International Journal of Computer Applications (0975 – 8887)
Volume 46– No.23, May 2012

Software Performance Quality Evaluation of MINPHIS Architecture using ATAM

Ishaya Gambo
Obafemi Awolowo University
Computer Science &
Engineering Department
Ile-Ife, Nigeria

Abimbola Soriyan
Obafemi Awolowo University
Computer Science &
Engineering Department
Ile-Ife, Nigeria

Philip Achimugu
Federal Univ. of Agriculture
Computer Science Department
Abeokuta
Abeokuta, Nigeria

ABSTRACT
Software architecture evaluation plays an important role in the validation of quality models of software systems. This paper is based on the research carried out where the Architecture Trade-off Analysis Method (ATAM) was used. ATAM was chosen and used because it provides insight into the way quality attributes are mapped onto architecture and also shows the trade-offs existing between the identified quality and others. The evaluation was based on the developed Software Architecture Scenario-Based Performance Quality Model (SASPUM). The paper presents the results of the analysis with ATAM by providing the set of scenarios and their prioritization from brainstorming, the utility tree, the risks discovered and non-risk documented; the sensitivity points and trade-off points found. The evaluation supports the fact that performance can be identified as a software quality attribute, which is part of the execution model of software system determined by the architecture of the software system, and that is suitable for software architectural evaluation.

General Terms
Software architecture, ATAM, MINPHIS.

Keywords
Software quality, ATAM, Software architecture, MINPHIS, SASPUM

1. INTRODUCTION
Software architecture and software quality are important subjects in the emerging discipline of software engineering, and it is concerned with improving the approach to software quality. Software systems' quality can be equal to "error free" system. In today's software development process, quality requirements during architectural design and decision are fundamental issues to the stakeholders (developers, analysts, programmers, users etc.). Even though the software engineering community has paid closer attention and done a lot of work around software architecture in recent years, it is still an evolving area. The software engineering community has not really looked at software quality thoroughly most especially from the perspective of software architecture. Till date it is still like news to many involved in software development processes and practices, most especially in the developing nations like Nigeria. They are more particular about the functionality of the system, and never bothered about the non-functional attributes which are more paramount to the stakeholders (end-users most especially). The challenge in software development is to develop software with the right quality levels. The main problem is not to know if a project is technically feasible concerning functionality, but if a solution exists that meets the software quality requirements. The problem resulting from this is if the architecture determines the quality of the system, do existing systems follow any specific architecture for the quality to be determined at all times? If architectural decisions determine a system's quality attributes, what is the possibility of evaluating architectural decisions with respect to their impact on those attributes?

However, we observed that within the engineering practices, the architectural design is very crucial in the attainment of quality goals. Quality issues have been a fundamental focal target in the development of most information systems. This is because most stakeholders involve in the usability of information systems are particular about quality in terms of functionality, ease of use and satisfaction. In this sense, we are of the opinion that this quality attributes of software systems can be highly constrained by a system's software architecture. The work in [1] opined that "the importance of the right software architecture to a development effort is widely recognized" and [2] have done justice to establish the prominent role software architecture plays in the overall system quality. Thus, it is good to determine the time a system's software architecture is specified whether the system will have the desired qualities and whether it will follow any specific architecture for the quality to be determined at all times. To do this, an architectural evaluation will be necessary to validate such.

Furthermore, the work in [3] 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 that have been typically performed in isolation. However, being able to evaluate the quality of software is very important, not only from the perspective of a software engineer, but also from a business point of view in order to determine the level of the provided quality. Some of the analysis and evaluation techniques used to achieve this includes: the SAAM (Software Architecture Analysis Method) [4], the QAW (Quality Attribute Workshop) [5], the ADD (Attribute Driven Design) method [2], ATAM (Architecture Trade-off Analysis Method) [6] and the CBAM (Cost Benefit Analysis Method) [7] among others. This paper addresses quality issues at the architectural level by using ATAM. The paper is based on the research carried out to evaluate the architecture of the Made in Nigeria Primary Healthcare Information System (MINPHIS) currently running in some tertiary and specialist hospitals in Nigeria. The choice of ATAM for the evaluation was because it provides insight into the way quality attributes are mapped onto architecture and also shows the trade-offs existing between the identified quality and others. ATAM was therefore used to provide the set of scenarios and their prioritization from brainstorming, the utility tree was provided, the risks were discovered (the risks are alternatives that might create future problems in some quality attributes), the non-risks were documented, the sensitivity points and trade-off points were found. Sensitivity here are alternatives for which a slight change makes a significant difference in a quality attribute, while trade-offs are decisions affecting more than one quality attribute.

2. MOTIVATION

Because software architecture is a major determinant of software quality, it follows that software architecture is critical to the quality of any software system. For the Made in Nigeria Primary Healthcare Information System (MINPHIS), the ability to know whether the system conforms to the existing architecture for the quality to be determined at all times is very important. In the work of [8], the performance attribute of MINPHIS was considered. Their work approaches performance issues qualitatively and developed the performance quality model called Software Architecture Scenario-Based Performance Quality Model (SASPUM). [8] noted that "the validity of the model characterized into three categories: stimuli, architectural decisions, and responses, can be tested on any existing software architecture using PASA (Performance Assessment of Software Architecture) and ATAM". Obviously, the achievement of quality attributes is critical to the success of a system. Therefore, it would then make sense to evaluate the architecture of a system in order to ensure that it is going on the right track.

3. MADE IN NIGERIA PRIMARY HEALTHCARE INFORMATION SYSTEM AND ARCHITECTURE

MIHPHIS is a software package developed within the INDEHELA projects (Methods for Informatics Development for Health in Africa) in Nigeria and has been utilized in a number of Nigerian Teaching and Specialist Hospitals. It is principally a hospital patient information system. The development of MINPHIS started back in 1989, by a doctoral student from the University of Kuopio in Finland who visited Computer Science and Engineering Department Obafemi Awolowo University in Nigeria as a researcher. A very rudimentary hospital information system, running on a stand-alone PC, was then jointly developed. MINPHIS was developed as a joint project between University of Kuopio in Finland (UKU), Computer Science and Engineering Department, OAU (Obafemi Awolowo University) and OAUTHC (Obafemi Awolowo University Teaching Hospital Complex) as a test bed. The system was originally installed on a PC server with 3 dumb terminals in 1991. The 2nd generation of the system implemented in 1998 was based on more powerful servers running Microsoft NT, Intersystems Cache, the VA Kernel and FileMan, and the FixIT software developed in Finland. MINPHIS has spanned through a thorough Information System Development Process with clinical and patient information well taken care of via a wide range of reports that could aid health policy and decision makers.

Today, one of the hospitals currently using the system is the OAUTHC. At the OAUTHC, MINPHIS is used as the application system for medical record and other health information exchange processes. During the research, MINPHIS was used to make investigation on how it is being used at the OAUTH. The investigation of the system (MINPHIS) also serves as the requirements of OAUTH for the redesign of the system, while the MINPHIS-enabled work system(s) in OAUTH was considered as the case studies in generating the scenarios.

The MINPHIS architecture in "figure 1" is a 2-tier architecture. There are four layers, separated from each other by well-defined interfaces depicted by dotted lines. As a 2-tier architecture, it consists of data server (i.e. the FileMan database and the M software, the legacy system on MINPHIS, which can access the database directly) and the client application. The database server is where the database serves up data based on queries submitted by the application using the hierarchical database system as the case is with MINPHIS, while the application on the client computer consumes the data and presents it in readable format..

4. RESEARCH DESIGN AND METHODOLOGY

Relevant information on the relationship between a software architecture and software quality, and requirements of a system and its architecture were studied. It was discovered that addressing performance quality attribute among others is crucial for the system (MINPHIS), and this justifies the fact that it is the most common quality attributes according to the SEI-ATAM evaluations [9]. Different software architectural views were studied and the Krutchen"s 4+1 Views model with the different views were adopted and used to provide the basis for reasoning about the appropriateness and quality of the architecture in achieving the system quality goals on the developed quality model. The software architectural views, tactics and design patterns were broken down into their structural parts with the aim of providing the needed architectural quality for evaluation. In this regard, a module view of MINPHIS architecture was shown, the refinement process for performance quality characteristics was shown as reported in [8], and the performance tactics were generated. The scenario-based approach to analyze and evaluate software architecture was used after a thorough review of various analysis and evaluation methods. The Architecture Trade-off Analysis Method (ATAM) was chosen and used because it provides insight into the way quality attributes are mapped onto architecture and also shows the trade-offs existing between the identified quality and others.

During the research, the OAUTH was visited six (6) times to make investigation into how the system (MINPHIS) functions and behaves during execution. The investigation was carried out using interview and direct observation of the system at run time. From the behavior of the system and how it functions, scenarios were generated. Other scenarios were generated by the stakeholders (these includes the system administrator, system analyst, programmer, requirement engineer, and medical record officers as users) based on the business requirements of the system. All these were carried out with a view to ensuring that existing software systems (for example MINPHIS) follow a specific architecture for the quality to be determined exactly at all times.

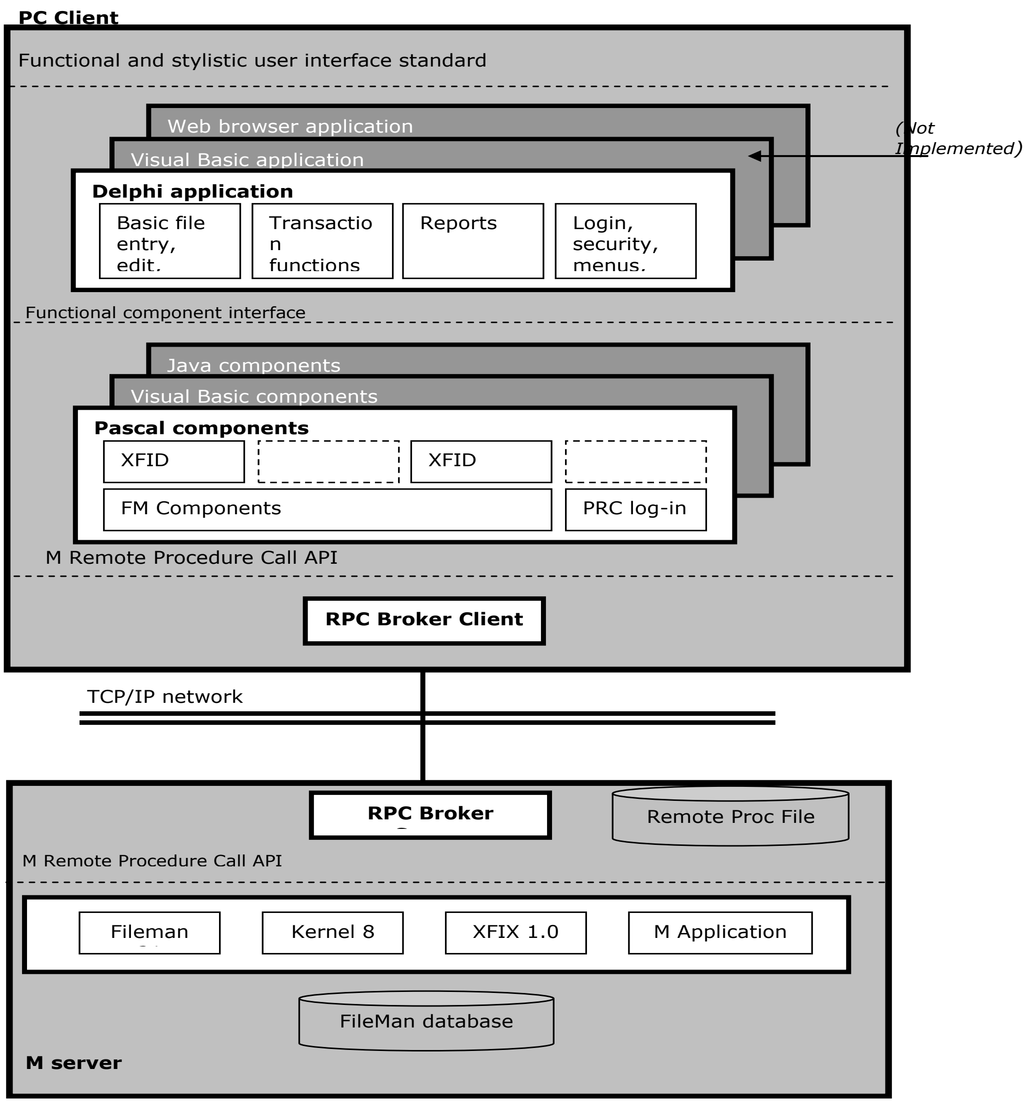

Figure 1 : MINPHIS Architecture (Source: [10])

5.EVALUATION OF THE MINPHIS ARCHITECTURE

ATAM is a scenario-based and model-based analysis technique for software architectures that analyses a software architecture with respect to multiple attributes and explicitly considers the trade-offs inherent in the design. The goal of ATAM is to learn where a quality characteristic of interest is affected by architectural design decisions, so that careful reasoning about those decisions is possible in order to possibly model them more completely in subsequent analyses. According to [6] "ATAM process is organized around the idea that architectural styles are the main determinants of architectural quality attributes". ATAM has two main phases, each of which consists of several intermediate steps. Totally, the method contains nine steps [5, 11]. In support of [12], ATAM requires a software architecture (SA) documented with different views. During the evaluation with ATAM, the SASPUM in [8] was used as the quality model. ATAM is considered a mature approach, as it has been validated in different domains. The purpose of ATAM to MINPHIS is to assess the consequences of MINPHIS architectural decision alternatives in light of the performance quality attributes. The "figure 2" shows the phases and corresponding steps in of ATAM.

The evaluation of the MINPHIS architecture focused on determining the performance quality attribute. This quality was identified among other non-functional qualities as important characteristics for the system's stakeholders. The purpose of MINPHIS application is for keeping and handling electronic patient records and generating various reports for health management and research purposes. The reports include the patient status, medical history and admissions plus indicators like length of stay per patient, discharge summaries, mortality and morbidity data, and operations. So, the important scenario is the time and resource behaviours. At the start of the evaluation process, the present MINPHIS system was considered and seen as a large system expected to comprise several thousand lines of codes and is the third version.

.

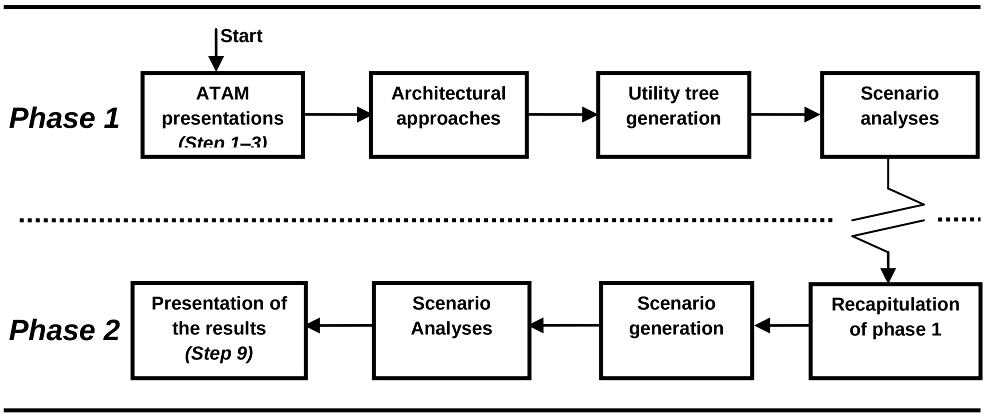

Figure 2: The phases and steps in ATAM

The system is fully implemented and running in some hospitals in Nigeria. The system is serving as the information backbone for the health care institutions in which it was installed. It provides data about patients' treatment history as well as tracking their insurance and other payments (i.e the billing system). The system produces a large number of on-demand and periodic reports, each tailored to the institutions or hospitals specific needs. At this phase of evaluation, the evaluator partners with the developers and end users of the system in order to gather detail information needed for other phases of the evaluation process. The evaluator served and worked as the team leader, evaluation leader, questioner, timekeeper, scenario scribe, data gatherer, process enforcer, proceedings scribe and process observer. Three among the developers of the system were contacted for detailed information about the system. A one-day discussion was held with one of the developer to give an overview of the system from the architectural point of view on how the system behaves during execution. The following steps of ATAM were strictly followed.

5.1 Step 1: Present the ATAM

ATAM steps and expected outputs were outline and explained to the team. Here, the process that everyone will be following to answer questions and to set the context and expectations for the other activities were explained by the team leader.

5.2 Step 2: Present business drivers

At the evaluation, one of the system stakeholders (a developer) presented the business objectives for the MINPHIS system from the development point of view, as well as from the viewpoints of end users from the different hospitals using the system. Some of the MINPHIS application business requirements that were addressed include:

- Keeping patient records
- Answering ad hoc queries from medical researchers, end users (e.g. cases of cholera for a period per geographical location for specific age group or sex or both).
- Providing performance information relevant to particular health care professionals, such as the mortality rates for patients treated by a particular staff member, as well as number of patients attended to by particular medical staff.
- Resource management decisions, by improving the understanding of indicators such as the number of consultations per day handled by medical professionals, the number of patients per ward, and the number of professionals who fails to write discharge summaries for their patients, etc.
- Generating various reports for health management and research purposes. The report the system generates include the patient status, medical history and admissions plus indicators like length of stay per patient, discharge summaries, mortality and morbidity data, and operations.
- Creation of a new version of the system (e.g., to be web enabled, include web services, technologically compliant with current needs etc.) that the development organization could market to customers and other health/hospital institutions (teaching, private and public/government hospitals).

For the end users of the system which include but not limited to: Obafemi Awolowo University Teaching Hospital (OAUTH), Ladoke Akintola University of Technology Teaching Hospital (LAUTECH-TH), the MINPHIS system was to replace the manual system of handling Hospital/Health Management and Information Systems (HMIS), which were observed to be difficult to run, operate and maintain as well as unresponsive to the current and projected business and management needs of the healthcare practices. At the point the end user's business requirement grew considering the current trend of technological advancement and availability of more software tools that can be used to improve the architecture and system. Some of these business requirements are:

- the ability of the system to deal with diverse cultural and regional differences.
- the ability of the system to deal with multiple languages (especially Foreign, English and other local dialect like Yoruba, Hausa and Igbo as the 3 main language in the country) and currencies (especially foreign currencies for foreigners who might want to transact with the system when it has been deployed on the web platform).
- a new system at least as fast as any legacy system being replaced.

- an improved web based system with web services.
- a new single system combining distinct legacy financial management systems.

However, the current MINPHIS system architecture needs to be re-engineered in order to accommodate the above requirements.

5.3 Step 3: Present architecture

During the evaluation and interactions with the developers who happen to be the architect, before as well as during the evaluation exercise, several views of the architecture and the architectural approaches emerged.

5.4 Step 4: Catalog architectural approaches

The MINPHIS architectural approaches identified include:

- layering, especially the FileMan, RPC Broker Server and readymade components.
- cache object orientation.
- client-server transaction processing.
- a data-centric architectural pattern, with a FileMan database at its heart.

These and other approaches gave the evaluation a conceptual footing from which to begin asking probing questions when scenario analysis began.

5.5 Step 5—Generate quality attribute utility tree

In this step, performance quality attribute of the MINPHIS system was considered, identified, prioritized, refined and used to show how it has been affected in particular scenarios. From the performance attribute selected, one or more specific descriptions and scenarios were produced. Each scenario is classified according to their priority on importance and difficulty. Consequently, "table 1" shows the performance quality attribute, the attribute refinement and scenarios. These scenarios are generated for the stakeholders: the end users, the architect and the application developer. The scenarios in "table 1" are annotated with the priority rankings assigned by the decision makers of the system present. The first of each ordered pair indicates the importance of the capability; the second indicates the architect's estimation of the difficulty in achieving it. "Figure 3" shows the performance utility tree. The utility tree contains 'utility' as the root node, with the performance quality attribute forming the secondary level of the utility tree. The prioritization in the utility tree is based on relative rankings: High (H), Medium (M) and Low (L). The utility tree contains utility as the root node, which shows the overall "goodness" of the system. Typically, the quality attribute performance is the high-level node immediately under utility. Performance is broken down into "data latency", "transaction throughput" and "transaction response time". This is a step towards refining the attribute goals to be concrete enough for prioritization. Latency and throughput are two of the types of response measures noted in the attribute characterization as described in [8]. Data latency was refined into "Minimize storage latency on patient database" and "Generate patient report within 10ms". Transaction response time is refined into "A user updates a patient's account in response to a change-of-address notification while the system is under peak load, and the transaction completes in less than one second" and "A user updates a patient's account in response to a change-of-address notification while the system is under twice the current peak load, and the transaction completes in less than 4 seconds), while throughput is refined into "At peak load, the system is able to complete 150 normalized transactions per second". This is meant to maximize average throughput to the authentication server".

Table 1. Tabular form of the utility tree for the MINPHIS ATAM

Quality Attribute	Attribute Refinement	Scenarios
Performance	Transaction response time	A user updates a patient's account in response to a change-of-address notification while the system is under peak load, and the transaction completes in less than 1 second. (H,M)
		A user updates a patient's account in response to a change-of-address notification while the system is under twice the current peak load, and the transaction completes in less than 4 seconds. (L,M)
	Throughput	At peak load, the system is able to complete 150 normalized transactions per second. (M,M)

From the utility tree "Minimize storage latency on patient database" has priorities of (M, L), meaning that it is of medium importance to the success of the system and low risk to achieve, while "Generate patient report within 10ms" has priorities of (H, M), meaning that it is highly important to the success of the system and achievement of this scenario is perceived to be of medium risk. "A user updates a patient's account in response to a change-of-address notification while the system is under peak load, and the transaction completes in less than 1 second" has priorities (H, M), meaning that it is highly important to the success of the system and achievement of this scenario is perceived to be of medium risk. "A user updates a patient's account in response to a change-of-address notification while the system is under twice the current peak load, and the transaction completes in less than 4 seconds" has priorities (L, M), meaning that it is of low importance to the success of the system and medium risks to achieve. "At peak load, the system is able to complete 150 normalized transactions per second" has priorities (M, M), meaning that it is of medium importance to the success of the system and is perceived to be of medium risk.

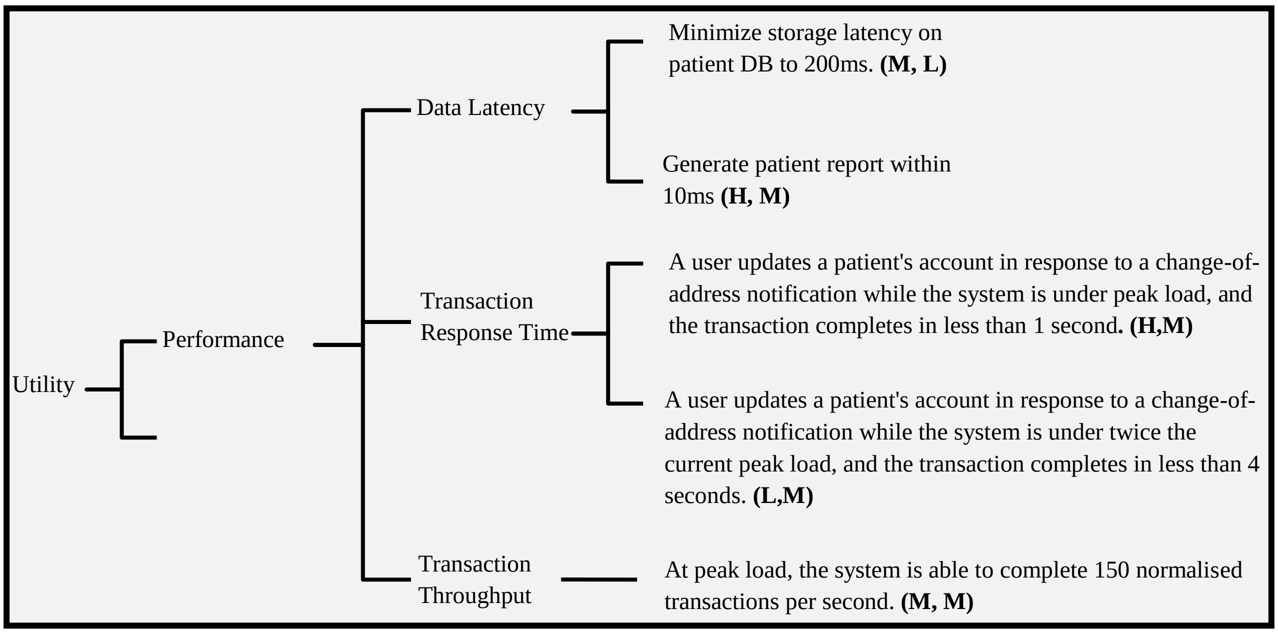
Utility
Performance
Data Latency
Minimize storage latency on patient DB to 200ms. (M, L)
Generate patient report within 10ms (H, M)
Transaction Response Time
A user updates a patient's account in response to a change-of-address notification while the system is under peak load, and the transaction completes in less than 1 second. (H,M)
A user updates a patient's account in response to a change-of-address notification while the system is under twice the current peak load, and the transaction completes in less than 4 seconds. (L,M)
Transaction Throughput
At peak load, the system is able to complete 150 normalised transactions per second. (M, M)

Figure 3: Performance utility tree

5.6 Step 6: Analyze architectural approaches

In this step, the architectural approaches analyzed were related to the scenarios and their rankings. The utility tree exercise produced no scenarios ranked (H, H), which indicates high-importance, high-difficulty scenarios that merit high analytical priority. The (H,M) scenarios was targeted, a cluster of which appeared under "Data Latency" and “Transaction Response Time” hypothesizing the generation of patient report within 10ms (H, M) and updating a patient’s account in response to a change record while the system is under peak load, and the transaction complete in less than 1 second. (H,M). From the prioritization of these scenarios, shown by the (M, L), (H, M), (H, M), (L, M) and (M, M) beside the scenarios, it is decided that the architectural quality attribute - performance is important to the system. The second and third scenario is chosen because it is of high importance to the success of the system, and of medium level of difficulty to the stakeholders. The fourth scenario is not considered because it is of low importance to the system.

5.7 Step 7: Brainstorm and prioritize scenarios

At the level the some of the stakeholders were actively engaged and they were so productive and resourceful. They contributed to the about 5 scenarios in this step. These scenarios are the ones at the leaves of the step 5's utility tree but were not analyzed. At this point, the stakeholders expressed their views on the fact that some scenarios deserved more attention as in steps 1 and 3 above. Some of the selected scenarios are outlined in “table 2”:

Table 2. Lists of selected scenarios resulting from step 7.

Number of event	Scenario
1	Data in the database is replicated to another department, and performance is degraded
2	Decide to support the local dialet (e.g .Yoruba and Hausa).
3	Add an NHIS service and supporting functionality.
4	MINPHIS is installed in a hospital, and the hospital's existing database must be converted.
5	A report needs to be generated using information from two hospitals that use different configurations.

Some of the prioritized list of brainstormed scenarios is compared with the prioritized scenarios obtained from the utility tree in step 5. At this point, three sessions of brainstorming were held using different scenarios at each session. This includes: (1) Use case scenarios, where the stakeholder is the end-user, (2) Growth scenarios, which represents the way in which growth in architecture is perceived and (3) Exploratory scenarios, which represent extreme forms of growth in the architecture.

5.8 Step 8: Analyze architectural approaches

In step 8, the risk, non-risk, sensitivity and trade-off points were identified. Considering the performance quality attribute, the performance factor is expressed as number of transactions per unit time or the time taken to perform one transaction. This property enables the understanding of the responsiveness of the system. At this level we sub-divided this step into four stages consisting of:

a) **Investigation of architectural approach**
The MINPHIS architecture is command driven at the interface level, so the performance of the entire system cannot be measured as number of transactions per unit time.

b) **Creation of analysis question**
At this level we had the following questions for analysis: (i) Does the architecture process any task in the fastest possible speed? (ii) How are priorities assigned to processes? (iii) What are the message arrival rates? And (iv) What are transaction processing times?

c) **Risk and Non-Risk**
The decision to keep backup is a risk if the performance cost is excessive, while decision to keep backup is a non-risk if the performance cost is not excessive.

d) **Sensitivity and Trade-off points**
The average speed at which the task is performed is sensitive to the number of components involved in processing the task. We arrive at the at the trade-off point of number of components involved in processing the task. It was also seen that keeping the backup database affects performance also. So, it is a trade-off between reliability and performance.

Conclusively, evaluating the system showed that the database system need to be migrated from hierarchical to relational structure. This convinced the evaluator that a well-thought-out procedure was in place, with known strengths and reasonable limitations.

5.9 Step 9: Analyze architectural approaches

This is the final step of the ATAM evaluation. The information collected during the evaluation was presented. The main findings of the ATAM evaluation include: A utility tree, Set of generated scenarios, Set of analysis questions, Set of identified risks and non-risks, and the identified architectural approaches.

6. CONCLUSION

It may be argued that software quality is part of the execution model of software system which is determined by the software architecture. So, quality control and management must be carried out through the whole development process of software systems to ensure the implementation of required quality characteristics. More so, the importance of software architecture to support the required quality characteristics has been recognized many times by many different people. Since quality is in the eye of the beholder, it then means more work is needed to show the relationship that could exist among the different perspectives in future analysis of software systems from the architectural perspective.

7. ACKNOWLEDGMENT

I will like to appreciate Paul C. Clement (Ph.D) who was formally at the Software Engineering Institute (SEI), Carnegie Mellon University (CMU), Pittsburgh in USA for comments, research materials and guidance during the period of my research from which this paper is written. Your hospitable spirit really gave me confidence to come closer and to have more confidence in the subject area of software architecture and software quality research. You made me feel proud of what I have done and what I will yet do.

8. REFERENCES

[1] Kazman, R., Bass, Len., and Klein, M. 2006. The essential component of software architecture design and analysis. The Journal of Systems and Software 79 (2006) 1207-1216.

[2] Clements, P., Kazman, R., and Klein, M. 2002. Evaluating Software Architecture: Methods and Case Studies: Addison-Wesley, Boston, MA.

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

[4] Kazman, R., Abowd, G., Bass, Len., Webb, M. 1994. SAAM: a method for analyzing the properties of software architectures. In: Proceedings of the 16th International Conference on Software Engineering. Sorrento, Italy, May 16-12, 1994. IEEE Computer Society, Los Alamitos, CA pp.81-90.

[5] Barbacci, M., Ellison, R., Lattanze, A., Stafford, J., Weinstock, C., and Wood, W. (2003). Quality Attribute Workshops (QAW), third ed. (CMU/SEI-2003-TR-016), third ed. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

[6] Kazman, R., Barbacci, M., Klein, M., Carriere, S., 1999. Experience with performing architecture tradeoff analysis. In: Proceedings of the 21st International Conference on Software Engineering, May 1999, Los Angeles, CA, pp. 54-63.

[7] Kazman, R., Asundi, J., Klein, M., 2001. Quantifying the costs and benefits of architectural decisions. In: Proceedings of the 23rd International Conference on Software Engineering, May 2001, Toronto, Canada, pp. 297-306.

[8] Gambo, I., Soriyan, A., and Achimugu, P., 2011. Software Architecture Performance Quality Model: Qualitative Approach. ARPN Journal of Systems and Software, Vol.1. April 2011, pp. 28-33.

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

[10] Korpela, M. 1990. The Ife Project: Report 1989. Health care informatics in the context of a developing country. University of Koupio, 1990.

[11] Kazman, R., Klein, M., and Clements, P. 2000. ATAM: A Method for Architecture Evaluation. Technical Report CMU/SEI-2000-TR-004, Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA.

[12] Bass, L., Clements, P., and Kazman, R. (2003). Software Architecture in Practice, second ed. Addison-Wesley, Reading, MA.