Importance of Software Documentation

Noela Jemutai Kipyegen[1] and William P. K. Korir[2]

[1]Department of Computer Science, Egerton University
Njoro, Kenya

[2]Department of Computer Science, Egerton University
Njoro, Kenya

Abstract

Software Documentation is a critical activity in software engineering. Documentation improves on the quality of a software product. It also plays significant roles in software development environment and system maintenance. Several software development best practices are ignored. This paper looks at the importance of software documentation, creation, applications, tools and standards.

Keywords: *Software Documentation, Importance, Role, Applications, tools and standards*

1. Introduction

Many factors contribute to the success of a software project; documentation included. *Software documentation is an artifact whose purpose is to communicate information about the software system to which it belongs* [1]. Parnas [2] defines *a document as a written description that has an official status or authority and may be used as evidence. In development, a document is usually considered binding, i.e. it restricts what may be created. If deviation is needed, revisions of the document must be approved by the responsible authority.*

Systematic approaches to documentation increase the level of confidence of the end deliverable as well as enhance and ensure product's success through its usability, marketability and ease of support [3]. "The dominant factor between a successful project and an unsuccessful project reduces to the effective dissemination of key information and successful software projects become successful because they give the right level of attention to clearly communicating the key concepts and requirements" [4].

Capri defines a successful documentation as one that makes information easily accessible, provides a limited number of user entry points, helps new users learn quickly, simplifies the product and helps cut support costs.

Poor documentation is the cause of many errors and reduces efficiency in every phase of a software product's development and use [2]. Documentation is an activity that needs to commence early in development and continue throughout the development lifecycle. It acts as a tool for planning and decision making.

2. Motivation

After assessing students' projects for a period of time, we realized majority of the students are neither enthusiastic nor motivated in the area of documentation. Most of them prefer only one phase of software development which is, coding. From this issue, we developed a desire to deeply understand software documentation: applications, benefits, creation and role in software development environment. *Does it contribute to the success of a project?* This led us to explore the existing documentation practices in software engineering.

3. Document Creation

Capri [3] describes in Figure 1, eight processes (analysis, design, development, validation, production, manufacturing, delivery and customer satisfaction) that guide in document creation. *Document preparation is the process of creating a document and formatting it for publication* [5]. Other researchers [6] gave seven rules for sound documentation. These rules include; 1. Documentation should be written from the point of view of the reader, not the writer, 2. Avoid repetition, 3. Avoid unintentional ambiguity, 4. Use a standard organization, 5. Record rationale, 6. Keep it current and, 7. Review documentation for fitness of purpose.

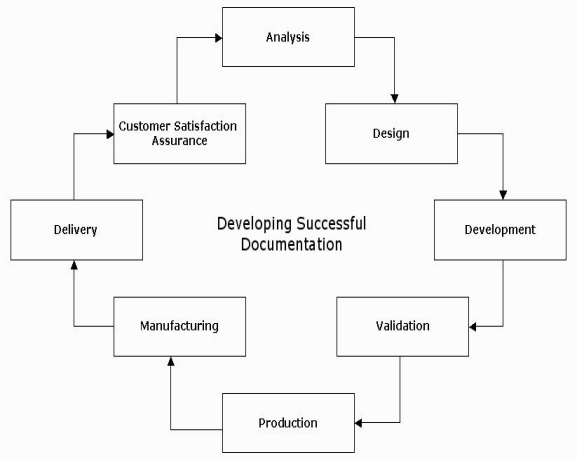

Fig. 1 Eight Phases to successful documentation (source: [3])

In the first Phase, analysis, possible audience that could potentially need documentation for the product and the tasks they will perform on the software are identified. Next phase, design, involve taking all the documentation items identified during the Analysis Phase and contents for each are designed/planned. The third phase entails creation of the actual document to be delivered. Validation, phase five, entail testing the documentation to ensure it meets its performance objectives, and the needs of its target audiences. The purpose of phase six is to produce high-quality finished goods (paper, videotape, audio, CD, online, etc.) to meet demand. In the seventh phase, final product (software and documentation) is then delivered to the customer. The last phase is customer satisfaction. In this phase, the document is improved based on customer's needs [3]. Sommerville [5], described document preparation process in three stages namely document creation, polishing and production as shown in Figure 2.

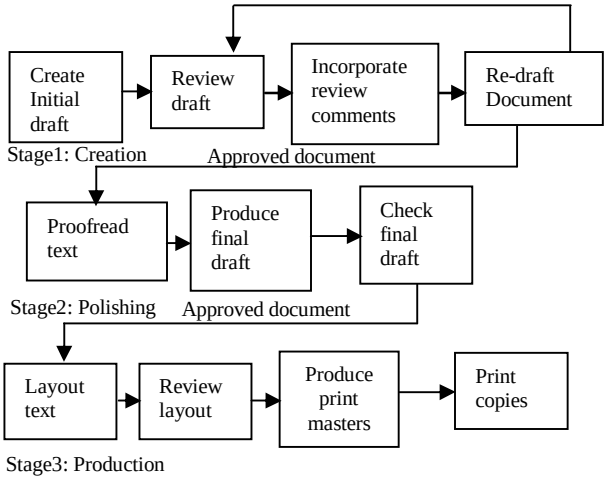

Fig. 2 Stages of document preparation (Source: [5])

Document creation involves initial input of the information in the document. The second stage which is document polishing entails improving the writing and presentation of the document to make it more understandable and readable. The last stage is document production which defines the process of preparing the document for professional printing [5].

The two models, (Capri's and Sommerville's) define software document creation as a process which involve continual understanding, review, and modification throughout the development lifecycle [7] as shown in Figure 3 below.

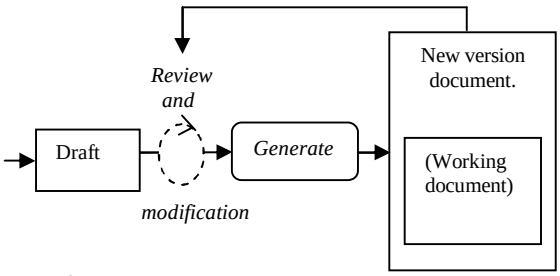

Fig 3 Document creation

Apart from being documentation cost and benefit model [8], Bo sun's model (Figure 4 below) describe in-depth the processes involved in software documentation. The model gives clear illustration of various actors (e.g Requirements Engineer, Business analyst, document reviewers, software developers and maintainer of the system) involved in software development, and uses of the documents. Bo Sun's model is comprehensive and can be efficiently employed in the description, creation, applications, as well as understanding the costs and benefits of documentation.

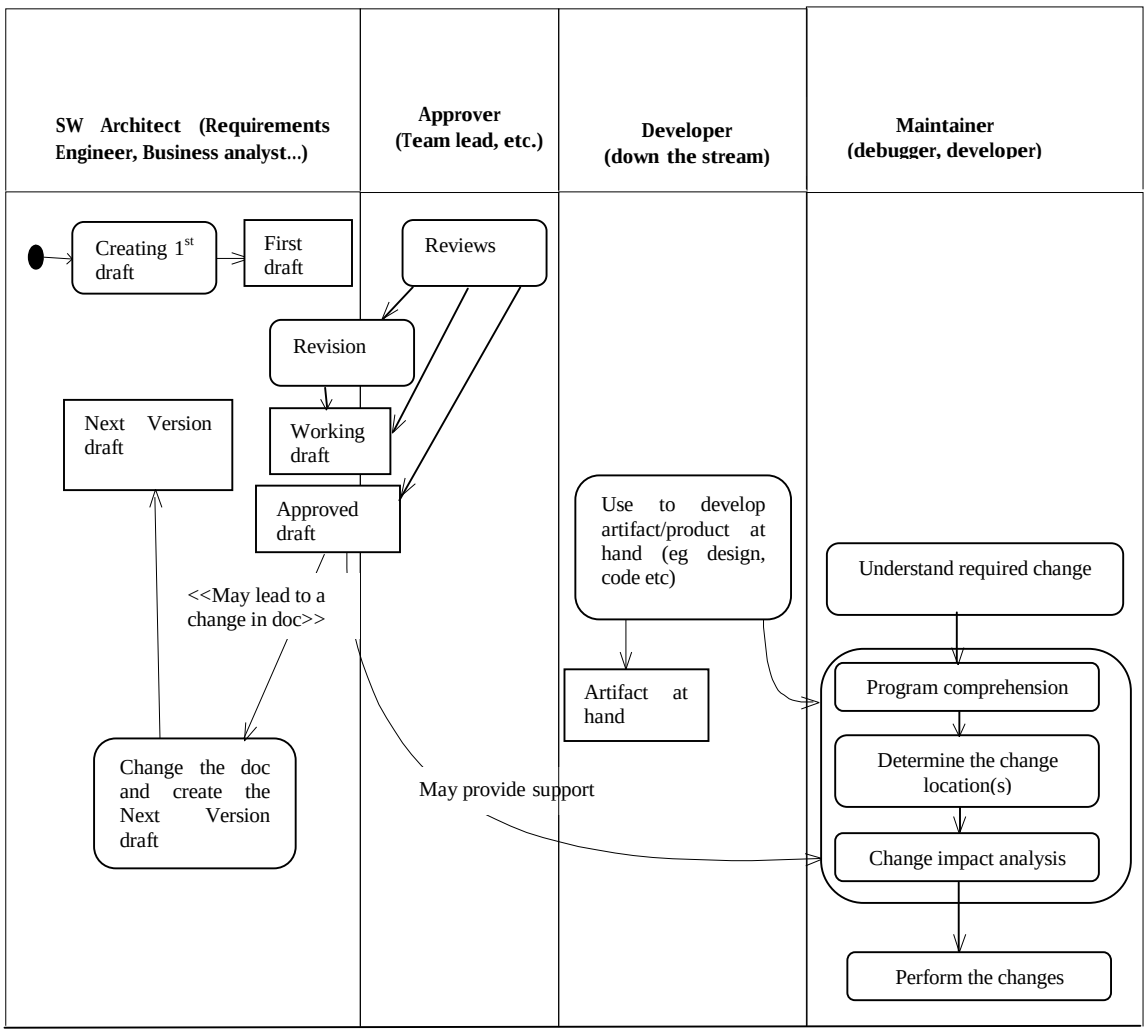

Fig. 4 Documentation cost and benefit model (Source: [8])

4. Application and benefits of documentation

The role of documentation in a software engineering environment is to communicate information to its audience and instill knowledge of the system it describes [1]. As seen in Bo Sun's model, system documentation plays another role namely software maintenance.

Parnas [2] identified several benefits of documentation. These include: easier reuse of old designs, better communication about requirements, more useful design reviews, easier integration of separately written modules, more effective code inspection, more effective testing, and more efficient corrections and improvements.

Researchers and practitioners also have looked at the uses of software documentation and just to compile a few, [9] notes that documentation helps at software development, keeps software-quality at high levels and makes it easy to transfer projects. Documentation can also be used for [10] learning a software system, testing a software system, working with a new software system, solving problems when other developers are unavailable to answer questions, looking for big-picture information about a software system, maintaining a software system, answering questions about a system for management or customers, looking for in-depth information about a software system, working with an established software system. Akin-Laguda [11] lists other uses which include; facilitates effective communication regarding the system

between the technical and the non technical users, training new users, solve problems like trouble shooting, evaluation process, and quantify the financial ramifications/footprint of the system.
All the above mentioned uses of documentation, can be simplified and summarized as [5] puts; 1. Documentation is used as a communication medium between members of the development team and probably the clients, 2. Used for maintenance, 3. Provide information for management to help them plan, budget and schedule the software development process and, 4. Tell users how to use and administer the system.

5. Types of Documentation

Sommerville describes two main categories of software documentations; process and product documents. Process documentations are used to manage the development process for example planning, scheduling and cost tracking, standards among others. Product documentations describe the main deliverable (software product) and some of the documents in this category form part of deliverables. These include; Requirements Specification, Design documents, Commented Source Code, Test Plans including test cases, Validation and Verification plan and results, List of Known Bugs and user manual [5].

6. Tools for documentation

Forward and Lethbridge [12] in their survey found the following documentation tools more helpful; MS Word (and other word processors), Javadoc and similar tools (Doxygen, Doc++), Text Editors, and Rational Rose.
Doxygen is a documentation system for ***C***++, C, ***Java***, Objective C, Fortran, VHDL, PHP, C#. Doxygen generates; on-line documents in HTML, off-line manual in latex and output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and Unix man pages. It extracts documentation directly from the sources, which makes it much easier to keep the documentation consistent with the source code. Doxygen can be configured to extract the code structure from undocumented source files. This is very useful to quickly locate elements in large source distributions. Doxygen include; dependency graphs, inheritance diagrams, and collaboration diagrams, which are all generated automatically. These help to visualize the relations between the various elements. Doxygen can be used for creating normal documentation. Another important aspect of Doxygen is that, it is highly portable [13].
A word processor is a program that is used to produce, edit and format text. "Word processing systems are screen based. This means that the image of the document on the user's terminal is, more or less, the same as the final form of the printed document. Layout can be improved before printing the document"[5].
LATEX is a TEX macro package that simplifies the use of TEX. It is portable and come with its set of fonts. Most LATEX commands are "high-level" and specify the logical structure of a document. This simplifies the authors work by taking care of document layout details. LATEX provides several standard document classes from which to choose. The document class determines how the document will be formatted. Other benefits of LATEX include; its flexibility, gives the user complete control, handles big, complex documents with ease, and never crashes or corrupts users' files [14].

7. Software Documentation standards

Standardized documentation can be defined as documents having a consistent appearance, structure and quality. This means should be easier to read, understand and usable [5], [15]. Standards act as a basis for document quality assurance.
"Using a standard means that documentation producers and customers have a consistent accepted reference for the format and content that they will find in the documentation. For example, what documentation must be printed? What does it mean to say that the documentation is "complete"? Does it have to include every function and screen shot?" [15].
Reilly discusses various ISO Software documentation standards which include;
ISO/IEC/IEEE 26514:2008, *Systems and software engineering-Requirements for designers and developers of user documentation*. This standard details both process and product standards.
ISO/IEC/IEEE 26513:2009, *Software and systems engineering-Requirements for testers and reviewers of user documentation*. This standard covers the activities and responsibilities for planning and conducting documentation reviews and managing the results of the review. It also addresses how to plan, measure, and conduct usability tests for documentation, along with tests of accessibility and of localized or customized versions.
ISO/IEC/IEEE 26512, *Software and systems engineering-Requirements for acquirers and suppliers of user documentation.* It lays out the processes for acquiring user documentation services and for monitoring and managing contractors.

IEEE Std 1063-2001, *IEEE Standard for Software User Documentation.* This standard is a revision of IEEE std 1987. It provides minimum requirements for the structure, information content, and format of user documentation, including both printed and electronic documents used in the work environment by users of systems containing software. This standard is limited to the software documentation product and does not include the processes of developing or managing software user documentation; it applies to printed user manuals, online help, and user reference documentation [16].

8. Conclusion

Software documentation is an activity of creating documents which are used in software development environment to communicate functions, operations and events to various stakeholders, for example software Requirements Engineers, Reviewers, Developers, operators, Maintainers of the system among others. Documents describe the product at all levels of development including the finished product. The documents also act as evidence of all the procedures and activities involved in software development therefore, documents need to be up-to date, complete, consistent and usable. To achieve consistency, systematic ways of document creation should be employed.

References

[1] A. Forward, "Software Documentation – Building and Maintaining Artefacts of Communication". MS thesis. Institute for Computer Science, Ottawa-Carleton. Canada. 2002.

[2] Parnas, D.L.: Precise documentation: The key to better software. In: Nanz, S. (ed.) The Future of Software Engineering, 2011, pp. 125–148. Springer, Heidelberg

[3] S. Capri, Developing Successful Software Documentation, 2006. [Accesed: 14th August,2013], Available online:<http://www.softwareceo.com/downloads/files/sceo/white papers/DevelopingSuccessfulSoftwareDocumentation.pdf>

[4] Elowe, The role of documentation in software development. 2006. [Accessed: 9th July,2013]. Available online:< https://blogs.oracle.com/elowe/entry/the_role_of_documentation_in>

[5] I. Sommervile, "Software Documentation", 2001.Available from World Wide Web: <http://www.literateprogramming.com/ documentation.pdf>.

[6] F. Bachmann, L. Bass, J. Carriere, P. Clements, D. Garlan, J. Ivers, R. Nord, R. Little, Software Architecture Documentation in Practice: Documenting Architectural Layers. CMU/SEI-2000-SR-004. Carnegie. Mellon University. 2000.

[7] F. Shull, "Developing Techniques for Using Software Documents: A Series of Empirical Studies". PhD thesis, Computer Science Department, University of Maryland, USA. 1998.

[8] Bo Sun, "A Methodology for Analyzing Cost and Cost-Drivers of Technical Software Documentation". MS thesis, University of Calgary. Alberta. 2012.

[9] A. Berger Guidelines: Technical Documentation Standard for Software development. Specific Group software solutions. v 1.3. 2010.

[10] T. C.Lethbridge, J.Singer, A.Forward, How Software Engineers Use Documentation: The State of the Practice, IEEE Software, v.20 n.6, 2003.p.35-39, [doi>10.1109/MS.2003.1241364]

[11] F. Akin-Lagida, Documentation in programming. NASSCOM. 2013. [Accessed online: 18th July, 2013]. Available:<http://www.slideshare.net/itniketan/importance-of-documentation>

[12] A. Forward and T.C. Lethbridge, "The Relevance of Software Documentation, Tools and Technologies: A Survey," *Proc. ACM Symp. Documentation Eng.*, ACM Press, 2002, pp. 26-33.

[13] Doxygen [Accessed: 16th July, 2013],Available:<http://www.stack.nl/~dimitri/doxygen/>

[14] Academic and Research Computing, Text Formatting with LATEX. A Tutorial. RPI, 2007. [Accessed on 14th August, 2013]. Available online:<www.unc.edu/depts/econ/egsa/LaTeX.pdf>

[15] A. Reilly, Audience-Oriented Standards for Software Documentation from ISO. Intercom; Vol. 58 Issue 3, 2011, p14-17

[16] IEEE, *Standard for Software User Documentation*, IEEEStd1063-2001. New York: Institute of Electrical and Electronics Engineers. 2001.

Noela Jemutai Kipyegen holds Master of Science in Software Engineering, awarded by Jomo Kenyatta University of Agriculture and Technology, Kenya in 2010. Bachelor of Science in Computer Science, from Egerton University, Kenya, awarded in 2006. Currently, she is an Assistant Lecturer and Researcher in the Department of Computer Science, Egerton University, Kenya. Research interests include Software Engineering Project Management, ICT for Development, looking at Human Computer Interaction (HCI). Publication: Kipyegen Noela J., Waweru Mwangi, Stephen Kimani,

Risk Management Adoption Framework for Software Projects: A Case Study for Kenyan Software Project Managers and Developers. International Journal for Computer Science Issues (IJCSI) Vol 9, Issue 3, May 2012.

William Paul Kiplangat Korir is a senior Lecturer at Egerton University. He holds MSc. in Computer Science from University of Regina, Canada, BSc. (Hons), University of Nairobi, Kenya. He has published several papers found in his profile, <http://www.egerton.ac.ke/index.php/Computer-Science/kiplangat-korir-william-pau.html>.