Qualities of Relevant Software Documentation: An Industrial Study

Andrew J. Forward and Timothy C. Lethbridge
University of Ottawa, 800 King Edward, Ottawa, Ontario, Canada, K1N 6N5
aforward@site.uottawa.ca and tcl@site.uottawa.ca

Abstract

This paper highlights the results of a survey of software professional conducted in March and April, 2002. The results are compiled from 48 software professionals ranging from junior developers to managers and project leaders. One of the goals of this survey was to uncover the perceived relevance (or lack thereof) of software documentation, and the tools and technologies used to maintain, verify and validate such documents. Another goal was to uncover how software documentation is used in industry and the extent to which, and under what circumstances, documentation can be effective. The data suggest somewhat conflicting views of the importance of documentation maintenance. In particular, participants responded that not-so-up-to-date documents could still be an effective resource. Conversely, the extent to which a document is up-to-date was selected as one of the most important factors in determining its effectiveness. The results suggest that the software industry and academia may overemphasize the importance of document maintenance relative to a software professional's tolerance of out-dated content.

1. Introduction

This paper presents the results of a survey of professionals in the software industry conducted in April and May of 2002. This survey was constructed to uncover:

- The current industrial application of software documentation.
- How documentation attributes and artefacts influence both usefulness and relevance to the software team.

The software documentation we will discuss in this paper includes any artefact whose purpose is to communicate information about the software system to which it belongs. These artefacts include requirement, specification, architectural, and detailed design documents. These documents are geared to individuals involved in the production of that software, such as managers, project leaders, developers and customers. We will not be considering end-user documentation

We will refer to the term *documentation attributes*; these describe information about a document beyond the content provided within. Example attributes include the document's writing style, grammar, extent to which it is up to date, type, format, visibility, etc. We will also discuss *documentation artefacts*; these consist of whole documents, or elements within a document such as tables, examples, diagrams, etc. An artefact is an entity that communicates information about the software system.

1.1. Motivation

Since most of a Software Engineer's time will be spent doing maintenance [15], it seems appropriate that software documentation should be an important aspect of the software process.

But what constitutes good documentation? Most individuals believe the two main requirements for good documentation are that it is complete and up-to-date. We hypothesise that other factors may have a larger impact on documentation relevance than has been previously thought.

We must also consider the issue of the applicability and usefulness of a document. Can complete and up-to-date documents that are rarely referenced or used be considered of high quality? What about incomplete, highly referenced documents that appropriately communicate to their intended audience?

To gauge the quality of documentation, factors beyond its completeness and being up to date should at least be considered when discussing documentation relevance.

Our work is driven to uncover factors, preferably measurable ones that contribute to (or hinder) documentation relevance. We hope to exploit the effects of these factors to better predict the effectiveness of a document based on the current environment where that document exists.

In search of answers, we performed a systematic survey to question the thoughts of software practitioners and managers. Our approach is to build theories based on empirical data; possibly uncovering evidence that questions our intuition and common sense about documentation and its role in software engineering.

1.2. Related Work

Curtis et al [3] interviewed personnel from 17 large software projects. Their analysis was 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.

Our work provides statistical data that affirm some of the documentation issues Curtis identified.

Abdulaziz Jazzar [8] conducted an empirical investigation using a comparative case study research method. The basis for the work was concerned with the requirements for information system documentation.

Jazzar's work resulted in eight hypotheses that attempt to model the requirements for achieving effective, high quality documentation products and processes.

Our work complements Jazzar's as we focused on the attributes of quality documents, whereas Jazzar focused on the process of quality documentation.

In addition, our work contributes knowledge about several other facets of documentation including the current use and perception of software documentation tools and technologies.

1.3. History

Before conducting the main survey described in this paper, we conducted a pilot study to help develop and refine the questions.

The pilot-study participants were sampled from a fourth year software engineering course offered at the University of Ottawa in the winter 2002. Most participants had some experience in the software industry.

The official survey, conducted in April 2002, featured fewer and more concise questions with an improved sampling approach. All participants had at least one year of experience in the software industry; several had over ten years experience.

A summary of the data used in this report is available on-line [4]. Individual responses and identifying information have been withheld to protect confidentiality. The University of Ottawa's Human Subjects Research Ethics Committee approved the conducting of the survey.

1.4. Importance

The survey results presented in this paper are important for various reasons and to several audiences:

- Individuals interested in documentation technologies can use the data to better understand how it is used in practice and what attributes of a document are perceived as contributing most (as well as those that contribute least) to its effectiveness.
- Software decision makers can use the data to justify modifications to established documentation processes (such as those found in [14]) to improve the creation, maintenance and approval of documentation.
- Using the results of our work, better documentation tools and technologies might emerge; as designers will have a better grasp of the role of documentation and under what circumstances it is, or can be, used.

1.5. Outline

The remainder of this paper is organized as follows:

Section 2 describes the method under which the survey was conducted and the way in which we categorized participants based on their responses.

Section 3 highlights several interesting (and potentially controversial) findings from the gathered data.

Section 4 summarizes the participant demographics based on professional experience in the software industry.

2. Survey Method

2.1. Question Topics

The survey consisted of 50 questions of various types including multiple-choice, short answer, ratings, and free-form questions.

The question topics included:

- The role of software team members in the process of writing, maintaining and verifying different types of documents.
- The participant's personal preference for different types of documentation, and their effectiveness.
- The ability of a document's attributes, as opposed to its content, to promote (or hinder) effective communication.

- The state of software documentation in the participant's organization.
- Comparison of past projects to current ones.
- The effectiveness of documentation tools and technologies.
- Demographics of the participants.

2.2. Participants

Participants were solicited in three main ways. The members of the research team approached:

- Management and human resource individuals of several high-tech companies. They were asked to approach employees and colleagues to participate.
- Peers in the software industry.
- Members of software e-mail lists. They were sent a generic invitation to participate in the survey.

Most participants completed the survey using the Internet. A few replied directly via email. There were a total of 48 participants that provided responses that were complete and contained valid data. The participants were categorized in several ways based on software process, employment duties and development process as outlined below.

We divided the participants into two groups based on the individual's software process as follows:

- Agile. Twenty-five individuals that somewhat (4) to strongly (5) agree that they practice (or are trying to practice) agile software development techniques, according to Question 29 of the survey.
- Conventional. Seventeen individuals that somewhat (2) to strongly (1) disagree that they practice agile techniques, or indicated that they did not know about the techniques by marking 'n/a' for not applicable, according to Question 29 of the survey.

Our rationale for the above division is that the proponents of agile techniques promote somewhat different documentation practices from those recommended in conventional software engineering methodologies.

In addition, we divided the participants based on current employment duties as follows:

- Manager. Twelve individuals that selected manager as one of their current job functions, according to Question 44.
- Developers. Seventeen individuals that are non-managers and selected either senior or junior developer as one of their current job functions, according to Question 44.

Finally, we divided the participants based on management's recommended development process as follows:

- Waterfall. Thirteen individuals that selected waterfall as the recommended development process, according to Question 46 of the survey.
- Iterative. Fourteen individuals that are non-waterfall participants and who selected either iterative or incremental as the recommended development process, according to Question 46 of the survey.

3. Survey Results

3.1. Is Documentation Maintained?

This section illustrates the extent to which documentation is maintained. The data presented below substantiates the claim that software documentation is rarely, if ever, updated. This information will serve as the basis for several other sections in this paper.

Question 4 asked the participants from personal experience how long it takes for supporting documentation to be updated when changes in the system occur. The documents in question include: requirements, specifications, detailed design, low level design, architectural, and testing / quality documents

The participants selected from fixed values ranging between '*updates are never made*' (score of 1) and '*updates are made within a few days of the changes*' (score of 5).

Table 1 illustrates the preferred (mode) score, the percentage of responses of that score as well as the textual meaning of the score.

Table 1: How often is documentation updated when changes occur in a software system?

Document Type	Mode	% of Mode	In Words
Requirements	2	52 %	Rarely
Specifications	2	46 %	Rarely
Detailed Design	2	42 %	Rarely
Low Level Design	2	50 %	Rarely
Architectural	2	40 %	Rarely
Testing / Quality Documents	5	41 %	Within days

Similarly, question 20 asked if the participants agreed that documentation is always outdated.

Many participants somewhat agreed (43%) with that statement, and a considerable number strongly agreed (25%).

The fact that documentation is infrequently updated does not imply that our sample participants work on projects of lower quality or that proper software engineering practices are not in place. In fact, another part of our survey indicates that software quality seems to be improving despite little to no improvement in the quality of software documentation [5].

The evidence that documentation is rarely updated is important from a technology perspective. Since our results imply that the usage of tools that support documentation maintenance will be sporadic at best, such tools must enable users unfamiliar with a document to quickly comprehend its structure and content so they can make consistent and correct changes. The tools must also be efficient from a task perspective, helping users to quickly accomplish what they intended to achieve.

Based on the participant categorization (refer to Section 2.2), several statistically relevant differences occur between agile and conventional participants, as well as those grouped as iterative and waterfall.

Table 2 summarizes the differences between agile and conventional individuals with respect to document maintenance, and Table 3 between iterative and waterfall individuals.

In general, most agile and iterative participants believe documentation is at best updated within a few months of changes to the system. Conversely, conventional and waterfall participants believed documents were maintained within a few months to within a few weeks of system changes.

Table 2: Documentation Update Time (Agile vs. Conventional); a higher the score means a document is more quickly updated following changes to the system

Document Type	Agile		Conventional	
	Mean	St. Dev	Mean	St. Dev
Requirements	2.76	1.30	2.75	1.16
Specifications	3.07	1.16	3.25	1.16
Detailed Design*	2.60	1.06	3.88	1.25
Low Level Design	2.93	1.33	3.57	1.13
Architectural*	2.81	1.05	3.75	1.16
Testing / Quality Documents*	3.31	1.38	4.25	0.89

* Statistically significant differences between the means with 95% confidence

Table 3: Documentation Update Time (Iterative vs. Waterfall), a higher the score means a document is more quickly updated following changes to the system

Document Type	Iterative		Waterfall	
	Mean	St. Dev	Mean	St. Dev
Requirements*	2.00	1.10	3.25	1.28
Specifications	3.00	1.15	3.63	1.19
Detailed Design**	2.20	1.30	3.50	1.60
Low Level Design**	2.25	1.26	3.83	1.47
Architectural*	2.17	0.75	3.38	1.30
Testing / Quality Documents	3.50	1.29	3.86	1.21

* Statistically significant differences between the means with 95% confidence (** 90% confidence)

Question 20 asked to what degree the participants agreed that documentation is always out of date. The following observations compare the results of agile and conventional participants.

As expected, the results ranged from strongly disagree to strongly agree. Many participants (43%) somewhat agreed with that statement, but a considerable number (25%) also strongly agreed. In particular, the agile participants were statistically more likely to agree with this statement (mean of 3.83, st. dev 1.20) compared to the conventional participants (mean of 3.08, st. dev 1.29). There is also suggestive, but not statistically significant evidence that iterative participants are also more likely to agree that documentation is always out of date compared to the waterfall participants. These results help affirm the results shown in Table 2 and Table 3 that iterative and agile participants are less likely to update documentation. As well, the data support the consensus that documentation is rarely updated.

The evidence that agile and iterative individuals work in projects where documentation is less frequently updated and almost always outdated does not imply that these projects are of lower quality or that proper software engineering practices are not in place. Software project quality may be high despite a lack of documentation maintenance and, as many have suggested, despite [[1], [6], [10], [11], [15]] a lack in documentation quality. One might argue that since software engineering's primary role is to deliver software (with its secondary role to facilitate future software development) then it may be that the role of documentation, based on current practices, in software engineering is of little importance with respect to software quality. Unfortunately our data suggests otherwise, as most individuals in our survey believe in the importance of documentation as well as the fact that the documentation available to them is importance (see Section 3.4). As well, most participants agreed that both software project quality and documentation quality are improving [5].

It seems reasonable to believe there should be a high correlation between a software project's success and the quality of the documentation available. However, we believe that many individuals may have an inappropriate *parsing pattern*, (defined in [2], p. 3, as *the techniques that an individual uses to processes his or her surrounding')* to identify documentation quality. Instead of investigating the correlation of software quality to documentation quality, we will investigate the relationship between documentation quality and the extent to which a document is up-to-date. Our reason for taking this approach will become clearer in the following section.

3.2. Up-to-date Double Standard?

This section discusses the importance of keeping software documentation up to date. Two similar questions were posed in the survey with seemingly contradicting results.

Question 21 of the survey asked participants if they believe that documentation can be useful even though it is not always up-to-date. The results are summarized in Table 4.

There is overwhelming agreement that out-dated documentation is still quite useful. In all but two categories the mean was at or above 4.0 (the average participant somewhat agrees with the question statement). This observation questions both common intuition as well as past statements that documentation is practically useless unless accurate and kept up to date.

Table 4: Can out-dated documentation be useful? A higher score indicates a high level of agreement.

Participant Category	Mean	St. Dev.	Percentage that Strongly Agree
All	4.0	0.98	28 %
Waterfall*	4.1	0.75	38 %
Iterative*	3.5	0.44	15 %
Agile	3.9	0.74	28 %
Conventional	4.1	0.87	29 %
Manager*	3.3	0.76	8 %
Developer*	4.1	0.67	35

* Statistically significant differences between the means of a pair of participant categories with 95% confidence

The conflict might be the assumed association between being up to date and correctness, and inversely not-so-up-to-date with incorrect. Some sources argue that being out-dated implies the information is incorrect and thus not reliable. This unreliability then affects the document's credibility and hence its effectiveness [11].

The above reasoning is based on the notion that documentation must present facts, and facts are only useful when they are accurate and up to date. Conversely, some argue that the purpose of documentation is to convey *knowledge or information* [2]. The system's source code is the artefact that presents the facts, whereas the supporting documents facilitate higher-level views 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 [2].

The above data in support of useful out-dated documentation might convince some that the extent to which a document is up-to-date is not that important of a factor to create effective documentation. The survey data illustrated a somewhat different perspective.

Extent to which a document is up-to-date was one of the document attributes listed in Question 9. Question 9 asked to what degree certain attributes contribute to effective software documentation.

Table 5 outlines the mean, standard deviation and the percentage of participants who rated this item a 5 (the item is one of the *most* important factors to determine a documents importance).

Table 5: The importance of up-to-date documents, the higher the rating the more important the attribute

Participant Category	Mean	St. Dev.	Percentage of participants rating 'up-to-date' as one of the most important attributes
All	4.3	0.89	46 %
Waterfall	4.4	1.25	50 %
Iterative	4.5	3.33	50 %
Agile	4.3	0.73	44 %
Conventional	4.3	1.45	43 %
Manager	4.0	2.23	20 %
Developer	4.4	1.14	56 %

The data above illustrate the perceived correlation between a document's maintenance and effectiveness. Alone this result is intuitive, but in combination with the previous results, we present a slightly different interpretation.

Many participants responded that out-dated documentation could be useful. At the same time, many individuals rated the extent to which a document is up to date crucial to determine its usefulness.

This disagreement that out-dated documentation can be useful but that being up-to-date is a crucial element of useful documentation may influence individuals to over-estimate the importance of the up-datedness of a document relative to several other factors including content, availability and use of examples. Although, we can conclude that it would be very nice if our documents were up to date, we should not necessarily consider them useless solely because they are out-dated. More importantly, it seems wasteful to attempt to consistently and regularly assure that all documents are up-dated for the mere sake of keeping the documents current. Software documentation should evolve with the project team based on the needs and available resources of those team members. Maintenance for its own sake detracts software professionals for other potentially more important activities including software construction.

It is important to understand that our findings, based on the data from the survey as well as from the existing literature, do not promote improving the documentation process by merely ignoring it. Rather, our goal is to help software project team members to analyse their own needs for documentation and to integrate documentation into the software development process. The act of documentation should be scrutinized in the same manner that new features are added to a system: with care.

Our primary argument is that documentation should not be updated *merely* because it is updated. Instead, documentation should be updated when a real, not perceived, benefit will be achieved by maintaining the document. The survey data showed that individuals believe that out-dated documentation can still be useful, and therefore is not always a valid argument for documentation maintenance. A better rationale for documentation maintenance is the combination of the situation where artefacts that not effectively convey information with a need for the information to be conveyed. Information needs change in software projects and the fact that an artefact may no longer convey information is usually not enough to warrant maintenance. Individuals must also need and use the artefact's information.

3.3. Rating Documentation Attributes

This section discusses how certain attributes contribute to a document's effectiveness.

Question 9 asked the participants how important particular document attributes contribute to its overall effectiveness. Participants gave rating between 1 (least important) and 5 (most important).

Table 6 lists the attributes considered in Question 9 in descending order based on the attributes perceived contribution to a document's effectiveness.

Table 6: Document attributes and effectiveness

Document Attribute	Mean of Q9	Std. dev.	% Rate 5	% Rate 1 or 2
Content – the document's information	4.85	1.57	85 %	0 %
Up-to-date	4.35	0.89	46 %	0 %
Availability	4.19	0.79	41 %	4 %
Use of examples	4.19	0.85	37 %	4 %
Organization – sections / subsections	3.85	0.64	30 %	4 %
Type – req, spec, design, etc.	3.78	0.63	26 %	11 %
Use of diagrams	3.44	0.60	15 %	22 %
Navigation – quality of internal / external links	3.26	0.44	19 %	33 %
Structure – arrangement of text, diagrams, figures	3.26	0.60	11 %	22 %
Writing Style – sentence / paragraph structure, grammar	3.26	0.67	7 %	19 %
Length – not too long or short	3.15	0.64	7 %	22 %
Spelling and grammar	2.93	0.85	0 %	22 %
Author	2.63	0.41	7 %	48 %
Influence to use it	2.62	0.48	12 %	50 %
Format – pdf,, doc, txt, xml, etc.	2.42	0.58	0 %	54 %

Interesting observations from this data include:

- Content is the most important factor. All document tasks (creation, maintenance, verification, validation) should always keep the target audience in mind. Effective content is the key to effective documentation.
- Extent to which a document is up-to-date is the second most important factor. Although seemingly intuitive and accepted [[1], [6], [10], [11], [15]], the previous sections provided a different interpretation of this result.
- A document's format (.pdf, .doc, .html), its author, influence from management to use it and the quality of spelling and grammar have low correlation with the document's effectiveness.
- The style of writing does not have much impact on effectiveness. This result supports the argument that readability formulas are not an effective metric to determine the usefulness of a document [12].

Relating to documentation engineering in the large, documentation technologies should strive to facilitate the above qualities that promote a document's usefulness. In particular, our data suggest that technologies should:

- Focus on content. Allow the author to easily create and maintain content-rich documents. This should be the primary intent of most documentation technologies.
- Focus on availability. Allow for larger-scale publishing capabilities to assure the most up-to-date documents are readily available and easily located.
- Focus on examples. Allow for better features to support examples and their integration within a document.

3.4. Agile Vs. Conventional Thinking

This section describes in more detail how the agile and conventional participants use and perceive documentation in practice. Table 7 compares the results of the questions listed below.

Question 14 asked if the participants felt that documentation is important, but not that useful in their current organization.

Questions 15 and 16 asked if the software documentation available to participants is easy to understand, easy to cross-reference, brief and to the point.

Question 17 asked the participant if the appropriate documentation was easy to locate when required. Conversely, Question 34 asked if the appropriate documentation was difficult to find and navigate due to the large number of documents available.

The data in Table 7 below suggest that agile participants are statistically more likely to:

- Agree that the documentation they reference is easier to understand, navigate and cross-reference.
- Agree that documentation is brief and to the point.
- Disagree that the collection of documentation is poorly organized and difficult to navigate due to the size and number of available documents.

Table 7: Perception of Documentation (Agile Vs. Conventional), A higher score means a higher agreement of the statement

Question	Agile		Conventional	
	Mean	St. Dev	Mean	St. Dev
14 - not useful in our organization	2.35	1.40	2.41	1.54
15** - easy to understand and navigate	3.56	1.19	3.06	1.39
16* - brief and to the point	3.56	1.04	2.94	1.03
17 - easy to locate	2.80	1.12	2.82	1.13
34* - too many documents available	2.88	1.19	3.41	1.33

* Statistically significant differences between the means with 95% confidence (** 90% confidence)

Despite the observation that documentation is rarely updated (refer to Section 3.1), agile participants feel that the documentation they reference is brief, to the point and easy to navigate; more so than conventional participants. Also, a considerable number of agile participants strongly agree (39%) and somewhat agree (22%) that the documentation in their projects is useful. Finally, agile participants indicated that they were less likely to maintain documentation relative to conventional participants (again, refer to Section 3.1). Combined, these results support our hypothesis stated earlier that present mental parsing patterns for software documentation may not be the most appropriate; that is, being up-to-date may not be the most appropriate metric when analyzing the relevance of software documentation.

3.5. Project Size Independence

This section provides evidence that the conclusions drawn in previous sections appear to be independent from the project size (based in thousands of lines of code, KLOCs).

Question 41 asked what for the size of the participants current project in thousands of lines of code (KLOCs). The available sizes were less than 1, 1-5, 5-20, 20-50, 50-100, over 100 KLOCS or N/A.

Table 8 illustrates the project size distribution for all categories outlined in Section 2.2.

Table 8: Participants Project Size in KLOCs

Participant Category	Percent of projects between 1 and 20 KLOCS	Percent of projects >= 50 KLOCS	Number of Individuals considered
All	29 %	35 %	45
Waterfall	36 %	44 %	13
Iterative	31 %	39 %	13
Agile	36 %	44 %	25
Conventional	24 %	18 %	16
Manager	33 %	50 %	12
Developer	35 %	35 %	17

It is interesting to point out that a larger then expected portion of agile participants are working on large projects (agile development is typically associated with small projects). Our phrasing of practicing agile techniques helps explain this high percentage. In the context of our research, individuals were asked if they practice agile techniques. Agreement with this statement does not necessarily imply that the project itself is agile. As such, it is not unfounded to have such a large portion of agile techniques applied to large projects.

Using Spearman's Rank Correlation [7], the correlation between project size and the individual's software techniques (ranging from highly conventional to highly agile) was very low (-0.09). Similarly, the correlation of project size to the individual's role (ranging from highly managerial to highly developmental) was quite low (0.19).

The low correlation above, and the fair representation of the software categories outlined in Section 2.2 suggest that the results cited in previous sections should hold regardless of project size.

4. Demographics

In this section, we will describe the participant demographics. The divisions separate individuals based on software experience, current project size and software duties. The purpose of this section is to show that the survey was broadly-based, and therefore more likely to be valid in a wide variety of contexts.

Table 9 illustrates the participant experience in the software field (based on number of years in the industry).

Table 9: Participant Software Experience

Software Experience (years)	Number of Participants	Percentage
< 1	0	0 %
1 to 4	11	23 %
5 to 10	14	30 %
> 10	22	47 %

Table 10 indicates the current job functions held by the participants. Please note that one individual can have several functions.

It appears from the data in Table 10 that most employment areas in the software field have been well represented. The two somewhat under-represented categories are Junior Developers and Software Support. This survey was not directed at students since they probably would have lacked the experience to provide useful results.

Table 10: Participant Employment in the Software Field*

Job Functions	Number of Participants	Percentage
Sr. Software Developer	19	40 %
Software Architects.	17	36 %
Project Leader	14	30 %
Manager	12	26 %
Technical Writers	10	21 %
Quality Assurance	9	19 %
Jr. Software Developers	5	11 %
Other	4	9 %
Software Support	3	6 %
None of the above	3	6 %
Student	1	2 %

* Note that many participants performed one or more function.

5. Summary

The data from the April 2002 survey of software professionals provides concrete evidence that debunk some common documentation misconceptions and lead to the following conclusions.

- Document content can be relevant, even if it is not up to date. (However, keeping it up to date is still a good objective).

- Documentation is an important tool for communication as opposed to simply a fact sheet about the source code that is only relevant if well maintained.

The conclusions cited above will help decision makers choose more appropriate documentation strategies and technologies based on needs as oppose to generic expectations.

Once we can admit that documentation is often out-dated and inconsistent, we can then appreciate and utilize it appropriately as a tool of communication. This tool can then be judged based on its ability to communicate as opposed to merely presenting facts.

Software documentation should focus more on conveying meaningful and useful knowledge than on precise and accurate information.

5.1. Future Work

Based on these findings as well as the additional questions raised from this survey, the following lists some possible avenues for continued research in this field.

- How do people use documentation? More in-field research is required to substantiate some of the observations in the paper. What attributes of documentation hinder / facilitate its use?

- How can documentation maintenance be improved? Although maintenance may not be a critical contributor to useful documentation, it would be useful to understand techniques and tools that improve this process.

- What effect does time and change have on a document's relevance? For example, as a document ages, its relevance most likely decreases. Why? To what extent? How do external factors affect this decay in relevance?

- How else can we document a system? How effective are these methods with respect to creation, maintenance, use and quality of the content?

Acknowledgments

Our thanks to all participants and participating companies (who must remain anonymous). Thank you to members of the Knowledge Based Reverse Engineering (KBRE) group at the University of Ottawa. Your feedback and support have been greatly appreciated.

We would also like to thank Ayana Nurse and Jayne Forward for helping edit this paper.

References

[1] Angerstien, Paula. *Better quality through better indexing*, SIGDOC '85, Cornell University, Ithaca, New York, USA, ACM Press, p 57.

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

[3] Curtis, Bill, Herb Krasner, and Neil Iscoe. A field study of the software design process for large systems.

[4] Forward, A. Survey data website available at www.site.uottawa.ca/~aforward/docsurvey/

[5] Forward, A. Software Engineering Documentation Priorities: An Industrial Study, submitted to CASCON 2002, available from [4].

[6] Glass, R. *Software maintenance documentation*, SIGDOC '89, Pittsburg, Pennsylvania, USA, ACM Press, p18 – 23.

[7] Institute of Phonetic Sciences (IFA). http://fonsg3.let.uva.nl/Service/Statistics/RankCorrelation_coefficient.html

[8] Jazzar, Abdulaziz and Walt Scacchi. Understanding the requirements for information system documentation: an empirical investigation, COOCS `95, Sheraton Silicon Valley, California, USA, ACM Press, p268 – 279.

[9] Klare, George R. *Readable computer documentation*, ACM JCD, Volume 24, Issue 3 (August 2000), p148 – 167.

[10] Medina, Enrique Arce. *Some aspects of software documentation*, SIGDOC '84, Mexico City, Mexico, p57 – 59.

[11] Ouchi, Miheko L. Software Maintenance Documentation, SIGDOC'85, Ithaca, New York, USA, ACM Press, p18 – 23.

[12] Redish, Janice. *Readability formulas have even more limitations than Klare discusses*, ACM JCD, Volume 24, Issue 3 (August 2000), p132 – 137.

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

[14] Stimely, Gwen L. *A stepwise approach to developing software documentation*, SIGDOC '90,

Little Rock, Arkansas, USA, ACM Press, p122 – 124.

[15] Thomas, Bill and Scott Tilley. *Documentation for software engineers: what is needed to aid system understanding?*, SIGODC '01, Sante Fe, New Mexico, USA, p 235 – 236.

[16] Tilley, Scott R. Documenting-in-the-large vs. Documenting-in-the-small