Int. J. Advanced Networking and Applications
Volume: 06 Issue: 06 Pages: 2563-2572 (2015) ISSN: 0975-0290

Software Project Documentation - An Essence of Software Development

Vikas S. Chomal
[1]Assistant Professor
[2]Research Scholar
[1]The Mandvi Education Society Institute of Computer Studies,
Mandvi, Gujarat, India
[2]Singhania University, Pacheri Bari, District – Jhunjhunu, Rajasthan
Email: vikschomal80@gmail.com

Dr. Jatinderkumar R. Saini
[1]Director I/C & Associate Professor
[2]Research Supervisor
[1]Narmada College of Computer Application,
Bharuch, Gujarat, India
[2]Singhania University, Pacheri Bari, District – Jhunjhunu, Rajasthan
Email: saini_expert@yahoo.com

----------ABSTRACT----------

Software documentation is a critical attribute of both software projects and software engineering in general. Documentation is considered as a media of communication among the parties involved during software development as well the one who will be using the software. It consists of written particulars concerning software specifications as well as what it does, in which manner it accomplishes the specified details and even how to exercise it. In this paper, we tried to focus on the role of documentation in software projects.

Keywords: Documentation, Software Engineering, Software Project Documentation, Software Projects

Date of Submission: March 07, 2015 Date of Acceptance: May 07, 2015

I. INTRODUCTION

Software documentation is an essential feature of both software projects and software engineering in common. In piece of evidence, documentation engineering has become an accepted sub-domain in the software engineering society. The task of documentation in a software engineering milieu is to commune information to its spectators and instils knowledge of the system it describes [2][19]. Documentation is requisite in software development. Even though every software development project is exclusive and produces diverse categories of documents, different amount of documentation, and may employ different documentation methods and notations, we need to be able to control the documentation produced in software development projects in a uniform manner [3][30]. Documentations is the process of collecting, organizing, storing and maintaining historical record of programs and other documents used or prepared during the different phases of the life cycle of the software [13][14].Software development is partly a learning and communication process. Software developers need to communicate with each other and also with various interest groups of the system to be developed, such as customers, marketing people, end users, service personnel, and authorities. Documentation is the basis for communication in software development organizations as well as between development organizations and the interest groups of the system to be developed [28][41].

II. LITERATURE REVIEW

Ambler et al [1] describes the issues concerning the changing needs of documentation. In particular, Ambler says that "during development you're exploring both the problem and solution spaces, trying to understand what you need to build and how things work together. Post development you want to understand what was built, why it was built that way, and how to operate it". Cockburn [18], as well as Ambler et al [1] presents an alternate view concerning the role of documentation. They argue that the purpose of documentation is to convey knowledge – something that can be different from merely providing information. Cockburn argues that source code presents the facts of a system and the supporting documents facilitate higher-level interpretation of those facts. A document that instils knowledge in its audience can then be deemed effective, somewhat regardless of its age and the extent to which it is up-to-date [18].

Laitinen [28] puts forward that software development is supposed to be documentation-oriented, which means that documents are considered to be the most essential and valuable products of the development process. Documentation-orientedness involves considering such computer-process able products as source program modules and batch-files as documents. On the other hand, a product such as executable machine code is regarded as

Volume: 06 Issue: 06 Pages: 2563-2572 (2015) ISSN: 0975-0290

a by-product in the development process, because in a development environment we can always derive correct executable machine code when we have the correct documents. The executable machine code is essential to the user of a computer system, but it is considered less important to the software developers.

Software development is partly a learning and communication process. Software developers need to communicate with each other and also with various interest groups of the system to be developed, such as customers, marketing people, end users, service personnel, and authorities. Documentation is the basis for communication in software development organizations as well as between development organizations and the interest groups of the system to be developed. To ensure efficient communication, all communicating parties need to be able to identify various software documents, and, to ensure that the right information is found, all communicating parties should be able to anticipate what information is in each document [26][27].

Hager & Kellner et al [25][29] states that documentation is probably most crucial to the maintenance phase, which accounts for 60%75% of the total cost of the software. Osborne [37] reports that documentation accounts for more than 60% of maintenance costs, and is involved in about one-third of the maintenance tasks. A quick understanding of the existing software is a key activity of the maintenance process. Chapin [11] asserts that maintenance people spend 40% of their time dealing with documentation. Fjel et al [23] showed that when making a program modification 47% of a maintenance programmer's time is spent studying the program source code and associated documentation. They also found that when correcting errors, the time increases to 62%. Documentation has appropriately been called the castor oil of software process-it is good for you but tastes awful. Far too often documentation may not exist, or if it does exist, it may be incomplete, inaccurate, or out of date.
Basili et al [4] studied an industrial maintenance environment and found that 20% of the maintenance problems are due to bad documentation, with the most frequent problems being documentation faults and documentation clarifications. They claim that better documentation can solve a big percentage of maintenance problems. According to Chapin [10], maintenance programmers report that for most maintenance tasks the source code is the only available documentation. Buckley [8] claims that in most cases maintainers discover that the available documentation is not current; Poston [38] asserts that flawed or outdated documentation is more costly than no documentation.

Poor quality documentation is a major problem. In a survey of 487 data processing organizations, Lien et al [33] found documentation quality ranked 3rd in the list of 26 maintenance problem items. They identify documentation quality and adequacy of design specs as accounting for 70% of product quality. Guimaraes et al [24] claims that the documentation rating has an inverse relationship with the average yearly maintenance expenditures and that maintenance programmers felt that the most important document was an "English narrative describing what the programs and modules are supposed to do". Documentation impacts the analysis and development phases as well. Boehm [7] estimated that documentation costs run about 10% of total Software Development costs. Scheff et al [40] found that 85% of all software development errors are introduced during requirements, analysis and design. Basili et al [5] conducted a study to analyze the factors that cause errors and found that misunderstanding of a module's specifications or requirements constituted the majority of detected errors. Card et al [9] studied a production environment to evaluate the effectiveness of different technologies and their impact on productivity and reliability. They found that high-use of documentation improves productivity by 11% and reliability by 27% compared to low-use of it. To improve quality, they suggest effective documentation of each phase of development. Fagan [22] claims that documentation quality inspections are as important as program inspections when the goal is to increase productivity and final software quality.

The effectiveness of documentation within a development process is determined by the way in which the intentions of the authors correspond to the expectations of the potential readers. In a typical software development process, many different kinds of documents are produced and consumed at various points in time. The contents of those documents necessarily exhibit a certain amount of overlap. People may lose track of the meaning of individual documents; which information it contains and what its role is in the development process. When the expectations of the consumers of the documentation drift too far from the intentions of its producers, the ultimate consequence might be a need to rediscover already documented knowledge. In such a situation, customers for example may need to explain their situation and requirements over and over again to different parties in the development process [2][15][16].

Laitinen [28] identified the following task when document is applied in practice:

1) Certain quality control policies (e.g. walkthroughs, reviews, and inspections) must be assigned to certain types of documents. For example, some Development Plans need to be reviewed with customers, and Utilization Documents usually need a different kind of quality control than Software Descriptions.
2) For every document or document-type that is considered necessary in software development documentation guidelines, document templates, and checklists for validating a document should be created.

Volume: 06 Issue: 06 Pages: 2563-2572 (2015) ISSN: 0975-0290

3) When a specific development method is used the correspondence between the documents and/or models produced by a method and the documents needs to be specified.
4) An appropriate version control policy must be established for every document class. A general rule is that all document classes excluding Quality Control and Administrative Documents need version control.
5) Various roles are needed in software development [14]. It may be beneficial to decide that certain documents or document types are created, maintained, stored and/or collected by a certain role according to the software quality system.

Bill et al [22] believe that documentation should focus on how requirements and design decisions were made, represented, communicated and changed over the lifespan of a software system. As well, documentation should describe the impact of the current system on future development processes. Their study involved interviewing personnel from seventeen large software projects. Their analysis focused on the problems of designing large software systems; but many results report directly about the use (and misuse) of documentation in a software project. Thomas [42] raises several fundamental questions in their discussion about software documentation.

1) What types of documentation does a software engineer (or support staff member) need?
2) Who should produce, maintain and verify documentation to assure an appropriate level of quality?
3) Why should documentation be produced at all?

Forward et al [3] specify procedures for managing user documentation throughout the software life cycle. It applies to people or organizations producing suites of documentation, to those undertaking a single documentation project, and to documentation produced internally, as well as to documentation contracted to outside service organizations. It provides an overview of the software documentation and information management processes, and also presents aspects of portfolio planning and content management that user documentation managers apply. It covers management activities in starting a project, including setting up procedures and specifications, establishing infrastructure, and building a team. It includes examples of roles needed on a user documentation team. It addresses measurements and estimates needed for management control, and the use of supporting processes such as change management, schedule and cost control, resource management, and quality management and process improvement. It includes requirements for key documents produced for user documentation management, including documentation plansand documentation manage ment plans. ISO/IEC/IEEE 26511:2012 is independent of the software tools that may be used to produce or manage documentation, and applies to both printed documentation and on-screen documentation. Much of its guidance is applicable to user documentation for systems including hardware as well as software. While there is no universally recognized standard for software documentation, there is a standard for documenting engineering and scientific software. Developed by the American National Standards Institute (ANSI) and the American Nuclear Society (ANS) in 1995, it is called the ANSI/ANS 10.3-1995 Standard for Documentation of Computer Software. The standard provides a flexible, robust framework for documentation needs. One of its goals is to encourage better communication between developer and user and to facilitate effective selection, usage, transfer, conversion and modification of computer software. The standard is not a rigid set of specifications but a guide that can apply to most software projects intended for internal or external use. While the standard cannot cover all documentation problems, it is a good starting point, even for the most complex software. Similarly, while the standard provides recommendations for documenting scientific and engineering software, it doesn't offer guidance for online monitoring, control or safety systems, and doesn't specifically address the unique requirements of consumer-oriented software. As a general guideline for clear, well-organized documentation, however, the ANSI/ANS 10.3-1995 standard can serve as a place for developers to begin a documentation methodology. The standard is fairly comprehensive, and it allows for individual developer differences and unique software documentation problems [39].

Quian [32] states that, the software documentation is unpopular among many developers at present while the documents are important for the staffs who work for secondary development and software maintenance. For this phenomenon, Quain proposed a teaching method of writing software documentation, in which the software maintenance is the driving force to make students fully understand and grasp the method of software documentation writing through an upgrade and maintenance software project. And students learn to write effective software documentation to establish the level and structure of the document.

Lethbridge [17] put forward that, software engineering is a human task, and as such we must study what software engineers do and think. Understanding the normative practice of software engineering is the first step toward developing realistic solutions to better facilitate the engineering process. We conducted three studies using several data-gathering approaches to elucidate the patterns by which software engineers (SEs) use and update documentation. Their objective is to more accurately comprehend and model documentation use, usefulness, and maintenance, thus enabling better decision making and tool design by developers and project managers. Our results confirm the widely held belief that SEs typically does not update documentation as timely or completely

Volume: 06 Issue: 06 Pages: 2563-2572 (2015) ISSN: 0975-0290

as software process personnel and managers advocate. However, the results also reveal that out-of-date software documentation remains useful in many circumstances. Choudhary [17] disclose the challenges faced by the documentation team in a globally distributed setting have not received much attention. In their work they highlighted on the challenges faced by the software documentation team in a globally distributed product development team. Further the paper elaborates on the solutions implemented, successes and failures of the team. This also includes the learning of the team as it aligned with the Lean model of Software Development. Magyar [34] state the work carried out at Spatial Technology Inc. which has 45-60 developers whose main task is of writing code. The Technical Publications Department has three writers to keep up with them, plus two programmers who develop and maintain software tools that support the documentation. This team puts out paper and online documentation, on time, reasonably complete, for one major and several minor releases a year. It's the tools that make it possible. Magyar first describes the tools and processes that were in place, explain the issues that led us to change them, and discuss the changes that were put forward in place both for tools and process. Magyar also discusses some of the problems that were faced and how they dealt with them.

Chomal and Saini [12][13] in their work considered documentation of software projects prepared by students as a source for data collection. Specifically, documentations of large software projects of only final year students of Masters level course have been considered for the research purpose. The duration of these software projects is six months. The said documentations of software projects were procured from college libraries. These documentations include complete project profile along with the following elements:
1) Requirement analysis
2) Technology used
3) Database design
4) Structural and Object Oriented Modelling Techniques
5) Screen layouts
6) Testing techniques along with test case and data

They analyzed and reviewed 505 large software project documentations developed during a period of academic years from 2001-2002 to 2011-2012. During our exploration we considered all of the above described elements. For simplicity and better exhaustive analysis of the documentations, the phased process was followed. As each project is uniquely different definition from other projects, it is noteworthy here that this was repeated for each of the 505 project reports under study. These phases are presented below:
1) Exploration of Project Profile
2) Exploration of Existing System and Proposed System
3) Exploration of Requirement Gathering Techniques
4) Exploration of Requirement Analysis done by Students
5) Exploration of Technology on which Software Project carried out
6) Exploration of Process Model adapted for Software Project Development
7) Exploration of Data Dictionary (including Database Design)
8) Exploration of various Structural and Object Oriented Modelling Techniques
9) Exploration of Screen Layouts
10) Exploration of Generated Reports
11) Exploration of Testing Techniques and Test data

In their present work, they identified 103 software attributes from software project documentations.

Forward [3] discusses how certain attributes contribute to a document's effectiveness. They conducted a survey and asked the participants how important particular document attributes contribute to its overall effectiveness. Participants gave rating between 1 (least important) and 5 (most important). Dragicevic et al [21] in their work considered problems of elicitation, documentation and validation of user requirements, and implicates the need for method that enables the stakeholders to resolve problems of incomplete, incorrect and contradictory requirements in the earliest possible phase of project. In their review of research literature they showed that the existing methods are primarily intended for requirement engineering and software engineering professionals and that there is a lack of method that will ensure the active role of business users. They define new method of elicitation, documentation and validation of users requirements based on complementary application of Event Process Chain method and UML language. Its experimental part verifies and evaluates the suggested method on specific project of customized software development. This method is suitable for early software size estimation. New metric for estimation of software size and complexity is developed. Lepasaar et al [31] articulate that, in a small software organization, a close and intense relationship with its customers is often a substitute for documentation along the software processes. Nevertheless, according to the quality standards, the inadequacy of the required documentation will retain the assessed capability of software processes on the lowest level.
Their article describes the interconnections between software process documentation and the maturity of the organization. The data is collected from the SPICE assessment results of small and medium sized software organizations in Finland. The aim of their article is to visualise the necessity of documentation throughout the software engineering processes in order to achieve a higher capability level. In addition they pointed out that processes with insufficient documentation decrease the chance to improve the quality of the processes, as it is impossible to track and analyse them.

Volume: 06 Issue: 06 Pages: 2563-2572 (2015) ISSN: 0975-0290

Bayer [6] speaks about documentation by stating that, documentation is an integral part of a software system. It contains the information that is necessary to effectively and successfully develop, use, and maintain a system. In practice, however, the creation of appropriate documentation is largely neglected. In their paper they investigated the reasons for this neglect, presents view-based software documentation; their approach was to improve the current situation, and reports on empirical evidence in support of the presented approach. Because the quality of documentation depends on its usage, view-based software documentation exploits existing software modeling techniques to provide all users of documentation with the documentation they require f or performing their tasks. View-based software documentation has been empirically validated in a series of experiments and case studies that showed that the approach improves the completeness, correctness, and usefulness of produced and maintained documentation.

Liu et al [20] verbalize that; studying software development processes can help us to understand the software development models which in turn can help programmers to build high-quality software products. Software is not all homogeneous, and industrial software and software developed in academia seem to be different. In order to understand the characteristics of academic software development, they surveyed ten student programmers in five research fields and conducted content analysis. They found that although academic software is highly diverse, the development processes are fairly similar to some extent. They also found some common weaknesses, such as lacking of code management and documentation, and proposed some suggestions to improve the process.

Nasution et al [36] implies that, agile software development methods seem inherently suitable for today's quick-paced business environment as they shorten the time to develop new systems and typically incur lower development costs compared to the conventional systems development life cycle approach. Software development project failures using conventional SDLC are often attributed to project delays, resulting in budget overruns. On the other hand, a well planned and documented systems development project is more likely to result in a system that meets the expectations of both the intended users and the software engineers. In their work they takes another look at conventional SDLC methodology by focusing on an aspect that is often overlooked in systems development practice, namely the significance of good documentation.

Wallace et al [43] illustrates that most undergraduate information Systems courses use some sort of Computer-Aided Systems and Software Engineering (CASE) tool to help the System Designers (cadets) graphically depict the proposed System under construction. Currently, at the United States Military Academy, they were in the process of identifying, evaluating and selecting an appropriate CASE tool for use by their Computer Science Engineering Sequence cadets. The cadets who will use the CASE tool are seniors completing a capstone design project with a local client. They have become system designers who must build an Information System to meet the needs of their client. The cadets only have 2 semesters to learn how to use a CASE tool and apply it to their system design using the six phases of the Systems Development Life Cycle (SDLC). The current CASE tool available to them is very robust and non user-friendly. As a result, little value is currently gained from the use of this CASE tool. That is why it is vital that a new user friendly CASE tool is acquired. They developed a ten step method that will evaluate and select the most users friendly and cost efficient CASE tool for the cadets, which will ultimately improve present and future information System Designs. This method can take up to ten months from developing an initial scoring criterion to the final selection and procurement of a meaningful CASE tool

Mitchell et al [35] conveyed an approach for designing multi-factor scoring systems for evaluating and selecting early stage innovation projects. A project is a piece of work of finite duration with finite resources, aimed at a defined outcome. Innovation projects have the extra complication that all of these aspects will be somewhat uncertain and knowledge of them is liable to change as the project proceeds. Clearly different assessment factors are required for different organizations, and for different types and stages of project. There is little guidance in the literature on how to choose the factors and how best to structure the scoring process. They presented approach in the form of managerial guidelines, targeted at those who have to implement innovation project selection systems. Design aspects are discussed, including structure of the tool, choosing the factors, scaling statements, weightings, risk, uncertainty and confidence. Management aspects are considered, including preparation, scoring, decisions and outputs. The method is positioned in terms of theory and practice, with reference to both literature and industrial case studies.

III. CONCLUSION

From the surveyed work premeditated by us on documentation of software project, we wrap up the highlights about considering documentation as the essence of software project.

Volume: 06 Issue: 06 Pages: 2563-2572 (2015) ISSN: 0975-0290

Table I. Conclusion

Sr No.	References	Highlights
1	[1][18]	Describes the issues concerning the changing needs of documentation. Also presents an alternate view concerning the role of documentation.
2	[2][19]	The task of documentation in a software engineering milieu is to commune information to its spectators and instils knowledge of the system it describes.
3	[3]	Specify procedures for managing user documentation throughout the software life cycle. It applies to people or organizations producing suites of documentation, to those undertaking a single documentation project, and to documentation produced internally, as well as to documentation contracted to outside service organizations.
4	[4]	They claim that better documentation can solve a big percentage of maintenance problems.
5	[5]	Conducted a study to analyze the factors that cause errors and found that misunderstanding of a module's specifications or requirements constituted the majority of detected errors.
6	[6]	Speaks about documentation by stating that, documentation is an integral part of a software system. It contains the information that is necessary to effectively and successfully develop, use, and maintain a system. In practice, however, the creation of appropriate documentation is largely neglected.
7	[7]	Estimated that documentation costs run about 10% of total Software Development costs.
8	[8]	Claims that in most cases maintainers discover that the available documentation is not current.
9	[9]	Studied a production environment to evaluate the effectiveness of different technologies and their impact on productivity and reliability. They found that high-use of documentation improves productivity by 11% and reliability by 27% compared to low-use of it.
10	[10]	Maintenance programmers report that for most maintenance tasks the source code is the only available documentation.
11	[11]	Asserts that maintenance people spend 40% of their time dealing with documentation.
12	[12]	In their work considered documentation of software projects prepared by students as a source for data collection. Specifically, documentations of large software projects of only final year students of Masters level course have been considered for the research purpose.
13	[13][14]	Documentations is the process of collecting, organizing, storing and maintaining historical record of programs and other documents used or prepared during the different phases of the life cycle of the software.
14	[15][16]	The effectiveness of documentation within a development process is determined by the way in which the intentions of the authors correspond to the expectations of the potential readers. In a typical software development process, many different kinds of documents are produced and consumed at various points in time. The contents of those documents necessarily exhibit a certain amount of overlap.
16	[18]	Argues that source code presents the facts of a system and the supporting documents facilitate higher-level interpretation of those facts. A document that instils knowledge in its audience can then be deemed effective, somewhat regardless of its age and the extent to which it is up-to-date.
17	[20]	Verbalize that; studying software development processes can help us to understand the software development models which in turn can help programmers to build high-quality software products.
18	[21]	In their work they considered problems of elicitation, documentation and validation of user requirements, and implicates the need for method that enables the stakeholders to resolve problems of incomplete, incorrect and contradictory requirements in the earliest possible phase of project.

Volume: 06 Issue: 06 Pages: 2563-2572 (2015) ISSN: 0975-0290

19	[22]	Claims that documentation quality inspections are as important as program inspections when the goal is to increase productivity and final software quality.
20	[23]	Showed that when making a program modification 47% of a maintenance programmer's time is spent studying the program source code and associated documentation. They also found that when correcting errors, the time increases to 62%.
21	[24]	Documentation impacts the analysis and development phases as well.
22	[25][29]	States that documentation is probably most crucial to the maintenance phase, which accounts for 60%75% of the total cost of the software.
23	[26][27]	To ensure efficient communication, all communicating parties need to be able to identify various software documents, and, to ensure that the right information is found, all communicating parties should be able to anticipate what information is in each document.
24	[28][41]	Documentation is the basis for communication in software development organizations as well as between development organizations and the interest groups of the system to be developed.
25	[30]	Documentation is requisite in software development. Even though every software development project is exclusive and produces diverse categories of documents, different amount of documentation, and may employ different documentation methods and notations, we need to be able to control the documentation produced in software development projects in a uniform manner.
26	[31]	Articulate that, in a small software organization, a close and intense relationship with its customers is often a substitute for documentation along the software processes.
27	[32]	States that, the software documentation is unpopular among many developers at present while the documents are important for the staffs who work for secondary development and software maintenance.
28	[33]	Found that documentation quality ranked 3rd in the list of 26 maintenance problem items. They identify documentation quality and adequacy of design specs as accounting for 70% of product quality.
29	[34]	This team puts out paper and online documentation, on time, reasonably complete, for one major and several minor releases a year. It's the tools that make it possible. Magyar first describes the tools and processes that were in place, explain the issues that led us to change them, and discuss the changes that were put forward in place both for tools and process.
30	[35]	Conveyed an approach for designing multi-factor scoring systems for evaluating and selecting early stage innovation projects.
31	[36]	A well planned and documented systems development project is more likely to result in a system that meets the expectations of both the intended users and the software engineers. In their work they takes another look at conventional SDLC methodology by focusing on an aspect that is often overlooked in systems development practice, namely the significance of good documentation.
32	[37]	Reports that documentation accounts for more than 60% of maintenance costs, and is involved in about one-third of the maintenance tasks. A quick understanding of the existing software is a key activity of the maintenance process.
33	[38]	Asserts that flawed or outdated documentation is more costly than no documentation.
34	[39]	A general guideline for clear, well-organized documentation, however, the ANSI/ANS 10.3-1995 standard can serve as a place for developers to begin a documentation methodology. The standard is fairly comprehensive& it allows for individual developer differences & unique software documentation problems.
35	[40]	Found that 85% of all software development errors are introduced during requirements, analysis and design.
36	[42]	Raises several fundamental questions in their discussion about software documentation. These questions including matters regarding types of documentations, contents, maintenance and so on.

Volume: 06 Issue: 06 Pages: 2563-2572 (2015) ISSN: 0975-0290

37	[43]	Illustrates that most undergraduate information Systems courses use some sort of Computer-Aided Systems and Software Engineering (CASE) tool to help the System Designers (cadets) graphically depict the proposed System under construction. Currently, at the United States Military Academy, they were in the process of identifying, evaluating and selecting an appropriate CASE tool for use by their Computer Science Engineering Sequence cadets.

REFERENCES

[1] Ambler, Scott and Ron Jeffries. Agile Modelling: Effective Practices for Extreme Programming and the Unified Process, Chapter 14, John Wiley & Sons, 2002.

[2] Andrew J. Forward, "Software Documentation – Building and Maintaining Artefacts of Communication", presented to the Faculty of Graduate and Postdoctoral Studies in partial fulfilment of the requirements for the degree Master in Computer Science, Ottawa – Carleton Institute of Computer Science, University of Ottawa, Canada, 2002.

[3] Andrew J. Forward, Timothy, "Qualities of Relevant Software Documentation: An Industrial Study", available from psu.edu.

[4] Basili, Victor R. and Musa, John D. The future engineering of software: a management perspective. IEEE Computer, September, 1991.

[5] Basili, Victor R. and Perricone, Barry T. Software errors and complexity: An empirical investigation. Communications of the ACM, Vol.27, No. 1, 1984.

[6] Bayer, J., A view-based approach for improving software documentation practices Published in: Engineering of Computer Based Systems, 2006. ECBS 2006. 13th Annual IEEE International Symposium and Workshop on Date of Conference: 27-30 March 2006 Page(s): 10 pp. – 278 Print ISBN: 0-7695-2546-6 INSPEC Accession Number: 9017024 Publisher: IEEE

[7] Boehm, Barry W. The high cost of software. From Horowitz, Practical Strategies for Developing Large Software Systems, Addison-Wesley, Reading, Mass, 1975.

[8] Buckley, J. Some standards for software maintenance.Standards, IEEE Computer, November 1989.

[9] Card, David N., MC Garry, Frank E. and Page, Gerald T. Evaluating software engineering technologies. IEEE Transactions on software engineering, Vol. SE13, No. 7, 1987.

[10] Chapin, Ned. Software maintenance life cycle. Proceedings Conference on Software Maintenance, IEEE, 1988.

[11] Chapin, Ned. The job of software maintenance. Proceedings Conference on Software

[12] Chomal V.S. , Saini J.R., "Finding Trend of Both Frequency and Type of Errors from Software Project Documentation‖, International Journal of Emerging Trends and Technology in Computer Science (IJETTCS)ISSN 2278-6856, Volume 2, Issue 5, September – October 2013

[13] Chomal V.S. , Saini J.R., "Software Quality Improvement by Documentation – Knowledge Management Model‖, National Journal of System And Information Technology ISSN : 0974 – 3308, Volume 6, Number 1, June 2013, Page Number: 49 – 68

[14] Chomal V.S. , Saini J.R., "Software Template to Improve Quality of Database Design on basis of Error Identification from Software Project Documentation‖, International Journal of Engineering and Management Research ISSN No.: 2250-0758,Volume-4, Issue-1, February-2014, Page Number: 168-179

[15] Chomal V.S. , Saini J.R., "Significance of Software Documentation in Software Development Process" International Journal of Engineering Innovation and Research, ISSN: 2277 – 5668, Volume 3, Issue 4

[16] Chomal V.S. , Saini J.R., "Identification, Categorization and Weighting of Software Engineering Attributes for Quality Evaluation of Software Project Documentation", International Journal of Advanced Networking Applications (IJANA), ISSN No: 0975 – 0290, page – 53, 2014.

[17] Choudhury, J. Software Documentation in a Globally Distributed Environment aligned with the Lean model of Software Development. Published in: Global Software Engineering (ICGSE), 2014 IEEE 9th International Conference on Date of Conference: 18-21 Aug. 2014 Page(s): 90 – 94 INSPEC Accession Number: 14649864 Publisher: IEEE

[18] Cockburn, A. Agile Software Development, Addison-Wesley Pub Co, 2001.

[19] Curtis, Bill, Herb Krasner, and Neil Iscoe. A Field Study of the Software Design Process for Large Systems. Communications of the ACM 31(11):1268-1287, November, 1988.

[20] Dapeng Liu ; Shaochun Xu ; Brockmeyer, M. Investigation on Academic Research Software Development Published in: Computer Science and Software Engineering, 2008 International Conference on (Volume:2) Date of Conference: 12-14 Dec. 2008 Page(s): 626 – 630 Print ISBN: 978-0-7695-3336-0 INSPEC Accession Number: 10427109 Publisher: IEEE

[21] Dragicevic, S. ; Split Airport, Kastela, Croatia ; Celar, S. Method for elicitation, documentation and validation of software user requirements (MEDoV) Published in: Computers and Communications (ISCC), 2013 IEEE Symposium on Date of Conference: 7-10 July 2013 Page(s): 000956 – 000961 INSPEC Accession Number: 14146909 Conference Location Publisher: IEEE

[22] Fagan, M.E. Design and code inspections to reduce errors in program development. IBM Systems Journal, Vol. 15, No. 3, 1976.

[23] Fjelstad, R.K and Hamlen, W.T. Application program maintenance study report to our respondents. Proceedings GUIDE 48, Philadelphia, PA, 1979.

[24] Gamalel-Din, Shehab A. and Osterweil, Leon J. New perspectives on software maintenance processes. Proceedings Conference on Software Maintenance, IEEE, 1988.

[25] Hager, James A. Software cost reduction methods in practice: a post mortem analysis. Journal of systems and software, Vol. 14, No.2, 1991.

[26] Janicki, Ryszard, David L. Parnas, and Jeffery Zucker, "Tabular representations in relational documents", In C. Brink, editor, Relational Methods in Computer Science. Springer-Verlag, 1996.

[27] June S. Hopkins and Jean M. Jeroow, "Documenting The Software Development Process", 1990 ACM 0 - 89791- 414 -7/90/1000 – 0125.

[28] Kari Laitinen, "Document Classification for Software Quality Systems", Technical Research Centre of Finland (VTT) Computer Technology Laboratory, ACM SIGSOFT Software Engineering Notes vol 17 no 4 Oct 1992 Page 32

[29] Kellner, Marc I. Non-traditional perspectives on software maintenance. Panel, Proceedings Conference on Software Maintenance, IEEE, 1989.

[30] Klare, George R, "Readable computer documentation", p148 – 167, ACM JCD, Volume 24,Communications of the ACM 31(11):1268-1287, November, 1988.

[31] Lepasaar, M. ; Varkoi, T. ; Jaakkola, H., Documentation as a software process capability indicator, Published in: Management of Engineering and Technology, 2001. PICMET '01. Portland International Conference on (Volume:1) Date of Conference: 2001, Print ISBN: 1-890843-06-7 INSPEC Accession Number: 7174877 Publisher: IEEE

[32] Lethbridge, T.C., How software engineers use documentation: the state of the practice Published in: Software, IEEE (Volume:20 , Issue: 6) Page(s): 35 – 39 ISSN : 0740-7459 INSPEC Accession Number: 7957107 Date of Publication : Nov.-Dec. 2003 Date of Current Version : 03 November 2003 Issue Date : Nov.-Dec. 2003 Sponsored by : IEEE Computer Society Publisher: IEEE

[33] Lientz, Bennet P. and Swanson, E. Burton. Problems in application software maintenance. Communications of the ACM, Vol. 24, No. 11, 1981.

[34] Magyar , Automating software documentation: a case study Published in: Professional Communication Conference, 2000. Proceedings of 2000 Joint IEEE International and 18th Annual Conference on Computer Documentation (IPCC/SIGDOC 2000) Date of Conference: 2000 Page(s): 549 – 558 Print ISBN: 0-7803-6431-7 INSPEC Accession Number: 6762904 Publisher: IEEE

[35] Mitchell, R. ; Phaal, R. ; Athanassopoulou, N. , Scoring methods for prioritizing and selecting innovation projects Publication Year: 2014 , Page(s): 907 – 920 IEEE CONFERENCE PUBLICATIONS

[36] Nasution, M.F.F. ; Weistroffer, H.R., Documentation in Systems Development: A Significant Criterion for Project Success Published in: System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on Date of Conference: 5-8 Jan. 2009 Page(s): 1 – 9 ISSN : 1530-1605 Print ISBN: 978-0-7695-3450-3 INSPEC Accession Number: 10467088 Publisher: IEEE

[37] Osborne, Wilma M. Building and Sustaining Software Maintainability. Proceedings Conference on Software Maintenance, IEEE, 1987.

[38] Poston, Robert M. When does more documentation mean less work?. Software Standards, IEEE Software, October 1984. Published in: Computer (Volume:30 , Issue: 10) Page(s): 97 – 98 ISSN : 0018-9162 INSPEC Accession Number: 5725215 Date of Publication : Oct 1997 Date of Current Version : 06 August 2002 Issue Date : Oct 1997 Sponsored by : IEEE Computer Society Publisher: IEEE

[39] Qian Hu Software documentation writing on the project of software upgrade and maintenance Published in: Computer Science and Information Processing (CSIP), 2012 International Conference on Date of Conference: 24-26 Aug. 2012 Page(s): 1400 – 1403 Print ISBN: 978-1-4673-1410-7 INSPEC Accession Number: 13055318 Publisher: IEEE

[40] Scheff, Benson H. and Georgon, Thomas. Using documentation blueprints to produce mandated DOD data items. Journal of Systems and Software, Vol. 14, No. 2, 1991.

[41] Scheff, Benson H. and Tom Georgon., "Letting software engineers do software engineering or freeing software engineers from the shackles of documentation", p81 – 91, SIGDOC '88, Ann Arbor, Michigan, USA, ACM Press, 1988.

Volume: 06 Issue: 06 Pages: 2563-2572 (2015) ISSN: 0975-0290

[42] Thomas, Bill, Dennis Smith and Scott Tilley, "Documentation for software engineers: what is needed to aid system understanding?", p 235 – 236, SIGDOC '01, Sante Fe, New Mexico

[43] Wallace, M.K.; Crow, J.A. Improving student information system design through evaluation and Selection of an appropriate CASE tool Published In: Frontiers in Education Conference, 1995. Proceedings, 1995 (Volume: 1) Date of Conference 1-4 Nov 1995 Page(s):2c3.16-2c3.19 vol.1 ISSN: 0190-5848 Print ISBN: 0-7803-3022-6 INSPEC Accession Number: 5225740 Publisher: IEEE

AUTHORS

Vikas Sitaram Chomal – M.Phil, MCA and Research Scholar at Faculty of Computer Science, Singhania University, Pacheri Bari, District – Jhunjhunu, Rajasthan – 333515. He has more than 7 years of rich teaching experience. Presently he is working as Assistant Professor at The Mandvi Education Society Institute of Computer Studies – MCA, Mandvi, District – Surat, Gujarat, India. Formely he was Assistant Professor (Ad hoc) at Narmada College of Computer Application – MCA, Bharuch, Gujarat, India. He worked as Principal (I/C) & Assistant Professor at Shri Manilal Kadakia College of Management & Computer Studies, Ankleshwar, Gujarat, India.

Jatinderkumar R. Saini was awarded Ph.D. in Computer Science by Veer Narmad South Gujarat University, Surat, Gujarat, India in the year 2009. He has more than 8 years of rich professional experience including working at Ministry of Info. Tech., New Delhi licensed CA under PKI at Ahmedabad, Gujarat, India. Presently he is working as Director (I/C) & Associate Professor at Narmada College of Computer Application, Bharuch, Gujarat, India. He is also the Director (I.T), GTU's Ankleshwar – Bharuch Innovation Sankul. Formely, he was Associate Professor & GTU Coordinator & HOD at S.P. College of Engineering, Visnagar, Gujarat, India.