Chunking Complexity Measurement for Requirements Quality Knowledge Representation

David C. Rine[1] and Anabel Fraga[2(✉)]

[1] George Mason University, Fairfax, VA, 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. A formal requirement inspection identifies low quality knowledge representation content in the requirements document. In software development projects where natural language requirements are produced, a requirements document summarizes the results of requirements knowledge analysis and becomes the basis for subsequent software development. In many cases, the knowledge content quality of the requirements documents dictates the success of the software development. The need for determining knowledge quality of requirements documents is particularly acute when the target applications are large, complicated, and mission critical. The goal of this research is to develop knowledge content quality indicators of requirements statements in a requirements document prior to informal inspections. To achieve the goal, knowledge quality properties of the requirements statements are adopted to represent the quality of requirements statements. A suite of complexity metrics for requirements statements is used as knowledge quality indicators and is developed based upon natural language knowledge research of noun phrase (NP) chunks. 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 and suggested to be taken into account.

Keywords: Requirements inspections · Chunking and cognition · Complexity metrics · Cohesion · Coupling · NP chunk · Requirements · Software quality · Information retrieval · Natural language understanding and processing

1 Introduction

Steven R. Rakitin [30] 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

A. Fred et al. (Eds.): IC3K 2013, CCIS 454, pp. 245–259, 2015.
DOI: 10.1007/978-3-662-46549-3_16

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 include, functional requirements, non-functional requirements, system requirements, user requirements, and so on [34]. 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 [12–15] and applies a set of metrics as quality indicators of requirements statements in an SRS. Many research studies on software quality [6, 11, 16, 21, 23, 26], 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 [12–15] evaluated knowledge 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 [3, 7, 8, 10, 18, 31, 36].

The evidence provided by Din and Rine [15] concludes that the "NPC complexity metrics indicate the content goodness properties of requirements statements." The contribution of the research from [12–15] 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 Sect. 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 and conclusions.

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 [17] 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" [26, 34].

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 [12–15] 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 [12–15].

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" [12–15]. 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 [32, 33], 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 [19, 22, 25, 27] 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" [23].

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

Unfortunately, readability indexes, including Coh-Metrix, are not comparable with this research [12–15] for the following reasons:

(1) The readability metrics are designed for the whole documents, instead of sections of documents.
(2) The readability scores are not reliable indicators when the document under evaluation has less than 200 words (McNamara, 2001). However, many of the requirements statements have less than 50 words.
(3) Although Coh-Metrix attempts to measure the cohesion of texts, the definition of cohesion used by Coh-Metrix is different from the definition of cohesion used in Computer Science, and there are no coupling metrics in Coh-Metrix.
(4) Coh-Metrix does not have a single metric to represent the size, cohesion, or coupling complexity. Coh-Metrix includes more than 50 metrics to measure very specific aspects of texts. No composite metric that combines those specific aspects of a document has been proposed.
(5) Coh-Metrix attempts to measure the cohesion of texts. Future work of Coh-Metrix may address comprehension, or understandability. However, Coh-Metrix will never address the issue of testability and many other goodness properties.

4 NP Chunk Based Complexity Metrics

4.1 Chunking, Cognition and Natural Language Quality

Humans tend to read and speak texts by chunks. Abney [1] proposed chunks as the basic language parsing unit. Several categories of chunks include but are not limited 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 [20, 24]. To deal with the inherent difficulty in software complexity, a myriad of indirect metrics of software complexity have been proposed [29].

Multiple empirical studies indicate that Line of Code (LOC) is better or at least as good as any other metric [2, 16, 35]. All these evidence and findings indicate that counting should be one of the core software metrics. Zuse [37] 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. Darcy and Kemerer believe that cohesion and coupling are effective metrics and they can represent the essential complexity measures for the general software design tasks [9]. Hence, NPC-Cohesion and NPC-Coupling are chosen to represent the complexity of requirements. To assist the identification of low quality requirements, a composite metric (NPC-Composite) that combine cohesion and coupling measures is also proposed and studied in the research.

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 [31], and [3, 7, 8, 10, 18, 36].
(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.

$$\mathrm{NPC-Sentence}(\mathrm{sentence_j}) = \sum_{1 \le i \le N} \frac{\mathrm{Entry(i,j)}}{\sum_{1 \le j \le C} \mathrm{Entry(i,j)}}, \tag{1}$$

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.

$$\mathrm{NPC-Req}(\mathrm{req_j}) = \sum_{\mathrm{sentence_i} \in \mathrm{req_j}} \mathrm{NPC-Sentence}(\mathrm{sentence_i}), \tag{2}$$

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., "IM-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.

IM-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.

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

IM-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 results of the chunk parsing are as follows.

```
(S:
        (0: <imags/NN> <ii/NN>)
        <shall/MD>
        <track/VB>
        <and/CC>
        <maintain/VB>
        (1: <users/NNS> <'/POS> <assignment/NN>)
        <to/TO>
        <waas/VB>
        <./.>
)

(S:
        (2: <a/DT> <user/NN>)
        <can/MD>
        <be/VB>
        <assigned/VBN>
        <to/TO>
        (3: <one/CD>)
        <or/CC>
        (4: <more/JJR><waas/NNS>)
        <,/,>
        <and/CC>
        (5: <a/DT> <waa/NN>)
        <can/MD>
        <have/VB>
        <more/JJR>
        <than/IN>
        (6: <one/CD> <user/NN>)
        <assigned/VBN>
        <./.>
)
```

```
(S:
        (7: <imags/NN> <ii/NN>)
        <shall/MD>
        <assign/VB>
        <incoming/VBG>
        (8: <addresses/NNS>)
        <to/TO>
        (9: <users/NNS>)
        <based/VBN>
        <upon/IN>
        (10: <their/PRP$><waas/NNS>)
        <./.>
)

(S:
        (11: <imags/NN> <ii/NN>)
        <shall/MD>
        <provide/VB>
        (<12: <a/DT> <way/NN>)
        <to/TO>
        <assign/VB>
        (<13: <a/DT> <list/NN>)
        <of/IN>
        (<14: <waas/NNS>)
        <to/TO>
        (<15: <multiple/NN> <users/NNS>)
        <at/IN>
        <once/RB>
        <./.>
)
```

The chunk parsing results (1-2) 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.

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

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, Fig. 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).

$$\begin{aligned} \text{NPC} - \text{Cohesion}(\text{S1}) &= 2/(3-1) = 1, \\ \text{NPC} - \text{Cohesion}(\text{S2}) &= 1/(2-1) = 1, \\ \text{NPC} - \text{Cohesion}(\text{S3}) &= 1 \end{aligned} \tag{4}$$

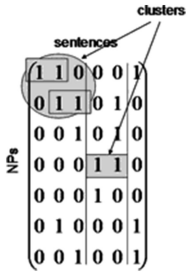

Fig. 1. Clusters of NP chunks.

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.

The NPC-Coupling metric of section S_j:
x = 0,
remarks: handle clusters coupling
for every cluster i in section S_j,
  for every NP_k in cluster i,
  for every sentence l outside section S_j,
  if sentence l contains NP_k,
  calculate the distance between sentence l and the centroid of cluster i,
  add the distance to x,
remarks: handle non-clusters coupling
for every NP_k of section S_j that does not belong to any cluster of S_j,
  for every sentence l in section S_j,
  if sentence l contains NP_k,
  for every sentence m outside section S_j,
  if sentence m contains NP_k,
  calculate the distance between sentence l and sentence m,
  add the distance to x,
return x

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.

$$\begin{aligned} \text{NPC - Coupling}(S1) &= 4 + 3 + 2 + 4 + 3 = 16, \\ \text{NPC - Coupling}(S2) &= 3 + 2 = 5, \\ \text{NPC - Coupling}(S3) &= 4 + 4 + 3 = 11 \end{aligned} \tag{5}$$

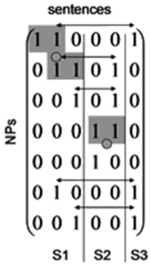

Fig. 2. Calculation of coupling metrics.

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 [22].

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).

$$\text{NPC} - \text{Composite}(S_i) = \text{NPC} - \text{Cohesion}_i - a * (\text{NPC} - \text{Coupling}_i - b), \tag{6}$$

for all i, $1 \leq i \leq S$, S is the total number of requirements sections, where $b = \min(\text{NPC - Coupling}_i)$ and $a = 1 / (\max(\text{NPC - Coupling}_i) - b)$

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 Fig. 3, together with the cohesion measures published in [28]. It is clear that the two metrics are consistent with each other except in one section– Sect. 11 of the FAA requirements document.

Although NPC-Cohesion is able to identify low cohesion requirements in the above example using syntactic categories of words, syntactic categories can sometimes mislead

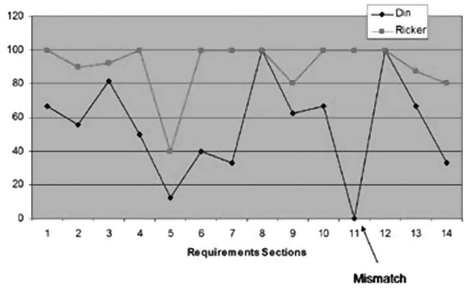

Fig. 3. Cohesion values between two methods.

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.

The mismatch between the two cohesion metrics can be explained as follows. Section 11 of the FAA requirements document consists of two sentences. Here are the sentences and the corresponding chunk parsing results.

Sentence 1: "The system shall provide flight plan outputs to a variety of operational positions, collocated processors, and remote facilities."

```
(S:
        (0: <the/DT> <system/NN>)
        <shall/MD>
        <provide/VB>
        (1: <flight/NN> <plan/NN> <outputs/NN>)
        <to/TO>
        (2: <a/DT> <variety/NN>)
        <of/IN>
        (3: <operational/JJ> <positions/NNS>)
        <,/,>
        <collocated/VBN>
        (4: <processors/NNS>)
        <,/,>
        <and/CC>
        (5: <remote/JJ> <facilities/NN>)
        <./.>
)
```

Sentence 2: "The ACCC shall output data periodically, on request, or in accordance with specified criteria (NAS-MD-311 and NAS-MD-314)."

```
(S:
        (6: <the/DT> <accc/NN>)
        <shall/MD>
        <output/VB>
        (7: <data/NNS>)
        <periodically/RB>
        <,/,>
        <on/IN>
        (8: <request/NN>)
        <,/,>
        <or/CC>
        <in/IN>
        (9: <accordance/NN>)
        <with/IN>
        <specified/VBN>
        (10: <criteria/NN>)
        <(/(>
        (11: <nas-md-311/NN>)
        <and/CC>
        (12: <nas-md-314/NN>)
        <)/)>
        <./.>

)
```

There are 13 NP chunks. It is clear that there are no common NP chunks between the two sentences. This is why the NPC-Cohesion metric gives a low cohesion measure for the above requirements section. On the other hand, Ricker uses terms to measure the cohesion of the section, and the word "output" appears in the first sentence as a noun, while the word "output" appears in the second sentence as a verb. Ricker's algorithm does not consider syntactic categories and hence links the two sentences. It is believed that a word in different forms, i.e., verbs and nouns, in different sentences should not always be considered as cohesive, since the two words in the two forms can refer to two totally different objects. By closely examining the two sentences, it can be found that the word "output" in the two sentences indeed refers to two different things or two different concepts. Hence, the proposed cohesion metrics is more effective.

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

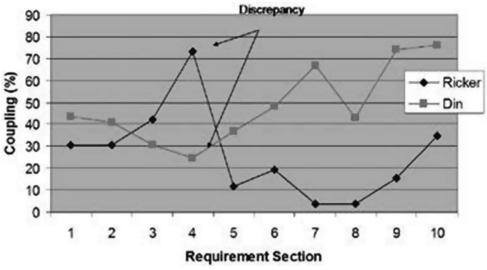

Fig. 4. Coupling values between two methods.

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.

Sensitivity/Accuracy: The NPC-Cohesion metrics are relative measures. They are normalized and fall in the range of 0 to 1. Comparing such relative measures derived from different requirements documents is not logical. In other words, it is not appropriate to compare the sensitivity and accuracy of the NPC-Cohesion metrics with Ricker's metrics.

Although the NPC-Coupling metrics are based upon spatial distance between NP chunks and they are not normalized, comparing it with Ricker's metric which uses different units of measurement does not seem to be logical either.

Summary: Based upon the above analysis, it can be concluded that the derived data from the case study met the evaluation criteria for the consistency hypothesis.

5 Summary of Contributions and Conclusions

This research derived from [12–15] made two contributions: (1) the invention of a suite of complexity metrics to measure the content goodness properties of requirements documents and (2) the empirical case study to evaluate the invented suite of complexity metrics.

The invented complexity metrics are researched and developed to identify low quality requirements statements in SRS's. These metrics are based on the NP chunks in SRS's. In the empirical two phased case study, it is concluded that the proposed metrics can measure the content goodness properties of requirements statements.

This research provides evidence for the feasibility of using NP chunks as the elements of measurement for complexity metrics. In addition the invented suite of complexity metrics provides requirements engineers and managers with a tool to measure the quality of the requirements statements. These metrics can be use to identify low quality requirements statements. They can also be used to identify requirements statements and requirements sections that may require more rigorous testing. Potential flaws and risks can be reduced and dealt with earlier in the software development cycle.

At a minimum, these metrics should lay the groundwork for automated measures of the quality of the requirements statements in SRS's. Because those metrics are constructed by a software tool, their measures are easy to collect, a vital characteristics for a quality measurement program [28].

During the research, and as stated by Genova et al. [22] and Fanmuy et al. [18], a set of metrics to complement the chunk based metrics can be included in the set of requirements metrics indicators in order to create a complete set of metrics of interest for any systems requirement engineer. It includes a set of semantic metrics based on natural language processing (NLP) and semantic notions assisted by a knowledge-based system and requirements patterns.

References

1. Abney, S.: Parsing by chunks. In: Berwick, R., Abney, S., Tenny, C. (eds.) Principle-Based Parsing. Kluwer Academic Publishers, Dordrecht (1991)
2. Basili, V.R.: Qualitative Software Complexity Models: A Summary, Tutorial on Models and Methods for Software Management and Engineering. IEEE Computer Society Press, Los Alamitors (1980)
3. Bøegh, J.: A new standard for quality requirements. IEEE Softw. **25**(2), 57–63 (2008)
4. Briand, L.C., Daly, J.W., Wust, J.K.: A unified framework for cohesion measurement in object-oriented systems. IEEE Trans. Softw. Eng. **3**(1), 65–117 (1998)
5. Briand, L.C., Daly, J.W., Wust, J.K.: A unified framework for coupling measurement in object-oriented systems. IEEE Trans. Softw. Eng. **25**, 91–121 (1999)
6. Cant, S., Jeffery, D.R., Henderson-Sellers, B.: A conceptual model of cognitive complexity of elements of the programming process. Inf. Softw. Technol. **37**(7), 351–362 (1995)
7. Chung, L., do Prado Leite, J.C.S.: On non-functional requirements in software engineering. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 363–379. Springer, Heidelberg (2009)
8. Costello, R.J., Liu, D.-B.: Metrics for requirements engineering. J. Syst. Softw. **29**(1), 39–63 (1995)
9. Darcy, D.P., Kemerer, C.F., Software Complexity: Toward a Unified Theory of Coupling and Cohesion, 8 February 2002
10. Davis, A., Overmyer, S., Caruso, J., Dandashi, F., Dinh, A.: Identifying and measuring quality in a software requirements specification. In: Proceedings of the First International Software Metrics Symposium, 21–22 May, pp. 141–152 (1993)
11. Demarco, T.: Controlling Software Projects. Yourdon Press, Englewood Cliffs (1982)
12. Din, C.Y.: Requirements content goodness and complexity measurement based on NP chunks. Ph.D. thesis, George Mason University, Fairfax, VA, 2007, Reprinted by VDM Verlag Dr. Muller (2008)
13. Din, C.Y., Rine, D.C.: Requirements content goodness and complexity measurement based on NP chunks. In: Proceedings, Complexity and Intelligence of the Artificial Systems: Bio-inspired Computational Methods and Computational Methods Applied in Medicine, WMSCI 2008 Conference (2008)
14. Din, C.Y., Rine, D.C.: Requirements metrics for requirements statements stored in a database. In: Proceedings of the 2012 International Conference on Software Engineering Research and Practice, SERP 2012, July 16–19, pp. 1–7 (2012)
15. Din, C.Y., Rine, D.C.: Requirements Statements Content Goodness and Complexity Measurement. International Journal of Next-Generation Computing. **4**(1) (2013)
16. Evangelist, W.: Software complexity metric sensitivity to program structuring rules. J. Syst. Softw. **3**(3), 231–243 (1983)
17. Fagan, M.: Advances in Software Inspections. IEEE Trans. Softw. Eng. **12**(7), 744–751 (1986)
18. Fanmuy, G., Fraga, A., Llorens, J.: Requirements Verification in the Industry. CSDM, Paris, France (2011)
19. Farbey, B.: Software quality metrics: considerations about requirements and requirement specifications. Inf. Softw. Technol. **32**(1), 60–64 (1990)
20. Fenton, N.E., Neil, M.: Software metrics: roadmap. In: Proceedings of the International Conference on Software Engineering (ICSE), pp. 357–370 (2000)
21. Fenton, N.E., Pleeger, S.L.: Software Metrics: A Rigorous and Practical Approach, 2nd edn. International Thomson Computer Press, Boston (1997)

22. Genova, G., et al.: A framework to measure and improve the quality of textual requirements. Requirements Eng. **18**(1), 25–41 (2013). doi:10.1007/s00766-011-0134-z. Url: http://dx.doi.org/10.1007/s00766-011-0134-z
23. Graesser, A.C., Mcnamara, D.S., Louwerse, M.M., Cai, Z.: Coh-Metrix: analysis of text on cohesion and language. Behav. Res. Methods Instrum. Comput. **36**(2), 193–202 (2004)
24. Henderson-Sellers, B.: Object-Oriented Metrics textendash Measures of Complexity. Prentice Hall PTR, New Jersey (1996)
25. Kemerer, C.F.: Progress, obstacles, and opportunities in software engineering economics. Commun. ACM **41**, 63–66 (1998)
26. Kitchenham, B.A., Pleeger, S.L., Fenton, N.E.: Towards a framework for software measurement validation. IEEE Trans. Softw. Eng. **21**, 929–943 (1995)
27. Klemola, T.: A cognitive model for complexity metrics, vol. 13 (2000)
28. Mcnamara, D.S.: Reading both high and low coherence texts: effects of text sequence and prior knowledge. Can. J. Exp. Psychol. **55**, 51–62 (2001)
29. Mcnamara, D.S., Kintsch, E., Songer, N.B., Kintsch, W.: Are good texts always better? Text coherence, background knowledge, and levels of understanding in learning from text. Cogn. Instr. **14**, 1–43 (1996)
30. Pleeger, S.L.: Lessons learned in building a corporate metrics program. IEEE Softw. **10**(3), 67–74 (1993)
31. Purao, S., Vaishnavi, V.: Product Metrics for Object-Oriented Systems. ACM Comput. Surv. **35**(2), 191–221 (2003)
32. Rakitin, S.: Software verification and validation: a practitioner's guide (Artech House Computer Library). Artech House Publishers, Norwood (1997). ISBN-10: 0890068895 ISBN-13: 978-0890068892
33. Ricker, M.: Requirements specification understandability evaluation with cohesion, context, and coupling. Ph.D. thesis, George Mason University, Fairfax, VA (1995)
34. Schneider, R.E., Buede D.,: Criteria for selecting properties of a high quality informal requirements document. In: Proceedings of the International Conference on Systems Engineering, Mid-Atlantic Regional Conference, INCOSE-MARC, 5–8 April 2000a, pp. 7.2-1–7.2-5 (2000)
35. Schneider, R.E., Buede D.: Properties of a high quality informal requirements document. In: Proceedings of the Tenth Annual International Conference on Systems Engineering, INCOSE, 16–20 July, 2000b, pp. 377–384 (2000)
36. Weyuker, E.: Evaluating software complexity measures. IEEE Trans. Softw. Eng. **14**(9), 1357–1365 (1988)
37. Wnuk, K., Regnell, B., Berenbach, B.: Scaling up requirements engineering – exploring the challenges of increasing size and complexity in market-driven software development. In: Berry, D. (ed.) REFSQ 2011. LNCS, vol. 6606, pp. 54–59. Springer, Heidelberg (2011)