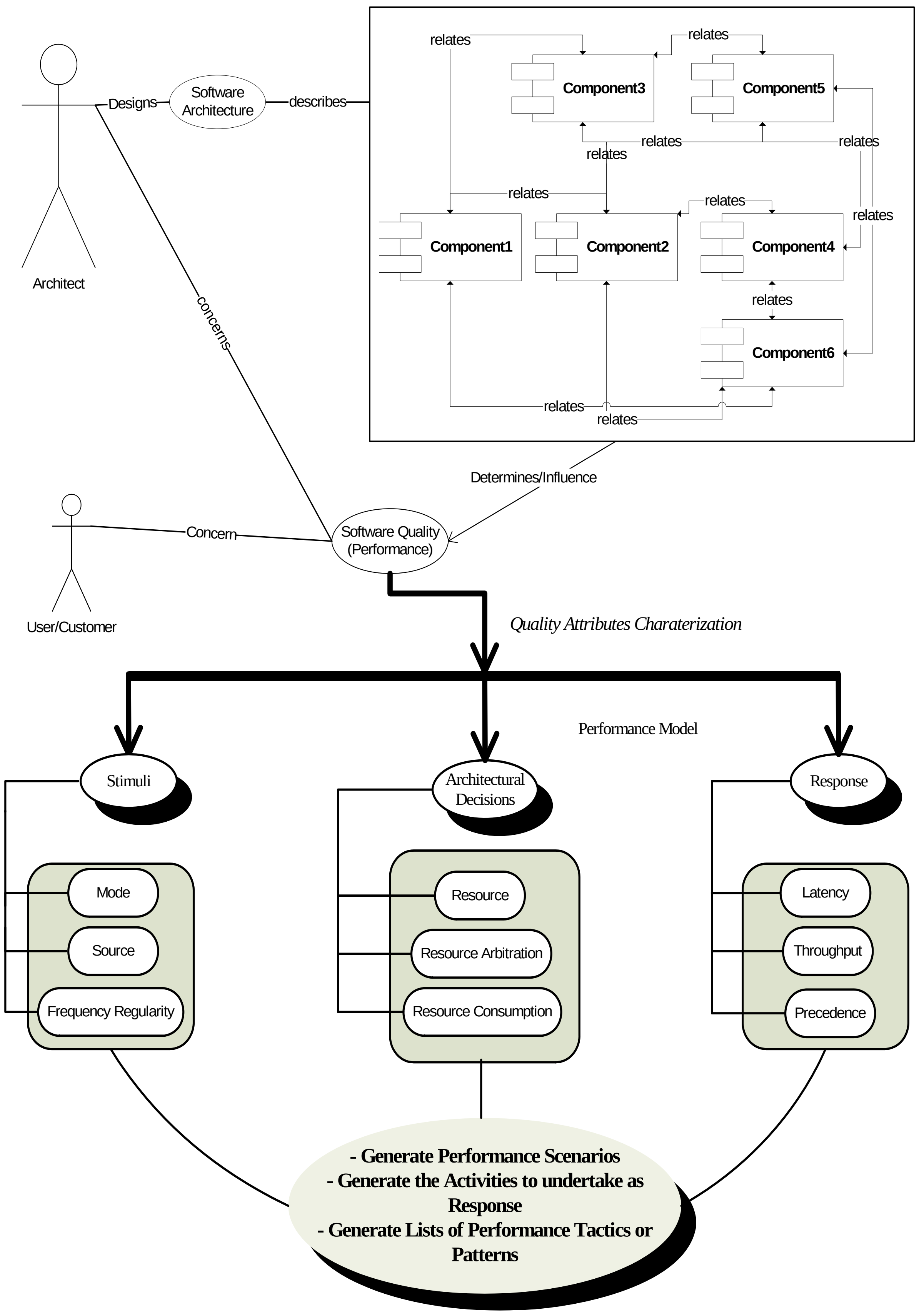
4. THE PERFORMANCE QUALITY MODEL

The model is developed in Unified Modeling Language (UML) using AgroUML Computer Aided Systems Engineering tool. The effectiveness of the model was tested and validated using the software architecture evaluation method/techniques - ATAM (Architecture Tradeoff Analysis Method). The model is based on the identified architectural quality attribute – performance that can be used along side with ATAM for qualitative reasoning given the different views of the architecture and architectural patterns, tactics and quality sensitive scenarios. With the model, the right questions about an architecture can be asked by telling us what artifacts (such as specific design decisions also known as architectural tactics and patterns, which are the tools used) to examine in achieving the quality attribute (performance) with the scenarios generated.

The model is concerned with the performance of an existing system from the architectural perspective. Performance as the major requirement considered is one among other quality attributes that is appropriate for an architecture and it’s often originates from the organization’s business goals. The achievement of this is significant to the success of the system. The proposed Software Architecture Scenario-Based Performance Quality Model is presented below in figure 1:

From the model in figure 1, each quality attribute characterization is divided into three categories: *stimuli, architectural decisions*, and *responses*. *Stimuli* are the events that cause the architecture to respond or change. To analyze architecture for adherence to quality requirements, those requirements are expressed in terms that are concrete and measurable or observable. These measurable or observable quantities are described in the *responses* section of the attribute characterization. *Architectural decisions* are those aspects of architecture—components, connectors, and their properties—that have a direct impact on achieving attribute responses. For example, the stimuli for performance are events such as messages, interrupts, or user keystrokes that result in computation being initiated. Performance architectural decisions include processor and network arbitration mechanisms; concurrency structures including processes, threads, and processors; and properties including process priorities and execution times.

Responses are characterized by measurable quantities such as latency and throughput. When apply to existing software architecture, the model will look at what architectural decisions have direct impact on performance. These decisions are called tactics. The model can then be populated with a list of tactics or pattern and some stimulus/response pair for each defined. This will enable drawing of conclusion about the performance for each. Still on the model, some of the performance quality attributes that can be of interest include;


Architect
Designs
Software Architecture
describes
relates
Component1
Component2
Component3
Component4
Component5
Component6
concerns
Determines/Influence
User/Customer
Concern
Software Quality (Performance)
Quality Attributes Charaterization
Performance Model
Stimuli
Architectural Decisions
Response
Mode
Source
Frequency Regularity
Resource
Resource Arbitration
Resource Consumption
Latency
Throughput
Precedence
- Generate Performance Scenarios
- Generate the Activities to undertake as Response
- Generate Lists of Performance Tactics or Patterns

a. The set of processes (with given deadlines and period) schedulability
b. The maximum, minimum, or average throughput of a transaction-based system, and;
c. The maximum time a particular end-to-end processing chain will take.

All these can be determined from the existing architecture.

5. INTERPRETATIONS AND APPLICATION OF PROPOSED MODEL

The model is concerned about the performance of an existing architecture. In the model the two actors (Architect and User/Customer) are both concern about the software quality. The Architect designs the software architecture, which is a description of the various components of the system in relation. The software architecture determines or influence the software quality. As shown in figure 1 above, performance is one among other quality attributes that is appropriate for an architecture. It originates from business goal. The model is used to asked the right questions about the architecture, telling us what artifacts (such as the specific design decisions) to examine with the questions generated, and then measures the result from those questions. This performance quality attribute is based on the most common quality attributes according to the SEI-ATAM evaluations (Ozkaya et al., 2008). The model is called Software Architecture Scenario-Based Performance Quality Model (SASPQuM). In the Performance model section, the performance has three (3) major concern or things to do. They are:

a. Stimuli
b. Architectural Decisions
c. Response

The ATAM method can be use in testing and validating the model with some scenarios that express the performance requirements. The steps involve in ATAM evaluation is similar with the three part formulation or three major concern considered in the proposed model. The ATAM evaluation method uses a six-part scenario structure: Stimulus source, stimulus, artifact stimulated, environment (e.g., peak load, normal operation), response and response measure. These makes it similar with the three major concern considered in the proposed model. The architectural decisions that will have direct impact on performance are the tactics.

6. CONCLUSION

The major thrust of this paper is to propose a performance quality model for software architecture using qualitative approach. The qualitative approach result in asking questions about architecture by generating qualitative questions using scenarios. This is because quality attributes or requirements of software systems at the architectural level during architectural designs are fundamental issues to the stakeholders (end users, developers etc), and not all stakeholders are vast enough to understand the quantitative approach when it has to do with measurement and rigorous mathematical equation(s) that need to be analyzed and understood. It therefore means that great attention need to be paid on the refinement process of all considered quality attributes and also consideration need to be made by the software engineering community on how well to promote qualitative approach to quality model design for software architecture, most especially on the performance characteristics of any software system.

REFERENCES

[1] Chung, L. Nixon, B. Yu E. and Mylopoulos, J. *Non-functional requirements in software engineering*. Kluwer Academic, Boston, 2000.

[2] Garlan, D., and Shaw, M. (1996). Software Architecture: Perspective on an emerging discipline. *Prentice Hall.*

[3] Clements, P., and Northrop, L. (1996). Software Architecture: An Executive Overview. CMU/SEI-96-TR-003.

[4] Clements, P., and Northrop, L. (2002). Software Product Lines: Practice and Patterns. Addison-Wesley, Boston.

[5] Cavano Joseph P. and McCall James A. A framework for the measurement of software quality. Proceedings of the software quality assurance workshop on Functional and performance issues Pages: 133 – 139: 1978

[6] Boehm, W., Brown, J., and Lipow, M. (1976). Quantitative Evaluation of Software Quality: International Conference on Software Engineering, Proceedings of the 2nd international conference on Software engineering

[7] ISO 1998). International Organization for Standardization, "ISO 9000-3:1998 -- Quality management and quality assurance standards – Part 3: Guidelines for the application of ISO 9001_1994 to the development, supply, installation and maintenance of computer software (ISO 9000-3:1997)".

[8] Ozkaya, I., Bass, L., Sangwan, R., and Nord, R. (2008). Making Practical Use of Quality Attribute Information. *IEEE Software*, vol 25, no.2, pg. 28-31, March/April.

[9] Perry, D., and Wolf, A. (1992). "Foundations for the Study of Software Architecture," *ACM SIGSOFT*

Software Engineering Notes, Vol. 17, No. 4, pp. 40-52.

[10] Barbacci, M., and Klein, M. (1995). Quality attributes. Technical report, 1995.

[11] Denzin, N., and Lincoln, Y. (2002). Handbook of Qualitative Research. In Sale, J. E. M., Lohfeld L. Brazil K. Revisiting the Quantitative-Qualitative Debate: Implications for Mixed Methods Research. Quality and Quantity, 36.

[12] Guba, E., and Lincoln, Y. (1994). Competing paradigms in qualitative research. In Denzin N. K. Y Lincoln Y. S, Handbook of qualitative Research, Thousands Oaks, Sage.

[13] Kazman, R., Clements, P., et al., 1997. Classifying Architectural Elements as a Foundation for Mechanism Matching, Washington, DC.

[14] Chastek, Gary & Donohoe, Patrick. *Product Line Analysis for Practitioners* (CMU/SEI-2003-TR-008). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.

[15] Kazman, R.; Abowd, G.; Bass, L.; Clements, P.; Scenario-based analysis of software architecture Software, IEEE Volume: 13 Issue: 6 p: 47 - 55 ISSN: 0740-7459: 1996