A READABILITY CHECKER FOR IT-RELATED TEXTS

Research paper

Abstract

A major task in information technology (IT) is communication. Difficult-to-read texts hinder the communication between stakeholders and can lead to expensive consequences. We aim to design a software application reducing time and resources needed to improve the readability of IT-related texts. To identify the requirements of the tool, we conducted 13 interviews with employees of a software company followed by a quantitative survey. The results of the interviews confirmed the assumptions of previous work: Difficult-to-read texts hinder communication. We introduced the term readability anomaly as an indicator of difficult-to-read text passages that may negatively affect communication. Based on empirically determined requirements, we designed and implemented the readability analysis tool (RAT). The software determines readability anomalies with an average precision of 69% with high variation. Practitioners considered 64% of the findings as relevant and would correct 59% immediately. The analysis of our readability checker takes an average of 40 seconds for 10,000 words. Although some readability anomalies need to be adjusted or have to be supported by richer linguistic features, the checker provides effective means to improve the readability of IT-related texts and can support during review cycles.

Keywords: Natural Language Processing, Readability Assessment, Readability Checker, UIMA.

1 Introduction

As long as people have communicated through written language, the notion of text difficulty has been important. Two millennia ago, scholars in ancient Athens noted a concern about text comprehensibility: Legal arguments or analyses are of little persuasive value if their audience cannot understand them (Collins-Thompson, 2014; Zakaluk & Samuels, 1988). However, the scientific field of understanding the subjective and objectives factors associated with text comprehensibility and readability was only developed a century ago. The term *text readability* has been more formally defined by Dale & Chall (1948) as the sum of textual elements that affect readers' understanding, reading speed, and level of interest. These elements include – among others – linguistic features such as lexical richness, syntactical complexity or cohesion, which are the focus of this work. Besides linguistic features, the perceived difficulty of a text is also a function of the readers; it depends on their domain knowledge, their cognitive capabilities, and situational factors (Collins-Thompson, 2014). The assessment of text readability based on linguistic features was first applied by Flesch (1948) and Dale & Chall (1948). They developed readability formulas to classify the readability of school books in order to assign them to an appropriate grade level. These formulas were superior when using surface textual features, e.g. sentence or word length, despite more sophisticated text processing, such as coherence, dependency, and constituent having emerged in the 70s and 80s. This development resulted in a decline of readability research in the 90s. However, over the last decade, the fields of Natural Language Processing (NLP) and Machine Learning (ML) have made significant improvements, which have led to the resurgence of research interest (Francois & Miltsakaki, 2012; Tonelli, Manh, & Pianta, 2012). In this work, we transfer concepts of static code analysis to the readability assessment and improvement of IT-related texts, as ***Figure 1*** illustrates. The terminology of IT-related text, as used in this work, describes all text that comes into existence through the communication of stakeholders participating in the engineering lifecycle of an information system.

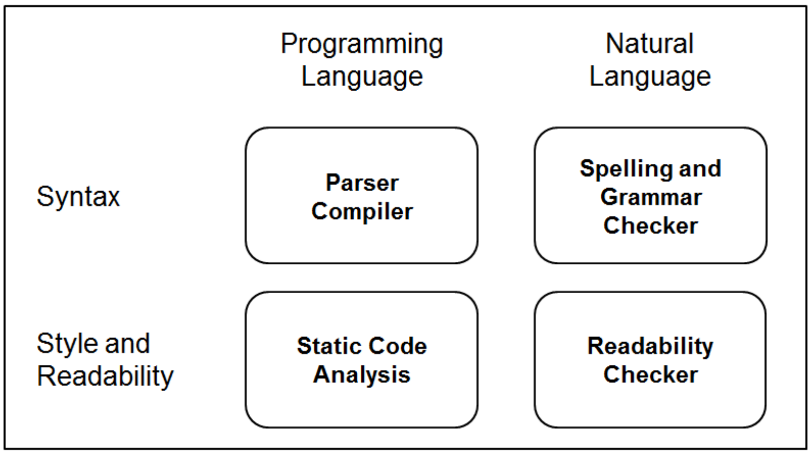


Figure 1. Analogy between static code analysis and the readability assessment of text

2 Research Method and Objectives

Throughout this paper, the information system research process, originally proposed by Hevner, March, Park, & Ram (2004), has been used. In ***Figure 2*** we present an overview of our research process.

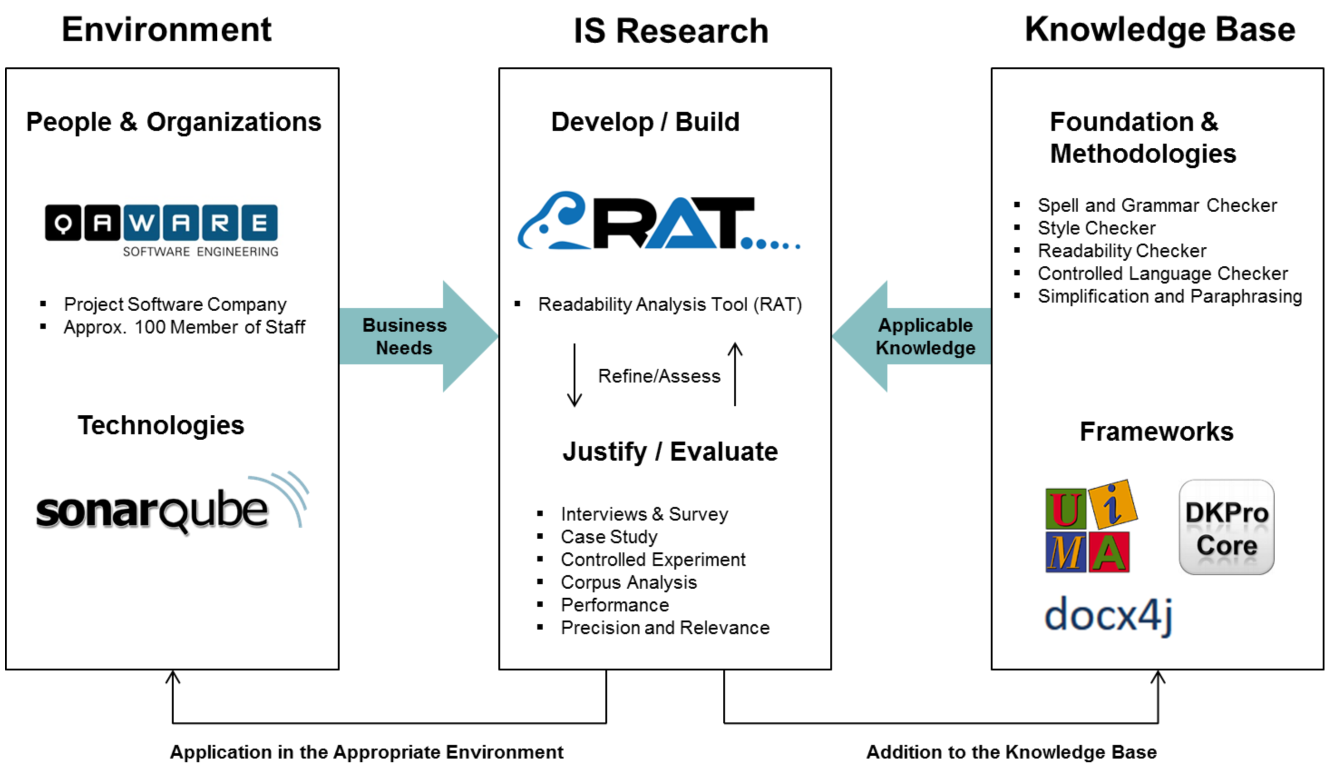


Figure 2. Information system research process proposed by Hevner et al. (2004)

To understand how a readability checker can be used similar to static code analysis in the quality assurance process of a software company, we have to find answers to the following questions:

RQ1 What problems are caused by difficult-to-read texts in IT?

First, we want to understand what problems are introduced by texts that are difficult to read. Thereby, we can infer solutions to design an artifact to approach these problems.

RQ2 How can a readability checker be integrated into the workflow of an IT company?

We want to identify the requirements that a readability checker has to fulfill to be accepted by users and determine how the tool can be integrated into the workflow of users.

RQ3 How can we improve the readability of IT-related texts?

We need to understand the components involved in readability and how we can improve them with tool support. In addition, we examine whether IT-related texts contain certain types of errors.

RQ4 How does a prototypical implementation of a readability checker for IT-related text look?

To evaluate our artifact and prove the utility of our approach, we implement the designed readability checker and thereby contribute technical solutions to the knowledge base.

RQ 5 How accurate is the readability anomaly detection?

To gain an understanding of the utility of our artifact, we want to determine the precision and recall of implemented readability rules for IT-related text.

RQ6 How many readability anomalies are relevant?

The findings of a readability checker are subjective – in contrast to findings of spelling or grammar checker. We aim to determine how many readability anomalies are relevant to users and how many they were aware of.

3 Knowledge Base

First attempts at readability assessment were made by Dale & Chall (1948) and Flesch (1948) with the development of readability formulas. The primary objective was the classification and assignment of school books to an appropriate grade level. Traditional readability formulas estimate the degree of readability based on superficial text features, such as the word length, sentence length or number of syllables per word. Dale & Chall (1948) argue that more complex textual features do not improve the quality of the measurement because superficial features correlate with more complex syntactic and semantic features. However, the results of readability formulas must be taken with caution. Overfitting can indicate good readability although this might not be the case (Davison & Kantor, 1982; Green & Olsen, 1986; Siddharthan, 2014). Furthermore, the result of a readability formula does not provide concrete information for improvement, which is the inherent problem of document-level readability assessment. Vajjala & Meurers (2014) examined how readability features that work at the document level can be applied to the sentence level.

In computational linguistics, the research field of controlled languages has garnered considerable attention due to the interest of the industry in creating user manuals that are less ambiguous and easier to translate (Wojcik & Hoard, 1997; Wojcik, Hoard, & Holzhauser, 1990). Controlled languages form a subset of our natural language by restricting vocabulary, syntax, and grammar, and thereby reduce ambiguity and complexity. Controlled language checkers have many goals in common with grammar, style, and readability checkers (Naber, 2003).

The MULTILINT project is a controlled language approach and style checker developed for technical documentations in the automotive sector. Reuther (1998) has collected requirements mentioned by technical authors who have used the tool. The technical authors stated that the integration of the tool into the workflow and local editing system is important. Furthermore, the tool should have reasonable processing times and rules dependent on the text category. Moreover, rules should be configurable. In addition, the tool should present an overview of the analysis and the overall quality of a text. Perin, Renggli, & Ressia (2010) argue that poorly written text fails to deliver ideas to the reader no matter how relevant those ideas may be. The authors claim that the concept of static code analysis can be applied to the readability assessment of text to alleviate this problem. They see improvements to their approach in developing rules for specific domains, in the usage of more sophisticated linguistic features, in allowing a user to decline findings, e.g. mark them as false positives, and in the improvement of the user interface. Femmer et al. (2016) transferred the concept of code smells to requirement engineering and investigated violations of language criteria in requirement artifacts derived from ISO 29148 (Software & Systems Engineering Standards Committee of the IEEE Computer Society, 2011). They conducted an extensive case study with the objective of understanding how many smells are present in requirement artifacts and how many of them are relevant. For prospects of future work, they propose to clarify and extend the rules applied to detect language violations, to understand how smell detection can be integrated as a supporting tool in an organization’s quality assurance process, and to obtain a thorough understanding of the impact that a quality defect has.

4 Environment

We elicited our requirements on the basis of 13 interviews and a survey with 46 participants in a project company for software technology that currently employs approximately 100 members of staff. We focused on interviewing a sufficient number of employees from the management of the company, and not only employees who are directly concerned with software development. Our objective is to obtain an understanding of the problems that difficult-to-read texts cause, the text processing programs and file types in use, the categories of text that IT-related texts are composed of, how employees write and edit text, whether IT-related texts contain specific error classes, whether the tool will be used if the findings are presented in a separate document (i.e. not in the text processing program), how long an analysis should take, how findings should be displayed, and how the readability checker can be integrated into the workflow of employees.

Problems caused by difficult-to-read texts

During the interviews, the key problems of difficult-to-read texts depicted in ***Table 1*** emerged. Afterward, we asked this question in our survey with the possibility of additional comments. Two employees added that readers might stop reading a difficult-to-read text or do not read it at all, which can lead to severe consequences.

Option	**Result**
Difficult-to-read texts take more time to read.	100.00%
	46
I do not understand parts of the content in difficult-to-read texts.	80.43%
	37
The editing of difficult-to-read texts takes more time.	71.74%
	33
Communication with team members is negatively affected by difficult-to-read texts.	69.57%
	32
Communication with customers is negatively affected by difficult-to-read texts.	65.22%
	30
Writing texts that are easy to read is difficult for me.	19.57%
	9
A text that is difficult to read does not cause problems, or if it does, they are not worth mentioning.	0.00%
	0
Total	**46**

Table 1. Problems caused by difficult-to-read texts according to our interviewees

Text processing programs and file types

Microsoft Word is the most used text processing program in QAware. However, AsciiDoc and Markdown are the preferred choices of text processing programs if the customer does not prescribe a specific program. We found that PowerPoint and e-mail clients are the main text processing programs for the communication of managing directors. Furthermore, Confluence was referred to as relevant text processing program. On the basis of our results, we developed the readability checker for Microsoft Word and the .docx file format that is standardized by the Office Open XML standard.

Text categories of IT-related texts

By sample inspections of the QAware corpus and our interviews, we determined nine categories of texts being written: Commercial documents, professional concepts, technical concepts, professional documentations, technical documentations, presentations, scientific articles, online text (advertising texts, blog posts and text on the QAware website), and e-mails. The quantitative survey revealed that e-mails and presentations are written the most. However, the use cases of these categories of texts require fewer rigors in proofreading and are therefore of lesser interest for readability assessment.

Use cases of writing text

Some interviewees mentioned that they would perform an analysis several times during the creation of a text. Others stated that they would want to use our tool after they finish a chapter. Still others indicated that they would use the software only once at the end of their writing. Hence, we could not infer a process model that generalizes.

Use cases of editing text

We found that a text undergoes a thorough editing if it has an external audience, a large audience, or a long life span. Furthermore, the editing is divided into a technical correction phase and a stylistic correction phase. On the basis of this result and the statement of a managing director, we target the text categories of technical documentations and concepts and neglect e-mails and presentations.

Error classes in technical documentations and concepts

The following error classes were reported by the interviewees: Missing redundancy and vague references, project implicit vocabulary and abbreviations, vocabulary that is too technical for the target readership, misinterpretations due to passive voice, long and nested sentences, modal verbs that mitigate critical statements and content that extends the text without providing information.

Time spent by employees in writing and editing texts

A considerable total of 15 participants (32.61%) write texts for more than 6 hours per week, which corresponds to 15% of their weekly working hours. Furthermore, 4 out of 46 participants are occupied by writing for more than two of their five workdays. Most employees write for 2-6 hours per week, comprising 5-15% of their workload. The editing of texts occupies 24 (52.17%) employees 1-2 hours and 11 employees (23.91%) for 2 to 6 hours per week.

Present findings in a separated document

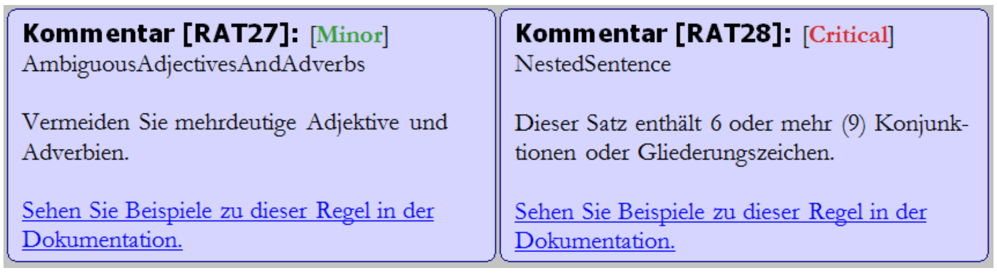
In case we could not programmatically embed findings into the text processing program in use, we aimed to present the findings within an HTML file. This means that a user would have to compare the original text with the HTML file and search for the findings in his or her text. Our survey revealed that 23 out of 46 participants (50%) would use the software regardless of findings being presented in a separate file. The tool would occasionally be used by 22 participants and 1 participant would not use the tool. In summary, participants responded that the inconvenience of a separate document could be ignored as long as the editing of a text is important enough and the findings are relevant enough.

How long should an analysis take

The more effort is justified to edit a text, the longer an analysis might take. Furthermore, depending on how frequently the user wants to use the software during the writing and editing process, the performance requirement varied between 1 second, 3-4 minutes, 15 minutes, and an hour or more. Three interviewees argued that real time might be counterproductive, since one may focus too much on the readability instead of the content of a text. These interviewees would not apply the analysis until a text reaches a certain degree of maturity. A period of between 5 and 10 minutes is acceptable in such a case. During the final check of a text, however, low turnaround times are desirable. On the basis of these results, we aimed for a performance of less than one minute for 10,000 words.

How findings should be displayed

At the end of the interviews, we showed our interviewees a mockup, presenting possible readability rules and how they could be displayed. By that, we determined how a readability anomaly should be displayed. ***Figure 3*** illustrates how readability anomalies are indicated by RAT in Microsoft Word. Each anomaly has a configurable severity level of minor (green), major (orange), or critical (red). Furthermore, the name and an explanation of the anomaly are depicted. The hyperlink leads to a detailed explanation of the readability anomaly and example sentences to aid the user in correcting the anomaly.



Kommentar [RAT27]: [Minor]
AmbiguousAdjectivesAndAdverbs

Vermeiden Sie mehrdeutige Adjektive und Adverbien.

Sehen Sie Beispiele zu dieser Regel in der Dokumentation.

Kommentar [RAT28]: [Critical]
NestedSentence

Dieser Satz enthält 6 oder mehr (9) Konjunktionen oder Gliederungszeichen.

Sehen Sie Beispiele zu dieser Regel in der Dokumentation.

Figure 3. Indication of a readability anomaly as a Microsoft Word comment

How a readability checker should be integrated into employee's workflow

Most interviewees stated that a plugin in the text processing program would be the most convenient way to use the tool, but that it is not required. In summary, the interviewees mentioned the following options to integrate the software support: Text processing program plugin, version control integration, web service, and command line tool. Two interviewees considered that developing an artifact tailored to a particular text processing program might result in less portability to other programs. We designed our artifact as a command line tool. In this way, we avoided the aforementioned concerns and achieved portability as well as extensibility. In addition, by designing clear interfaces, we can use the core of the command line tool implementation for the version control and the web service integration.

5 Design and Implementation

The readability checker RAT is built on the Apache UIMA architecture. Using UIMA in conjunction with the DKPro Core component collection allowed us to instantiate and configure different NLP pipelines at runtime, without the need for manual integration or configuration. RAT extracts text from a Microsoft Word file using the docx4j library, annotates the text with linguistic annotations and applies readability rules. Thereafter, the findings are incorporated as comments in the Microsoft Word file. To meet our performance requirement of one minute for a text containing 10,000 words, we restricted the linguistic analysis to part-of-speech annotations.

5.1 Workflow of an Analysis

The workflow of our readability analysis tool is depicted in ***Figure 4*** and explained in this section.

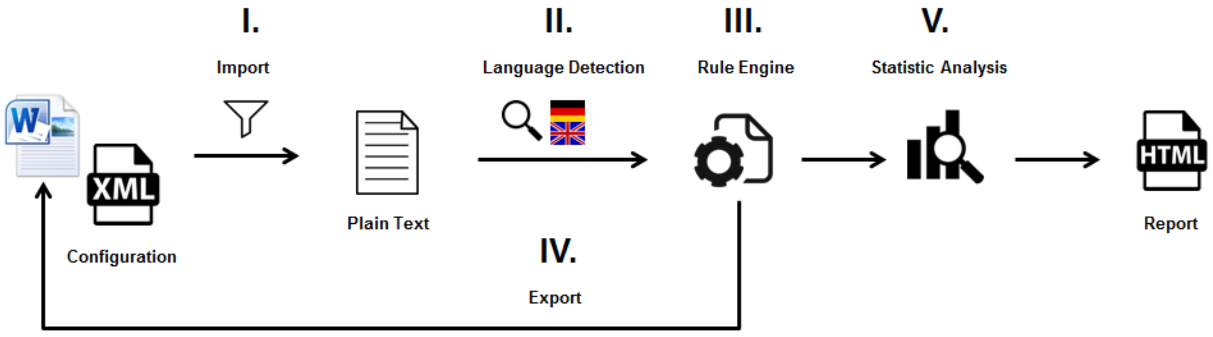


Figure 4. Processing steps of an analysis by RAT

I. Import

In the first step, the arguments that refer to the file(s) to be analyzed, the path of the configuration file and the output directory are parsed and validated. Thereafter, the file extension of each file is detected and a service locator searches for an importer module implementation that supports the detected file type. If such an implementation does not exist, the user is notified and the analysis of the current file is skipped. If an implementation exists, the relevant text from the supported file is extracted and filtered.

The filtering of text sections is achieved by searching for content that is commonly introduced by these sections, e.g. references in a bibliography. The result of the first step is a plain text representation of the file that can be analyzed by the linguistic and rule engine in step (III). Since we have a plain text representation of the text, we lose a great deal of information about the original file. Therefore, we designed an abstraction of the core elements of the .docx file format that allows us to locate text passages in step (IV).

II. Language Detection

The second step performs the detection of the language of the extracted text. On the basis of this information, we assemble an appropriate NLP pipeline and apply readability rules for the given language. If the detected language is not supported, the user is notified and the current file is skipped.

III. Linguistic Engine and Rule Engine

In the third step, the linguistic analysis is performed. Subsequently, readability anomalies are detected by RAT.

IV. Export

Once readability anomalies have been detected, they are classified as redundant, declined, incorporated, or anomalies to apply. The information about the classification of readability anomalies is stored in a custom XML file within the .docx file. The information is also provided to the user via the HTML report of step (V). Afterward, the readability anomalies to apply are located in the original text and are applied as Microsoft Word comments. Each comment provides information about the name of the anomaly, the severity, the violations that have caused the anomaly and a link to the GitHub documentation with further information about the anomaly.

V. Statistics and Quality Gate

In the fifth step, RAT computes 19 quantitative statistics of the text, e.g. the average sentence length, the reading time, or the most used conjunctions. In addition, readability formulas are calculated. The statistics are aggregated into a quality gate. The quality gate provides information about the overall readability of a text and can be configured using the configuration XML file. The concept of the quality gate is similar to that of static code analysis. The output of the statistical analysis (V) is an HTML report that summarizes the statistics, readability formulas, and information of readability anomalies.

5.2 Conceptual Overview

RAT consists of four major components: The importer, the linguistic engine, the rule engine, and the exporter. We use Service Provider Interfaces (SPIs) to encapsulate our components. By that, the components can be exchanged without affecting one another. DKPro Core allows us to integrate existing spelling, grammar or style checker into our tool. For example, we can integrate the open-source checker LanguageTool[1] by adding the component to our linguistic engine and make use of the existing functionality to show the findings in a .docx file. ***Figure 5*** illustrates the components of RAT and the common analysis structure (CAS) of UIMA's component-based architecture. Different components only interact with the CAS object, which allows for interchangeability, as long as the required input and output types of the components comply. For the development of our readability rules, we use Java and UIMA Ruta, which both allow for configuration of rules.

[1] LanguageTool: https://languagetool.org/, last access on 11-26-2016.

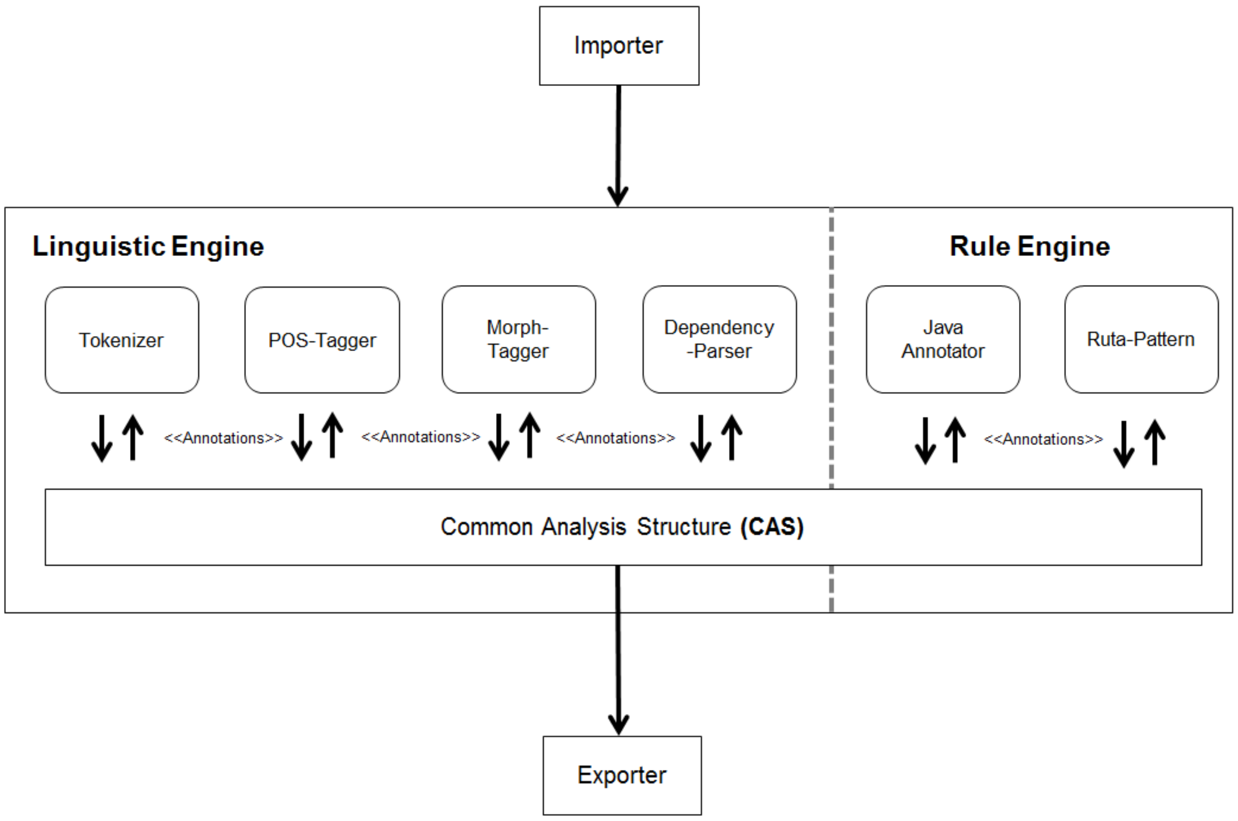


Figure 5. Conceptual overview of the readability checker RAT

6 Evaluation

In this section, we present the results of the evaluation of our work.

6.1 Precision and Relevance of Readability Rules

RAT detects by a second analysis of a previously analyzed text whether the user had rejected a finding or incorporated a finding. In this way, we can comprehend the user's editing process. This allows us to evaluate our readability checker not only in a practical setting but also under realistic conditions, in which an employee could be under pressure. Based on this approach, we determined the relevance of 258 findings (including false positives) during the application of RAT. Furthermore, we evaluated the precision by manually classifying 425 findings in texts of the QAware corpus. Both results are depicted in ***Table 2***.

In addition, we asked three employees to classify a sample of 52 true-positive anomalies regarding their relevance. First, we asked our participant whether he or she was aware of the anomaly. Subsequently, we asked if the finding is relevant. If a finding was classified as relevant, we asked whether the participant would correct the finding immediately, in the short term, or the long term. Overall, our participants considered 64% of the findings to be relevant. They were not aware of 48% of the findings. Lastly, they would act on 59% of the presented findings immediately, on 23% in the short term, and on 18% in the long term.

#	Readability Rule	Precision	Relevance
1	AdjectiveStyle *Five or more adjectives in a sentence.*	96 %	67 %
2	AmbiguousAdjectivesAndAdverbs	64 %	68 %
3	ConsecutiveFillers	82 %	60 %
4	ConsecutivePrepositions	33 %	33 %
5	DoubleNegative	43 %	25 %
6	FillerSentence *Three or more fillers in a sentence.*	57 %	60 %
7	LeadingAttributes *Four or more words between an article and its corresponding noun.*	38 %	24 %
8	LongSentence *A sentence with more than 35 words.*	65 %	73 %
9	LongWord *A word with more than 8 syllables.*	52 %	31 %
10	ModalVerbSentence *Two or more modal verbs in a sentence.*	86 %	67 %
11	NestedSentence *A sentence with six or more conjunctions or punctuation marks.*	32 %	40 %
12	NominalStyle *Three or more nouns ending on "heit", "keit", or "ung" in a sentence.*	97 %	89 %
13	PassiveVoice	87 %	42 %
14	SentencesStartWithSameWord *Two sentences start with the same word.*	69 %	43 %
15	SubjectiveLanguage	67 %	33 %
16	Superlative	100 %	17 %
17	UnnecessarySyllables	100 %	75 %
	Average	**69 %**	**50 %**
	Overall	**68 %**	**49 %**

Table 2. Precision and relevance of implemented readability rules

6.2 Reflection on Research Questions

RQ1 What problems are caused by difficult-to-read texts in IT?

We found that difficult-to-read texts cause problems in IT-related texts. The most common problem is the increased reading time. All 46 participants of our quantitative survey agreed with this. Furthermore, 37 participants indicated that they do not understand parts of the content of texts that are difficult to read. Moreover, 33 employees mentioned that editing takes longer for texts that are difficult to read. In addition, communication in the team and with customers is negatively affected according to 30 and 33 participants, respectively. Besides time expenses, several employees mentioned that the occurring problems could lead to misunderstandings and defects in the development process.

RQ2 How can a readability checker be integrated into the workflow of an IT company?

Throughout our interviews, we found different preferences depending on employees' individual workflow. To gain an understanding of how to integrate the tool into their workflow, we first wanted to understand how fast an analysis has to be. Some of our interviewees stated that the analysis should perform in real time and that they would not use the tool otherwise. Conversely, others responded that they would use the tool even if an analysis took several hours. In fact, they even claimed that real-time analysis might be distracting. We received similar results for the technical integration. While most employees want to use the tool as a plugin in their text processing program, few see this integration as a decisive requirement. In addition, we asked whether employees would use the tool if the findings were not presented in the analyzed text. We found that half of the 46 participants of our survey would use the tool even when the results were presented in a separate document. All three results depend on similar factors: Number, precision, and relevance of detected anomalies and the category, relevance, and maturity of the text.

RQ3 How can we improve the readability of IT-related texts?

We derived the readability rules for our artifact from five different sources: The error classes mentioned by our interviewees, rules from related work in academia (Femmer et al., 2016; Graesser, McNamara, & Kulikowich, 2011; O'Brien, 2003), recommendations and specifications in the domain of requirement engineering and administration language (Bundesverwaltungsamt, 2002; Software & Systems Engineering Standards Committee of the IEEE Computer Society, 2011), and linguistic and journalistic books on the subject of readability (Rechenberg, 2006; Schneider, 2001, 2011).

We found that the underlying problem of many readability anomalies can be explained by the theory of working memory. The working memory is „the temporary storage of information in connection with the performance of other cognitive tasks such as reading […]" (Baddeley, 1983, p. 311). Miller (1956) found that we can only store 7 (±2) information units in our working memory. A readability anomaly occurs when these information units are unnecessarily occupied by a text. On the lexical level, our working memory takes more time to process infrequent or long words (Just & Carpenter, 1980). On the syntactic level, long and complex sentences increase the syntactic processing that our working memory has to perform (Graesser et al., 2001). On the dependency level, pronouns and vague references occupy our working memory with dissolving these references. Readability anomalies cause less working memory to be devoted to higher-level semantic processing that we need to understand a text (Mason & Kendall, 1979).

RQ4 How does a prototypical implementation of a readability checker for IT-related text look?

We implemented the readability analysis tool (RAT).[2] The tool extracts text from .docx files, detects the language of a text, assembles an appropriate linguistic pipeline, and applies readability rules. Findings are presented in the original .docx file as Microsoft Word comments. The comments explain the anomaly and include a hyperlink to an online documentation. The rules can be configured (e.g. the threshold of long sentences or words) and disabled. RAT detects readability anomalies that have been declined and incorporated by users. This enables a semi-automated evaluation of readability rules – that is, we can comprehend a user's editing process without manual interaction. An HTML report presents quantitative statistics, readability formulas, and findings of readability rules. In addition, the report summarizes all measurements and provides insight into the overall readability of the text.

RQ5 How accurate is the readability anomaly detection?

[2] Readability Analysis Tool (RAT): https://github.com/qaware/readability-analysis-tool, last access 11-24-2016.

Precision: We determined an average precision of 69% and an overall precision of 68% with high variation. Seven out of seventeen readability rules had a precision greater than 70%, which is considered acceptable in static code analysis (Bessey et al., 2010). The `AmbiguousAdjectivesAndAdverbs` rule's precision was negatively affected because terms were used in mathematical expressions, e.g. „minimal" and „maximal". `ConsecutiveFillers` and `FillerSentence` were erroneous when filler words were used as conjunctions or idiomatic expression. We found that two prepositions separated by a comma are not difficult to read and hence not fulfill the rule definition. The implementation of the `DoubleNegative` rule is solely based on lexical text features. If two words indicating a negation are found, a finding is assumed. Furthermore, the `LeadingAttributes` rule led to false positives, when anglicisms are not recognized as nouns. Anglicisms are particularly common in IT-related texts. Therefore, the rule led to many false positives. The `LongSentence` rule's precision was flawed due to errors in text extraction. In particular, erroneous filtering of figures led to false positives. Long words were detected incorrect due to errors in the filtering of hyperlinks and file names. We examined syntactic complexity in sentences with the `NestedSentence` rule. We had erroneous results for the `NestedSentence` rule because words after a delimiter (i.e. commas, semicolons, or dashes) are frequently annotated as conjunctions by the `OpenNlpPosTagger`. In addition, we have not considered enumerations. The `SubjectiveLanguage` was flawed when a word was used in another context or idiomatic expression.

Similar to Femmer et al. (2016), we found false positives due to grammatical errors in the text, imprecisions in NLP libraries, and the fact that readability rules do not take the context into account. In addition, anglicisms, sentence boundary detection, and the definition of our word lists led to false positives. Furthermore, many false positives were caused by errors in text extraction. In particular, when sections of a text were not correctly detected, i.e. cover sheet, content, or bibliography, or when references or hyperlinks were not filtered correctly.

Recall: We tested the implemented rules against sample texts from the QAware corpus with various border cases. Thereby we found problems which affect the recall of readability rules. First, the sentence boundary detection was flawed. Abbreviations, hyperlinks, or other constructions that include punctuation marks caused this. We found that typical German abbreviations were not detected correctly by Java's `BreakIterator` and OpenNLP's `Segmenter`. Second, the readability rules `PassiveVoice` and `DoubleNegative` are only supported by lexical features. These rules have to be supported by more sophisticated linguistic features. However, this would result in longer processing times.

RQ6 How many readability anomalies are relevant?

Our participants considered 64% of true-positive anomalies as relevant. Moreover, they were not aware of 48% of the findings. Lastly, they would act on 59% of the presented findings immediately, on 23% in the short term, and on 18% in the long term. We also examined the relevance of findings during the application of our artifact. By that, participants were also confronted with false-positive findings. We found an average relevance of 50% in this scenario. The relevance of readability rules is influenced by two factors: First, there are no obligatory writing guidelines in QAware. This means that individual decisions are made about the writing style. As we noted in our interviews, the opinions on good writing style differ. Second, context-specific relevance applies to several readability rules. For example, if the passive voice is used to introduce a topic, it rarely has negative effect on readability.

7 Conclusion

In this paper, we presented an open-source readability checker for IT-related texts and its application in an IT company. To identify problems caused by difficult-to-read texts, we conducted 13 interviews with employees of an IT company, followed by a quantitative survey. The requirements for our readability checker (RAT) were derived from our empirical studies and the research of related approaches. We defined the term *readability anomaly* as an indicator of difficult-to-read text passages that may negatively affect communication of stakeholders in IT. Furthermore, we provided definitions of 17 concrete readability rules to detect readability anomalies. We evaluated our approach in terms of performance, precision, and relevance. The analysis performed by RAT takes an average of 40 seconds for 10,000 words. This is achieved by conducting the linguistic analysis only up to part-of-speech annotations. The readability rules are defined accordingly. We obtained an average precision of 69% and relevance of 64% with high variations. Since rules can be configured and disabled, many practitioners were already satisfied with both the precision and relevance.

We see three advantages of our approach. First, the editor can focus on the content of a text instead of stylistic errors. Second, we create awareness about the importance of readability. Third, common writing guidelines can be established. However, the low precision of some readability rules can cause unnecessary work. To alleviate this problem, we allow users to configure or disable individual readability rules, as proposed by (Naber, 2003; Reuther, 1998). Furthermore, an anomaly can be declined by users, thereby excluding it in subsequent analyses, as suggested by (Perin et al., 2010). Several behavioral studies show the impact of text difficulty on comprehension (DuBay, 2007). Through our empirical studies, we confirmed these results. Moreover, we found specific error classes in IT-related texts. In addition, our study showed that it is necessary to tailor the detection of readability anomalies to the category of the text and the workflow of users.

It should be borne in mind that not all of our requirements and evaluation results can be generalized, since they are subjective in that they emerged from an application of RAT in QAware.

7.1 Limitations and Future Work

Based on our application in a practical environment, we found the following requirements and prospects for future work: Improvement of the precision and relevance of anomalies, domain-specific readability rules, configurability of the anomaly detection, paraphrasing of detected anomalies, performance of an analysis, integration into the workflow of a company, support of multiple file formats, and the extent of integration in text processing programs.

Some of these aspects are contrary. For example, a sophisticated linguistic analysis to support precise readability rules impedes performance. We found that practitioners had different opinions regarding this conflict. However, improvements in both areas can be achieved without negatively affecting one another: First, the performance of NLP components can be increased through efficient algorithms and parallelization techniques; second, readability rules can be refined through empirical studies. On the basis of our interviews, we identified error classes and functional requirements that can be integrated into future versions of RAT, for example, structural error classes and paraphrasing suggestions. To achieve paraphrase suggestions for lexical anomalies, we want to apply language modeling. Furthermore, we like to examine the impact of word sense disambiguation on readability. By evaluating our approach, we found problems that impair the precision of readability rules. To improve the precision, we pursue working on the context sensitivity of readability rules, the sentence boundary detection, and the text extraction. Whether the reported precision and relevance of our readability checker is sufficient for industry also requires additional research.

Femmer et al. (2016) argue that „spell and grammar checkers are used on a daily basis, although they are far away from 100% recall". Consequently, the precision of a readability checker might be more important than its recall. Yet, empirical evidence in readability research is difficult to obtain because many findings depend on subjectivity. Moreover, the link between difficult-to-read texts and the impact on communication between stakeholders in IT must be thoroughly examined by future work. Both academic approaches and industry solutions suggest that the extent of integration in the workflow and text processing program are decisive requirements for the acceptance of a readability checker (Perin et al., 2010; Reuther, 1998). The results of our interviews indicated that these requirements depend on the accuracy of the anomaly detection and properties of the text. Therefore, we need to further investigate how a readability checker can be integrated into the workflow of practitioners.

8 References

Baddeley, A. (1983). Working memory. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences, Vol. 302, No. 1110*. (1110), 311–324.

Bessey, A., Block, K., Chelf, B., Chou, A., Fulton, B., Hallem, S., … & Engler, D. (2010). A few billion lines of code later: using static analysis to find bugs in the real world. *Communications of the ACM*, *53*(2), 66–75.

Bundesverwaltungsamt. (2002). *BBB-Arbeitshandbuch „Bürgernahe Verwaltungssprache“* (4th ed.): Bundesverwaltungsamt – Bundesstelle für Büroorganisation und Bürotechnik (BBB).

Collins-Thompson, K. (2014). Computational assessment of text readability: A survey of current and future research. *ITL-International Journal of Applied Linguistics*, *165*(2), 97–135.

Dale, E., & Chall, J. S. (1948). A formula for predicting readability: Instructions. *Educational research bulletin*, 37–54.

Davison, A., & Kantor, R. N. (1982). On the failure of readability formulas to define readable texts: A case study from adaptations. *Reading research quarterly*, 187–209.

DuBay, W. H. (2007). *Smart Language: Readers, Readability, and the Grading of Text*: ERIC.

Femmer, H., Méndez Fernández, D., Wagner, S., & Eder, S. (2016). Rapid quality assurance with Requirements Smells. *Journal of Systems and Software.* doi:10.1016/j.jss.2016.02.047

Flesch, R. (1948). A new readability yardstick. *Journal of applied psychology 32.3*. (3), 221.

Francois, T., & Miltsakaki, E. (2012). Do NLP and machine learning improve traditional readability formulas? *Proceedings of the First Workshop on Predicting and Improving Text Readability for target reader populations. Association for Computational Linguistics*, 49–57.

Graesser, A. C., Karnavat, A. B., Daniel, F. K., Cooper, E., Whitten, S. N., & Louwerse, M. (2001). A computer tool to improve questionnaire design. In *Paper presented at the funding opportunity in survey research seminar on June 11, 2001.*

Graesser, A. C., McNamara, D. S., & Kulikowich, J. M. (2011). Coh-Metrix providing multilevel analyses of text characteristics. *Educational researcher*, *40*(5), 223–234.

Green, G. M., & Olsen, M. S. (1986). Preferences for and Comprehension of Original and Readability-Adapted Materials. Technical Report No. 393.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information System Research. *MIS Quarterly Vol. 28 No. 1*, 75–105.

Just, M. A., & Carpenter, P. A. (1980). A theory of reading: from eye fixations to comprehension. *Psychological review*, *87*(4), 329.

Mason, J. M., & Kendall, J. R. (1979). Facilitating Reading Comprehension through Text Structure Manipulation. *Alberta Journal of Educational Research*, *25*(2), 68–76.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, *63*(2), 81.

Naber, D. (2003). *A rule-based style and grammar checker* (Diplomarbeit). Technische Fakultät, Universität Bielefeld, Bielefeld.

O'Brien, S. (2003). Controlling controlled english. an analysis of several controlled language rule sets. *Proceedings of EAMT-CLAW, 3*, 105–114.

Perin, F., Renggli, L., & Ressia, J. (2010). *Natural Language Checking with Program Checking Tools.* University of Bern, Switzerland, Bern.

Rechenberg, P. (2006). *Technisches Schreiben: (nicht nur) für Informatiker* (3., erw. und aktual. Aufl). München [u.a.]: Hanser.

Reuther, U. (1998). Controlling language in an industrial application. In *Proceedings of the Second International Workshop on Controlled Language Applications, CLAW* (Vol. 98, pp. 174–184).

Schneider, W. (2001). *Deutsch für Profis: Wege zu gutem Stil* (Überarb. Taschenbuchausg). *Goldmann Mosaik bei Goldmann: Vol. 16175*. München: Goldmann.

Schneider, W. (2011). *Deutsch für junge Profis: Wie man gut und lebendig schreibt*. *Rororo: Vol. 62629*. Reinbek bei Hamburg: Rowohlt-Taschenbuch-Verl.

Siddharthan, A. (2014). A survey of research on text simplification. *ITL-International Journal of Applied Linguistics, 165*(2), 259–298.

Software & Systems Engineering Standards Committee of the IEEE Computer Society. (2011). ISO/IEC/IEEE 29148:2011(E), Systems and software engineering — Life cycle processes — Requirements engineering.

Tonelli, S., Manh, K. T., & Pianta, E. (2012). Making readability indices readable. *Proceedings of the First Workshop on Predicting and Improving Text Readability for target reader populations. Association for Computational Linguistics,* 40–48.

Vajjala, S., & Meurers, D. (2014). Assessing the relative reading level of sentence pairs for text simplification. In *EACL* (pp. 288–297).

Wojcik, R. H., & Hoard, J. E. (1997). Controlled languages in industry. In *Survey of the state of the art in Human Language Technology* (pp. 238–239).

Wojcik, R. H., Hoard, J. E., & Holzhauser, K. C. (1990). The boeing simplified english checker. In *Proc. Internatl. Conf. Human Machine Interaction and Artificial Intelligence in Aeronautics and Space, Centre d'Etude et de Recherche de Toulouse* (pp. 43–57).

Zakaluk, B. L., & Samuels, S. J. (1988). *Readability: Its Past, Present, and Future*: ERIC.