Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol 3, No.7, 2012
www.iiste.org

The Critical Need for Software Architecture Practices in Software Development Process

Ishaya Gambo[1*] Rhoda Ikono[2] Olaronke Iroju[2] Theressa Omodunbi[1]

1. Computer Science and Engineering Department, Obafemi Awolowo University, Ile-Ife, Nigeria
2. Computer Science Department, Adeyemi College of Education, Ondo State, Nigeria

* E-mail of the corresponding author: ipgambo@gmail.com

Abstract

Software architecture is the master plan of every reliable software system. It is the building block of any kind of software system which greatly determines the success of the system. This paper argues that every system needs a good architecture and that requires the use of good architecture engineering practices in a software development process. The paper recognized software architecture practice as a discipline pervading all phases of software development and then identifies some of the pertinent areas where architectural practice can be used based on a framework. In addition a model showing how software architecture fits into the phases of a generic software development process lifecycle was presented. The model is to enable software developers and acquirers to use effective software architecture practices during software development in order to exert significantly greater control over software product qualities.

Keywords: Software architecture, Software Development, Software, Quality, Stakeholders, Software engineering

1. Introduction

With a view to finding a lasting solution to problems emanating from software complexities so as to ensure the delivery of high quality systems, provide strong management support, maximise available resources, enhance stakeholders' communication and increase productivity; a dynamic design approach is needed. For decades, software designers have been taught to build systems based exclusively on the technical requirements. Conceptually, the requirements document is tossed over the wall into the designer's cubicle, and the designer must come forth with a satisfactory design. Consequently, requirements are meant to beget design, which in turn begets the system. However, this practice still poses a lot of complexity on software systems. According to Bass et al. (2003), "modern software development methods recognize the naïveté of this model and provide all sorts of feedback loops from designer to analyst". Even with this notion, they still make the implicit assumption that design is a product of the system's technical requirements. These technical requirements can be architecturally represented where the relationship of the various components and elements can be shown.

In practice, effective software engineering requires facility in architectural software design. Architectural practices have emerged as a crucial part of the design process, which is critical to any software intensive system. The architecture serves as building block of any kind of software system and determines the degree of success of a system. Software architecture in software engineering (SE) practices was introduced to help manage complexity and risk during development and maintenance of software systems. It guides and documents the structure of the system as well as the role each system's components play within the structure. It embodies the earliest software design decisions that enables or preclude the achievement of a desired system quality, such as reliability, modifiability, security, performance, testability and usability etc. The architecture also forms the basis for the development approach and acts as the blueprint that guides the teams building the system. Architectural decisions are the most critical to get right and the most difficult to change downstream in the development life cycle. It is the result of assembling a certain number of architectural elements in some well-chosen forms to satisfy the major functionality and performance requirements of the system as well as some other non-functional requirements such as the quality attributes listed above and described in Chung (2000). According to Clements et al. (2002), "software architecture is

developed as the first step toward designing a system that has a collection of desired properties". Perry and Wolfe (1992) put it very nicely in this formula "(Software architecture = {Elements, Forms, Rationale/Constraints})". This was modified by Boehm (1995) with the opinion that "software architecture deals with abstraction, with decomposition and composition, and with style and aesthetics."

However, the role software architecture plays in the overall system quality has been established by Clements *et al.* (2002). If the architecture is not right, the system will not meet its requirements. Software architecture has been identified as an increasingly important part of software development. With the architecture, software developers are able to define the internal structure of any system. The requirements towards software systems usually go beyond the correct functionality, the presence of certain quality demands are also very essential for the systems' acceptance by the stakeholders and users. All these are the promises from architectural practices in a software development process. Therefore, in this paper we recognized software architecture as the master plan of every reliable software system, which greatly determines the success of the system. The paper opined that every system needs a good architecture and that requires the use of good architecture engineering practices in a software development process. We also recognized that software architecture practice as a discipline should be integrated into all phases of software development in a generic software process lifecycle.

2. Software Architecture Practice

To practitioners, researchers, educationist and/or academia in industries and government, software architecture is an area of growing importance. The April 1995 issue of IEEE Transactions on Software Engineering and the November 1995 issue of IEEE Software were all devoted to software architecture. Industry and government working groups on software architecture are becoming more frequent. Workshops and presentations on software architecture are beginning to populate software engineering conferences. Most science and technology based reputable journals always welcome contributions on software architectural issues. Another example is the October 1996 ACM Symposium on the Foundations of Software Engineering, which strictly had a focus on software architecture. Today, software architecture practice is one sub-discipline within software engineering that is concerned with the high-level design of the software of one or more systems. Software architecture are created, evolved, and maintained in a complex environment. In Bass et al. (2003) we have the architecture business cycle illustrated as shown in figure 1 below. From the left we have different factors that influence a software architecture through an architect. We also have from figure 1 the factors that are formed by requirements which are gotten from the stakeholders and developing organization.

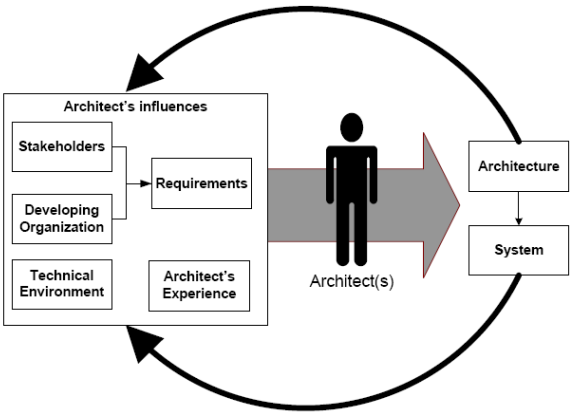

Figure1. The architecture business cycle (Source: Bass et al., 2003)

The requirements clearly states what the system is expected to achieve, and the architect need to capture this by making sure the architecture defines how it could be achieved. From the business lifecycle, we have a feedback loop where the perceptions of the system and architecture influences are conveyed to the stakeholders. With this, the architecture is able to serve as a communication vehicle among stakeholders. The feedback loop is depicted in figure 1 with the arcing arrows from the system and architecture to the stakeholders' influences. The architecture business

lifecycle shows that the architecture can be used;

a) to outlined a design for the software system
b) to plan ahead the evolution of the software system, where it can be used as part of a technology roadmap
c) to communicate major decisions among stakeholders, which can influence the software system
d) to decompose software into smaller parts, which enables developers and/or engineers to work comparably on the software.
e) to predict the quality of the system

3. Software Development Process and Architectural Design

Soriyan (2004) opined that "software development process is a transformation process executed by series of agents (people and machine) which perform a variety of activities and whose association results in the production of a software system. Software development process is an engineering practice that is concern with the analysis, design, deployment/implementation and maintenance of software system using a framework and/or model with a view of sociotechnical influences. This further support the idea by Soriyan (2004) that "the wide use of software systems has made software engineering an integral part of human existence". As an engineering process practice, it traditionally focuses on the practical means of developing software (Warsta, 2002). Salo (2006) emphasizes that software development process is aimed at providing the technological framework for applying tools and people to the software task. The essence of systems development process include ensuring that high quality systems are delivered, providing strong management controls over the projects, and maximizing the productivity of the development team (Bender, 2003). However, in today's software development process for example, quality requirements is a fundamental issues to the stakeholders (developers, users, architect, analyst etc), and it is in our opinion that this can be determined during architectural design and decision. For example, developers and acquirers of software systems need their systems to be modifiable and perform predictably. They also need them to be secure, interoperable, portable, usable and reliable. This quality attributes depend on choosing the correct software architecture. As systems become larger and more complex, software architecture takes on an even more important role in software development. Thus, software architecture drives software development throughout the life cycle. In literature different development process model exists such as the Waterfall, Prototype, Agile, Incremental, Iterative, and Spiral models etc. In the phases of each process model in the development cycle, software architecture can come in between the requirement and design phases, and then we can also establish it roles in each phase of the development process. This is shown in figure 3 where software architecture is integrated into all phases of the lifecycle. In the model we note that during the requirements, an architecture may be used to identify, prioritize, and record system concerns and desired. During design and analysis, an architecture may be used to model, visualize, and analyze design decisions chosen to address the principal concerns and achieve the desired qualities. Decisions may be guided by adopting one or more architectural styles. During implementation and testing, an architecture may be used to drive testing, instantiate a product, support runtime dynamism, or enforce security policies. Rather than thrown out an architecture at this point, as is often done, an architecture remains part of the product. During maintenance, an architecture may be used as a basis for incorporating new features, or increasing modeling detail.

At the software development process level, the architectural design can be determined based on what the architecture is use for. The criterion to determine this whether it is a component, or a relationship between components, or a property (of components or relationships) that needs to be externally visible in order to reason about the ability of the system to meet its quality requirements or to support decomposition of the system into independently implementable pieces.

4. Utilizing Software Architecture Practice

The description of software architecture has taken different views which can be utilized for several indispensable tasks during the development process of any system. Consequently, modern approaches to software architecture have taken a multi-view approach. However, some literatures like Kazman *et al.* (2003), Kruchten (1995), and Hofmeister *et al.* (1995) agree at a point that for a software architecture description, different views are necessary for describing

the different aspects of a software system. Among these views are the description of the static organisation of the software system and the development environment as well as the description of the hardware architecture. A view is the representation of one or more structures present in the system. The use of multiple views allows to address separately the concerns of the various 'stakeholder' of the architecture, and to handle separately the functional and non-functional requirements, which are referred to as software quality. A typical example of this could be the architecture of a building where various stakeholders (such as the construction engineer, the plumber, and the electrician) all have an interest in how the building is to be constructed. Although they are interested in different components and different relationships, each of their views is valid. Each view represents a structure that maps to one of the architecture of the construction goals of the building, and these multiple views are necessary to represent the architecture of the building fully. Similarly, a software architecture has a variety of stakeholders, including developers, maintainers, testers, integrators, system administrators, project managers, analysts, certification authorities, and end users. Each of these stakeholders has a vested interest in different system properties and goals that are represented by different structural views of the system. The views provide the basis for reasoning about the appropriateness and quality of the architecture for achieving system quality goals. The views also address one specific set of concern using several concurrent views.

Furthermore, the architecture can be utilized in the aspect of defining what exactly has to be developed. This makes it a guideline and a means of control for the system development. In terms of relating quality issues of a system, the architecture can be used as a reference tool for evaluation. This makes it possible to use the architecture for quality control and assurance. Further utilization of the architecture should be primarily engineered by the architects who codify the design decisions by providing a machine-readable design artifact. The design artifact when imagined can be used to enhance communication and understanding by providing different perspective for interaction among stakeholders.

5. Pertinent Areas of Software Architecture Practices

In the conceptual framework shown in figure 2, we established a flow of possible areas where software architecture practice can be used. As shown in the framework, the architect designs the architecture of the system, which has to satisfy some quality requirements from the stakeholders' perspective. Secondly, the architect will need to design the system to meet the quality requirements of stakeholders for the system. Thirdly, the system need to be evaluated using the architecture as the reference tool. On the evaluation for quality issues, the question of "what method" and "on what model" arises. From literary source, there exists lots of architecture evaluation methods and/or tools such as; Software Architecture Analysis Method (SAAM) by Kazman et al. (1994), Architectural Trade-off Analysis Method (ATAM) by Kazman et al. (1998), Scenario-Based Architecture Reengineering (SBAR), Architecture Level Prediction of Software Maintenance (ALPSM), A Software Architecture Evaluation Model (SAEM), Architecture Level Analysis Method (ALMA), Family Architecture Assessment Method (FAAM), Cost-Benefit Analysis Method (CBAM), Active Reviews for Intermediate Designs (ARID), PASA (Performance Assessment of Software Architecture), QAW (Quality Attribute Workshop) and SALUTA (Scenario-based Architecture Level UsabiliTy Analysis), which can be used for evaluation purposes. Following the framework in figure 2, we deduced some of the pertinent areas of software architecture practice which include:

- Development of software system using architecture-based development methodologies and design tools. This includes; the styles, patterns, and tactics.
- Evaluating, analyzing and/or assessing software architecture for suitability, conformance or fitness of purpose. To support this, the work of Jan Bosch in Bosch (2000) gave three types of architecture assessment methods. This includes; Scenario based assessment, Simulation and mathematical modeling. All of these methods can be used in practice.
- Predicting of quality attributes of systems to satisfy the developed system and meet the quality requirements of stakeholders. This can be done using software quality models by relating it at the architectural level. An example is the relationship between software architecture and testing. This gives an idea into the investigation of testability. Another example could be investigating the usability of the system from the architectural perspective. This is because quality attributes of a software system are to a large extent

determined by a system's software architecture.

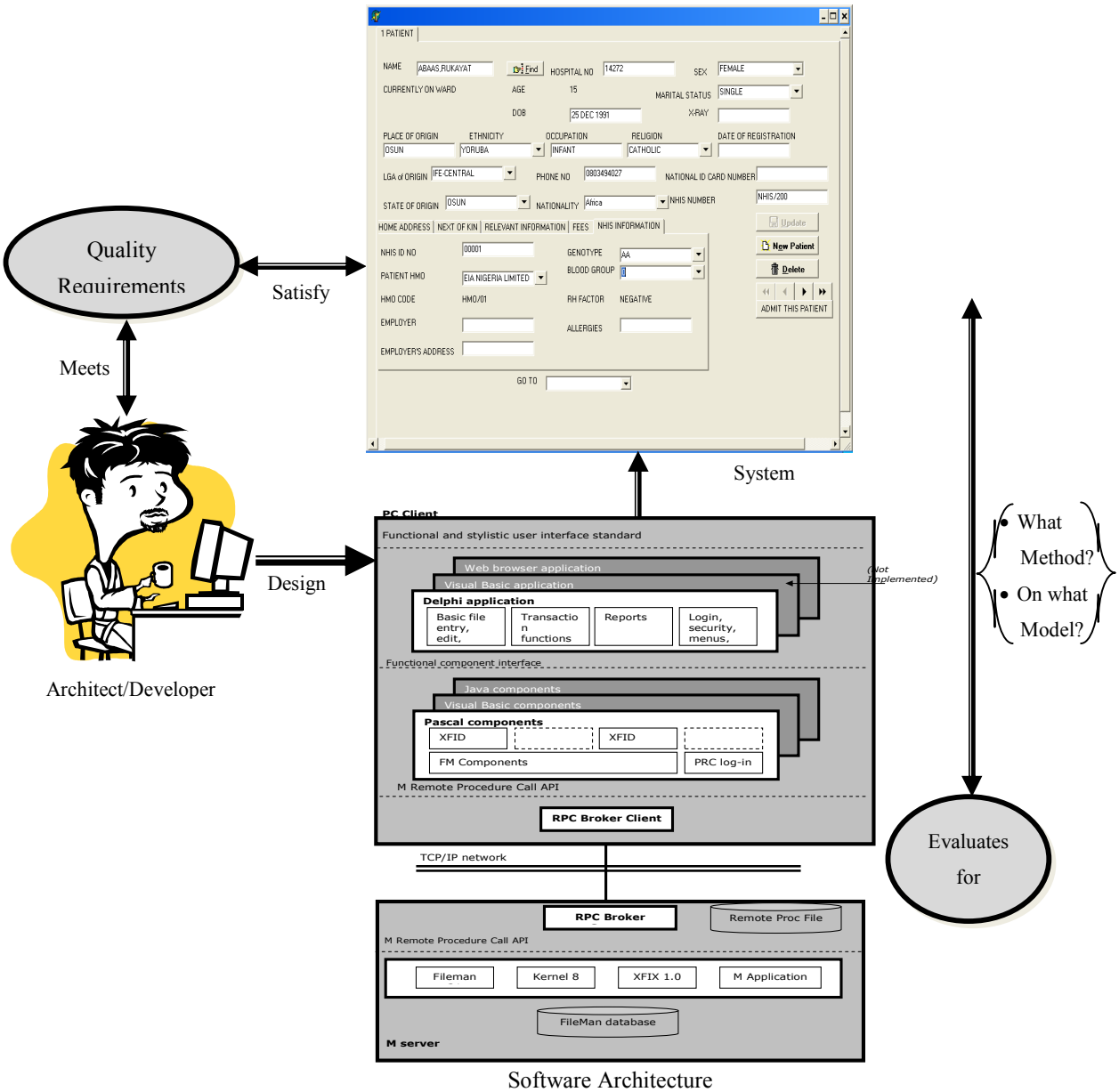

Figure 2. Conceptual framework of software architecture practice

6. How Software Architecture Practice fits into Software Development Process Lifecycle

Software architecture practice can be integrated into all the phases of software development methodologies and models. This is used to distinguish it from particular analysis and design methodologies. Since the architecture determines the quality of the system, it then makes a lot of sense to have architectural design built into the software development process. As shown in figure 3 below, software architecture is integrated into all the phases in development process. The role of software architecture in each phase of the software development process is established. The model shows that during the requirements phase of development, an architecture may be used to identify, prioritize, and record system concerns and desired. During design and analysis, an architecture may be used to model, visualize, and analyze design decisions chosen to address the principal concerns and achieve the desired qualities. Decisions may be guided by adopting one or more architectural styles. During implementation and testing, an architecture may be used to drive testing, instantiate a product, support runtime dynamism, or enforce security policies. Rather than thrown out an architecture at this point, as is often done, an architecture remains part of the product. During maintenance, an architecture may be used as a basis for incorporating new features, or increasing modeling detail.

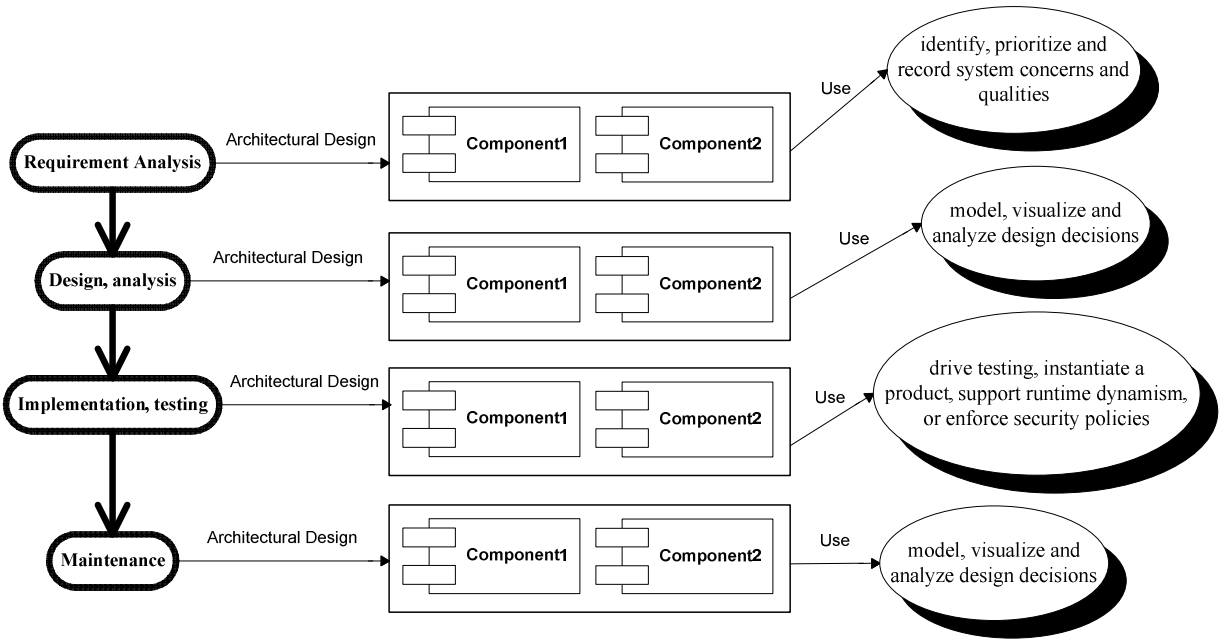

Figure-3. A model integrating architecture into software development process

7. Conclusion

Just like any other complex structure, all software must be built on a solid foundation. Obviously, the failure to consider fundamental scenarios, failure to gratify the long term consequences of key decisions can put most software applications at risk. In this regards, software architecture should stand as the solid foundation upon which software systems are built. Software architecture should be seen and taken as a key business asset for an organization. The development process is well known as a contributor to software product quality, leading to application of software process improvement as a key technique for the overall improvement of the software product. This can be said for any form of software development. Consequently, software developers and system acquirers can use effective

software architecture practices across the life cycle to ensure predictable product qualities, cost, and schedule. We establish in this paper that software architecture is the bridge between mission/business goals and a software system. Secondly, software architecture drives software development throughout the life cycle, and finally the paper identifies some of the issues to consider in order to make software architecture useful in practical software development process for developers, researchers, and acquirers, thereby enabling them to embrace the central role of software.

References

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

Chung, L. Nixon, B. Yu E. and Mylopoulos, J. (2000). "*Non-functional requirements in software engineering"*. Kluwer Academic, Boston, 2000.

Clements, P., Kazman, R., and Klein, M. (2002). "Evaluating Software Architecture: Methods and Case Studies": Addison-Wesley, Boston, MA.

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.

Boehm, B. (1995): "Cost Models for Future Software Processes: COCOMO 2.0," Annals of Software Engineering.

Soriyan (2004): Soriyan, H. (2004). "A Conceptual Framework for Information Systems Development Methodologies for Education And Industrial Sector In Nigeria". PhD Thesis, *Obafemi Awolowo University Ile-Ife.*

Warsta, J. (2002). "Agile Software Development Methods Review and Analysis". *VTT Publications.*

Salo, O. (2006). "Enabling Software Process Improvement in Agile Software Development Teams and Organizations". *VTT Publications.*

Bender RBT Inc. (2003). "Systems Development Lifecycle: Objectives and Requirements". *Bender RBT Inc, Queensbury, New York.*

Kazman, R., Bass, L., Clements, P. (2003). "Software Architecture in Practice". Addison Wesley, second edition.

Kruchten, P. (1995). "Architectural Blueprints - The .4+1. View Model of Software Architecture". IEEE Software 12 (6), pp. 42-50.

Hofmeister, C., Soni, D., and Nord, R. (1995). "Software architecture in industrial applications". *Proceedings of the 17th international conference on Software engineering*. pages 196-207, Seattle, Washington, USA.

Kazman, R., Abowd, G., and Webb, M. (1994) "SAAM: A Method for Analyzing the Properties of Software Architectures", *Proceedings of the 16th International Conference on Software Engineering*, pp. 81-90.

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., and Carriere, J. (1998) "The Architecture Tradeoff Analysis Method", *Proceedings of ICECCS'98*.

Bosch, J. (2000). "*Design and Use of Software Architectures: Adopting and Evolving a Product Line Approach"*, Pearson Education (Addison-Wesley and ACM Press).2000.