# Analysis and Improvement of Software Documentation Processes for Software Developers in Nigeria

**Theodora Isola**

Department of Information Systems and Technology, University of Liverpool

**Abstract**

Developing quality software should be the priority for software organizations and developers. Software attributes can be defined using software documentation which allows for effective communication among the software team and stakeholders towards the delivery of quality software applications. The aim of the project is to analyze and improve software documentation processes for software developers in Nigerian software developers with a focus on the general study of previous work on software documentation carried out over the years, carrying out literature review on software documentation, finding drivers and barriers to good documentation, developing survey distributed to Nigerian software developers and to users as questionnaires for gathering information on software documentation. To develop a framework/guideline which will include, methodologies, usage, procedures, and work patterns that can be easily used by software developers to ensure good documentation to deliver cost-effective and better-quality software. This research work analyzes the processes in software documentation using literature review and questionnaire. The focus was on finding drivers and barriers to good software documentation from the literature survey and getting responses from questionnaire on usage by software developers. This project has researched and discussed the documentation process, with resulting best practices and framework. Some guidelines have been proposed which can be used; and because there are no strict standards on software documentation, adhering to software development life cycle standards will ensure that documentation practices are well followed. Software documentation is helpful in defining useful and usable specifications and functionalities of software for the delivery of quality software as well as helping in the transfer of knowledge among team members and stakeholders.

**Keywords:** Software, Documentation, Agile, Developers, Development

## 1. Introduction

Documentation is an important step in the software development lifecycle (SDLC) and creating software document can facilitate effective communication amongst software developers and "with various interest groups of the system to be developed, such as customers, marketing people, end users, service personnel, and authorities." [1]. Understanding the need for documentation in a software project is just the beginning and is not as much important as knowing exactly which documents to create or how to create and maintain software documentation for an application and this has been a challenging aspect leading to no proper documentation.

A software application that is correctly documented will communicate the appropriate information to the team allowing for better records of development specifics, tracking of project progress to thorough testing support, validation, and verification, as well as provide guide for users and owners on the functionalities of the application [2]. Sadly, software documentation amongst some software organizations in Nigeria is not taken as seriously as rolling out applications for use and is mostly done poorly or not properly updated causing the development of poor-quality software and challenges in maintenance, probably because they do not know "how to create and maintain a program documentation system that meets the need of the organization" [3]

Software documentation is often perceived differently by each member of a project team. While the project manager will depend on the documentation to rate the project's success, in some cases, a team member or developer may not at all care for documentation especially if its role in the project is not well understood. Documentation problems as well as infrastructure and processes in software application projects can cause lack of software quality, this project focuses on the use of Software documentations such as system requirements, specifications, design, and architecture as well as using information from the findings of this research to analyze and improve documentation to help Software Developers in Nigerian develop better quality applications.

By researching from already existing work the extent of study of the importance of documentation to developing a software application with focus on the (1) if the size of the project determines if documentation is carried out or not? (2) How is software documentation managed? (3) Who takes responsibility for producing software documentation? From these research questions what has been done so far will be noted and will be used to suggest framework for software developers in Nigeria.

## 2. Analysis and Design

The data collection for this project followed the following steps. Step one involved literature review, and from the literature survey, the following were some of the drivers and barriers to good documentation used to determine how documentation is carried out and used.

### Good Drivers and Barriers to Software Documentation

Table 1: Drivers and Barriers to Good Documentation

| Sr. No. | Good Drivers | Barriers |
|---|---|---|
| 1 | The extent of software documentation being produced and used in software organizations. | Outrightly measuring the extent of software documentation in a project can be ambiguous. |
| 2 | Understanding the types of software documentation and the importance of each document. | Researching all the types of documents in a project can make the survey too broad. |
| 3 | Ensuring that documentation is complete and updated and maintained will help software team in developing high quality software. | Documenting and maintaining all the process of a software project being carried out is not entirely possible. |

| | | |
|---|---|---|
| 4 | The experience of the development team can affect how software documents are being referenced. | There are no metrics for measuring team experience especially in the first projects. |

This was followed by questionnaire design. The Questionnaire was designed with focus on the literature review to bring together perception and usage of software documentation among software developers in Nigeria. Questionnaire details can be found in Appendix B. The questionnaire was designed from already validated research work by [4] using Google forms.

The questions were mainly common questions on general use of software documentation where it applied, and distribution of the questionnaire was unstructured. In question 1 – 3 (Q1 – Q3) participants were asked on working experience, job function and the use of software documentation in their organization. Questions 4 – 8 (Q4 – Q8) covered the recommended type of development process, types of software documents written/edited/verified, whose primary responsibility is it to create and maintain and verify and validate software documentation, length of time before updates are done on software documentation after there is a change in the software system. In Questions 9 – 16 (Q9 – Q16) participants had to agree or disagree on a scale of 1 to 5 with 5 being they agree completely. The questions cover user involvement in software documentation, solving problems in software application using requirement documents, differences in the documents used between development and maintenance phase, level of experience in software documentation usage, maintenance cost exceeding cost benefit of up to date documents, effort in formatting and preparing software document compared to providing actual content, poorly referenced document and difficulty in navigation based on size and number of documents and individual usefulness in software documentation. The last three questions (Q 17 - Q19) required participants to comment based on the following: software tools most and least helpful, types of software artifact most and least effective in software documentation.

## 3. Implementation

Models and processes can continuously be developed based on each phase of the SDLC. These processes as described in previous research although not standardized are believed to help in software documentation. To gain proper understanding of software documentation, it will help to check out already known standards. For example, it is known that documentation is a supporting life-cycle process toward quality framework and although “the benefits and advantages of following a defined process is sometimes not immediately evident to the project team” [5] Software developers must ensure to follow defined process with aim at quality software.

Based on individual customers, requirement for software application may differ requiring more “Formal Program Management Plans (PMPs), Configuration Management Plans (CMPs), Quality Assurance Plans (QAP), and some other programs may require less formal version of document plans etc.”[6].

Although these processes above will aid in carrying out software documentation, there is no guarantee that they and, in some cases, the main issues are not with requirement gathering or poor documentation management, but “with out-of-date or incomplete documentation.” [7] There have been seemingly numerous documentation practices as a result of the advent of web development. These practices range from unstructured projects to structured projects (contracted and planned). Also,

there has been an increase in the practice of the use of digital libraries where the project team uses data management system for document and communication. This system is necessary in gathering and conveying information from everyone involved to allow for easy decision making. Successful practice in documentation has been known to deliver quality in the development of software [8] especially in the use of technical documentation which improves development and maintenance practice. [9].

- Software documentation of an application should be developed in an understandable format for all involved in the project with little or no explanation when read. In ensuring that SD is not ambiguous, the structure as well as the contents of software documentation should be simple and clear resulting to the development of quality applications according to "To achieve greater documentation relevance, we need to find ways to increase its power, simplicity, or preferably both." [7] Quality assurance in the development of software requires that documentation is not redundant and has no anomaly. Relevant requirement and architecture information should be easily retrievable from software documentation when needed. Software documentation should be easily modified to effect necessary changes. [10]

- Software documentation of an application should be developed in an understandable format for all involved in the project with little or no explanation when read. In ensuring that SD is not ambiguous, the structure as well as the contents of software documentation should be simple and clear resulting to the development of quality applications according to "To achieve greater documentation relevance, we need to find ways to increase its power, simplicity, or preferably both." [7] Quality assurance in the development of software requires that documentation is not redundant and has no anomaly. Relevant requirement and architecture information should be easily retrievable from software documentation when needed. Software documentation should be easily modified to effect necessary changes. [10]

- Prioritizing documentation and treating it like any of the other requirements for developing software applications helps in creating explicit documentation which allows easy and better consideration from the stakeholders. A project team, should be aware that quality documentation can help in the transfer of this knowledge to users as well as new members of the team and in treating documentation as a requirement, they will be sure to develop clear and appreciable documents.

- Adhering to Standards: We can begin from first considering the Software Development Lifecycle standards of which documentation is a process in the life cycle. That can help set the stage for developing software documentation. And although there are "no universally recognized standard for software documentation, there is a standard for documenting engineering and scientific software. This standard developed by American National Standards Institute (ANSI) and the American Nuclear Society (ANS) developed in 1995 formally called ANSI/ANS 10.3-1995 Standard can be adopted for the Documentation of Computer Software." [1] It use as a guide can serve as the starting point for software organization to produce quality software by following the documentation methodology.

The ANSI/ANS 10.3-1995 Standard has flexible set of specifications which serves as a guide in the delivery of quality software for most projects. Therefore, it helps to improve communication between software developers, stakeholders, and users, assisting in the effective selection, usage, transfer, conversion, and modification of computer software. [1] On its limitations, the ANSI/ANS 10.3-1995

Standard does not offer guidance for online monitoring, control and safety systems and does not handle customer-oriented requirement software. [1]

Also, the Institute of Electrical and Electronics Engineers (IEEE) standard for software requirement specification (IEE87, IEE94) provides an available structure that can be understood by many. Although these standards require various aspects to be specified, it allows SRS to fit in considering that "different project may require their requirements to be organized differently" [13] and by agreeing on what constitute and how the system should be represented, developers and stakeholders can identify and agree on sets of requirements to characterize the system. (IEEE Std 1059, 1993)

Other standards that may be considered include IEEE Std 829-1983 - IEEE Standard for Software Test Documentation (ANSI), IEEE Std 830-1993 - IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 1016-1987 - IEEE Recommended Practice for Software Design Descriptions (ANSI). (IEEE Std 1059, 1993).

- A software development team must work as a team to achieve software project goals, with proper understanding that documentation is important in delivery quality software and knowing the extent of documentation required and that the stakeholders need. This will help the team focus on the key areas to be documented instead of reporting on every happening during the project delivery and maintenance. Software organization should make software documentation is requirement. A software development team may come up with its own "documentation and coding standards that define the content, order and format of the documents, and the code created by team members." [14] Following these standards will ensure that all team members are working together. These standards can always be modified, and everyone is encouraged to comply.

- With the growing trend come new technologies, there are software-based tools that can basically reverse-engineer existing code and generate views. [15] These tools can help in creating effective documentation as well as save time and cost. Project managers should therefore seek out for the best documentation tool that suits the project and can be used by the development team without having to spend time training them to use it. Having to train a project team to use a tool they are not familiar with might delay development process therefore this should be duly considered when planning the project.

## Software Documentation Tools Used

In the survey, participants were asked to comment on the type of software tools they found most useful to create, edit, browse, or generate software documentation. The responses show that 48% of the respondents use word processor, while 16% use JavaDoc, the others were significantly low.

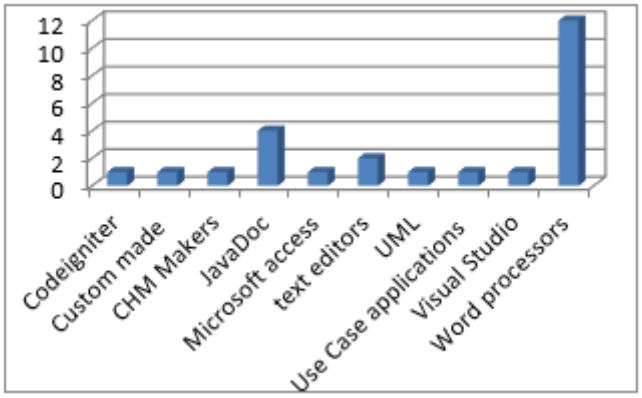


Figure 1: Summary of Most Used Documentation Tools

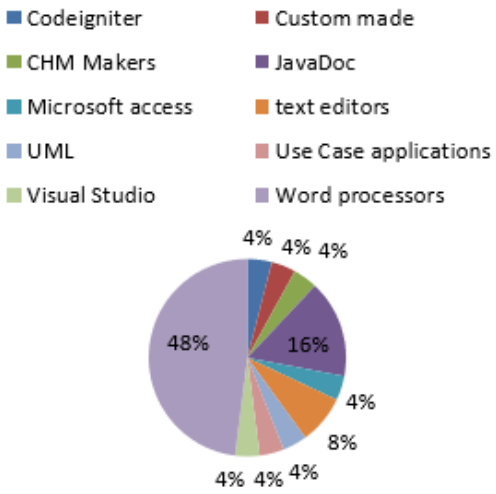


Figure 2: Summary of Most Used Documentation Tool

Example of some of the available tools includes:

**a) Unified Modeling Language (UML):** The UML is known to give a "visual representation of a system's specification at various levels of design" [16] which enhances communication and understanding of the software system properties. The benefits and limitations of UML tool below can help software developers to decide on its adoption. UML has been known for enhancing communication among project team and stakeholders. It provides good advantage for its visual properties which complements the code. As well, its ability to manage growing complexities associated with software development. This is done through high level abstraction, and its ability to trace different stages from requirement to low level design. Also, it is recognized for its use in constructing and documenting object-oriented software system's artefacts. Although with all its benefits, if adequate training is not gotten, its use will not be efficient. A UML deployment diagram, along with user interface flow diagrams and physical data diagram are valuable; models' software developers can base documentation on when kept up to date.

**b) Input – Process – Output (IPO):** This is mostly represented in diagrams used to document a process. The IPO model is an interactive model relevant during decision making among groups. During communication, the goals (input) are discussed which then influence subsequent decision

(process) thereby affecting the team (output). The model requires understanding of what occurs during communication [17. Rue [18] used influence diagrams to document the relationship between process parameters of a project. When these diagrams are drawn in the requirement phase, the identified inputs and outputs must be noted, and then are to be captured in the influence diagrams.

**c) Delete Document of Action (DofA):** an example being a design document goes through many successive versions, with corresponding evolving status in addition with individual fragments. A documents level of standardization plays a role in the indexing and classification of DofA. [19]

**d) Contract Deliverable Requirement Lists (CDRLs):** *T*his help to put together the documents which will be needed together based on what the client wants thereby creating a framework for software systems which is then agreed on by all parties involved. The contract documents are the first documents to be prepared and reviewed before a project is awarded and these documents are updated as changes are made. [20] These documents can also serve as ready materials for new projects.

## 4. Framework

The changing needs of documentation and the extent to which documentation may be carried out in software development organization will require numerous and different documents because each project is different leading to increase time and cost. From the research so far, it has been established that requirements specification document is the mostly done documents. Therefore, the developed software requirement documentation outline below will serve as guide in managing documentation in uniformed manner for quality software. As well serve as a basic structure on which further development of documentation is based thereby, helping software developer in easy and consistent documentation of software system leading to quality system. The following is an example of a Software Development Documentation framework:

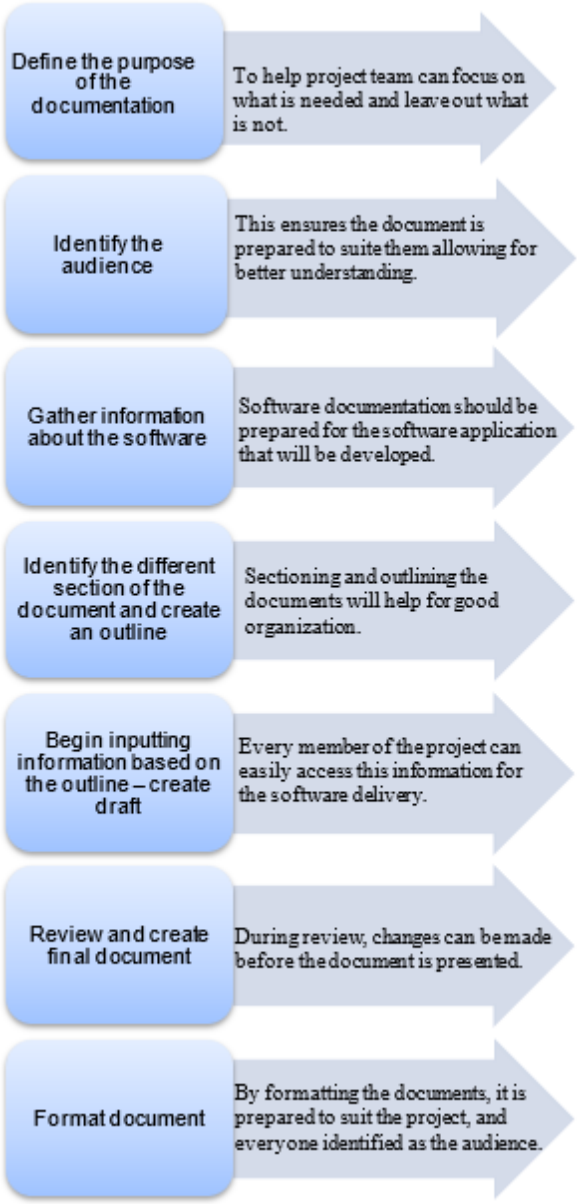


Figure 3: Framework for Software Documentation

Below is an example of documentation outline for a software application coined from Restaurant Point of Sale Project documentation outline [21]. Based on the application and the team, the outline can be revised to their standard.

Table 2: Sample Software Documentation Outline

| | | |
|---|---|---|
| 1 | Introduction | • Document purpose<br>• Scope<br>• Objectives<br>• Intended audience |
| 2 | Product Description | • Functions<br>• Operating environment<br>• Performance requirements |

| | | |
|---|---|---|
| | | • Security requirements<br>• Policies<br>• Industry standards<br>• Business rules |
| 3 | External Interface Requirement: | • Program interface<br>• User interfaces<br>• Hardware and software interfaces<br>• Communication interfaces. |
| 4 | Functional Specification Requirements | • Software inbuilt features |
| 5 | Product Schedule | • Resources allocated for the project |
| 6 | Risk Identification | • Potential risks<br>• Risks prioritization<br>• Mitigating risk<br>• Potential resolution<br>• Risk monitoring |
| 7 | Change Management | • Evaluation plans<br>• Tracking changes |

**Framework Description**

**e) Introduction:** The introduction will cover document purpose, its scope, objectives and intended audience. This will allow anyone to understand what the document is all about and the extent to which the document will be developed.

**f) General Idea of the product:** Every project is different, and by providing a general idea of the product, what is needed to achieve the project is known specifically for the product. This general idea will include its functions, the operating environment, performance requirements, security requirements, policies, industry standards, business rules.

**g) External Interface Requirement:** This will discuss any program that the system being created will interface with. This could include user interfaces, hardware and software interfaces and communication interfaces.

**h) Functional Specification Requirements:** This section should explain extensively the software inbuilt features covering will do and how it will be done. This should be well explained to intended audience identified above.

**i) Product Schedule:** This will breakdown the schedules and resources that will be allocated for the project, including time and cost required for the product to be developed.

**j) Risk Identification:** This section will identify all potential risks, how the risks will be prioritized, plan on mitigating risk, potential resolution, and monitoring means.

**k) Change Management:** How change will be managed throughout the project period with plans on evaluating and tracking changes.

## 5. Result and Evaluation

Software documentation is commonly used to describe the attributes of a software and when done properly can help a software project team reach its full potential, in developing and delivering timely software applications. The use of documentations tools for especially for requirement and functional

specifications can however improve documentation usage in a project although this is depended on the system capabilities as well as the adaptability of the team to use the system. In the case of language documentation, these documents are collected and with the help of language tools are transcribed into more understandable documents. According to Bird and Simons [22], these language tools are mostly known to have portability problems and software developers are required "to represent their rationale information with ad hoc vocabularies and format" [23] which this project, with the best practices discussed and framework can help software developers do better documentation.

The goal of this study was to analyze and improve software documentation processes for software developers in Nigeria. Form the literature review, drivers and barriers to good documentation were defined in this project. That in addition to the responses from the questionnaire were used in finding out existing practices from which the best practices have been discussed above. While the best practices will help software developers deliver quality software, it is important to note that there are no strict standards and processes but adhering to known standards of software development lifecycle.

In the cause of evaluation of the framework/guideline for improved software documentation for use by Software Developer to deliver cost effective and better-quality software, 3 software developers were recruited to evaluate the framework/guidelines. By evaluating the project artefact with a small group of 3 software development experts helped to check how the processes and framework can benefit software developers in Nigeria and the framework.

Highlights of the best practices were discussed with 3 software development (SD) experts and then they were asked to rate on a scale of 1 - 5 the following the questions generated from the following best practices.

Table 3: The Result of the Evaluation

| Best Practices | SD Expert 1 | SD Expert 2 | SD Expert 3 |
|---|---|---|---|
| Rate and the extent with which an understandable format will benefit all in the project development? | 5 | 4 | 4 |
| Will clear enough documentation allow for easy interpretation? | 5 | 5 | 4 |
| How important is prioritizing documentation in a software project? | 5 | 4 | 4 |
| How will you rate the importance of adhering to known standards in developing software documentations? | 4 | 5 | 5 |
| Rate the importance the importance of teamwork in | 5 | 5 | 4 |

| | | | |
|---|---|---|---|
| ensuring quality documentation | | | |
| Rate the use of software-based tools in ensuring good documentation | 4 | 4 | 5 |
| Comments | These best practices can help any software development team carry out documentation. | The importance of software documentation in delivery quality software projects should compel any project team to seek out best practices and the above seem to be good enough. | Even though these best practices will enhance the development of software documentations, it will be easier if the focus is just on using software documentation tools. |

Table 4: Framework as Described by 3 SD Experts

| Framework | SD Expert 1 | SD Expert 2 | SD Expert 3 |
|---|---|---|---|
| • Define the purpose of the documentation<br>• Identify the audience<br>• Gather information about the software<br>• Identify the different section of the document and create an outline<br>• Begin inputting information based on the outline – create draft<br>• Review and create final document<br>• Format document | By following this procedure, software developers can easily put together software documentation for stakeholders and the team. | Any software developer can always use this framework for a start but should not be so fixed on it that they cannot come up with better procedures for software documentation. | The information for framework could be improved, for better understanding because anyone using this software should have some form of experience on software documentation. |

From the evaluated best practices and framework, it can be said that the three software development experts, agreed to the practices although, one expressed concerned on the number of best practices and mentioned that the focus should be on documentation tools for easy documentation to allow for quick delivery of software applications. All three of them agreed that the framework and outline will be useful in forming the basis for documenting software applications which can be improved on based on the project and its team.

## 6. Discussion

Software documentation can be referred to formal document whether in print or electronic format for use in effectively and efficiently developing a software application fit for its intended environment [10]. Its importance in the software development process helps provide the appropriate information to the stakeholders while enhancing communication among team members thereby ensuring everyone involved in the software project work together.

It is difficult to come up with a single definition for software documentation considering that there are many descriptions and in discussing some of the definition of Software Documentation [24] expressed that the many meanings of Software Documentation would not help one come to a specific definition, the notion of Software Documentation being an artifact for communicating information about a

specific product providing such description that is expected to give precise information about the system being developed should be considered. As well as documentation being a description of authority or evidence of work emphasized.

Documentation as a basis for communication among project team, stakeholders and users need to be effective and the documents needs to be well understood. [1] The diagram below (Figure 2) encourages project team to choose the best form of communication option available based on the team's current situation. Apparently, conversation with stakeholders and involving them can help handle issues which is more valuable than the best documentation. [15] Having face to face conversation between team members and stakeholders can help to give better understanding of what the team is communication and that way, misinterpretations are prevented. [19]

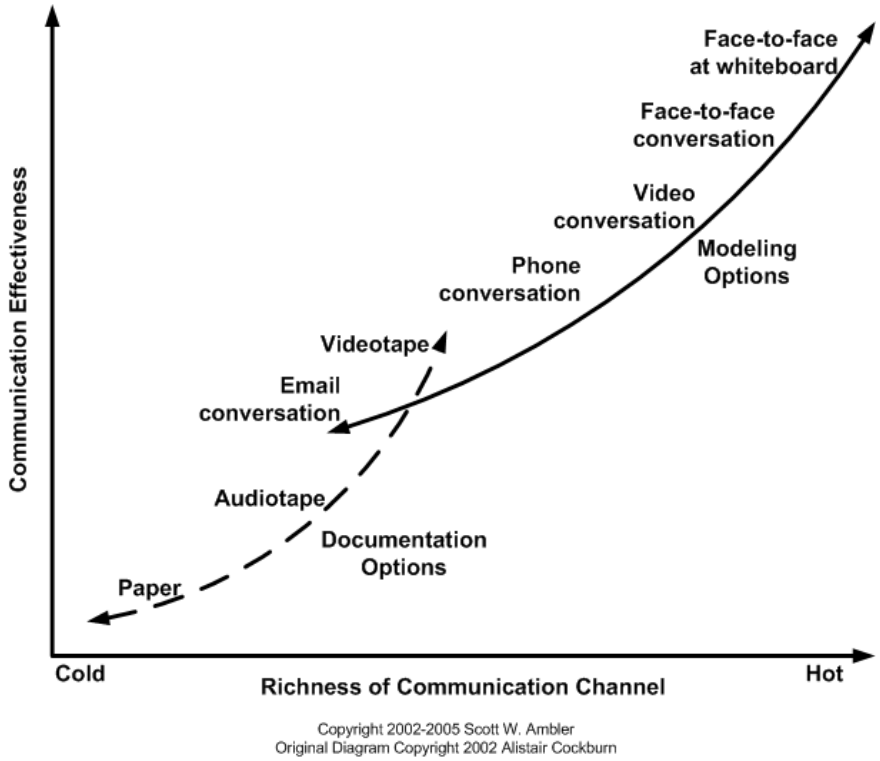


Figure 4 Comparing the Effectiveness of Various Communication Modes [15]

Software Documentation can be used as a factor to rate the value of a software application except in cases where the documentation is incomplete, outdated or of poor quality, despite that if the software documents are well communicated and understood by the team, this will still allow for the development of quality software. Documentation can help in the proper understanding of software process during a system's modeling and with the use of representations the customers can understand properly what the software engineers are communicating. [18] Project managers must ensure complete and open communication amongst team members as well as early and frequent feedback to lower cost and improve delivery time. A more efficient review of requirement during the requirement phase of the SDLC will be well carried out by test engineers but "If requirements are so ambiguous that test engineers do not know what to test, how can developers be expected to design and code to those requirements?" [6]

Just like Software development process ensures proper coordination and better communication among all involved in developing the software application with the sole aim of achieving the goal for a working software, software documentation as an important part of the software development process, can help stakeholders communicate asynchronously without "the time and geographical restriction during software development process." [10]

Some developers have misinterpreted Agile's manifesto of having a working system over documentation as an excuse not to carry out documentation. On the contrary, the "Agile development method does not stop developers from carrying out documentation and it is important to know the kind of documentation that your project requires and when it is the best time to produce it." [25] In a summary of results of DDJ's 2008 Modeling and Documentation survey pertaining to documentation deliverables, it was found that instead, the extent of documentation deliverable documentation like operations documentation, user manuals etc. were just about the same in the agile teams and the traditional teams. [15]

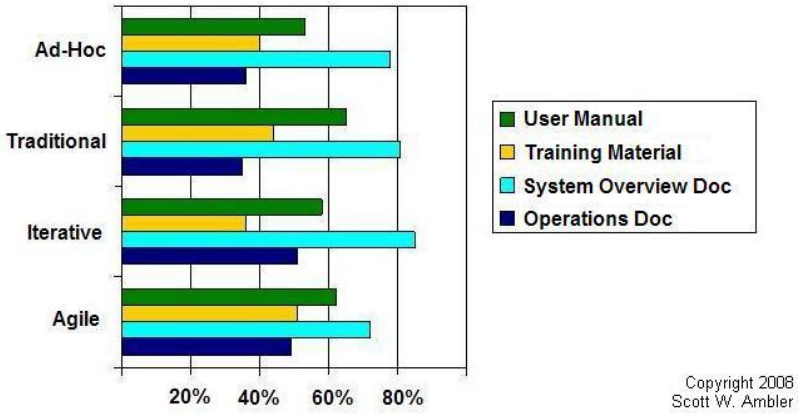


Figure 5: Percentage of Teams Creating Deliverable Documentation [15]

The agile methods encourage "face-to-face communication over formal and precise documentation and also over tele-/video-conferences or email conversations" [26] and this process have been found to be "most workable on small projects and relatively low at-risk outcomes, highly capable personnel, rapidly changing requirements [27] and in striving to continuously improve software products, adequate documentation must always be in focus [28]. According to Turk et al "Good documentation of requirements and designs, produced and maintained in a timely manner, are essential to ensure that the distributed team members all maintain the same vision of the product to be built."

Software documentation can describe a wide variety of aspect in software development [29]. There are several types of software documents needed by software developers, end users and for maintenance and these documents are broadly used in software development lifecycle. For example, the development documents (requirements specifications, design, or architectural documentation) allow for easy coordination among team members helping them to effectively develop the software. Also, the maintenance documentation guides the team in improving or fixing software failures while the user's documentation help users get to understand the suitable ways to use software application. [14]

Software Requirement Specification (SRS) document is the most common and often stands as a guide for reviewing software's requirement. When SRS documents are well defined, it can cover the attributes of software and all aspects of and maintenance of software in ensuring usability, reusability and maintainability as well as ensuring end users satisfaction. The requirement phase of a software project is meant to define the purpose and the usage of the software model, determining the questions that the software model will have to answer, [18] and then these software requirements documents are produced, validated, and used to describe behaviors and characteristics usually visible. When developers deviate from the documented requirements, sometimes due to no proper analysis of the changes in fulfilling new requirements, by omitting required functionalities due to times and budget constraint, [14] these may "lead to compromise in the dependability of the software systems being developed." [30] Problems detected early during the development have the chances of being corrected and fixed, whereas if the problem is detected at the end of the software development lifecycle, there are no chance of fixing as the problems may already have led to a complete compromise of the software system.

The requirement specification phase in software development can be quite difficult to achieve because it is an important area of interest for software team, stakeholders and end users and must be made to be easily understood for all parties. By getting SRS to a reasonable quantity, software developers can achieve the goal of helping everyone understand the different aspect of the project through a well written and specified document and as well ensuring that what is specified is what is required to achieve the project. Complimenting requirement document with some visual representation like use case, flow diagrams and mind maps thereby saving prospective client the stress of reading long documents without visual aids.

According to Jalote [13], there are three main approaches when analyzing requirement specification. These approaches include:

1. The unstructured approach where all requirements are discussed among analysts, customers, and users and afterwards the requirements are documented.
2. The Modelling Oriented approach where available information is used to develop a model which is then used to determine proper understanding and thereby determining the project's requirements.
3. Prototype: In this approach, a prototype is developed and used in validating the correctness and completeness of requirements.

Developers who may worry about software requirement specification not in accordance with standards need to understand that there are "no standard and a few conventions exist regarding the content of reference documentation." [31] As long as the customer is involved with validation of requirements and agree with the content of the resulting model specification document. [18] Software developers should ensure to use documentation systems or tools based on the platforms for the system to be developed an example will be using Javadocs for Java Application Programming Interfaces.

The software design documents account for the architecture information. These documents require are common for giving development and maintenance team a difficult time especially when they "contain incorrect dimensions, inadequate references to drawings, standards and building/engineering codes

and conflicting specification". [23] Therefore, having correct and complete documents will help software developers in meeting all the system's requirement as well as the runtime constraints.

These two types of documents are the most common and are usually written in natural language "for the communication between various stakeholders of the project, such as customers, managers, requirements engineers, architects, and developers." [19] The use of controlled natural language will help in representing and communicating the software's structure for easy understanding.

A project's technical requirement basically applies to the functional behaviors of the software which may be completely textual or graphical or with the use of visual language based on the preference of stakeholders and project team and as well helping to joining their "needs and perceptions to the technological solutions designed by the software experts." [32]

The design (high level and detailed or low-level design) documentation is used to define the functional and operational interfaces. However, determining the level of precision of a document to check if it is of high quality may be difficult especially if the document is not dependent on any process or organizational context. Creating documents independent of any process keeps the documentation without any criteria for determining its precision level. Also creating documents that requires the interpretation of a skilled interpreter can often be difficult to access. "Instead, information should be documented in such a way that the reader can easily retrieved" [33] and the use of diagrams can help reduce the complexity of documentation.

Software documentation is an important aspect in software development process. While this study targets the processes that can help software developers in Nigeria deliver better quality software, from the literature review, there are undoubtedly several studies carried out so far with most focused on its effect on cost, benefits, and quality of software applications.

With much information on the importance of software documentation in the delivery of effective software, one would wonder why it remains a challenge for software developers to create software documents. The time spent trying in figuring out how a software works can be reduced with the availability of proper software documents and developers can simply move on to the next project without interruptions from end users.

The growing trend in technology and the use of software applications in almost all organizations makes it important that software developers continuously improve on software applications that are of ISO quality and this has not been taken seriously amongst most software developers in Nigeria.

## 7. Conclusion/Recommendation

This project analyzed software documentation through literature review and interactions among software developers and then suggests best practices and framework for improving software documentation for the delivery of quality software amongst software developers in Nigeria. The research questions as seen in the table below helped in gathering drivers, barriers and in designing the survey. The results have been discussed in chapter 5 after careful evaluation of the best practices and framework.

Table 4: Research Questions and Method of Responding

| Research Questions | How Research was done |
|---|---|
| The questions used for this project research are as follows:<br>• What is the importance of documentation to developing a software application?<br>• Does the size of the project determine if documentation is carried out or not?<br>• How do you carry out and manage your documentation?<br>• Who takes responsibility for producing software documentation? | The method used for researching includes:<br>• Literature review: investigating existing work done on software documentation focusing on the research questions.<br>• Interactions with a few software developers also helped to answer some of the research questions<br>• Questionnaires were also used, and participants responded to the questions asked. |

The importance of software documentation in the software development cycle can be seen as it improves quality of the application. By analyzing and improving the processes of software documentation, the terminologies used above, existing knowledge and practice, techniques and method discussed can be used to help software developers and their team to develop more quality software applications.

Maintaining good documentation of project specifications requirements, designs and other documents in the timeliest manner is important in ensuring all members of the team are in sync with the software plan for a quality product. Despite the much emphasis on software documentation in this project, it is important to note that "Documentation should be created and maintained only if they provide value to the project and project stakeholders." [26]

While searching for resources, they were no much work done on software development or documentation as related to Nigerian software developers. Notwithstanding, software documentation remains an important process in the SDLC and should be taken seriously. The primary focus of this project has been to analyze and improve software documentation processes among software developers in Nigeria. Considering that the level of software industry is not regulated, and more work should be done on software documentation according to standards and to software organizations should always ensure to develop quality software applications.

The limited number of respondents also helped to establish that overall software development is still in its growing phase and more needs to be done to encourage people to engage in developing quality software instead of limiting themselves to just web site creations.

Considering the increased use of technology tools in businesses today, it will be unfortunate if businesses still outsource development of software applications overseas. Therefore, organization and stakeholders should insist known and proven standards as well as project plans in the form of requirements and design documents that they can relate to.

If specification requirements documents remain the most important and Software Developers do not give same attention to creating and managing other documents, this could lead to defects in the software system.

This project focused mainly on the software documentation processes among Nigerian software developers. The foundation for software documentation been laid here can be used in the future to further analyze software documentation processes with more developers reached to get a wide overview of usage in software organization. Also, seeing that there is not much research work on software documentation among Software application in Nigeria, it is an area to be explored and the level of usage of software documentation processes should be considered and actual software engineers with experience should be recruited as research participants.

Another area to be considered is the usage of software documentation tools. By identifying useful tools, and by including their benefits and limitations, software engineers can compare and work with the most suitable tools. And because the more experienced software developers are significantly affecting the level of usage of software documentation, young developers should not feel bad about not been able to produce perfect documentation, instead they should keep improving.

## References

1. Chomal V. S., Saini J. R. (2015) Software Project Documentation - An Essence of Software Development. Int. J. Advanced Networking and Applications, ISSN: 0975-0290, Volume 6, Issue 6, Pages 2563-2572
2. Fowler K. R. (2015) Documentation – Chapter 5 Developing and Managing Embedded Systems and Products. Elsevier Inc. http://www.sciencedirect.com.ezproxy.liv.ac.uk/science/article/pii/B9780124058798000052 (Accessed on: 2015 July 17)
3. T. W. York, MacAlister D. (2015) Program Documentation and Performance Measures. Hospital and Healthcare Security. (Sixth Ed.) Elsevier Inc. ISBN: 978-0-12-420048-7. http://www.sciencedirect.com.ezproxy.liv.ac.uk/science/article/pii/B978012420048700012X (Accessed on: 2015 August 04)
4. Forward A. (2002) Software Documentation – Building and Maintaining Artefacts of Communication. Thesis, Ottawa-Carleton Institute for Computer Science. https://www.site.uottawa.ca/~tcl/gradtheses/aforward/aforward_thesis.pdf (Accessed on: 2015 Dec 19)
5. Kuhn T., Bergel A. (2014) Verifiable source code documentation in controlled natural language. Science of Computer Programming, Volume 96, Part 1, Pages 121–140. http://www.sciencedirect.com/science/article/pii/S0167642314000069/ (Accessed on: 2015 Sept 20)
6. Drabick R. (2013) Best Practices for the Formal Software Testing Process: A Menu of Testing Tasks (Dorset House eBooks). Addison-Wesley, ISBN 0133489329, 9780133489323
7. T. C. Lethbridge, J. Singer, A. Forward (2003) How Software Engineers Use Documentation: The State of the Practice. IEEE Software. https://ieeexplore.ieee.org/abstract/document/1241364 (Accessed on: 2016 Jan 13)
8. Krogstie L. et. al. (2014) On Knowledge-based Development: How Documentation Practice represents a strategy for Closing Tolerance Engineering Loops. Procedia CIRP, Volume 21,

Pages 318–323. http://www.sciencedirect.com.ezproxy.liv.ac.uk/science/article/pii/S2212827114007173 (Accessed on: 2015 June 17)

9. Garousi G. et. al (2015) Usage and Usefulness of Technical Software Documentation: An Industrial Case Study. Information and Software Technology, Volume 57, Pages 664–682. https://www.sciencedirect.com/science/article/abs/pii/S095058491400192X (Accessed on: 2015 June 17)
10. Ding W. et al. (2014) Knowledge-based approaches in software documentation: A systematic literature review. Information and Software Technology. Volume 56, Pages 545–567. https://www.sciencedirect.com/science/article/abs/pii/S0950584914000196 (Accessed on: 2015 June 17)
11. Henderson S., Feiner S. (2011) Exploring the Benefits of Augmented Reality Documentation for Maintenance and Repair IEEE Transactions on Visualization and Computer Graphics, Vol. 17, No. 10. https://ieeexplore.ieee.org/abstract/document/5620905 (Accessed on: 2015 Sept 20)
12. Rivera T., Tate A., Will S. (2004) Deadly Sins of Technical Documentation. IEEE SoutheastCon, Pages 297–301, DOI: 10.1109/SECON.2004.1287934 (Accessed on: 2015 July 08)
13. Jalote P. (1997) An Integrated Approach to Software Engineering. (2nd Ed.) Springer-Verlag New York, ISBN 978-1-4684-9312-2
14. Galin D. (2004) Software Quality Assurance: From Theory to Implementation. Pearson Education Limited, ISBN 0-201-70945-7
15. Scott W. A. (2014) Best Practices for Agile/Lean Documentation. http://www.agilemodeling.com/essays/agileDocumentationBestPractices.htm (Accessed on: 2015 May 25)
16. Dzidek W. J Arisholm, E and Briand, L. C (2008) A Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance. IEEE Transactions on Software Engineering, Vol. 34, No. 3. https://ieeexplore.ieee.org/abstract/document/4459340/ (Accessed on: 2015 Jan 15)
17. Pavitt C. (2014) An Interactive Input–Process–Output Model of Social Influence in Decision-Making Groups. Small Group Research, Vol. 45, No. 6, Pages 704–730. DOI: 10.1177/1046496414548353 (Accessed on: 2015 Feb 01)
18. Rus I., Neu H., Münch J. (2003) A Systematic Methodology for Developing Discrete Event Simulation Models of Software Development Processes. 4th International Workshop on Software Process Simulation and Modeling (ProSim 2003), Portland, Oregon, USA. http://arxiv.org/abs/1403.3559 (Accessed on: 2015 Jan 09)
19. Zacklad M. (2006) Documentarisation Processes in Documents for Action (DofA): The Status of Annotations and Associated Cooperation Technologies. Computer Supported Cooperative Work (CWSW), 15, Pages 205–228. https://link.springer.com/article/10.1007/s10606-006-9019-y (Accessed on: 2015 June 17)
20. McCrady S. G. (2013) Designing SCADA Application Software: A Practical Approach. Elsevier Inc. DOI: 10.1016/B978-0-12-417000-1.00009-9 (Accessed on: 2016 Jan 10)
21. Laureate Online Education (2009) Restaurant Point of Sale Group Project - Managing the Software Enterprise Module
22. Bird S., Simons G. (2002) Seven Dimensions of Portability for Language Documentation and Description. http://arxiv.org/abs/cs/0204020 (Accessed on: 2015 June 17)
23. Love P. E. D. et al. (2013) Documentation errors in instrumentation and electrical systems: Toward productivity improvement using System Information Modeling. Automation in

Construction, Vol. 35, Pages 448–459. https://www.sciencedirect.com/science/article/abs/pii/S0926580513000952 (Accessed 2015 June 17)

24. Zhi J. et al. (2015) Cost, Benefits and Quality of Software Development Documentation: A Systematic Mapping. The Journal of Systems and Software, Vol. 99, Pages 175–198 https://www.sciencedirect.com/science/article/abs/pii/S0164121214002131 (Accessed 17 June 2015)
25. Gottesdiener E. (2013) Is 'Agile software documentation' an oxymoron? Search Software Quality: TechTarget. http://searchsoftwarequality.techtarget.com/answer/Is-Agile-software-documentation-an-oxymoron (Accessed on 2015 June 24)
26. Turk D., France R., B. Rumpe. (2005) Assumptions Under lying Agile Software Development Processes. Journal of Database Management, Volume 16, No. 4, Pages 62-87, Idea Group Inc. http://arxiv.org/abs/1409.6610 (Accessed on: 2016 Jan 06)
27. Boehm B. (2006) A View of 20th and 21st Century Software Engineering. ICSE '06: Proceedings of the 28th International Conference on Software Engineering, Pages 12-29. https://dl.acm.org/doi/10.1145/1134285.1134288 (Accessed on: 2015 June 17)
28. Lisa C. (2010) Software documentation is important in Agile environments. Search Software Quality: TechTarget. http://searchsoftwarequality.techtarget.com/answer/Software-documentation-is-important-in-Agile-environments (Accessed on 20154 June 24)
29. Kuhn T., Bergel A. (2014) Verifiable source code documentation in controlled natural language. Science of Computer Programming, Volume 96, Part 1, Pages 121–140 http://www.sciencedirect.com/science/article/pii/S0167642314000069 (Accessed on: 2015 Sept 20)
30. Veras P. C. et al. (2015) A benchmarking process to assess software requirements documentation for space applications. Journal of Systems and Software, Vol. 100, Pages 103–116. https://www.sciencedirect.com/science/article/abs/pii/S0164121214002404 (Accessed on: 2015 Dec 18)
31. Maalej W., Robillard M. P. (2013) Patterns of Knowledge in API Reference Documentation. IEEE Transactions on Software Engineering, Vol. 39, No. 9, Pages 1264–1282. https://ieeexplore.ieee.org/abstract/document/6473801 (Accessed on: 2016 Jan 09)
32. Meli R. (1999) Risks, requirements and estimation of a software project ESCOM-SCOPE 99, Herstmonceux Castle, East Sussex, England. http://www.researchgate.net/profile/Roberto_Meli/publication/2464670_Risks_Requirements_and_Estimation_of_a_Software_Project/links/00b7d5273970cdfdc8000000.pdf (Accessed on: 2016 Jan 09)
33. Landes D., Schneider K., Houdek F.(1999) Organizational Learning and Experience Documentation in Industrial Software Projects. Int. J. Human-Computer Studies, Volume 51, Issue 3, Pages 643–661. https://www.sciencedirect.com/science/article/abs/pii/S107158199990280X (Accessed on: 2015 June 17)