Requirements Quality Knowledge Representation using Chunking Complexity Measurement: Prior to Formal Inspections

David C. Rine[1]
and Anabel Fraga[2]

[1] George Mason University, Virginia, USA
{davidcrine}@yahoo.com
[2] Carlos III of Madrid University, Madrid, Spain
{afraga}@inf.uc3m.es

Abstract. In order to obtain a most effective return on a software project investment, then at least one requirements inspection shall be completed. This means that a software project requirements inspection shall never be omitted. This is because the requirements, especially those written in natural language, is the most important form of software knowledge.A formal requirements inspection identifies low quality knowledge representation content in the requirements document. The knowledge quality of requirements statements of requirements documents is one of the most important assets a project must inspect. An application of the metrics to improve requirements understandability and readability during requirements inspections can be built upon the metrics shown.

1 Introduction

Steven R. Rakitin [37] states "If you can only afford to do one inspection on a project, you will get the biggest return on investment from a requirements inspection. A requirements inspection should be the one inspection that is never skipped." The formal inspection makes significant knowledge quality improvements to the requirements document, or formally a software requirements specification (SRS), which is the single artifact produced through the requirements engineering process. Kinds of knowledge in an SRS includes, functional requirements, non-functional requirements, system requirements, user requirements, etc [42]. The knowledge quality of the SRS document unavoidably is the core of requirements management of which the formal inspection is an important part. And the SRS, which is comprised of requirements, or requirements statements, is a basis for developing or building the rest of the software, including verification and validation phases. Despite abundant suggestions and

guidelines on how to write knowledge quality requirements statements, knowledge quality SRS's are difficult to find.

The goal of this research is to improve the knowledge quality of the SRS by the identification of natural language knowledge defects derived from that of prior research [13-16] and applies a set of metrics as quality indicators of requirements statements in an SRS. Many research studies on software quality [6, 12, 18, 24, 27, 30], and various quality factors have been proposed to represent the quality of software. The quality factors adopted in this research are developed by Schneider (2002, 2000a, 2000b) and are named as goodness properties.

Din and Rine [13-16] evaluatedknowledge quality by means of Noun Phrase Chunking complexity metrics. That research compared the NPC-Cohesion and NPC-Coupling metrics with the cohesion and coupling metrics proposed earlier [38, 3, 8, 9, 11, 20, 45].

The evidence provided by Din and Rine [16] concludes that the "NPC complexity metrics indicate the content goodness properties of requirements statements." The contribution of the research from [13-16] is "a suite of NP chunk based complexity metrics and the evaluation of the proposed suite of metrics."

The paper is organized as follows. Section 2 presents the research problem statement and the importance of the research problem. The background of the research is summarized in Section 3. Section 4 illustrates the detailed process of obtaining the elements of measurement, Noun Phrase (NP) chunks, and then presents the proposed suite of metrics. Section 5 summarizes the contributions of the research.

2 Research Problem and Importance

2.1 Research Problem

The research was designed to answer the following question: How can low natural language knowledge quality requirements statements be identified in an SRS? Although the research focuses on SRS's, the conclusions of the research can be applied to other requirements documents such as system requirements documents.

Certain requirements defects are hard to identify. The Fagan's requirements inspection [19] can be used to identify requirements defects, requirements inspections can in the present practice "be effective when sections of an SRS are limited to 8-15 pages so that a requirements quality inspector can perform an inspection of a given section within two hours' time frame" [30, 42].

Defects such as inconsistent or missing requirements statements can easily be missed due to the spatial distance. The current requirements inspection practice does not consider these kinds of requirements defects.

2.2 The Importance of the Research

The suite of NPC complexity metrics is supported by a software tool researched and developed as part of this research [13-16] to identify high knowledge complexity and hence low knowledge quality requirements.

Low quality requirements are not only the source of software product risks but also the source of software development resource risks, which includes cost overrun and schedule delay [13-16].

Quality software "depends on a software manager's awareness of such low quality requirements, their ability to expediently assess the impacts of those low quality requirements, and the capability to develop a plan to rectify the problem" [13-16]. The proposed suite of complexity metrics expedites the process of identifying low quality requirements statements. The subsequent risk analysis of those requirements can be performed earlier. The rectification plan can hence be developed and carried out in a timely manner. This process, from quickly identifying low quality requirements to developing and carrying out the corresponding rectification plan, provides the foundation for the development of high quality software.

3. Background

3.1 Quality and Content Goodness Properties

Schneider, in his Ph.D. Dissertation directed by Rine [39, 40], proposed eleven goodness properties as a better coverage of quality factors 36]: Understandable, Unambiguous, Organized, Testable, Correct, Traceable, Complete, Consistent, Design independence, Feasible, and Relative necessity. Representing quality with a set of properties that are each relatively easier to measure is an important step towards measuring quality. However, the context of the current research focuses upon Understandable, Unambiguous, Organized, and Testable, keys to natural language knowledge representation quality.

3.2. Complexity Metrics and Measurement

Complexity is a major software characteristic that controls or influences natural language knowledge representation quality. It has been widely accepted as an indirect indicator of quality [21, 26, 29, 31] and hence the content goodness properties.

3.3. Readability Index

Difficult words are necessary to introduce new concepts and ideas, especially in education and research.

Coh-Metrix has developed readability indexes based on cohesion relations, interaction between a reader's skill level, world knowledge, and language and discourse characteristics. “The Coh-Metrix project uses lexicons, part-of-speech classifiers, syntactic parsers, templates, corpora, latent semantic analysis, and other components that are widely used in computational linguistics “ 25].

The Coh-Metrix readability index is used to address quality of an entire written document, such as an essay, rather than indivisual sections of technical documents, such as software requiremenst documents.

4. NP Chunk Based Complexity Metrics

4.1. Chunking, Cognition and Natural Language Quality

Humans tend to read and speak texts by chunkc. Abney [1] proposed chunks as the basic language parsing unit. Several categories of chunks include but are not limiteds to Noun Phrase (NP) chunks, Verb Phrase (VP) chunks, Prepositional Phrase (PP) chunks, etc [1]. This research NP chunks and ignores other types of chunks.

4.2. Three Core Metrics

It has been recognized that it is not likely that a single metric can capture software complexity [22, 28]. To deal with the inherent difficulty in software complexity, a myriad of indirect metrics of software complexity have been proposed [36].

Multiple empirical studies indicate that LOC is better or at least as good as any other metric [2, 18, 43]. All these evidence and findings indicate that counting should be one of the core software metrics. Zuse [47] also identified simple counts in his measurement theory as one of the metrics that possesses all the desired properties of an ideal metric.

Many of the published metrics, either for procedural languages or for object-oriented languages, include some variation of the cohesion and coupling metrics [4, 5]. Furthermore, cohesion and coupling metrics are ubiquitous across a wide variety of measurement situations, including 4GLs, software design, coding and rework.

4.3. Requirements Documents Used

Two requirements documents are used as cases of study in this research:

(1) A public domain requirements document for Federal Aviation Agency (FAA). The requirements document is available in Ricker's dissertation [38], and [3, 8, 9, 11, 20, 45].

(2) Versions of the Interactive Matching and Geocoding System II (IMAGS II) requirements documents for U. S. Bureau of Census. The IMAGS II, or IMAGS, project has gone through several iterations of requirements analysis. Four versions of the requirements documents are available for the research.

4.4. Sentence/Requirements Statement Level Complexity

The calculation can be expressed as follows.

$$\text{NPC-Sentence}(\text{sentence}_j) = \sum_{1 \le i \le N} \frac{\text{Entry}(i,j)}{\sum_{1 \le j \le C} \text{Entry}(i,j)}, \quad (0)$$

where Entry(i,j) is the number of occurrence of NP chunk NPi in sentencej, $1 \le i \le N$, $1 \le j \le C$, N is the total number of NP chunks, and C is the total number of sentences. Intuitively, NPC-Sentence is a metric that measures the normalized count of NP chunks in a sentence of a document.

The requirements statement level complexity metric, or NPC-Req(reqj), is the aggregation of NPC-Sentence of the component sentences and can be expressed as follows.

$$\text{NPC-Req}(\text{req}_j) = \sum_{\text{sentence}_i \in \text{req}_j} \text{NPC-Sentence}(\text{sentence}_i), \quad (1)$$

where $1 \le i \le L_j$, $1 \le j \le M$, L_j is the total number of sentences of requirement j, and M is the total number of requirements.

Example - From partial parsing to sentence level complexity

The following three requirements (four sentences) are extracted from the IMAGS Version 4 requirements document. The requirements in the IMAGS requirements document are marked with a label (e.g., "IM2-WKASSIGN-4") and a short description (e.g., "Assign Users to WAAs"). Note that WAA stands for Work Assignment Area and is a collection of counties. WAA is defined to facilitate the assignment of workload to individual users.

IM2-WKASSIGN-4: Assign Users to WAAs

IMAGS II shall track and maintain users' assignment to WAAs. A user can be assigned to one or more WAAs, and a WAA can have more than one user assigned.

IM2-WKASSIGN-7: Assign Incoming Addresses on WAAs
IMAGS II shall assign incoming addresses to users based upon their WAAs.

IM2-WKASSIGN-8: Assign Multiple WAAs to Multiple Users
IMAGS II shall provide a way to assign a list of WAAs to multiple users at once.

The chunk parsing results (0-1) in 16 NP chunks (see Table 1), where sentence is abbreviated as "sent." and requirement is abbreviated as "req.". Note that the word "WAAs" in the first sentence is tagged as "VB", a verb. This is an error due to the nature of statistical NLP process, which considers words after "to" as verbs. The stop list indicated in the table is used to filter NP chunks that have little meanings.

Table 1. Example – Parsing Results.

Stop NP chunks: (a user), (one), (users), (a way), (a list)		req. 1		req. 2	req. 3
		sent. 1	sent. 2	sent. 3	sent. 4
<imags/NN> <ii/NN>	0,7,11	1	0	1	1
<users/NNS> <'/POS> <assignment/NN>	1	1	0	0	0
<more/JJR> <waas/NNS>	4	0	1	0	0
<a/DT> <waa/NN>	5	0	1	0	0
<one/CD> <user/NN>	6	0	1	0	0
<addresses/NNS>	8	0	0	1	0
<their/PRP$> <waas/NNS>	10	0	0	1	0
<waas/NNS>	14	0	0	0	1
<multiple/NN><users/NNS>	15	0	0	0	1

By applying the stemming and text normalization techniques, the result is depicted in Table 2.

Table 2. Example – Stemming and Text Normalization.

Stop words: (a user), (one), (users), (a way), (a list)		req. 1		req. 2	req. 3
		sent. 1	sent. 2	sent. 3	sent. 4
<imags/NN> <ii/NN>	0,7,11	1	0	1	1
<users/NNS> <'/POS> <assign/NN>	1	1	0	0	0
<waa/NN>	4,5,10,14	0	2	1	1
<user/NN>	6	0	1	0	0
<address/NN>	8	0	0	1	0
<multiple/NN><user/NN>	15	0	0	0	1

For the sake of example, it is assumed that the four sentences constitute the complete requirements document. The resulting NPC-Sentence and NPC-Req are shown in Table 3.

Table 3. Example – NPC-Sentence and NPC-Req.

Stop words: (a user), (one), (users), (a way), (a list)	req. 1		req. 2	req. 3
	sent. 1	sent. 2	sent. 3	sent. 4
<imags/NN> <ii/NN>	1	0	1	1
<users/NNS> <'/POS> <assign/NN>	1	0	0	0
<waa/NN>	0	2	1	1
<user/NN>	0	1	0	0
<address/NN>	0	0	1	0
<multiple/NN><user/NN>	0	0	0	1
NPC-Sentence	1/3+1 =1.3	2/4+1 =1.5	1/3+1/4+ 1=1.58	1/3+1/4+1=1 .58
NPC-Req	1.3 + 1.5 = 2.8		1.58	1.58

4.5. Intra-Section Level Complexity

The formula (3) for NPC-Cohesion is as follows.

$$\text{NPC-Cohesion}(S_j)=\begin{cases}\dfrac{\sum_{1\le i\le M_j}\text{ClusterSize}(i,j)}{L_j-1}, & L_j>1\\ 1, & L_j=1\end{cases} \quad (3)$$

where Mj is the total number of clusters in section Sj, and Lj is the total number of sentences in section Sj.

If a requirements section consists of a single sentence ($L_j=1$), the NPC-Cohesion is 1. If all the adjacent sentences have common NP chunks, then the NPC-Cohesion is also 1.

For example, Figure 1 shows three sections of the requirements. The first section contains three sentences, the second section contains two sentences, and the third section has one sentence. Section 1 has a cluster that covers the whole section, and the size of the cluster is two. Section 2 has a cluster that covers the whole section, and the size of the cluster is one. Section 3 does not have any cluster. Based upon the above formula (3), the values of the NPC-Cohesion metric are as follows (4).

NPC-Cohesion(S1)=2/(3-1)=1,
NPC-Cohesion(S2)=1/(2-1)=1, (4)
NPC-Cohesion(S3)=1

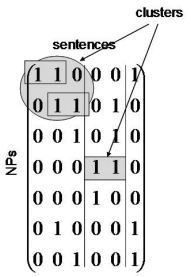

Fig. 1. Clusters of NP Chunks.

4.6. Inter-Section Level Complexity

The proposed NPC-Coupling metric value is the sum of the spatial distances among NP chunks and clusters that share the same NP chunks. Once a cluster is formed, the cluster represents all the components NP chunks inside the cluster. One possible algorithm to calculate the NPC-Coupling metric is as follows (5).

Figure 2 shows the calculation of NPC-Coupling using the same requirements sections in the previous example. The centroid of the cluster of the first section resides in

the second sentence. The centroid of the cluster of the second section resides between sentence 4 and 5. Based upon the above formula (4-5), the values of the NPC-Coupling metrics are as follows.

NPC-Coupling(S1)=4+3+2+4+3=16,
NPC-Coupling(S2)=3+2=5, (5)
NPC-Coupling(S3)=4+4+3=11

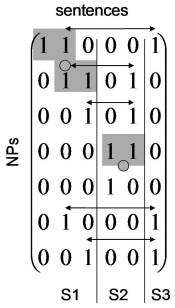

Fig. 2 . Calculation of Coupling Metrics.

4.7. A Composite Metric

It has been recognized that a single metric for a software product does not work. Multiple metrics provide multiple views of the subject, and each view serves its own purpose. Without carefully following engineering disciplines, operation, comparison, combination, or manipulation of different metrics can result in meaningless measures [26].

To identify the low quality requirements sections, this research combines the cohesion and coupling metrics into a single indicator. The formula used in the research is as follows (6).

(6)

$NPC\text{-}Composite(S_i) = NPC\text{-}Cohesion_i - a * (NPC\text{-}Coupling_i - b)$, for all i, $1 \leq i \leq S$, S is the total number of requirements sections, where $b = \min(NPC\text{-}Coupling_i)$ and $a = 1 / (\max(NPC\text{-}Coupling_i) - b)$.

The coefficients (6), a and b, are used to adjust the coupling metric so that (1) the measure falls in the range between –1 and 1, and (2) both the cohesion and coupling metrics use the same unit.

Cohesion Metrics: Based upon the proposed NPC-Cohesion metric defined previously, the NPC-Cohesion measures are depicted in Figure 3, together with the cohesion measures published in [35]. It is clear that the two metrics are consistent with each other except in one section – section 11 of the FAA requirements document .

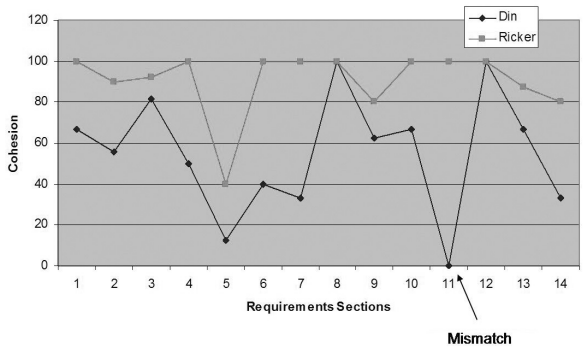

Fig. 3 .Cohesion Values Between Two Methods.

Although NPC-Cohesion is able to identify low cohesion requirements in the above example using syntactic categories of words, syntactic categories can sometimes mislead the analysis. For example, the following two sentences are low cohesion sentences according to NPC-Cohesion. "The computer program shall discretize the continuous function f(t). Then, it shall approximate the integral using the discretization."

One remedy to the weakness of NPC-Cohesion is to parse verb phrase (VP) chunks. Part of the text normalization process is to transform words into its root form. Then "discretize" and "discretization" can be normalized as the same chunk. However, the incorporation of a second type of chunks, i.e., verb phrase (VP) chunks, to the proposed metrics substantially increase the complexity of the parser and hence the processing effort and time. The addition of the parsing process for VP chunks to cope with the weakness that rarely occurs does not seem to be cost effective. Hence, it was decided to focus on NP chunks for the research.

Coupling Metrics: The coupling measures based on the NPC-Coupling metric and the coupling metric in (Pleeger, 1993) are depicted in Figure 4. The two metrics display consistent results except in one section - section 4 of the requirements document.

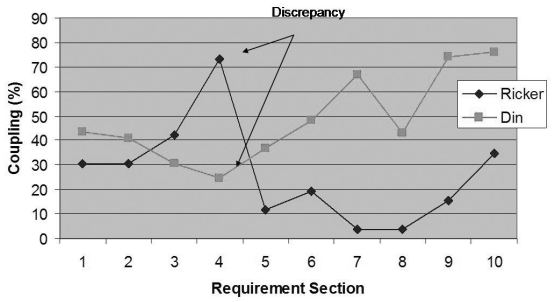

Fig. 4. Coupling Values between Two Methods.

The evaluation criterion for cohesion is whether the two sets of metrics are strongly consistent with each other. The derived data from this case study supports this consistency.

References

1. Abney, S. and Abney, S. (1991). Parsing By Chunks. In R. Berwick, S. Abney, and C. Tenny, editors, Principle-Based Parsing. Kluwer Academic Publishers, 1991.
2. Basili, V. R. (1980). Qualitative Software Complexity Models: A Summary, Tutorial on Models and Methods for Software Management and Engineering. 1980.
3. Bøegh, J. (2008). A New Standard for Quality Requirements. IEEE Software, March/April 2008 pp. 57-63.
4. Briand, L. C., Daly, J. W., and Wust, J. K (1998). A Unified Framework for Cohesion Measurement in Object-Oriented Systems. IEEE Transactions on Software Engineering, 3:65 – 117, 1998.
5. Briand, L. C., Daly, J. W., and Wust, J. K (1999). A Unified Framework for Coupling Measurement in Object-Oriented Systems. IEEE Transactions on Software Engineering, 25:91 – 121, 1999.
6. Cant, S. Jeffery, D. R., and Henderson-Sellers, B (1995). A conceptual model of cognitive complexity of elements of the programming process. Information and Software Technology, 37(7):351 – 362, 1995.
7. Card, D. N. and R. L. Glass (1990). Measuring Software Design Quality. Prentice-Hall, 1990.
8. Chung, L. and J. Cesar (2009). On Non-Functional Requirements in Software Engineering. Conceptual Modeling: Foundations and Applications, Lecture Notes in Computer Science, Volume 5600, 2009, pp 363-379.
9. Costello, R. and D. Liu (1995). Metrics for requirements engineering. Journal of Systems and Software, Volume 29, Issue 1, April 1995, Pages 39–63.
10. Darcy, D. P. and C. F. Kemerer (2002). Software Complexity: Toward a Unified Theory of Coupling and Cohesion. not working here, 8, February 8 2002.
11. Davis, A., S. Overmyer, J. Caruso, F. Dandashi, A. Dinh (1993). Identifying and measuring quality in a software requirements specification. Proceedings Software Metrics Symposium, 1993, First International, 21-22 May 1993, pp.141 – 152.
12. DeMarco, T. (1982). Controlling Software Projects. Yourdon Press, Englewood Cliffs, NJ, 1982.
13. Din, C. Y. (2008). Requirements Content Goodness and Complexity Measurement Based On NP Chunks. PhD thesis, George Mason University, Fairfax, VA, 2007, Reprinted by VDM Verlag Dr. Muller, 2008.
14. Din, C. Y. and D. C. Rine (2008). Requirements Content Goodness and Complexity Measurement Based On NP Chunks. Proceedings,Complexity and Intelligence of the Artificial Systems: Bio-inspired Computational Methods and Computational Methods Applied in Medicine, WMSCI 2008 Conference.
15. Din, C. Y. and D. C. Rine (2012). Requirements Metrics for Requirements Statements Stored in a Database. Proceedings of the 2012 International Conference on Software Engineering Research and Practice, SERP'12, July 16-19, 2012, pp. 1-7.
16. Din, C. Y. and D. C. Rine (2013). Requirements Statements Content Goodness and Complexity Measurement. International Journal of Next-Generation Computing.1:1. March 2013.
17. Dunsmore, H. E. (1984). Software Metrics: An Overview of an Evolving Methodology. Information Processing and Management, 20(1-2):183 – 192, 1984.
18. Evangelist, W. (1983). Software Complexity Metric Sensitivity to Program Structuring Rules. Journal of Systems and Software, 3(3):231 – 243, 1983.
19.0 Fagan, M. (1986). Advances in Software Inspections. IEEE Transactions in Software Engineering, 12, 7: 744-751 July 1986.
20. Farbey, B. (1990). Software quality metrics: considerations about requirements and requirement specifications. Information and Software Technology, Volume 32, Issue 1, January–February 1990, Pages 60–64.
21. Fenton, N. E. and M. Neil (2000). Software metrics: roadmap. In Proceedings of the International Conference on Software Engineering (ICSE), pages 357 – 370, 2000.

22. Fenton, N. E. and Neil, M. (1999). A Critique of Software Defect Prediction Models. IEEE Transactions on Software Engineering, 25:675 – 689, 1999.
23. Fenton, N. E. and S. L. Pleeger (1997). Software Metrics: A Rigorous and Practical Approach. International Thomson Computer Press, Boston, MA, 1997. 2nd ed.
24. Garson, G. D. (2006). Public Information Technology and E-Governance: Managing the Virtual State. Jones & Bartlett Pub., January 2006.
25. Graesser, A. C., D. S. McNamara, M. M. Louwerse, and Z. Cai (2004). Coh-Metrix: Analysis of Text on Cohesion and Language. Behavioral Research Methods, Instruments, and Computers, 36(2):193 – 202, 2004.
26. Henderson-Sellers, B. (1996). Object-Oriented Metrics textendash Measures of Complexity. Prentice Hall PTR, New Jersey, 1996.
27. Kemerer, C. F. (1998). Progress, Obstacles, and Opportunities in Software Engineering Economics. Communications of ACM, 41:63 – 66, 1998.
28. Kitchenham, B. A., Pleeger, S. L, and Fenton, N. E (1995). Towards a Framework for Software Measurement Validation. IEEE Transactions on Software Engineering, 21:929 – 943, 1995.
29. Klemola, T. (2000). A Cognitive Model for Complexity Metrics. volume 13, June 13 2000.
30. Lee, S. (2003). Proxy Viewpoints Model-Based Requirements Discovery. PhD thesis, George Mason University, Fairfax, VA, 2003.
31. Levitin, A. V. (1986). How to Measure Size, and How Not to. volume 10, pages 314 – 318, Chicago, Oct 8-10 1986. IEEE Computer Society Press.
32. Li, H. F. and W. K. Cheung (1987). An empirical study of software metrics. IEEE Transactions on Software Engineering, 13(6):697 – 708, 1987.
33. McNamara, D. S. (2001). Reading both high and low coherence texts: Effects of text sequence and prior knowledge. Canadian Journal of Experimental Psychology, 55:51 – 62, 2001.
34. McNamara, D. S., Kintsch, E., Songer, N. B, and Kintsch, W. (1996). Are good texts always better? Text coherence, background knowledge, and levels of understanding in learning from text. Cognition and Instruction, 14:1 – 43, 1996.
35. Pleeger, S. L. (1993). Lessons learned in building a corporate metrics program. IEEE Software, 10(3):67 – 74, 1993.
36. Purao, S. and V. Vaishnavi (2003). Product Metrics for Object-Oriented Systems. ACM Computing Surveys, 35(2):191 – 221, 2003.
37. Rakitin, S. (1997) Software Verification and Validation: A Practitioner's Guide (Artech House Computer Library). Artech House Publishers. ISBN-10: 0890068895 ISBN-13: 978-0890068892
38. Ricker, M. (1995). Requirements Specification Understandability Evaluation with Cohesion, Context, and Coupling. PhD thesis, George Mason University, Fairfax, VA, 1995.
39. Schneider, R. E. (2002). Process for building a more effective set of requirement goodness properties. PhD thesis, George Mason University, Fairfax, VA, 2002.
40. Schneider, R. E. and D. Buede (2000). Criteria for Selecting Properties of a High Quality Informal Requirements Document, Proceedings of the International Conference on Systems Engineering, Mid-Atlantic Regional Conference, INCOSE-MARC, April 5-8, 2000a, 7.2-1 to 7.2-5.
41. Schneider, R. E. and D. Buede (2000). Properties of a High Quality Informal Requirements Document, Proceedings of the Tenth Annual International Conference on Systems Engineering, INCOSE, July 16-20, 2000b, 377-384.
42. Sommerville, I. (2006). Software Engineering: update. Addison Wesley, 8 edition, 2006.
43. Weyuker , E. (1988). Evaluating Software Complexity Measures. IEEE Transactions Software Engineering, 14(9):1357 – 1365, 1988.
44. Wilson, W. M., Rosenberg, L. H., and Hyatt, L. E (1996). Automated Quality Analysis Of Natural Language Requirement Specifications. In Fourteenth Annual Pacific Northwest Software Quality Conference, October 1996.
45. Wnu, K., B. Regnell, B. Berenbach (2011). Scaling Up Requirements Engineering – Exploring the Challenges of Increasing Size and Complexity in Market-Driven Software Development. Requirements Engineering: Foundations for Software Quality, Lecture Notes in Computer Science Volume 6606, 2011, pp 54-59 .
46. Yin, R. K. (2003). Case study research. Sage Publications, Thousand Oaks, CA, 3 edition, 2003.
47. Zuse, H. (1998). A Framework of Software Measurement. Walter de Gruyter, Berlin/New York, 1998.