Automated Quality Assessment of Natural Language Requirements

Zur Erlangung des akademischen Grades eines

Doktors der Wirtschaftswissenschaften
(Dr. rer. pol.)

von der KIT-Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Patrick S. Kummler

1. Gutachter	Prof. Dr. Hansjörg Fromm
2. Gutachter	Prof. Dr. Harald Sack

Tag der mündlichen Prüfung: 17. August 2021

Karlsruhe, 2021

Abstract

High demands on quality and increasing complexity are major challenges in the development of industrial software in general. The development of automotive software in particular is subject to additional safety, security, and legal demands. In such software projects, the specification of requirements is the first concrete output of the development process and usually the basis for communication between manufacturers and development partners. The quality of this output is therefore decisive for the success of a software development project.

In recent years, many efforts in academia and practice have been targeted towards securing and improving the quality of requirement specifications. Early improvement approaches concentrated on the assistance of developers in formulating their requirements. Other approaches focus on the use of formal methods; but despite several advantages, these are not widely applied in practice today. Most software requirements today are informal and still specified in natural language.

Current and previous research mainly focuses on quality characteristics agreed upon by the software engineering community. They are described in the standard ISO/IEC/IEEE 29148:2011, which offers nine essential characteristics for requirements quality. Several approaches focus additionally on measurable indicators that can be derived from text. More recent publications target the automated analysis of requirements by assessing their quality characteristics and by utilizing methods from natural language processing and techniques from machine learning.

This thesis focuses in particular on the reliability and accuracy in the assessment of requirements and addresses the relationships between textual indicators and quality characteristics as defined by global standards. In addition, an automated quality assessment of natural language requirements is implemented by using machine learning techniques. For this purpose, labeled data is captured through assessment sessions. In these sessions, experts from the automotive industry manually assess the quality characteristics of natural language requirements.

The research is carried out in cooperation with an international engineering and consulting company and enables us to access requirements from automotive software development projects of safety and comfort functions. We demonstrate the applicability of our approach for real requirements and present promising results for an industry-wide application.

Acknowledgement

Normally, the content of a research thesis and its successful completion are also elementarily based on the cooperation of several people. Therefore, from a personal view, I would like to thank the relevant people who contributed to the success and execution of my research.

Reasonably, I would like to sincerely thank Prof. Hansjörg Fromm, who not only qualitatively supported the work with his excellent supervision, but also contributed unconditionally to the success of the thesis by providing scientific guidance and important advice. In particular, the detailed discussions and the exciting explanations of his personal experiences from industrial practice have provided important necessary impulses for my research.

Essentially, Prof. Gerhard Satzger and the many colleagues at the Karlsruhe Service Research Institute also contributed to my success. They introduced me to elementary methods of machine learning and natural language processing and gave research advice and valuable hints for which I am very grateful for.

Quite unquestionably, without the commitment of my employer, it would have been unfeasible to conduct my research in parallel with my job as a competence manager. I would therefore like to express my deep gratitude to Emmanuel Chronakis for his true support and for providing me with the necessary space for my research.

Finally, this work would not have been possible without the support of my entire family. My special gratitude goes to my wife Verena, who not only reviewed the vast amount of references, but also supported and encouraged me unconditionally. At the end, our daughter Amilia motivated me to finally finish this work.

Contents

1 Introduction

In short, software is eating the world.

— Marc Andreessen

In most areas of life, a total pervasion of software and digital services can be observed. "Before long, we can expect increasing public concern about the pervasiveness of software, not only in public services but also in consumer products like automobiles, washing machines, telephones, and electric shavers" (Kitchenham & Pfleeger, 1996, p. 1). This is referred to as the megatrend "digital transformation" and permeates areas and industries such as healthcare, railway services, and finance equally (Brennen & Kreiss, 2014; Collin, 2015; Kane et al., 2017). The phenomenon of digital transformation is the "employment of new digital technologies to foster major business improvements in organizations" (Hanelt et al., 2015, p. 1315).

Marc Andreessen stated in 2011, that “we are in the middle of a dramatic and broad technological and economic shift in which software companies are poised to take over large swathes of the economy” (Andreessen, 2011, p. 2). Software is everywhere. Large companies and entire industries focus on developing software and implementing enhanced digital services for their customers. This change is particularly noticeable as many new software-based companies have emerged in a short period and are growing rapidly. Today, Amazon, as a software-based company, is (not only) the largest book retailer in the world. Netflix is a software company known as the world’s largest video service. Spotify and iTunes, both software companies, are the most dominant music companies (Goodwin, 2015). Hardly an area of life is skipped in this transformation process.

These companies do not manufacture physical products or procure production materials. Their product is based on software and the ability to manage the thin layer between cost-intensive supply systems of others on the one hand and the access to potential consumer classes on the other. “The value is in the software interface, not the products” (Goodwin, 2015). Nearly all industries that are affected by this trend, and the automotive industry makes no exception, were mainly considered in the physical world. However, software is now taking over an increasing part of the value-added chains here as well.

1.1 Motivation

The digital transformation faces manufacturers in the automotive industry with new and unknown competitive challenges (Berman & Bell, 2011; Hanelt et al., 2015). “The global automotive industry is at the vanguard of a digital revolution” (Duncan et al., 2015, p. 1) and the value creation is increasingly shifting from hardware to software and services (Reidel, 2016).

Car manufacturers are faced with the "general trend of servitisation, which moves customers from being vehicle owners to using sharing services instead" (Moravek, 2020). They are evolving from product manufacturers to providers of integrated mobility services (Seiberth, 2015). Industrial boundaries are becoming blurred, value chains are being reconfigured, and new competitors from different industries are appearing. The transition to open digital ecosystems is taking place. The vehicle as a product is not at the center of value creation anymore, but rather provides access to a world of services. Digital platforms are the focus of vehicle development and the car will be part of an open and huge system.

Hardly any other industry is growing as fast and setting as many trends as the automotive industry. Yet electronics is the most important innovation and growth factor. Software-based services and functions have a significant influence on the customers’ purchasing decisions and, accordingly, have become a main differentiator in the market. Nowadays, customers no longer decide to buy a vehicle based on design and technical performance alone. The decision is largely driven by the user experience that is based on electronic functions and services provided to the customer (Ebert & Lederer, 2012). As a result, "the vehicle is also referred to as a third place in which people work, communicate and consume alongside their home and workplace" (de Buhr, 2015, p. 1).

Today, automotive manufacturers are offering numerous digital services for their customers. As one of many examples, the app-based service "Porsche Passport" allows subscribers to drive a new vehicle model of their choice on demand (Korosec, 2019). Another example is an over-the-air service which makes it possible to analyze problems that occur in the vehicle. The car owner does not have to go to a dealer’s workshop for analysis and even minor problems can be solved remotely (Volkswagen, 2020). In addition to digital services, customer functions are also a decisive criterion and contribute significantly to the success of today’s vehicles. Driving should be comfortable and safe. This leads to a high pressure to innovate on car manufacturers and poses extensive challenges in the development of such systems. The manufacturer with the first system assumes market leadership in a technological area that is only just emerging.

"The quick increase of software and the traditional structures of the car industry make it difficult for this old economy to adapt fast enough to the quite different requirements of software-intensive system, which cars become more and more" (Broy, 2006, p. 34). However, the automotive industry will have to continue to adapt, as the days of traditional car manufacturers are over (Diess, 2020). The car as a product is in the middle of the "transition from a hardware-driven machine to a software-driven electronics device" (Burkacky et al., 2018). An entire industry with a history of more than 100 years, originally focused on the production of physical goods, is forced to transform into a software-based industry with a multitude of new potential suppliers and competitors. Consequently, “the car will become a software product” (Diess, 2019).

More than 90 percent of automotive innovations are driven by mechatronic systems, in which software has developed as an essential differentiating factor (Lan et al., 2008; Seiberth, 2015). Software engineering is therefore becoming increasingly important for the automotive industry to meet current and future challenges. "Software and IT are the major drivers of modern cars—both literally and from a marketing perspective" (Ebert & Favaro, 2017). Additional demands for quality, cost efficiency, and compliance with norms and standards are rapidly increasing the complexity of a vehicle.

In figure 1.1, we[1] show how complexity has changed in the automotive industry, mainly due to the increase in software-driven innovations. Three complexity drivers can be identified that have a major influence on automotive development (Ebert & Favaro, 2017). First, the pervasion of software and digital services results in increasing liability risks and additional security and legal requirements. Second, the increasing complexity of software functions causes a significantly higher number of requirements in development projects. Third, distributed development leads to additional communication efforts. These drivers are briefly described in the following.

[1]For stylistic reasons, the generally accepted "we" is used in this thesis. The use increases the readability and involves the reader more on the topic. In addition, it draws attention to the fact that the successful completion of the thesis and the content of the research were based on the participation of more than one person. To further increase readability, we also use the third person in the masculine form "he". For this thesis, this usage should be considered independent and equally valid for all genders.

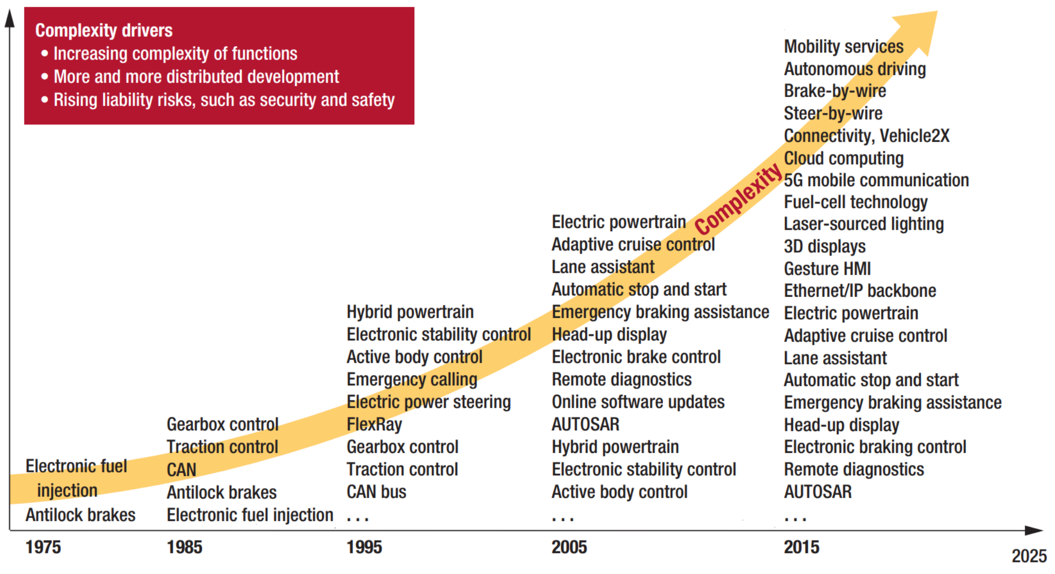


Fig. 1.1.: Software innovations and complexity drivers in the automotive development (Ebert & Favaro, 2017).

Demands.

Over the last decades, many electronic systems have become established in the automotive industry and their use is now also sometimes prescribed by law. This applies, for example, to systems such as the anti-lock braking system (ABS). The increasing demands on the safety of developed functions, on the availability of systems, and on the general information security can be observed generally in the automotive industry. To meet these demands, quality assurance measures must be taken into account during the entire development process and industry-specific standards must be considered and complied with.

One of the essential standards in the field of functional safety is represented by the ISO standard 26262 (Road vehicles - Functional safety; ISO 26262, 2011), which must be applied as a basis for the development of safety-critical functions and systems in the car. Technical and functional safety concepts ensure that the safety requirements are taken into account during development, thus considerably reducing the risk or danger of a faulty and insufficiently secured function (Ebert & Lederer, 2012).

Additional demands are also based on the requirements of rating companies. Euro NCAP is a European company that assesses the safety of vehicles. For the launch of a new vehicle, the automotive manufacturers strive for a high rating from Euro NCAP as a quality characteristic for potential customers: „Achieving a maximum five star award has proven to be a strong selling point for auto manufacturers. Its achievement relies on complex and sophisticated assisted driving systems“ (Chitkara et al., 2013, p. 12). Thus, when developing software-based functions, automotive manufacturers focus as well on the requirements of Euro NCAP to achieve a good assessment for new cars (Seiniger & Weitzel, 2015).

Besides, legal requirements have a major influence on the development of automotive software as well. As an example, to shorten the time it takes to get help in an accident, a software-based function called "eCall" has been mandatory in every new vehicle in Europe since 2018. According to Singh (2015), human error is responsible for more than 90 percent of car accidents. Advanced driver assistance systems (ADAS) help to improve passenger safety and to reduce the number of accidents. Although this number has been declining for years, accidents cannot be completely avoided today. The eCall function automatically detects an accident, provides improved assistance to the driver, and sends relevant data to the next emergency station.

Automotive development is faced with extensive and interdisciplinary challenges of software development and is characterized by increasing external factors. Legal requirements and global standards contribute to the increase of additional requirements in the development of software, particularly for software that is critical to safety. Besides, the increasing complexity of software and functions must also be taken into account when developing software. Thus, we briefly describe the second driver in the next section.

Complexity.

The automobile is one of the most complex products today. Whereas in the past, systems like engine control units (ECUs), airbag systems, and ABS were responsible for the increasing complexity, today electronic systems for autonomous driving, connectivity functions, and electric mobility are the main drivers for complex and challenging development (Ebert & Lederer, 2012). However, the complexity sometimes increases so quickly that it is hardly controllable. "As the importance of electronics and software has grown, so has complexity" (Burkacky et al., 2018). New security incidents involving automotive software are uncovered almost weekly.

If the number of software lines of code (LOC) is considered as a criterion for the complexity of automotive software, it is already apparent in 2009 how complex software can be in premium vehicles[2]. At that time, the software of a Mercedes

[2]We are aware that LOC (= lines of code) is not a reliable measure of software complexity. Nevertheless, complexity and volume of software are often used synonymously.

S-Class contained about 100 million LOC and up to 100 microprocessors (Charette, 2009). This figure is even more surprising as a Boeing 787 Dreamliner from the same year required about 6.5 million LOC to operate the avionics and on-board systems (Desjardins, 2017; MacDuffie & Fujimoto, 2010). The radio and navigation systems of the Mercedes alone required more than 20 million LOC (Charette, 2009). The number of ECUs required in the S-Class is comparable to the number that is installed in an Airbus A380.

Moreover, an immense increase can be observed between 2010 and 2016. While an average car contained around 10 million LOC in 2010, this number has increased to 150 million LOC only 6 years later (Burkacky et al., 2018). The trend is further reinforced by the increase in electric mobility, as the technology for electric vehicles is largely based on software[3]. Some even assume that the number will rise to over one billion LOC in the next ten years (Antinyan, 2020). Therefore it does not seem unreasonable to refer to cars as "computer on wheels" (Meissner et al., 2020).

In addition to extensive possibilities for improving safety, comfort, and entertainment in a vehicle, this development also poses a major challenge for the automotive industry. "Complexity, I would assert, is the biggest factor involved in anything having to do with the software field. It is explosive, far reaching, and massive in its scope" (R. Glass, 2002, p. 20). Industrial practice mostly shows a relation between the complexity of a system and its size, which can be supplemented based on further characteristics (Burkacky et al., 2018; MacDuffie & Fujimoto, 2010). Therefore, the number of LOC in a vehicle rises with growing complexity, which in turn is based on increased interconnections and underlying software requirements in automotive development projects. "As software grows so [do] the requirements" (Huertas & Juárez-Ramírez, 2013, p. 234).

With the discussion about the increasing complexity of automotive software functions, we presented the second driver. The complexity driver usually complements the previous driver, since, with additional demands from different areas, the complexity can also increase. The last driver considers the approach of distributed development and is shortly presented in the next section.

[3]The increasing development of electric cars is also justified by the fact that with the EU regulation 2019/631 defined limit values for CO_2 emissions are given. Otherwise, automotive manufacturers are threatened with severe fines. Therefore, the development of electric vehicles is currently the only way to meet the requirements of this regulation, which sets an average emission level for the entire vehicle fleet of an automotive manufacturer.

Distributed development.

The different challenges of software development force the automotive manufacturers to question even well-known and established processes and approaches to adapt them to current conditions and trends in the market. Starting points for this can be uncovered throughout the entire development process and be optimized through a distributed development approach.

In general, it is hardly possible to meet the challenges arising in the automotive industry on one's own. Therefore, new alliances are emerging between previous competitors, such as between the BMW Group and Daimler AG (Borgmann, 2019). "Ford and Volkswagen enter self-driving car joint venture" (Tobin, 2019) and invest together in the autonomous vehicle technology platform Argo AI. The automotive groups PSA and FCA are planning to join forces "to build a world-leading company" and to jointly master the challenges that lie ahead (Press Release, 2019). "Marking the beginning of a new era, past competitors now join forces and work together on highly strategic projects" (Klein & Ferres, 2019).

These partnerships are only some of many announcements to strengthen the market position of car manufacturers and to face new competitors, for example in the field of autonomous driving. "An extensive network of partnerships, alliances and joint ventures looking to shape the future of autonomous mobility" (Klein & Ferres, 2019). Figure 1.2 provides an extract of this network and shows the partnerships, collaborations, and participations between car manufacturers (red circle), traditional automotive suppliers (blue circle), and technology companies (green circle).

Besides direct partnerships and collaborations, larger alliances between companies are increasingly emerging. Figure 1.2 also shows the joint venture MONET, which defines the cooperation for the development of autonomous driving technology between several Asian car manufacturers (e.g. Toyota, Suzuki, and Mazda). Other examples are PAVE (Partners for Automated Vehicle Education) and NAV (Networking for Autonomous Vehicles), two approaches that define alliances between global automobile manufacturers and suppliers.

"Traditionally the car industry is highly vertically organized. In software engineering we would say it is modular" (Broy, 2006, p. 34). The major part of a car is developed by a number of different development partners and then integrated into the car by the manufacturer. This share is growing as more and more partners are involved due to the increasing software development. Such an approach, referred to as distributed development, has some decisive advantages (Weber & Weisbrod, 2002). In addition to an optimal allocation of costs and resources, e.g. through the support

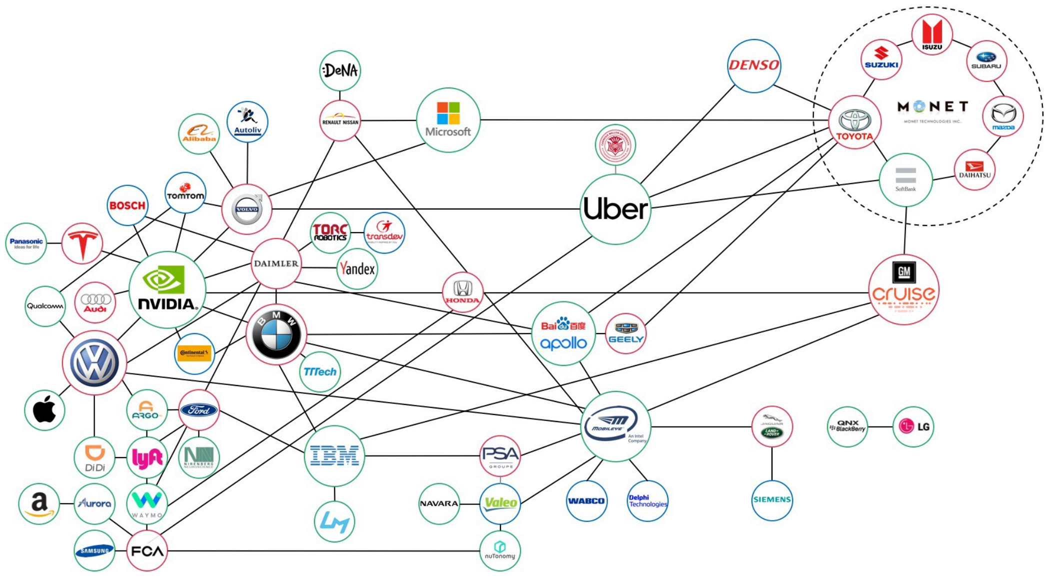


Fig. 1.2.: Overview of autonomous driving partnerships (Klein & Ferres, 2019).

of external engineering companies, it also enables the use of relevant knowledge and competencies from other industries.

Therefore, development is often carried out in cooperation with several partners (e.g. manufacturers, suppliers, engineering companies, mobility providers). An essential prerequisite for successful realization is reliable communication between the partners during the entire product life cycle, but especially in the phases in which the requirements are defined and communicated. However, "the communication between OEMs[4] and suppliers has to be organized via the requirements documents, which nowadays are often not precise and not complete enough" (Broy, 2006, p. 37).

We have identified three factors that have a major influence on the development of automotive software and its requirements: First, additional safety and legal requirements are a challenge for automotive manufacturers, which also results in an increase of software requirements in the development process. Second, due to increasingly complex functions, additional software requirements are necessary. Third, collaborative and distributed development in automotive software projects requires not only additional communication effort but also the assurance of reliable information exchange. For all three factors, software requirements play a decisive role and are crucial for the successful implementation and realization of automotive development projects.

[4]OEM (= Original Equipment Manufacturer) is synonymous with car manufacturer.

Conclusion.

In the beginning, the automotive industry had little to do with software development. Meanwhile, software is the most important innovation driver and with each new generation of a vehicle, the software volume almost doubles (Ebert & Lederer, 2007). At the same time, automotive development is shifting from a component-based approach to a software function approach. "Development will become a continuous process that fully decouples cars‘ rather stable hardware from their functionality driven by software upgrades" (Ebert & Favaro, 2017). Ensuring software quality is therefore becoming increasingly complex, time-consuming, and cost-intensive, which leads to growing attention to quality assurance methods in the software development process.

It has long been recognized that the success of a software development project strongly depends on the quality of the underlying requirements (Curtis et al., 1988; Kamata & Tamai, 2007; Knauss et al., 2009) and “requires fluid collaboration and communication between clients and software engineers” (Génova et al., 2013, p. 2). Failures in any development phase can often be traced back to miscommunication in the requirements phase. "The serious problems that have happened with software have to do with requirements, not coding errors" (Somers, 2017).

Car manufacturers are faced with the challenge of writing "good" software requirements. Failures in the requirement specification account for 70 to 85 percent of all the rework costs in the development process (Wiegers, 2003, cited by Kiritani and Ohashi, 2015). Consequently, the quality of requirements is in a new focus. "Given this central role of requirements as a mean to communicate intention, assuring their quality is essential to reduce misunderstandings that lead to potential waste" (Unterkalmsteiner & Gorschek, 2017, p. 77).

Improving requirements quality has been a concern of software engineering from the beginning on. Early improvement approaches concentrated on the management and discipline of the development process and on the assistance of developers in formulating software requirements. Several established approaches, such as formal specification methods to describe and specify software requirements, already exist. But, “despite the significant advantages attributed to the use of formal specification languages, their use has not become common practice” (Wilson et al., 1997, p. 1) and they have not been widely accepted. Requirements are still organized in text documents and used as the basis for the communication between stakeholders. “Natural language is key in requirements engineering” (Berry, 2007, p. 1).

As the majority of software requirements are still written in natural language, most discussions focus on the quality of natural language requirements. Consequently, methods from natural language processing have been used to analyze requirements. Besides, textual indicators were identified that are believed to be determinants of certain quality characteristics. In a first generation of requirements analysis tools, these textual indicators were analyzed and visualized to the developer for the purpose of quality improvement.

Due to the massive increase in requirements, new, reliable, and automated approaches are necessary to tackle the problem of requirements quality and to enable quality assurance for software development. "Such (semi-)automatic assessment can reduce the time needed for requirements analysis and validation in the requirements specification phase, and ultimately increase the quality" (Ormandjieva et al., 2007, p. 39). Therefore, a new generation of tools aims at quantitatively assessing the quality of requirements by the degree of fulfillment of its quality characteristics. For this purpose, machine learning algorithms are well suited. A single research study is known that uses classification algorithms to assess the quality of software requirements (Parra et al., 2015). The assessment, however, relates to a rather unspecific quality definition and only distinguishes "good quality" from "bad quality". What is missing so far is an attempt to quantitatively assess software requirements with respect to standardized quality characteristics on a finer scale. This is exactly the objective of our research, which we present in more detail in the following section.

1.2 Research design

In this thesis, we focus on the quality assessment of natural language requirements through characteristics. We identify such approaches that cope with the current challenges of the industry and that enable an automated quality assessment of requirements (e.g. Fabbrini et al., 2001; Fantechi et al., 2003; Génova et al., 2013; Huertas & Juárez-Ramírez, 2013; Ormandjieva et al., 2007; Wilson et al., 1997; H. Yang et al., 2012). Frameworks to measure and improve the quality of natural language requirements already exist (e.g. Parra et al., 2015).

Our research follows the idea to use machine learning techniques for the assessment of text quality. For this purpose, we use indicators for properties in natural language requirements that an expert takes into consideration when manually assessing a requirement. We draw upon the groundwork on requirements quality that has emerged from various standardization efforts within the software engineering community. The standard ISO/IEC/IEEE 29148:2011 (ISO 29148, 2011) provides a set of characteristics that are believed to determine the quality of an individual requirement. We consider these quality characteristics in our research.

In our first research question, we ask about the possibility to automatically assess requirements according to these quality characteristics by using machine learning techniques.

Research question 1

Can machine learning algorithms be used to accurately assess quality characteristics of a natural language software requirement?

The question is not only whether a machine learning algorithm can be used, but also whether the algorithm delivers sufficiently accurate results. In our research, the accuracy is related to the correct assessment of five discrete quality classes (1 = "very bad" till 5 = "very good"), which are defined for a classification problem.

In general, discrete values describe a limited number of intervals in a continuous range of values, while continuous values can be infinite. The former is more closely oriented to a representation on the knowledge level (Simon, 1981). Besides, data is reduced and simplified by a discrete approach, which is easier to understand and explain, not only for experts (Liu et al., 2002). Finally, discrete values can influence the speed and accuracy of some learning algorithms (Dougherty et al., 1995). Thus, for our research, we decide to use discrete class values.

As our machine learning approach is based on supervised learning and our first research question is similar to tasks known in the area of text analysis, some prerequisites should be taken into account. One of the most important preconditions is a certain amount of assessed requirements—so-called labeled data—to train and build a machine learning model and to enable an automated assessment of non-labeled requirements. We gather labeled data through the quality assessment of requirements by experts.

However, we can not be sure if and how well the experts can assess the quality characteristics of a natural language requirement. The analysis and the assurance of the reliability of the assessments are crucial for the accuracy of our machine learning model. Therefore, in a second research question, we ask whether such a manual assessment can be carried out reliably by experts at all.

Research question 2

Can experts reliably assess quality characteristics of a natural language software requirement?

We work out in this thesis how reliability is measured and when we consider the experts' assessments to be reliable. In our research, the second research question needs to be answered before the first research question can be addressed. This leads us to the overall outline of our research that distinguishes two consecutive research parts in the thesis. Both parts focus on requirements assessment in two ways: First, we let experts manually assess software requirements from our data set using defined quality characteristics; second, we use the labeled data as input to our machine learning model to implement an automated assessment of requirements.

For the first part, we have requirements assessed by experts based on the quality characteristics and present results from the assessment sessions. These sessions are conducted with experts from industrial practice that are working in automotive software development projects and are sensitized to questions of requirements quality. The results can be used to determine not only a reliable assessment of requirements between experts but also a reliable requirements assessment of a single expert. We also derive whether experts are at all capable of assessing each quality characteristic of a single natural language requirement. Further analyses ensure that the labeled data are appropriate for the implementation of an automated approach.

In the second part, we use the labeled data—i.e. assessed requirements from experts—to train a classifier by applying machine learning techniques and follow a classical method widely used for text mining tasks. We are able to identify textual indicators proposed from literature and industrial practice, that can be extracted from requirements text. Based on the textual indicators and the labeled data we implement supervised learning and demonstrate the feasibility of our approach.

With the proof of an accurate quality assessment based on quality characteristics that even experts can reliably assess, both research parts are covered. To the best of our knowledge, there is no comparable approach that fully incorporates the quality characteristics of the standard ISO/IEC/IEEE 29148:2011 for the assessment of requirements quality. In addition, we did not find any other research that analyzes whether the characteristics can be assessed by experts and, by means of indicators, that implements an automated assessment approach for these.

The research is done in cooperation with an international automotive engineering and consulting company and enables us to have access to software requirements from the automotive industry. Our approach is applied and evaluated on data from different automotive manufacturers. The use of industrial requirements and an assessment by experts from the automotive industry ensure reliable results.

The subsequent benefit of our research can be described from different points of view. A general increase in requirements quality leads to clear communication and understanding of requirements between manufacturer, supplier, and development partner. At the same time, the costs for testing and validation based on requirements that fulfill the quality characteristics can be reduced to such an extent that reliable test concepts can already be developed during the initial development phases. A prior analysis and assurance of requirements quality allow the identification and optimization of poorly specified requirements at an early stage of a development project. This ultimately leads to the early detection of errors and inaccurate requirements.

With our research, we contribute to different parts of relevant scientific areas. We analyze existing approaches that assess the communication quality based on informal text. We use input from experts' perspectives to enhance these and aim at creating an approach that automatically assesses text according to indicators. Finally, through the assessment, we also support practitioners in specifying "well-written" natural language requirements to improve requirements quality.

1.3 Structure of thesis

In this section, we describe the structure of the thesis that is also depicted in figure 1.3. The thesis is divided into five chapters, which are described in detail in the following.

Chapter 1
The first chapter of the thesis focuses on the introduction and motivation for our research. We identify three drivers that have an influence on the current software development particularly in the automotive industry. In this chapter, we also present the research design of the thesis and introduce our two research questions.

Chapter 2
Relevant foundations for our research and contributions to the quality assessment of natural language requirements are presented in chapter two. We analyze different quality characteristics and discuss the standard ISO/IEC/IEEE 29148:2011. Also,

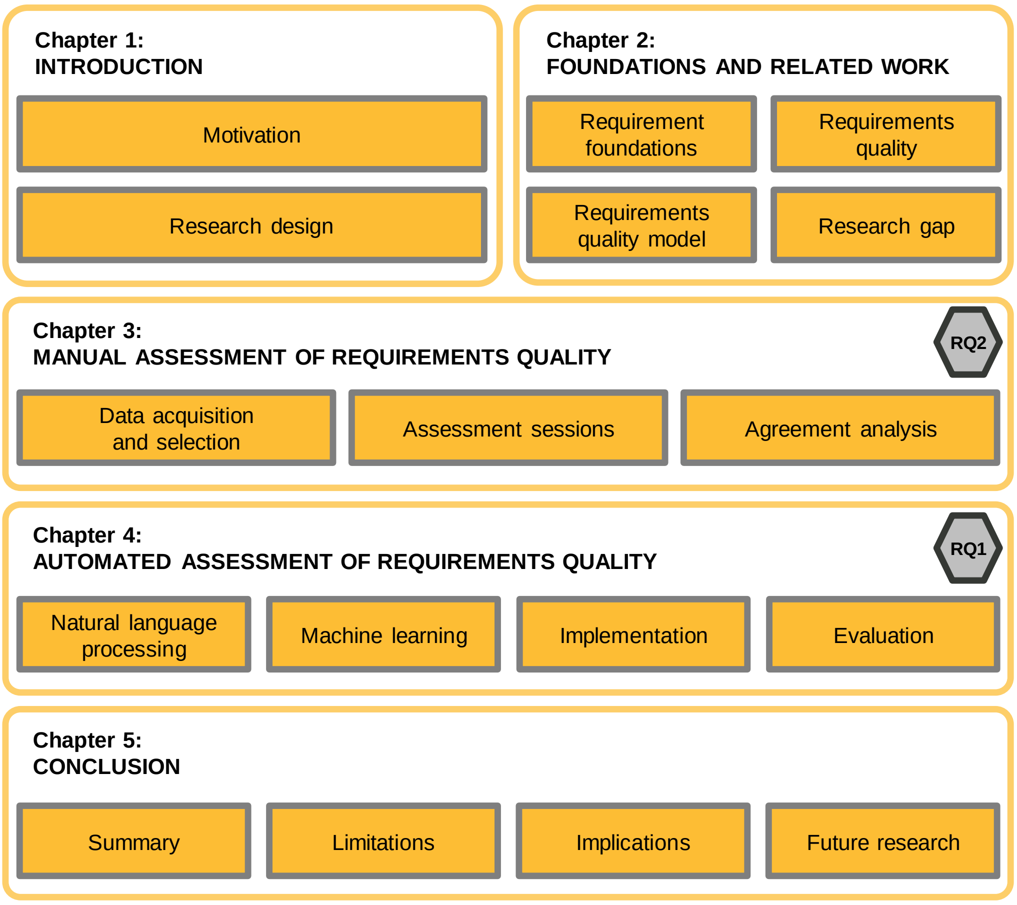


Fig. 1.3.: Structure of the thesis.

we present our requirements quality model, which is used for the requirements assessment. This chapter discusses approaches and contributions from related work that enable us to identify the research gap. Finally, we address existing challenges and issues for our research topic.

Chapter 3

In this chapter, we focus on our second research question. Based on the research gap and our requirements quality model from the previous chapter, we reveal the approach for the assessment of requirements quality by experts. In addition, the assessment framework is shown and our requirements data basis is presented. We describe in detail how the expert sessions are conducted and evaluate the results. The chapter is divided into three activities: Data handling, requirements assessment, and agreement analysis.

Chapter 4

The fourth chapter addresses our first research question and describes our approach to automatically assess the quality of natural language requirements. We present and discuss the development of features for measuring requirements quality, activities for data preprocessing, and statistical analyses. This chapter also describes the implementation of the machine learning models that enable an automatic assessment of requirements quality. It concludes with a performance evaluation, a model optimization, and the determination of the feature importance.

Chapter 5

Finally, this chapter contains a summary of the work and discusses the limitations and implications of our research. We also provide an outlook on future research activities.

Foundations and related work[1] 2

If you don't know where you're going,
any road will take you there.

— George Harrison

In this chapter, we lay the foundations of the thesis and present definitions for a common understanding of relevant terms. In the first part, we explain why requirements engineering plays a decisive role in today's software development and describe the relevance of requirements quality for the success of a project.

In the second part, we focus on the question, how requirements can be assessed from a quality point of view. Different contributions dealing with the definition and measurement of requirements quality are presented and discussed. We identify a relevant research gap and highlight our approach of a requirements quality model. Finally, we describe existing research issues and conclude with a summary of the chapter.

2.1 Requirement foundations

In the following section, we define the term "requirement" in the context of our research. We present the concept of requirements engineering and discuss techniques for the formulation of requirements. The section concludes with how requirements quality affects the processes and results of a development project.

2.1.1 Requirement definition

The term "requirement" is widely common in different areas. Generally, it is used to call for and to urge for something in a defined context. When we talk about minimum requirements, we specify technical or non-technical requirements that have to be fulfilled at least to realize pre-defined minimum targets. For example, for a new

[1]Parts of this chapter have already been published in Kummler (2017) and Kummler et al. (2018).

job, potential candidates have to fulfill requirements regarding skills and experience. A requirement can also be seen for regulations and rules. Legal requirements can serve as human protection. The same applies to hygienic requirements for a hospital, a supermarket, and a public swimming pool that includes the fulfillment of relevant standards. Thus, a requirement mostly "requires" something to be fulfilled.

If we take a closer look in the context of software development, a requirement is defined as what is expected of a software system as property by stakeholders (Balzert, 2010). Stakeholders, in general, are persons and organizations with a direct interest in the development of the software and its subsequent use. "The term stakeholder generalizes the traditional notion of customer or user in requirements engineering to all parties involved in a system's requirements" (Glinz & Wieringa, 2007, p. 18). Ebert (2019) extends the definition of a requirement and considers properties as conditions, attributes, and goals of the product. Consequently, requirements are the "wish list" of the customer (Ebert, 2019). In many, even current, literature, a more specific definition provided by the Institute of Electrical and Electronics Engineers (IEEE 610, 1990) is used to define a requirement as:

(a) A condition or capability needed by a user to solve a problem or achieve an objective;

(b) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document;

(c) A documented representation of a condition or capability of the previous two arguments.

Since our research refers to software requirements that are written in natural language, we partly refer to the approach from the IEEE and define a software requirement as follows:

Definition of a software requirement

A software requirement is a natural language representation of a condition or a capability of software that complies with defined quality characteristics and that is appropriately specified to enable developers to design software that meets that requirement.

In most cases, software requirements are further divided into different types, such as functional requirements, quality requirements, and basic conditions (Pohl & Rupp, 2015). For our research focus, however, such a distinction is not relevant. Usually, requirements are summarized in documents called specifications. This specification particularly refers to the standard ISO/IEC/IEEE 29148 (ISO 29148, 2011)[2], which contains a description for a software requirements specification (SRS). According to ISO 29148, an SRS is described as follows:

Definition of a software requirements specification (SRS)

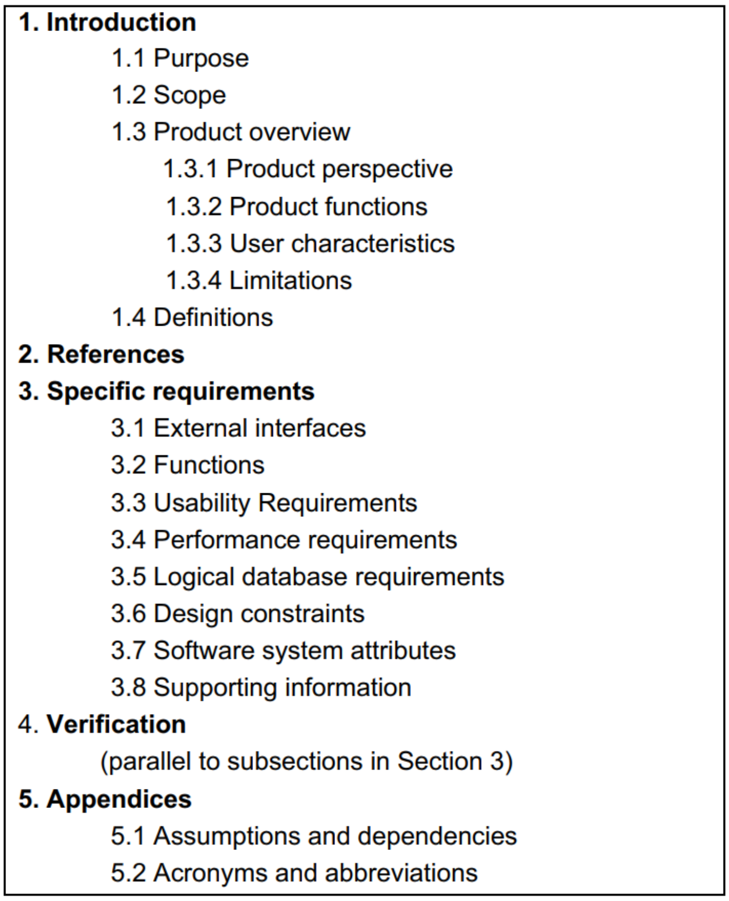
The SRS defines all of the required capabilities of the specified software product to which it applies, as well as documenting the conditions and constraints under which the software has to perform, and the intended verification approaches for the requirements.

The standard also provides a predefined structure that includes the chapters "Introduction", "References", "Specific requirements", "Verification" and "Appendices" (see figure 2.1). The chapter "Introduction" contains the purpose and scope of the software as well as the product overview, which lists all relevant interfaces to other systems. Definitions are provided, where specific words or phrases in the context of the software are explained. In "References" relevant documents for the software are included, which are not part of the SRS.

The chapter "Specific Requirements" describes "every input (stimulus) into the software system, every output (response) from the software system, and all functions performed by the software system in response to an input or in support of an output" (ISO 29148, 2011, p. 58). The requirements are assigned to different categories (e.g. functions, performance requirements). An example of a performance requirement is provided by the standard ISO 29148: "95% of the transactions shall be processed in less than 1 second" (ISO 29148, 2011, p. 59). This chapter contains the software requirements on which we focus our research.

In "Verification", approaches and methods for the verification of the software requirements are described. Lastly, "Appendices" contains assumptions and dependencies for software development as well as acronyms and abbreviations used in the context of the software development project.

[2]We apply the ISO/IEC/IEEE 29148:2011 standard as a basis for the definition of requirements quality. In the following, we refer to the standard as ISO 29148.



1. Introduction
- 1.1 Purpose
- 1.2 Scope
- 1.3 Product overview
  - 1.3.1 Product perspective
  - 1.3.2 Product functions
  - 1.3.3 User characteristics
  - 1.3.4 Limitations
- 1.4 Definitions

2. References

3. Specific requirements
- 3.1 External interfaces
- 3.2 Functions
- 3.3 Usability Requirements
- 3.4 Performance requirements
- 3.5 Logical database requirements
- 3.6 Design constraints
- 3.7 Software system attributes
- 3.8 Supporting information

4. **Verification**
- (parallel to subsections in Section 3)

5. Appendices
- 5.1 Assumptions and dependencies
- 5.2 Acronyms and abbreviations

Fig. 2.1.: Structure of an SRS according to ISO 29148.

The requirements in an SRS document are the basis for a development project and necessary for the estimation, planning, and implementation of activities. "Good requirements are essential if we are to be sure that we are building the system the users want, and that we are not doing more than is needed" (Hall, 1997, p. 3). Requirements also form a legal basis for contracts "between the customer and the developer or as a source of information for the project managers" (Berry et al., 2006, p. 1). Furthermore, the exchange of requirements between all relevant stakeholders makes it possible to ensure that each participant has the same level of information. Even after the successful completion of a project, requirements are used as a basis for the operation and maintenance of the software.

In this section, we have defined the term "requirement" and assigned it in the context of a software requirements specification (SRS). Requirements are of great importance for the development process and the subsequent product life cycle of software. We describe the relevance of requirements, particularly for the former, in the following section.

2.1.2 Requirements in the development process

In general, the high complexity in the development of a software system forces the division into subsystems as well as into different development phases. For this purpose, systematic approaches were created early on and are still developing further today (Benington, 1983). Such approaches are generally reflected in process models. These models consider the development process from the beginning, which usually starts with the analysis of requirements (Ebert, 2014; Rupp, 2014). Most of today's process models are based on the waterfall model from Royce (1987). It describes individual development phases that follow a flow-oriented process. At the end of each phase, specific documents are generated.

For the development of software, the V-model is usually used, which has established itself as a further development from the waterfall model. The V-model, which was originally published in 1992, defines activities along a chronological development process (Hakuli & Krug, 2015; IABG, 1993). The main difference to the waterfall model is the consideration and integration of quality assurance in the development process, as well as the parallel definition and synchronization of tests (Balzert, 2008). This enables the creation of suitable test cases for requirements already during the first phases of a project.

To take into account all the different characteristics of projects, the V-model comprises a total of four submodels: software development (SWD), quality assurance (QA), configuration management (CM), and project management (PM). In the submodel SWD, several fields of activity that are required for the development of software are described. In addition to a requirements analysis for a system, software, and hardware, a subdivision into a rough and detailed design on the software and hardware level is mandatory. Only afterward activities for implementation and integration take place. Each of the activity fields describes defined results by creating specific documents (e.g. software requirements specification), which are necessary for the implementation of subsequent activities.

Mostly, requirements are derived from specification documents, which become more concrete and detailed as the development process progresses. Figure 2.2 shows the simplified V-model and the individual phases and documents of a development project according to ISO 29148. Initially, based on stakeholder requirements, specifications (StRS = stakeholder requirements specification) are generated from which system requirements are derived and summarized in system requirements specifications (SyRS). The SyRS in turn form the basis for the specification of software requirements and serve as an information source for the creation of one or more software requirements specifications (SRS). Already in the development phase between the system and software specification, requirement documents are forwarded to different departments of a company. In industrial practice, it is also common that external engineering companies support the specification of requirements in these phases as well.

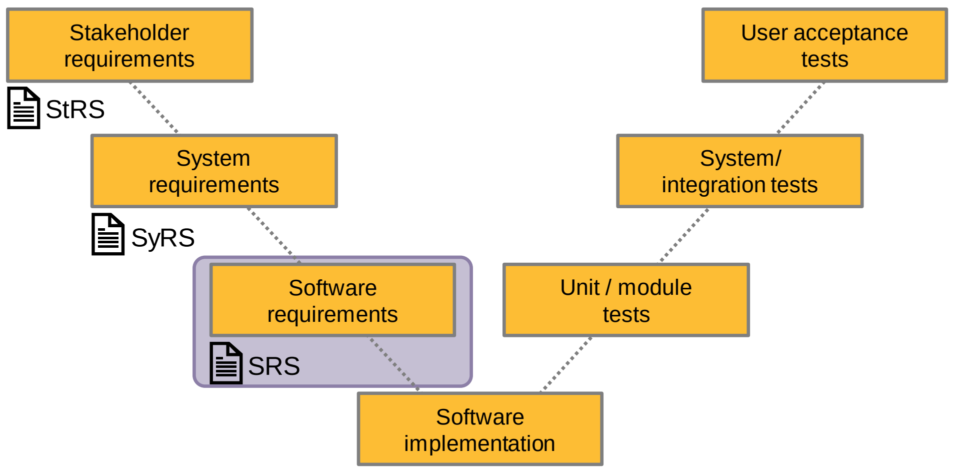


Fig. 2.2.: Simplified V-model according to ISO 29148.

Prior to the software implementation, the SRS is available in the most precise level of detail, in which, for example, specific signals, relevant parameters, and detailed specifications for function performance are specified. "Software requirements specification (SRS) documents are the medium used to communicate user requirements to the technical people responsible for developing the software" (Ormandjieva et al., 2007, p. 39). Nowadays the implementation of the software is often carried out by external development partners that use the SRS as a basis for communication with the client.

Therefore, between the software requirements phase and implementation phase, some sort of review or inspection should take place, since the concrete implementation is based on the SRS (Kassab et al., 2014). "Software requirements specification is a critical activity of the software process, as errors at this stage inevitably lead to problems later on in [...] implementation" (Ilieva & Ormandjieva, 2005, p. 1). Mostly, such a review can be very time-consuming for reasons we have presented in the motivation of our thesis (see section 1.1). However, the software requirements determine the basis on which the development partner implements the software. Besides other characteristics, that we present later in this chapter, an "SRS should be unambiguous both to those who create it and to those who use it. However, these groups often do not have the same background and therefore do not tend to describe software requirements the same way" (IEEE 830, 1998, p. 5). For that reason, we focus on requirements from the SRS and the assessment of requirements quality in our research.

There are further requirement activities that are reflected in the concept of requirements engineering. In general, the differences between software engineering and requirements engineering are based on respective disciplines for the product to be developed. Software engineering focuses on the creation of software systems based on methods and processes from a quality perspective. In contrast, requirements engineering, as a subtask of software engineering, is an important part of project planning and the fundament for every software development. In the following section, we provide a brief overview of requirements engineering, describe techniques for the specification of software requirements, and explore the question of why requirements engineering has a crucial influence on the success of a project.

2.1.3 Requirements engineering

It is now more than 40 years since "Requirements Engineering" was introduced as an independent discipline in an edition of the "Transactions of Software Engineering" (see D. Fernández, 2018). Due to its enormous relevance and impact on the success of a software project, it has received much international attention in research and practice. "Requirements engineering is a discipline which has emerged over the last few years in response to the realization that too many systems were being built which worked, in some sense, but did not do what their users wanted" (Moffett, 1999, p. 1). Many decisions in the course of a software project are based on the activities and the output of requirements engineering. Requirements are the basis for the success of a project. And at the same time the basis for problems.

The increasing relevance of industrial software and the aspiration for innovative and comprehensive systems necessitates effective requirements engineering. It ensures to reveal and prevent potential risks and to detect substantial gaps in requirements specifications. Nowadays, due to the increasing demands of a software product, companies must focus even more on requirements engineering and its core activities. "Requirements Engineering is a subtask of Software Engineering which deals with the discovering of that purpose by identifying stakeholders and their needs, and documenting them for their future analysis, communication, and subsequent implementation" (Saavedra et al., 2013, p. 240).

A general approach defines requirements engineering as "the part of development in which people attempt to discover what is desired" (Gause & Weinberg, 1989, p. XV).

Pohl and Rupp (2015) enhance this approach and describe requirements engineering as a systematic and disciplined approach for the specification and management of requirements. Also, requirements engineering focuses on understanding stakeholder needs, reducing relevant project risks, and ensuring reliable documentation. Ebert (2019) extends this definition by activities such as identification, documentation, analysis, testing, coordination, and management, taking particular account of customer orientation and economic and technical objectives. Requirements engineering is a construction kit to map the existing needs of different stakeholders to technical and economical solutions and is both problem- and solution-oriented.

In the literature and practice, the term is defined in many different ways (Balzert, 2010; Kotonya & Sommerville, 1998). However, all these definitions have in common that certain required activities are mentioned (Pohl & Rupp, 2015):

- **Elicitation:** Different techniques (e.g. interviews, brainstorming) are used to capture and detail information from stakeholders and other sources about potential requirements.
- **Specification and Documentation:** Identified requirements from elicitation are specified and documented. Several techniques to specify requirements, such as natural language and model-based approaches, are available and used in practice.
- **Validation and Agreement:** After the successful validation of specified requirements, the agreement phase with stakeholders is conducted.
- **Management:** The requirements management is an overall activity and includes tasks regarding the structure, the preparation, and the change process of requirements.

Since the focus of this work is on natural language requirements, we mainly concentrate on the activity "Specification and Documentation". For a better understanding of the overall context of requirements engineering, we give a short introduction to the elicitation of requirements.

Requirements are elicited from different sources. In most cases, stakeholders are the preferred source of information. The identification of relevant stakeholders is therefore a central task of requirements engineering in the beginning. However, there is no general approach for the collection of requirements (Hickey & Davis, 2003). The use of a particular technique (e.g. interviews, questionnaires, brainstorming, observation) correlates strongly with the requirement type, level of detail, and the experience of the involved persons (Pohl & Rupp, 2015). "Gathering correct and accurate requirements from customers requires a deep understanding of both the customer's business needs and the technical issues involved, which are often not communicated clearly or at all. These challenges often result in poorly written requirements that are unclear, verbose, and even inconsistent" (Verma & Kass, 2008, p. 751).

With the completion of the requirements elicitation, the activity "Specification and Documentation" is carried out[3]. Pohl and Rupp (2015) define the specification as a systematic collection of requirements that fulfills prescribed criteria. Several subsequent phases of a development project, such as implementation and testing, are based on the results from the specification phase. There are various techniques to specify requirements ranging from informal to formal languages (Fabbrini et al., 2000; Pohl & Rupp, 2015):

- **Natural language:** Requirements are specified without any restrictions. The statements are written like spoken language.
- **Structured natural language:** The terminology is limited and restricted and templates are often used for the specification.
- **Semi-formal language:** Special-purpose graphical notations are used with precise syntax and a non-rigorous semantic.
- **Formal languages:** Based on mathematics this language includes formally defined vocabulary, syntax, and semantics.

In industrial practice, natural language is often used for the specification of requirements (Berry et al., 2006; Fabbrini et al., 2001; Fantechi et al., 2003; Kamsties et al., 2001; Kassab et al., 2014; Kasser et al., 2006; Nikora et al., 2010; Pohl, 2008; Sikora et al., 2012; Wilson et al., 1997). A survey revealed that 79 percent of all requirements documents are written in natural language and only 21 percent use some form of formalism (e.g. requirement templates, formalized language) for the specification of requirements (Mich et al., 2004). Therefore, "when it comes to software requirements, natural language is the common means to communicate among stakeholders effectively" (Naeem et al., 2019, p. 1).

In general, natural language is inherently imprecise, incomplete and ambiguous (Fuchs & Schwitter, 1995; Ghosh et al., 2016; Macías & Pulman, 1995; Mich & Garigliano, 2000; H. Yang et al., 2012). Thus, the use of natural language could result in a subjective interpretation of the requirements content (Pohl, 2008).

[3]To improve readability and since it makes no difference in our research context, we use the terms "specification" and "documentation" synonymously.

It is also challenging to objectively validate that the requirements specified in natural language meet the needs of the customer (Ambriola & Gervasi, 1997; Unterkalmsteiner & Gorschek, 2017).

The more formality a language contains, the fewer the problems appear. Therefore, one way to reduce and avoid these problems is to increase the degree of formalization in the requirements. Besides removing ambiguity, this also allows the automatic detection of lexical, syntactic, and semantic errors. However, the use of formal languages can be very difficult and cost-intensive (Pohl & Rupp, 2015). Learning such a language takes a lot of time and lack of expertise makes it difficult to understand (Fantechi et al., 2003; IEEE 830, 1984).

Mich and Garigliano remind us, that "good linguistic competence is based on the adjustment of language to pragmatic purposes in order to minimize the effort of communication" (Mich & Garigliano, 2000, p. 3). Hence, requirements specified in natural language have decisive advantages despite the lack of formality. As natural language is based on the fundamental aspects of communication between people, specific training is not necessary to read the requirements[4] (Carew et al., 2005; Pohl, 2008). "Two advantages of this are: there is no limitation on the concepts that can be expressed; and sentences and grammatical structure provide means of tracing meaningful elements" (Parra et al., 2015, p. 181). Requirements specified in natural language can be used for different development areas, challenges, and domains and can represent various levels of abstraction and detail (Fabbrini et al., 2001; Ormandjieva et al., 2007). Despite numerous formal and semi-formal methods that are proposed by the research, natural language has therefore established itself as the most common technique for specifying requirements in industrial practice (Unterkalmsteiner & Gorschek, 2017).

At the same time, the specification of requirements is one of the most critical parts of a development project as requirements are the major source of errors for subsequent activities (Alshazly et al., 2014; Naeem et al., 2019; Sabriye & Zainon, 2018). "Where there is a failure, requirements problems are usually found at the heart of the matter" (R. Glass, 1998, p.21). Developers often interpret missing, incomprehensible, and inconsistent requirements in such a way that it would make sense from their point of view. However, the subjective interpretation of a requirement is a problem and the key to failure.

For that reason, a "key capability is the identification and management of requirements during the early phases of the system design process, where errors are cheapest and easiest to correct" (Arellano et al., 2015, p. 231). Poorly specified requirements

[4]Of course, this does not refer to the understanding of the content of requirements.

that are discovered late in the development process are the most expensive and most complicated to remove and handle. They are also cited as the main reason for project failure and cost overruns for more than 35 years (Abernethy et al., 2000; Belfo, 2012; Braude, 2000; Brooks & Kugler, 1987; Carson, 2001; A. Davis et al., 1993; Femmer, 2013; R. Glass & Becker, 2003; Gross & Doerr, 2012; Heck & Parviainen, 2008; Hooks, 1994; Ibanez & Rempp, 1996; Jacobs, 1999; Kasser & Schermerhorn, 1994; Lindquist, 2005; Parra et al., 2015; Polpinij, 2009; Saito et al., 2013; Salger et al., 2009; Scheffczyk et al., 2005; Wilson et al., 1997).

Boehm (1984) explored this phenomenon a while ago and linked the identification of requirement defects over the life cycle of software with error removal costs. Based on his analysis, defects that are already detected during the specification of requirements have a much lower cost factor than requirement defects that are detected in the production phase (factor of x10.000). A requirement defect identified in the coding phase requires ten times more effort to solve than a defect identified in the design phase. Thus, he claims that defects made during the requirements phase are the most expensive ones in the entire software development cycle if these are not discovered early (Ambler, 2004; McConnell, 1996). Figure 2.3 shows the costs for error correction according to the phase of the software life cycle in which the requirement defect is identified.

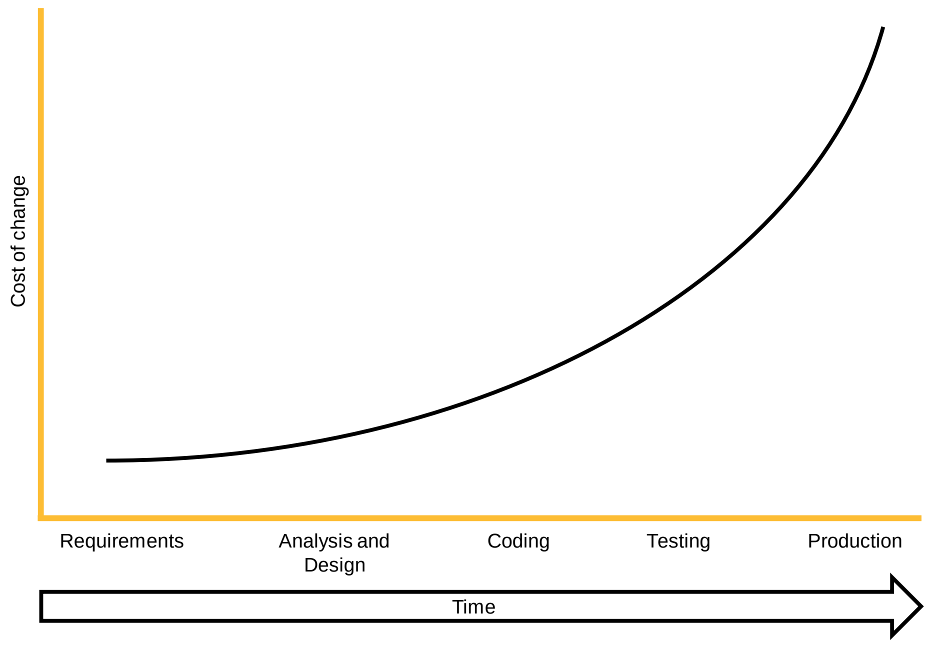


Fig. 2.3.: Own representation of the cost of change curve (inspired by Ambler, 2004).

Various studies prove that requirements defects are responsible for 50 to 70 percent of all software errors (Marasco, 2007; RTI, 2002; Suma & Gopalakrishnan Nair, 2009; Terzakis, 2013). In addition to purely financial aspects, such as cost overruns and extended project duration, these defects can also lead to worse consequences: "In an investigation of failed safety-critical systems, one study found nearly 1,100 deaths attributable to computer error. Many of these errors stemmed from poor or no specifications, not an incorrect implementation" (Hinchey & Coyle, 2012, p. 246).

In today's industrial practice, however, it is not uncommon for the requirement phase to be reduced significantly or even skipped completely. The effort is often not recognized, the result is not perceived as such, and engineers are allegedly deprived of too much of their creativity. Werner Gruhl revealed already in 1980 that this is a deceptive assumption. During his time as head of the analysis center of NASA, he was responsible for estimating project costs for new spacecraft and analyzed programs in comparison to cost overruns (Bowen & Hinchey, 2006). He observed that the amount of effort spent during the initial cost and planning phase of a space mission (Phase A and B) correlated with the cost overruns of a project. Figure 2.4 shows this correlation. In summary, projects, where requirements engineering accounts for less than five percent of the total cost of a project, have approximately 80 to 200 percent cost overruns. An increase of requirements engineering effort to eight to 14 percent of the total cost could result in projects with less than 60 percent cost overrun (Ebert, 2019). Today, between two and five percent of a project effort is spent on requirements engineering. Yet, errors resulting from requirements still have the greatest impact on a project and its successful completion (Mann et al., 2017 in Ebert, 2019).

A research study by the Standish Group has addressed these findings and tries to answer the question "Why software projects fail?". The results of the study, which are based on surveys and personal interviews of IT executives, are summarized in the "Chaos Report" (The Standish Group, 2006). The study identifies user involvement, management support, and clear requirements as essential elements for the success of a project. It also questions reasons for the cancellation of a project and detects insufficient involvement of the management and unclear requirements as main factors. According to the study, the number of failed software projects changed for the better between 1994 and 2006: whereas 30 percent of software projects were unsuccessful in 1994, the number decreased to 20 percent in 2006. Projects with increasing time and/or budget overruns and unsatisfied customers decreased from 53 percent to 46 percent (The Standish Group, 2006, cited in Pohl and Rupp, 2015).

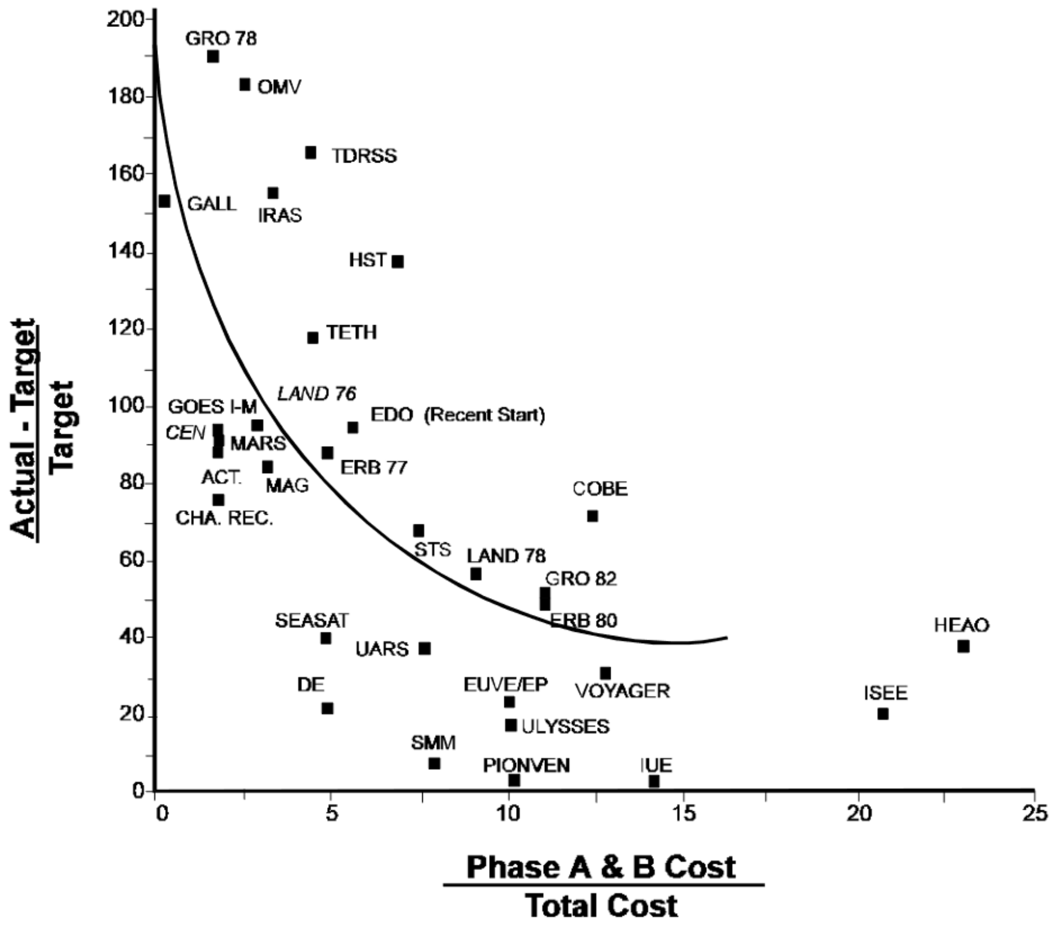


Fig. 2.4.: Percentage investment for requirements engineering (Phase A & B; x-axis) in relation to percentage cost overruns (y-axis) according to Hooks (2001).

The results from this report are repeatedly quoted when discussing the success and failure of software projects (Cerpa & Verner, 2009; A. Hussain et al., 2016). Even though there are some concerns, which are supported by scientific evidence (R. Glass, 2006; Jørgensen & Moløkken-Østvold, 2006), the effect on the publication was extensive: according to the report, the majority of software projects are not successful. In particular, the report identified that incomplete and unclear requirements have a decisive influence on the success of a project.

Summarized, the implementation of requirements engineering necessitates effort in different areas and is usually regarded as a time-consuming activity. Nevertheless, the advantages are obvious and it is considered as an essential prerequisite for the success of a project (Bucchiarone et al., 2005; Hu et al., 2010; Ormandjieva et al., 2007; Scheffczyk et al., 2005; Tjong et al., 2006). In particular, the specification of requirements is an important activity that must be considered with great attention in a software project. Challenges exist primarily due to the use of non-formal specification techniques like natural language (Chillarege et al., 1991; Juárez-Ramirez et al., 2011).

In general, companies still have difficulties and are usually not satisfied with the outcome of the requirements specification phase (Gross & Doerr, 2012). Therefore, "it is of major importance to provide this field with engineering discipline, particularly by means of quality controls since the very beginning of the process" (Génova et al., 2013, p. 25). Today, quality assurance processes often rely on manual reviews in which individual requirements are analyzed (Antinyan & Staron, 2017a; Salger, 2013). The success of a review depends heavily on the expertise and knowledge of the reviewer. Besides, a review can be very time-consuming and costly, especially if the reviewer is dealing with poorly specified requirements.

For that reason, quality controls should be reflected in methods and tools that take a particular account of the analysis of natural language requirements (Krogstie et al., 1995; Lehner, 1993; Macías & Pulman, 1995). In our research, we consider the assessment of requirements quality as a preliminary stage of a review to reduce the effort. This allows the reviewer to concentrate on the essential content of the requirements without being distracted by a poorly specified requirement (Femmer et al., 2017). In the following section, we address the main question of our research and answer how the quality of natural language requirements can be assessed.

2.2 Quality assessment of requirements

In this section, a set of characteristics for requirements quality is introduced. This set originates from the literature and is based on international standards. We present a requirements quality model, which is structured similarly to existing quality models from other domains. According to Heinrich et al. (2014), a quality model generally describes a system of quality characteristics and their relationships for specifying requirements and assessing the quality of an object.

In the further course, related work is presented that is dealing with the quality assessment of natural language requirements. We identify different approaches, which include rule-based and formalization methods as well as techniques from machine learning. Finally, a relevant research gap can be identified, which we aim to close with our research. We conclude the section with a discussion about existing research issues.

2.2.1 Requirements quality

Before we present our quality model, it is necessary to have a common understanding of the term "quality" in the context of requirements. In general, the quality of a product or a service always depends on the perception and expectations of the recipient. Traditional quality definitions use the term "fitness for use" and thus mainly consider attributes of functionality without concentrating on customer-relevant characteristics (Juran et al., 1974). In Haist and Fromm (1991), such characteristics are placed in the forefront when defining the term "quality". It is described as conformity with customer requirements, such as price, delivery time, safety, and reliability. Drucker (2014) also regards the concept of quality primarily from the customer's point of view: "Quality in a product or service is not what the supplier puts in. It is what the customer gets out and is willing to pay for" (Drucker, 2014, p. 280).

In the international standard ISO 9000, quality is defined as the "degree to which a set of inherent characteristics fulfills requirements" (ISO 9000, 2005, p. 18). The definition focuses on inherent characteristics that are required for quality measurement. If these characteristics are in "conformance to requirements" (Crosby, 1979), i.e. fulfilling the corresponding requirements[5], then high quality is achieved. If these characteristics do not fully meet all requirements, i.e. "there lies nonconformity or partial conformity between characteristics and requirements" (Sarkar, 2016, p. 33), then a low quality is present. The quality of requirements thus indicates the degree to which a requirement corresponds to measurable and defined quality characteristics (Rupp, 2014). Consequently, it is not possible to measure something as undefined as quality directly. "But even though Quality cannot be defined, you know what Quality is!" (Pirsig, 1999, p. 91).

If we take a look at other domains, we can identify the use of characteristics to describe quality. For SERVQUAL, a method for measuring service quality and customer satisfaction, five dimensions for the assessment are introduced (Parasuraman et al., 1988): reliability, assurance, tangibles, empathy, and responsiveness. The standard ISO 25010 (2010) defines eight criteria for determining software quality: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability. For requirements quality, we present a model that describes the quality by means of standardized and agreed characteristics. We identify the standard ISO 29148 as relevant source that defines a well-written

[5]In this context, the term "requirement" is not considered from the perspective of a software development project.

requirement based on the following characteristics: *Complete*, *Consistent*, *Feasible*, *Implementation Free*, *Necessary*, *Singular*, *Traceable*, *Unambiguous*, *Verifiable*.[6]

Our decision is based on several reasons. First, the ISO 29148 standard is the result of the harmonization of a large number of existing standards in the software field (including IEEE 830, 1998 and IEEE 1233, 1998). The standard provides an overview of different project phases as well as standard documents and structures. Second, characteristics are defined that describe the quality of an individual requirement. The broad acceptance of this standard confirms our decision to use these characteristics as the basis for our quality model. Related research with a similar objective comes to the same conclusion and refers to this standard (Femmer et al., 2017). Third, in Schneider and Berenbach (2013), the standard is described as "mother of all requirements standards", which contains basic principles of requirements engineering and criteria for requirements language (e.g. avoidance of vague pronouns). It is the standard "that every requirements engineer should be familiar with" (Schneider & Berenbach, 2013, p. 803). Therefore, we use the characteristics from ISO 29148 for the quality assessment of natural language requirements in our research. In the following, we present the description of each characteristic from ISO 29148 (2011).

Complete.

The stated requirement needs no further amplification because it is measurable and sufficiently describes the capability and characteristics to meet the stakeholder's need.

Consistent.

The requirement is free of conflicts with other requirements.

Feasible.

The requirement is technically achievable, does not require major technology advances, and fits within system constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk.

Implementation Free.

The requirement, while addressing what is necessary and sufficient in the system, avoids placing unnecessary constraints on the architectural design. The objective is to be implementation-independent. The requirement states what is required, not how the requirement should be met.

[6]To improve the readability, characteristics for requirements quality start with a capital letter and are in italics.

Necessary.

The requirement defines an essential capability, characteristic, constraint, and/or quality factor. If it is removed or deleted, a deficiency will exist, which cannot be fulfilled by other capabilities of the product or process. The requirement is currently applicable and has not been made obsolete by the passage of time. Requirements with planned expiration dates or applicability dates are clearly identified.

Singular.

The requirement statement includes only one requirement with no use of conjunctions.

Traceable.

The requirement is upwards traceable to specific documented stakeholder statement(s) of need, higher tier requirement, or other source (e.g., a trade or design study). The requirement is also downwards traceable to the specific requirements in the lower tier requirements specification or other system definition artefacts. That is, all parent-child relationships for the requirement are identified in tracing such that the requirement traces to its source and implementation.

Unambiguous.

The requirement is stated in such a way so that it can be interpreted in only one way. The requirement is stated simply and is easy to understand.

Verifiable.

The requirement has the means to prove that the system satisfies the specified requirement. Evidence may be collected that proves that the system can satisfy the specified requirement. Verifiability is enhanced when the requirement is measurable.

Based on the characteristics, we define the requirements quality model which we use in the further course of the thesis. Figure 2.5 shows the quality model and its characteristics from ISO 29148. However, the software engineering community struggled in the last decades to find appropriate characteristics. In a brief review, we show how the scope has changed over time.

One of the first references that provide characteristics for requirements can be found in ANSI/IEEE Std. 830 from 1984 (IEEE 830, 1984). The standard proposes seven characteristics to be considered when specifying requirements: *Complete*, *Consistent*, *Modifiable*, *Traceable*, *Unambiguous*, *Usable* and *Verifiable*.

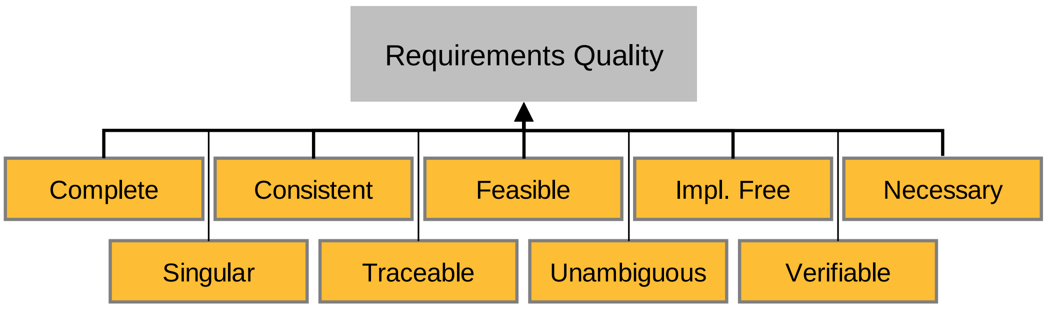


Fig. 2.5.: Our requirements quality model based on characteristics from ISO 29148.

In 1993, the standard ANSI/IEEE Std. 830-1984 was revised and supplemented with additional characteristics: *Correct* and *Ranked for importance and/or stability* (IEEE 830, 1993). The characteristic *Correct* refers to requirements that the software should fulfill, though there is "no tool or procedure that assures correctness" (IEEE 830, 1993, p. 5). Therefore, to verify this characteristic, the requirement and the related specification should be compared with higher-level specifications to ensure relevance and appropriateness. The characteristic *Ranked for importance and/or stability* demands that each requirement has a label of importance or stability. The previously introduced characteristic *Usable* has been removed.

One of the most comprehensive quality models is described by A. Davis et al. (1993). The authors propose 24 characteristics that result from a combination of different approaches[7]. Wilson et al. (1997) apply the characteristics of the standard IEEE 830 (1993) and add further characteristics: *Testable* and *Valid*. For the additional characteristic *Testable* it is necessary that quantitative or pass/fail evaluations can be derived from the specification. To fulfill the characteristic *Valid*, it is important that all project members (e.g. managers, engineers, and customers) have understood, accepted, and confirmed the requirements. "This is the primary reason that most specifications are expressed in natural language" (Wilson et al., 1997, p. 3).

Wiegers (2003) proposes a quality model for single requirements based on seven characteristics: *Complete*, *Correct*, *Feasible*, *Necessary*, *Prioritized*, *Unambiguous*, and *Verifiable*. Pohl (2008) provides a similar quality model and adds some characteristics: *Atomic*, *Complete*, *Comprehensible*, *Consistent*, *Correct*, *Rated*, *Traceable*, *Unambiguous*, *Up to date*, and *Verifiable* (extracted from Saavedra et al., 2013).

[7]*Achievable, Annotated by relative importance, Annotated by relative stability, Annotated by version, At right level of detail, Complete, Concise, Correct, Cross-referenced, Design independent, Electronically stored, Executable/interpretable, Externally consistent, Internally consistent, Modifiable, Not redundant, Organized, Precise, Reusable, Traceable, Traced, Unambiguous, Understandable, Verifiable.*

In Génova et al. (2013), a framework is presented with the aim to measure and improve the quality of natural language requirements. For this framework, characteristics from Wilson et al. (1997) are used and further are added. They are listed as "desirable properties of requirements": *Abstraction*, *Atomicity*, *Completeness*, *Consistency*, *Modifiability*, *Precision*, *Traceability*, *Unambiguity*, *Understandability*, *Validability*, and *Verifiability*. The characteristic *Abstraction* specifies that the requirement "tells what the application must do without telling how it must do it" (Génova et al., 2013, p. 28). The characteristic *Atomicity* demands that each requirement is clearly defined and that multiple requirements are not combined. The characteristic *Precision* describes that "all used terms are concrete and well defined" (Génova et al., 2013, p. 28). Finally, *Understandability* is defined as when "the requirements are correctly understood without difficulty" (Génova et al., 2013, p. 27).

In literature, there is a large number of further quality models that suggest various characteristics for the quality of a requirement. Based on exploratory analysis, Saavedra et al. (2013) have examined the multitude of existing quality models for requirements and identified similarities and differences. According to this study, the characteristics *Complete*, *Consistent*, *Unambiguous* and *Verifiable* appear most frequently in quality models. These characteristics are also reflected in the ISO 29148 standard, on which we have built the requirements quality model for our research.

Over the last decades, new characteristics have been established while others have been discarded or combined with existing characteristics. Even today people are still searching for the right terms. Some of the characteristics are persistent and appear in almost every approach that deals with the quality of requirements (e.g. *Complete*, *Consistent*, *Traceable*, *Unambiguous*, *Verifiable*). Other characteristics were introduced and shortly discarded afterwards (e.g. *Correct*, *Ranked for importance and/or stability*).

Thus, the quality aspect of requirements is a variable that adapts to changing needs, trends, and market situations over time. This may lead to the introduction of new characteristics in the future or the discarding of previous ones. Due to this continuous development of the characteristics, we are only able to refer to the standard (i.e. ISO 29148, 2011) that was valid at the time when we laid the foundations for our research.

2.2.2 Related work

A large part of industrial practice and research on the quality of requirements is dominated by wisdom and convictions rather than empirical evidence. This results in research on the one hand and practices in the industry on the other hand, which show only a low degree of congruence and lead to a constantly increasing separation between research and practice. For that reason, our contribution is based on personal experiences made during the interaction and collaboration while supporting industrial partners and enhanced by analyzing relevant research. Webster and Watson (2002) remind us, that such experiences help to justify and support a research topic.

We conduct a literature review as described by Webster and Watson (2002). The systematic approach ensures a comprehensive search to find relevant and current literature about the assessment of requirements quality. For our review, we primarily use "Google Scholar" as database. The identification of relevant approaches for the assessment of requirements quality is based on the selection of appropriate publications. We assess the relevance of the publications by the content, but we also take into account the number of citations. As there are many handbooks and seminal literature about requirements engineering, we start with analyzing this content. In a further step, we conduct searches for "requirements quality", "natural language requirements" and "textual requirements". With extensive forward and backward searches, we look for more detailed input for the identification of research methods and strategies that are used for the quality assessment of natural language requirements. In addition, we focus on publications including tools that also describe characteristics and indicators. The identification of key persons (e.g. Rupp, Ebert, Pohl) and organizations (e.g. INCOSE, IREB, Sophisten) in the domain of requirements engineering also ensures the consideration of practical literature to identify current approaches and contributions.

A systematic analysis of requirements quality must necessarily concentrate on the requirement text itself or on measurable indicators that can be derived from the text. Many efforts have been targeted at modeling, assessing, and measuring the quality of natural language requirements. Rather, there is a multitude of different approaches, all aiming in a similar direction: Determination of relevant quality characteristics and/or definition of indicators. This also enables the transfer of characteristics to the level of quantitative approaches. Furthermore, a strong emphasis has also been placed on natural language processing in recent decades to extract ontologies from requirements or to check the consistency of a specification (Ambriola & Gervasi, 1997; Rolland & Proix, 1992; Ryan, 1993).

In Femmer et al. (2017), the authors differentiate related research for the quality assessment of natural language requirements into manual and automated approaches. The former considers the application of manual techniques for the assessment of requirements quality by providing "analytical and constructive methods, as well as (varying) lists for defects" (Femmer et al., 2017, p. 191). The latter is assigned to the development of tools for the recognition and assessment of quality. In our research, we separate related work somewhat more distinctly from each other.

As a result of our review, the literature on requirements quality assessment largely falls into three research categories:

- **Rule-based systems.** The first category describes tools that are designed to support the requirements engineer in the specification of requirements. These tools identify weaknesses and give indications of how the requirements can be improved in quality. Examples are ARM - Automated Requirement Measurement (Wilson et al., 1997), QuARS - Quality Analyzer for Requirements Specifications (Fabbrini et al., 2001; Fantechi et al., 2003), and RQA - Requirements Quality Analyzer (Génova et al., 2013). Mostly, these tools follow a defined and static set of metrics for quality determination. Some of these systems are described as "spell or grammar checker" (Femmer et al., 2017; Génova et al., 2013; Verma & Kass, 2008) that refer to guidelines (Huertas & Juárez-Ramírez, 2013; Soeken et al., 2014; Tjong et al., 2006) or provide comprehensive checklists (Anda & Sjøberg, 2002; Berry et al., 2006; Kamsties et al., 2001; Kamsties & Peach, 2000; Van Lamsweerde, 2009).

- **Formalization and modeling systems.** The second category of research is devoted to the transformation of natural language requirements into formal models and logic specifications. This is typically done by a thorough linguistic analysis of the requirements language. Ontologies are used to transfer natural language requirements. Examples are Ilieva and Ormandjieva (2005), Holtmann et al. (2011) and Arellano et al. (2015).

- **Automated classification systems.** A third category is aiming at the automated classification of requirements into good or bad quality by using machine learning techniques and methods from natural language processing. Examples can be found in Ormandjieva et al. (2007), Soeken et al. (2014), H. Yang et al. (2012) and Parra et al. (2015).

A major part of the related work uses a quality model as starting point and applies methods and techniques which are typical for the corresponding research category.

Most of the papers in each category, but not all, know the concept of quality characteristics. Some papers address a complete set of quality characteristics (Gallego et al., 2016; Génova et al., 2013; Wilson et al., 1997), other papers focus on selected characteristics like *Ambiguity* (Berry, 2007; Ormandjieva et al., 2007; Shah & Jinwala, 2015), *Atomicity* (Huertas & Juárez-Ramírez, 2013), *Certainty* (H. Yang et al., 2012), *Correctness* (Parra et al., 2015), *Completeness* and *Consistency* (Arellano et al., 2015), and *Readability* (Holtmann et al., 2011).

Most of the latest papers can be assigned to the categories "Formalization and modeling systems" and "Automated classification systems". Especially the use of machine learning techniques to analyze and assess the quality of natural language requirements is a common topic. Mostly, these papers build upon the research of tools from the category "Rule-based systems" and enhance these approaches with methods from different disciplines. There is a trend towards developing algorithms and models that automatically assess the quality of natural language requirements by using characteristics and indicators to classify good or bad requirements. In the following, we briefly discuss some of the related work for each category, which we consider to be relevant for our research.

Rule-based systems.

Related work in the category "Rule-based systems" is intended to support the requirements engineer in the specification process. These approaches focus on the measurement of different characteristics by static indicators and the identification of defects in natural language requirements. Some of the publications use the term "spell checker", which not only provides evaluations but also points out concrete deficiencies to improve requirements quality.

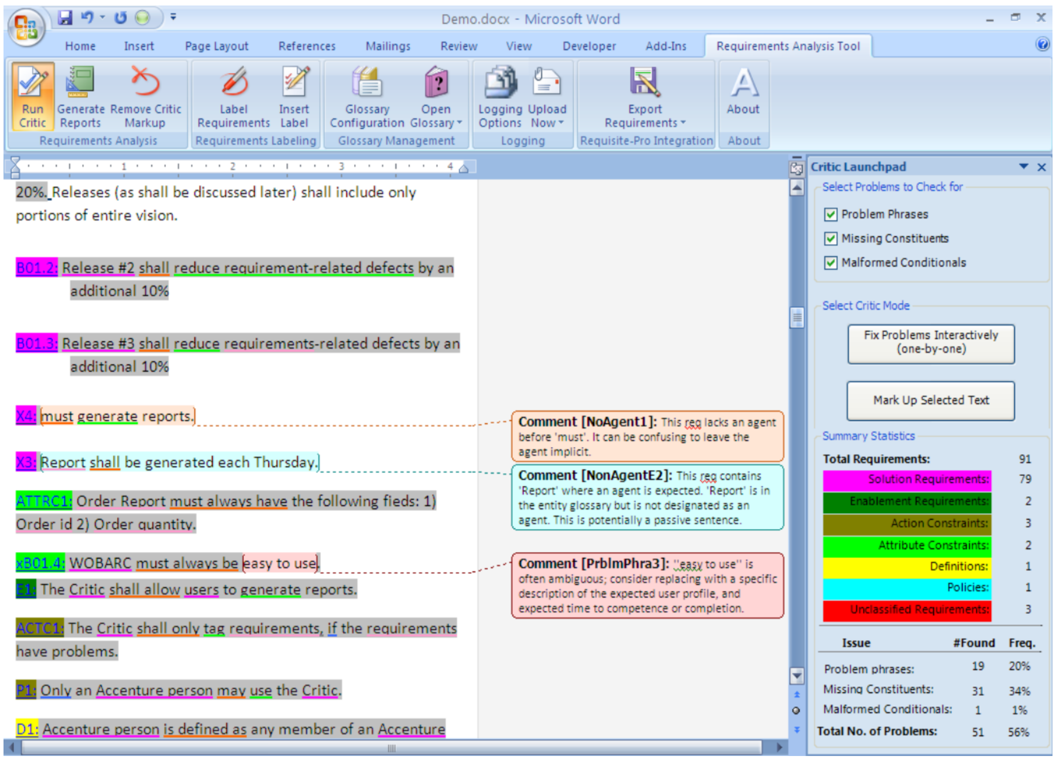
In Wilson et al. (1997), the authors describe an approach for the quality analysis of natural language software requirements and present a tool "Automated Requirements Measurement" (ARM) to support the requirement specification process. Quality is defined by "desirable characteristics" (shown in section 2.2.1) based on the standard IEEE 830 (1993)—one of the predecessors of the ISO 29148. The authors introduce different categories and indicators to assess the quality of requirements (imperatives, continuances, directives, options, weak phrases, size, and readability statistics). The tool counts potential defects based on the indicators. Simple dictionaries and lists of terms are used for this purpose. This research paper serves as groundwork and is mentioned in many other approaches as a starting point.

In Fabbrini et al. (2001), a tool called "QuARS" (Quality Analyzer of Requirements Specification) displays requirements together with automatically detected indicators. QuARS is a linguistic language tool that analyses the requirements for lexical, syntactic, structural, and semantic defects. The tool offers early detection and correction of costly errors by capturing the most common classes of defects. Four high-level quality properties are introduced: *Consistency*, *Non-ambiguity*, *Specification completion*, and *Understandability*. Nine text indicators are proposed, which are assigned to the properties and are largely based on specific keywords. Accordingly, words such as "clear", "easy" and "strong" are assigned to a vagueness indicator, which in turn partly affects the attribute *Non-ambiguity*. However, it is not possible to identify where this assignment comes from. Berry et al. (2006) extend the quality model and take into account further ambiguities that were not considered before. A total of 23 indicators are defined and additional keywords are presented. Both tools, ARM and QuARS, work in a similar way with a different abstraction and definition of the quality characteristics.

Kasser et al. (2006) propose five categories of requirement defects: *Multiple requirements in a requirement*, *Not verifiable*, *Possible multiple requirement*, *Use of wrong word*, and *User defined poor word*. A list of "poor words" gathered from experience and literature is presented. The presence of poor words in a requirement indicates a potential defect according to one of the five categories. The poor word "adequate" for example is linked to the category *Not verifiable*. The authors measure the quality by counting the number of defects, i.e. the occurrence of poor words, to the number of requirements examined.

Verma and Kass (2008) present the tool "RAT" (Requirements Analysis Tool) to detect and flag requirements that violate best practices. The tool builds upon "common requirements problems" like ambiguous terms, inconsistent and incomplete requirements. It highlights the recognized problem type, describes the problem in the requirement, and also offers suggestions for elimination. Figure 2.6 shows the tool that reminds of well-known spell and grammar checkers.

In Génova et al. (2013), the authors aim to answer the question "What should we measure?". They present desirable characteristics (see section 2.2.1) and indicators for assessing the quality of natural language requirements. A tool called "RQA" (Requirements Quality Analyzer) displays requirements together with automatically recognized indicators and an overall perceived quality assessment. The indicators are assigned to four categories: morphological, lexical, analytical, and relational. The analysis is carried out by statistical methods for morphological indicators, defined dictionaries or lists of terms for lexical indicators, approaches and tools from natural



20%. Releases (as shall be discussed later) shall include only portions of entire vision.

B01.2: Release #2 shall reduce requirement-related defects by an additional 10%

B01.3: Release #3 shall reduce requirements-related defects by an additional 10%

X4: must generate reports.

X3: Report shall be generated each Thursday.

ATTRC1: Order Report must always have the following fieds: 1) Order id 2) Order quantity.

xB01.4: WOBARC must always be easy to use.

E1: The Critic shall allow users to generate reports.

ACTC1: The Critic shall only tag requirements, if the requirements have problems.

P1: Only an Accenture person may use the Critic.

D1: Accenture person is defined as any member of an Accenture

Comment [NoAgent1]: This req lacks an agent before 'must'. It can be confusing to leave the agent implicit.

Comment [NonAgentE2]: This req contains 'Report' where an agent is expected. 'Report' is in the entity glossary but is not designated as an agent. This is potentially a passive sentence.

Comment [PrblmPhra3]: ''easy to use'' is often ambiguous; consider replacing with a specific description of the expected user profile, and expected time to competence or completion.

[✓] Problem Phrases
[✓] Missing Constituents
[✓] Malformed Conditionals

Total Requirements:	91
Solution Requirements:	79
Enablement Requirements:	2
Action Constraints:	3
Attribute Constraints:	2
Definitions:	1
Policies:	1
Unclassified Requirements:	3

Issue	#Found	Freq.
Problem phrases:	19	20%
Missing Constituents:	31	34%
Malformed Conditionals:	1	1%
Total No. of Problems:	51	56%

Fig. 2.6.: Requirements spell and grammar checker from Verma and Kass (2008).

language processing for analytical indicators, and the analysis of further properties of the requirement (e.g. *Number of versions of a requirement*) for relational indicators.

Huertas and Juárez-Ramírez (2013) examine functional requirements based on three quality attributes: *Atomic*, *Complete* and *Unambiguous*. Sentence and word tokenizer and associated tagging are applied to analyze the natural language requirements. For each attribute, indicators are proposed to identify, for example, each conjunction as a violation of the attribute *Atomic*.

Femmer et al. (2017) describe "Requirements Smells" as indicators "of a quality violation, which may lead to a defect, with a concrete location and a concrete detection mechanism" (Femmer et al., 2017, p. 194). Requirements smells are based on the concept of so-called code smells, which are intended to indicate poor code quality (Fowler et al., 1999). Nine requirement smells are defined, which are derived from the language criteria in the ISO 29148 standard. The authors present SMELLA, a tool for an automatic quality assessment based on requirements smells by using part-of-speech tagging, lemmatization, and dictionaries.

Formalization and modeling systems.

The second research category describes approaches that increase the degree of formalization of natural language requirements and transform them into logical models. The transformation aims to avoid problems that are typically encountered with natural language requirements (see section 2.1.3). We consider related work, that focuses on "checking for requirements quality criteria, and generating derived requirements specifications and models” (De Almeida Ferreira & Da Silva, 2012, p. 217).

Ilieva and Ormandjieva (2005) present an approach for the automatic transition of natural language software requirements into formal presentation and focus on "the automated extraction of semantics from [natural language]" (Ilieva & Ormandjieva, 2005, p. 1). The authors dissect each requirement sentence in its constituents subject, predicate, and object and arrange word groups in tabular form. An entity-relationship diagram is constructed from the tabular presentation using nouns as entities and verbs and prepositions as relationships. Characteristics are not used at all for the transition into the formal specification from a quality point of view.

Arellano et al. (2015) focus on consistent and complete natural language requirements by using "application-specific ontologies and natural language processing" (Arellano et al., 2015, p. 230). The authors analyze requirements by using natural language processing and extract objects and properties from the text. The properties are matched against a predefined ontology[8] (UML model). A property that is defined in the ontology should also appear in the requirement. A missing relation between entities could therefore indicate a completeness issue. Also, requirements that are redundant or have mutual conflicts could be identified. Since this approach requires a predefined ontology as input, it is considered as a semi-automatic solution (Kocerka et al., 2018).

A similar approach is presented by Bhatia et al. (2013). The authors use a part-of-speech tagger and a parser to perform lexical and syntactical analysis of natural language requirements (De Marneffe et al., 2006; Toutanova et al., 2003). The identification of requirement text, however, is limited to seven "grammatical knowledge patterns", which are defined by the authors. Based on these patterns, parameters of the requirement text are assigned (e.g. actor, action, object). If a defined parameter does not exist or is not recognizable, the requirement is evaluated as incomplete. In

[8]In general, an ontology is a "formal explicit specification of shared conceptualization" (Gruber, 1995, cited in Gawich et al., 2012, p. 1). The word "formal" in this context describes the possibility that the ontology is machine-readable, whereas the word "shared" aims at the acceptance of a group or community (Gawich et al., 2012).

cases where a requirement does not correspond to one of the predefined patterns, the assignment can also lead to incorrect results.

An "Automatic Requirements Specification Extraction from Natural Language" (ARSENAL) is implemented by Ghosh et al. (2016). The tool transforms natural language requirements into formal models and logic specifications for automated analysis. In the formal analysis, the authors focus on *Completeness* and *Correctness* of a requirement by performing various checks (consistency checks, satisfiability checks, and realizability checks). With the help of logical specifications in linear temporal logic, formalized requirements are used to check whether a suitable model is available. In the case of a match, the satisfiability check, for example, is successful.

In Kaiya and Saeki (2006), four quality characteristics are considered (*Completeness*, *Consistency*, *Correctness*, and *Unambiguity*) by applying semantic processing of natural language requirements and analysis through a domain ontology. The approach uses the ontology as a representation of domain knowledge and analyzes the relations between the mapped requirements into the ontology and the remaining requirements. In this paper, the use of an ontology allows semantic processing of requirements even without the use of natural language processing.

Automated classification systems.

Several contributions in the previous category build on text analysis and natural language processing. Techniques such as tagger, parser, and entity relations are used, which produce remarkable results. This shows us that these are generally applicable to natural language requirements. Some of the techniques can also be found in related work of our last category. The difference, however, is that algorithms from machine learning are additionally applied.

Ormandjieva et al. (2007) use 18 indicators that are fed into a decision tree to classify a natural language requirement into *Ambiguous* or *Non-ambiguous*. The indicators are selected by an analysis of different studies and are assumed to reflect ambiguity in requirements. Four experts assess a corpus of 168 requirement passages. A "gold standard" is created by the majority decision of all experts. The analysis of the pairwise inter-annotator agreement (kappa index from Cohen, 1960) reveals a "substantial" level of agreement between the experts and the gold standard.

A similar approach is provided by Parra et al. (2015), in which the authors use 25 textual indicators to classify requirements into "good or bad quality". Desirable properties of a good requirement are listed: *Complete*, *Consistent*, and *Correct*;

however, the first two properties are neglected in the further course of the paper. The tool developed by Génova et al. (2013) is used to extract metrics for each requirement. The method is composed of two tasks. In the first task, they let experts assess requirements and use a set of metrics to build a classifier. In the second task, they assess the quality of new requirements based on the same set of metrics.

The contributions from Ormandjieva et al. (2007) and Parra et al. (2015) describe an approach for the classification of requirements quality by using machine-learning techniques. Both compare the performance of the algorithm (i.e. a decision tree) against expert assessments (ambiguous/non-ambiguous, good/bad) on single requirements and present good results. This reinforces the idea to use machine learning techniques for our research scope since both papers also deal well with insufficient data or errors when predicting class values.

Soeken et al. (2014) present an approach for the "syntactic and semantic quality of a sentence" based on natural language processing techniques. For syntactic quality, the authors describe a phrase structure tree and a parser that uses probabilistic context-free grammars for tree computation. Isomorphic subtrees and "sentence length penalty" are considered, which take into account the number of words used for the sentences. For semantic quality, ambiguities are determined using WordNet. WordNet is a lexical database that contains semantic relationships of words and that offers synonyms, hyponyms, and meronyms. If a word in the WordNet dictionary has several meanings (i.e. synsets), it is considered to be ambiguous. In the first part, they let people assess a test set of 103 requirements. This results in a "matching percentage" between the algorithm and the human assessment of around 65 percent. In the second part, the authors present an approach to validate guidelines by proposing ten rules. For example, the first rule is described as "Define a requirement at a time" (Soeken et al., 2014, p. 3). These rules are then somehow transformed into features that are not explicitly described in the work. For rule three, which requires to "Use simple direct sentences", the authors only suggest checking for active or passive voice in the requirement sentences. Hence, if the requirement is written in the passive voice, this rule is violated. For the second part, accuracy values of over 80 percent are presented.

An approach for the detection of uncertainty in requirements is proposed by H. Yang et al. (2012). *Uncertainty* is characterized by an intended ambiguous specification compared to *Ambiguity*, which defines an unintended behavior. The authors present uncertainty cues (auxiliary verbs, conjunctions, epistemic verbs/adjectives/adverbs/-nouns) that are assumed to reflect uncertainty. With a supervised machine learning algorithm based on a conditional random field, they detect such clues at the term level and extract lexical and syntactic features from requirements. For the evaluation, a corpus is manually annotated based on the uncertainty cues and is used as a gold standard for developing and evaluating their system. The authors present promising results based on various performance measures.

Summary of related work.

Most of the related work focus on the characteristics *Complete*, *Consistent*, and *Unambiguous*. Many of the authors consider at least one of these. The characteristics *Feasible*, *Implementation Free*, and *Necessary* are not investigated at all. Other characteristics—like *Correctness*, *Modifiability*, *Testability*, *Validability* (Wilson et al., 1997), *Understandability* (Berry et al., 2006; Fabbrini et al., 2001), and *Abstraction* (Génova et al., 2013)—are used as well to determine the quality of requirements. The characteristics *Traceable* and *Verifiable* are focused mostly in research that deals with assisting tools in the category "Rule-based systems".

Some of the publications indicate that the quality attributes *Atomic*, *Complete*, and *Unambiguous* cause the "most problems" when specifying natural language requirements (Berry et al., 2003; Boyd et al., 2005; Gause & Weinberg, 1989; Goldin & Berry, 1997; Huertas & Juárez-Ramírez, 2013). Other researchers, however, claim that the quality of requirements is determined by the so-called three C's (*Completeness*, *Consistency*, and *Correctness*) and offer approaches to improve the characteristics (Kamalrudin et al., 2011; Parra et al., 2018).

In our literature review, we identify a strong focus on the characteristic *Unambiguous* (Berry, 2007; Denger et al., 2003; I. Hussain et al., 2007; Kamsties et al., 2001; Kiyavitskaya et al., 2008; Mich & Garigliano, 2000; Ormandjieva et al., 2007; Sabriye & Zainon, 2018; Shah & Jinwala, 2015; H. Yang et al., 2010). An extensive study by Pekar et al. (2014), which addresses the question of "what SRS problems are most frequently researched?", confirms this. The authors reveal that about one-third of all related research focuses on the study of *Ambiguity*[9]. The characteristics *Completeness* and *Consistent* are considered as the second and third most important challenges of natural language requirements with about 25 percent of publications. The characteristic *Correctness* is considered in every tenth research paper. The results of the study reveal that "ambiguity and the three C's are the most frequently researched issues" (Pekar et al., 2014, p. 244).

[9]There is a widely used test for an unambiguous requirement: "Imagine a sentence that is extracted from an SRS, given to ten people who are asked for an interpretation. If there is more than one interpretation, then that sentence is probably ambiguous" (A. Davis, 1993, cited in Berry, 2007.)

In our research, we do not exclude characteristics from the ISO 29148 standard for the quality assessment of natural language requirements, although the assessment of some characteristics for a single requirement does not seem to be conducive at first glance. For some of the characteristics, additional and context-related information would be helpful. A further discussion of this issue can be found in section 2.2.4. In addition, we analyze in chapter 3, whether experts are even able to assess all quality characteristics of a single natural language requirement. At first, in the following section, we present the research gap that results from the summary of related work.

2.2.3 Research gap

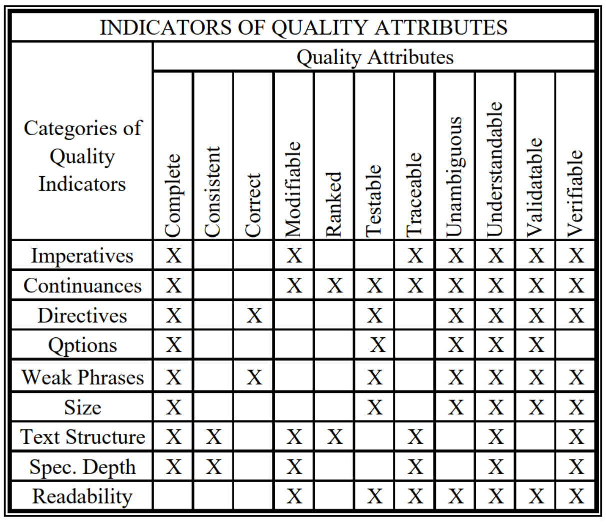
To the best of our knowledge, we do not find approaches that take into consideration all characteristics from the ISO 29148 for a quality assessment of natural language requirements. In addition, there are a number of proprietary quality models, some of which are either derived from the experience of individuals or are constructed in such a way that they can be directly quantified.

Even though characteristics have been further developed and refined over the last decades, quantification for the quality assessment of a requirement is a complex task. Several approaches describe textual indicators that are believed to relate to the quality of requirements. These indicators can be lexical, syntactic, or semantic (Antinyan & Staron, 2017b; Fantechi et al., 2003; Génova et al., 2013; Wilson et al., 1997).

Some papers even suppose a direct relationship between the indicators and the overall requirements quality (Femmer et al., 2017). In no case, these relationships are proven by empirical research. Mostly, "quantitative indicators [...] are more or less directly related to qualitative properties, so that the indicators can be used as an objective measure of subjective quality" (Génova et al., 2013, p. 27). An example of this is illustrated in figure 2.7.

Besides, we particularly identify a lack of empirical evidence under real-world conditions. Only some of the researchers validate their research on industrial data and those who had access to real requirements only present an overview of their results.

To address this research gap, we describe an approach to first analyze whether quality characteristics of requirements can be assessed by experts. Based on these findings, we implement an automated quality assessment of natural language requirements.



INDICATORS OF QUALITY ATTRIBUTES											
Categories of Quality Indicators	Quality Attributes										
	Complete	Consistent	Correct	Modifiable	Ranked	Testable	Traceable	Unambiguous	Understandable	Validatable	Verifiable
Imperatives	X			X			X	X	X	X	X
Continuances	X			X	X	X	X	X	X	X	X
Directives	X		X			X		X	X	X	X
Qptions	X					X		X	X	X	
Weak Phrases	X		X			X		X	X	X	X
Size	X					X		X	X	X	X
Text Structure	X	X		X	X		X		X		X
Spec. Depth	X	X		X			X		X		X
Readability				X		X	X	X	X	X	X

Fig. 2.7.: Relationship between quality attributes (i.e. characteristics) and quality indicators according to Wilson et al. (1997).

From a technical perspective, our approach is not fundamentally different from existing contributions. We use techniques from natural language processing, including dictionaries and part-of-speech tagging, to process the requirements. But, we take the knowledge of requirements quality from domain experts into account, as "one of the most decisive factors for the quality of requirement" (Parra et al., 2015, p. 191). This enables an implementation that is aligned with the needs of industrial practice. For our research we build upon the approach from Parra et al. (2015) and use results from several studies (e.g. Génova et al., 2013). We create a list of textual indicators extracted from related work and enhanced by guidelines and best practices from industrial practice for the automated quality assessment of natural language requirements.

In summary, we complement the current state of the discussed evidence on automatic quality measurement for natural language requirements by conducting a systematic assessment under realistic conditions with practitioners. We define a requirements quality model based on the characteristics of the ISO 29148 standard, let experts assess real-world requirements, and implement an automated assessment of requirements quality by applying natural language processing and machine learning. Thus, our approach enables a quality assessment of unlabeled requirements in the same way an expert would classify them.

However, our research is also accompanied by issues related to the expert assessment of requirements and our automated approach. We take these challenges into account in the following section.

2.2.4 Research issues

Based on the analysis of related work and the examination of the characteristics from ISO 29148, we identify research issues that we discuss in the following. The issues are of different nature and range from challenges with the use of natural language to possible interdependencies of individual characteristics in our requirements quality model. Also, general assumptions for the assessment of requirements quality are discussed.

Vagueness. The descriptions of the characteristics from ISO 29148 (and previous approaches) can be interpreted as vague. Our requirements quality model therefore already contains a certain fuzziness and allows a possible subjective interpretation. Also, the assessment of requirements based on these characteristics could lead to such fuzziness. This can be described by two influencing factors. First, experts can not understand the characteristics and are therefore not able to reliably assess requirements quality. Second, experts interpret these characteristics identically or at least similarly. Due to the different professional experiences of the experts, however, requirements are assessed differently. We consider this issue in our research as far as possible by analyzing the reliability of the experts' assessments and by ensuring a reasonable distribution of experts with regard to their professional experience.

Requirements context. Besides the vagueness of the descriptions, we recognize another issue related to the context of the requirements assessment through the experts. According to the ISO 29148 standard, the characteristics of our requirements quality model can be applied to an individual requirement. However, with the given descriptions, a distinction between characteristics that apply to individual requirements and characteristics that probably require further context-relevant requirements for the assessment is necessary. Characteristics of the first group can be applied to individual requirements where no further information is needed. We estimate *Feasible*, *Implementation Free*, *Singular*, *Unambiguous*, and *Verifiable* in this group. The characteristics *Necessary* and *Traceable* are estimated to relate to the second group. When assessing these characteristics, additional relevant requirements would be beneficial. For *Traceable*, information about linked requirements would be most helpful. The same applies to *Necessary* where the necessity of a requirement can be assessed when the whole requirements specification and all additional information and relevant documents are available. In the standard ISO 29148, the characteristics *Complete* and *Consistent* have a special role and relate to both groups. *Complete*, for example, can be applied to individual requirements and considers, whether the requirement "needs no further amplification" (ISO 29148, 2011, p. 11). Also, *Complete* can be assessed for a set of requirements where "it contains everything pertinent to the definition of the system or system element being specified" (ISO 29148, 2011, p. 11). Same applies for the characteristic *Consistent*. In our research, we estimate both characteristics for the first group since we use the description of the characteristics for an individual requirement. We are aware of these two groups and realize that additional contextual requirements would be helpful for the assessment of some of the quality characteristics of an individual requirement. Thus, we analyze in chapter 3 whether the experts are able to assess these characteristics at all.

Surface and concept understanding. The characteristic *Unambiguous* is described as the "Achilles‘ heel" (Berry et al., 2003) of software requirements specifications. Ormandjieva et al. (2007) also claim that *Unambiguous* is mainly responsible to enable the determination of the remaining quality characteristics. Hence, if a requirement is not *Unambiguous*, there is a reason for misunderstanding and misinterpretation. But, moreover, if a requirement is not *Unambiguous*, it can hardly be assessed to which degree it fulfills the other characteristics. This is an assumption that the authors describe with "surface understanding" vs. "concept understanding" (Ormandjieva et al., 2007). The surface level is concerned with the language of the requirement, or "what is stated", and the concept level with "what is meant or implied" (Ormandjieva et al., 2007). It is obvious that an assessment of the latter requires a much deeper insight into the structure of the requirement—both for a human and for a machine algorithm. The surface level has been often addressed by research. In particular, the tools from the category "Rule-based systems" to support the requirements engineer in writing good requirements work largely on this level. The hypothesis might be that an author who puts much effort into writing a requirement unambiguously, has put the same effort into assuring that the requirement is complete, consistent, and verifiable and fulfills the remaining quality characteristics as well. This would also imply, that the assessment of the characteristics can only be done if the requirement is *Unambiguous*. Therefore, in the context of expert assessments, we investigate to what extent the characteristics can be assessed in general, but we refrain from the analysis of mutual dependencies of characteristics.

Mutual influences. Besides the distinction between surface understanding and concept understanding, some researchers claim that there are further mutual dependencies between the quality characteristics (Génova et al., 2013). Saavedra et al. (2013) synthesize several lists of desirable properties and present assumptions about how the characteristics influence each other. The authors determine which of the characteristics "are most important to the project" (Saavedra et al., 2013, p. 254). However, empirical evidence for these results is not given. Some studies even extend this approach by asking experts to prioritize the characteristics (Unterkalmsteiner & Gorschek, 2017). We consider these results to be difficult to prove, as the priorities of the characteristics can particularly vary depending on different stakeholders. A software tester would presumably prioritize the characteristics *Unambiguous* and *Verifiable* in a different way than a stakeholder who does not have to cope with the verification of natural language requirements. In chapter 3, we apply an approach of prioritizing characteristics in our assessment sessions with the experts, although we do not use the results for our research. With the prioritization of characteristics at the beginning of an assessment session, we only want to establish a common understanding among experts for the description of the ISO 29148 characteristics. The prioritization task makes it necessary for the experts to deal with the descriptions of the characteristics before the assessment of requirements.

Practical contribution. Our research topic is located in an area where informal approaches are used to describe software requirements. Berry (2007) reminds us, that "in order to write software, ideas, which are inherently informal, have to be converted somehow to code, which is inherently formal" (Berry, 2007, p. 1). In general, it is not possible to fully describe and exactly specify ideas solely by text. However, we want to exactly assess requirements quality by using characteristics and textual indicators. Therefore, we must expect uncertainties in the assessment and strive for good results despite this inaccuracy. In addition, our research is based on several hypotheses: first, quality can be made objective and measurable; second, indicators can represent the manual quality assessment of experts; third, indicators are sufficient to interpret natural language. In later parts of this thesis, we provide empirical results to confirm these hypotheses. Besides, the analysis of natural language requirements can generally be an "aid for writing the requirements right, but not for writing the right requirements" (Génova et al., 2013, p. 26). An assessment can be an approach to improve the correct understanding of requirements. However, an overall challenge in the assessment of natural language requirements is to define indicators for content quality. A quantitative measurement does not necessarily give insights about this as the text is only the carrier of the content. Nevertheless, we are convinced that well-specified requirements improve the communication between stakeholders in a development project and lead to a higher quality of software products.

We conclude section 2.2 with the discussion of the research issues. We are aware of these issues and take them into account as much as possible in the manual and automated assessment of requirements quality. In a final section, we summarize chapter 2 of this thesis.

2.3 Conclusion

Research about requirements engineering is an old topic and a lot of literature is available. It is also a well-known concept that is described in numerous textbooks and articles. However, it took more than 40 years to establish it as an independent discipline in an edition of the "Transactions of Software Engineering". Although nowadays people are fully aware of the relevance and importance of requirements engineering for the success of a software project, the actual percentage of projects in which sufficient requirements engineering is included is still low (Ebert, 2019).

With an increase in the number of software projects and additional challenges (demands, complexity, distributed development), the focus on the quality of software requirements also became increasingly important. As "there is little excuse for specification errors" (A. Davis et al., 1993, p. 142), initial improvement approaches concentrated on supporting developers in the formulation and specification of good software requirements. Therefore, the question quickly arose as to what describes a "good" requirement.

Today, international standards for requirements engineering contain sections on the quality of requirements, although there is still no uniform definition. In general, quality can be described as the sum of individual characteristics, as can be seen from approaches in the context of software quality and service quality. Consequently, we define a requirements quality model based on nine characteristics from the standard ISO 29148 that describes a well-written requirement. In our research, we refer to this standard as it was valid at the time of the development of our quality model.

We further analyze the literature on requirements quality, which falls largely into three categories. "Rule-based systems" include research about static tools to support requirements engineers in writing good requirements; "Formalization and modeling systems" describe research on the transformation of natural language requirements into formal models and logic specifications; "Automated classification systems" focus on automated approaches to classify requirements quality by applying natural language processing in combination with machine learning techniques. Several researchers describe textual indicators that relate to the quality characteristics or even to the overall quality of a requirement. The relationship between textual indicators and characteristics has been suggested but not evidenced so far.

Consequently, two research gaps need to be investigated: first, we do not find assessment approaches that consider all quality characteristics from ISO 29148; second, no empirical research has been conducted on the relationship between indicators and characteristics. To address these gaps, we first analyze which of the characteristics can be assessed by experts at all. Based on these findings, we implement the automated assessment of natural language requirements and provide evidence on the relationship between indicators and characteristics. The identification of indicators from various studies dealing with a similar research problem enables us to perform linguistic analyses of natural language requirements. We create classification models to assess natural language requirements in terms of quality in the way an expert would do. Also, we do not only provide binary results for the "good" or "bad" quality of a requirement but allow a detailed analysis of (some of) the characteristics.

In chapter 3, we present our approach and results from the manual assessment of requirements quality with experts. We use these results for the automated assessment that is introduced in chapter 4.

3 Manual assessment of requirements quality

But how do we know what's good?
You just see it.

— **Robert Pirsig**
In: Zen and the Art of Motorcycle Maintenance

In this thesis, we apply a requirements quality model that we introduced in chapter 2. The model consists of nine quality characteristics described in the ISO 29148 standard. Our main target is to enable an automated assessment of these characteristics for natural language requirements by using supervised machine learning. An essential prerequisite for training and evaluating such machine learning models is labeled data, also called "gold standard" (Femmer et al., 2017; I. Hussain et al., 2007; Krisch et al., 2016; Rosenberg, 2008; Velupillai et al., 2009). For our research, we can obtain labeled data only by human judgment. Therefore, we ask experts to assess natural language requirements based on the characteristics of our quality model. The results of the assessments are then used as input for the implementation of an automated approach in chapter 4.

This chapter describes the expert assessments of requirements quality and the creation of a labeled data set. It consists of three sections: data handling, requirements assessment, and agreement analysis. In the first section, we examine the data set with more than 57,000 items (called "objects" in the following) from automotive development projects and create a subset for the assessment sessions. In the second section, an approach for the quality assessment of natural language requirements that allows simple and effective collection of labeled data is presented. The last section focuses in particular on the definition and determination of agreement regarding the experts' assessments. Finally, a summary and conclusion of the chapter are provided.

3.1 Data handling

In the first section, we describe how requirements are gathered for the assessment sessions with experts and examine the industrial data set consisting of natural language requirements. Additionally, we present the data selection process with which the data set for the requirement assessments is generated.

3.1.1 Data acquisition

In our research, we work together with an international consulting and engineering company. The cooperation is complemented by the execution of customer projects in the area of requirements engineering and provides us insights into the development departments of different automotive manufacturers. We also have access to software requirements specifications from industrial practice, which we can use for our research purpose. This allows us to gain real-world insights and also to derive reliable results for practical application.

We gather requirements of different automotive software development projects through exports from proprietary requirements management tools (mostly IBM Engineering Requirements Management DOORS[1]). These projects aim at function development, such as lane-keeping assist, collision avoidance, and the development of other safety and comfort functions[2]. A majority of the projects are implemented jointly with several external development partners. This makes the use of natural language in the software requirements specifications (SRS) the most reasonable option.

Typically, an SRS is written by different engineers, most of whom are domain experts in their field, and the content is formulated in natural language. An SRS generally describes (parts of) a software function and consists of a number of so-called "objects" in DOORS. An object can be thought of as an entry in the specification that is described by several attributes. In our data export, the objects in each software requirements specification contain at least the following attributes: "Module Name", "Object ID", "Object Type", and "Object Text". Each object results in one row in a spreadsheet with the attributes as columns.

The attribute "Module Name" contains the name of the software requirements specification to which an object belongs to. Our data set includes a total of 83

[1]DOORS = Dynamic Object Oriented Requirements System
[2]For reasons of confidentiality, we are not allowed to give detailed information about the projects.

different specifications that vary in size between 100 and 2,600 objects. Each object is also assigned to an "Object ID" that uniquely identifies an object. In total, our data set contains 57,801 objects.

The attribute "Object Type" further describes the type of an object. In a software requirement specification, a distinction is typically made between the types "Heading", "Information" and "Requirement". The types "Heading" and "Information" are used for structuring or providing additional information. A "Heading" delineates or summarizes the thematic areas of an SRS. Objects of the type "Information" supplement the context of a development project or parts of it without including requirement-relevant contents. The type "Requirement" therefore indicates requirement-relevant content. Our data set includes 9,365 "Heading" objects, 10,775 objects attributed as "Information" and 37,661 objects attributed to the type "Requirement".

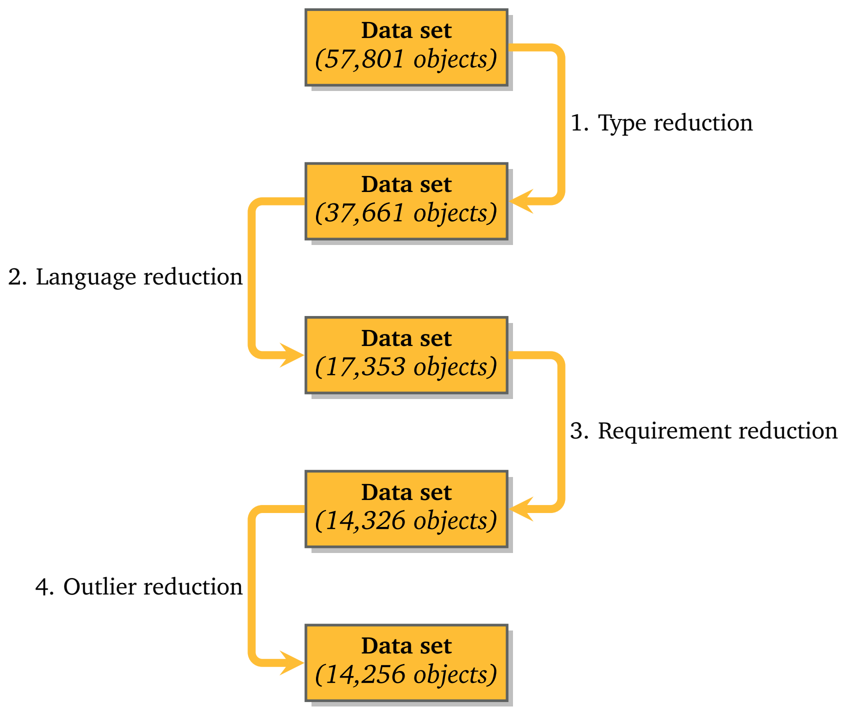
Finally, the attribute "Object Text" includes textual content for each object. Table 3.1 exemplarily shows the structure and the attributes of the exported requirements. The "Object Text" may be of different lengths and can also consist of several sentences. This is because the attribute typically allows text to be inserted without proper formal checks. Based on this data set we proceed with the selection process in the next section.

Tab. 3.1.: Objects in an SRS with a defined set of attributes.

Module Name	**Object ID**	**Object Type**	**Object Text**
Spec_01	ID_01	Heading	*Header text (e.g. "Functional requirements")*
Spec_01	ID_02	Information	*Here you will find textual information.*
Spec_01	ID_03	Requirement	*Here you will find requirement-related content in textual form.*
Spec_01	ID_04	Requirement	*Another text for a requirement. This requirement object consists of two sentences.*
Spec_01	ID_05	Information	*This object contains more textual information.*
Spec_02	ID_01	Heading	*Header text (e.g. "Performance requirements")*
Spec_02	ID_02	Information	*Here you will find more textual information. An information object can also consist of several sentences.*
Spec_02	ID_03	Requirement	*More textual content for a requirement.*
Spec_03	ID_01	Heading	*Header text (e.g. "Quality requirements")*
Spec_03	ID_02	Information	*Another text for an information object.*

3.1.2 Data selection

The export contains a lot of data, not all of which is relevant for our research. Therefore, we perform a four-step process for the selection of the data set. The process is illustrated in figure 3.1 and the steps are described in the following.



Data set (57,801 objects)
1. Type reduction
Data set (37,661 objects)
2. Language reduction
Data set (17,353 objects)
3. Requirement reduction
Data set (14,326 objects)
4. Outlier reduction
Data set (14,256 objects)

Fig. 3.1.: Selection process for the data set.

1. **Type reduction.** Our research focuses on the assessment of natural language requirements. Thus, in the first step, we start selecting objects which are attributed as such (Object Type = "Requirement"). This step reduces our data set from 57,801 to 37,661 objects[3].

2. **Language reduction.** We add the attribute "Module Language" to our data set and manually assign the language in which each specification is written. This attribute helps us to differentiate between English and German requirements. As our experts have German as their native language, we reduce the data set by English content[4]. This results in a reduction to 17,353 requirement objects.

[3]Each remaining object is assigned the type "Requirement". Therefore, we further speak of requirement objects, although a wrong assignment of the type could still be present.
[4]With the demonstration of the feasibility for German requirements, the approach can (and should) be adapted to English requirements as well.

3. **Requirement reduction.** We do not consider requirement objects in the data set that contain tables or that are part of enumerations, as the grammatical structure of a sentence is usually not given (Krisch & Houdek, 2015). We also identify and remove duplicates that can occur due to requirements included by default in any specification such as those prescribed by company policies. This step reduces the data set to 14,326 requirement objects.

4. **Outlier reduction.** The remaining objects are analyzed with regard to the number of words in the attribute "Object Text". We apply different ranges for the number of words to identify outliers in the data set. The analysis reveals that requirement objects with more than 84 words have no relevant content for our research. Some of these contain, for example, only a sequence of signal parameters or describe pseudo code. Figure 3.2 illustrates the frequency of the number of words per requirement object. Table 3.2 shows in detail that 99.50 percent of the requirement objects contain between five and 84 words. Therefore, in the final step, we reduce the requirement objects in the data set that contain more than 84 words. This applies to 70 objects.

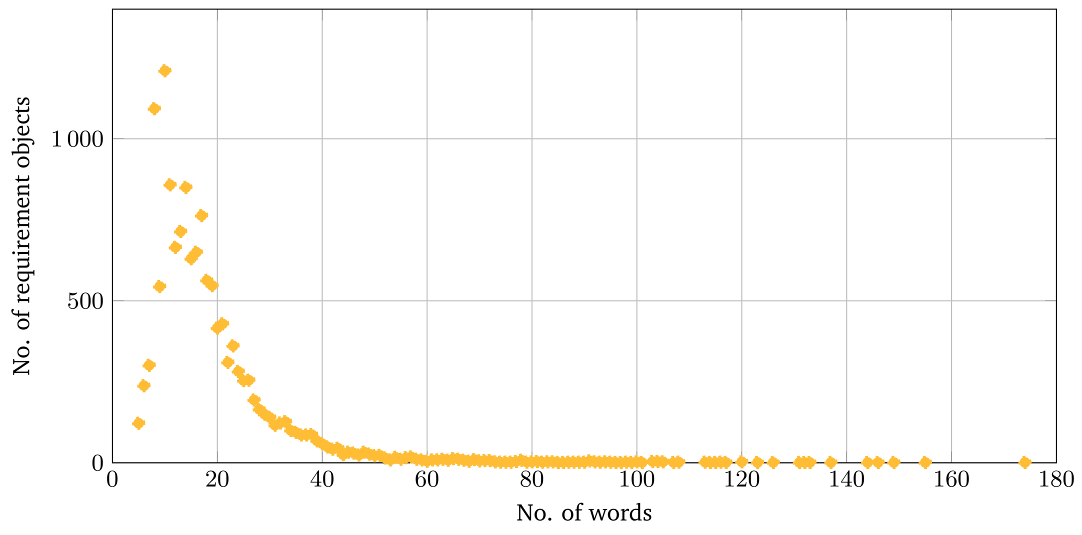


Fig. 3.2.: Analysis of the number of words in a requirement to identify outliers in the data set.

As a result, the final data set consists of 14,256 unique German requirement objects in natural language. The selection process can be seen as opportunistic. However, we assured that the most possible variety of relevant data is available for the assessment of requirements. In the next section, we use the selected data set as input for the assessment sessions with experts.

Tab. 3.2.: Detailed analysis of the distribution of the number of words.

No. of words	No. of objects	Percentage
5 – 24	11,546	80,59%
25 – 44	2,265	15,81%
45 – 64	340	2,37%
65 – 84	105	0,73%
85 – 104	43	0,31%
105 – 124	17	0,12%
125 – 144	6	0,04%
145 – 164	3	0,02%
165 – 184	1	0,01%

3.2 Requirements assessment

In this section, we focus on the labeling process based on expert assessments. Since there is no established methodology for assessing natural language requirements, we use methods from survey research for the design of our assessment tool. The tool facilitates the requirements assessment for the experts and collects relevant data for our "gold standard". We also provide an overview of the results before we finally discuss assessment issues.

In general, the quality assessment of requirements is a sophisticated and challenging task. If there are uncertainties in the assessment, this may be due to the different competencies of the experts, for example. The expert's background and level of experience could lead to different forms of implicit knowledge and the ability to assess requirements despite less information value. However, it is also crucial whether the characteristics of a requirement can be assessed based on the descriptions from ISO 29148. Therefore, a central question is whether these descriptions are appropriate to allow an assessment of the requirements. Besides, it is necessary to consider whether a single natural language requirement can provide enough information to enable the assessment of all these characteristics. These are further questions we discuss in this section.

3.2.1 Assessment tool design

Labeling is a complex process and often represents the major part of a machine learning project. Therefore, it should be done as competent and efficient as possible. We achieve competent execution by asking experts who already have professional experience in requirements engineering. To ensure that the labeling is also carried out efficiently, an assessment tool is required. This raises questions regarding the design of such a tool and the selection of a suitable scale for the assessment of the characteristics. These are all questions for which we can receive suggestions from the disciplines of survey research.

In general, survey research is a preferred approach for researchers (Palvia et al., 2004). "It is a quantitative research method that is commonly used in both empirical software engineering and information systems research" (Wohlin & Aurum, 2015, p. 1441). We take into account the use of a predefined and structured set of questions for data collection and aim to gather data that we can analyze quantitatively afterward (Palvia et al., 2004; Saunders et al., 2012). Besides, for the assessment design, we consider the guidelines and critical factors of Kitchenham and Pfleeger (2002), Marsden and Wright (2010), and Saunders et al. (2012). They describe, for example, that the experts' answer format should be in a standardized form.

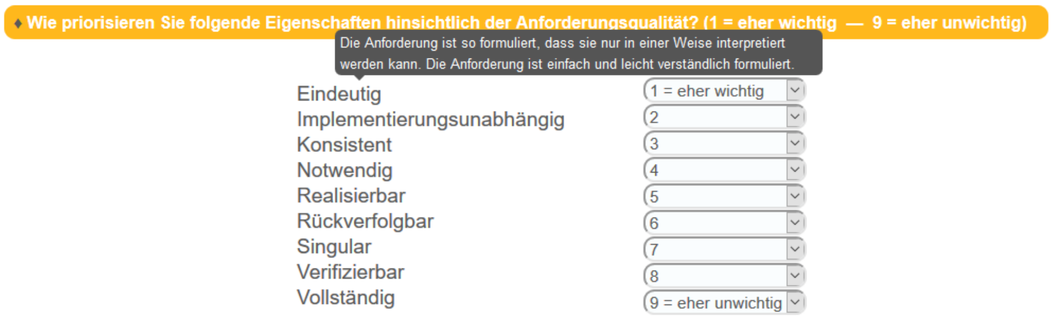
In most cases, a manual assessment can be time-consuming, which is why not all requirements[5] from our data set can be assessed by experts (Ormandjieva et al., 2007). For that reason, we select 1,000 requirements by random choice. For a simple and reliable requirements assessment, we developed a tool that consists of a website linked to a database. The tool enables a simultaneous requirements assessment of several experts by saving each assessment in the database with the help of SQL queries. The assessments are carried out in so-called "sessions". A session includes 25 natural language requirements, each of which is assessed according to the nine characteristics of our quality model.

"At the start [...] you need to explain clearly and concisely why you want the respondent to complete the survey" (Saunders et al., 2012, p. 446). Therefore, at the beginning of each session, we present general information and the motivation for the quality assessment of natural language requirements (see appendix A.1). This is followed by general questions to the expert, such as the gender and the number of years of professional experience the expert already has in requirements engineering.

Inspired by the approach presented in Génova et al. (2013), Saavedra et al. (2013) and Unterkalmsteiner and Gorschek (2017) to classify quality characteristics according to their relevance, we set up a prioritization task before the start of the assessment session. We ask the experts to prioritize the nine characteristics of our quality model—*Complete*, *Consistent*, *Feasible*, *Implementation Free*, *Necessary*, *Singular*, *Traceable*, *Unambiguous*, and *Verifiable*—regarding to the overall quality of a requirement. The experts can view the description of the characteristics which we translated into German and can choose between 1 (rather important) and 9

[5]To further increase readability, we generally refer to a requirement object as "requirement".

(rather unimportant). A value can also be used more than once, for example, if two characteristics have the same prioritization according to the expert's opinion. In figure 3.3, we show the approach for the prioritization of characteristics.



♦ Wie priorisieren Sie folgende Eigenschaften hinsichtlich der Anforderungsqualität? (1 = eher wichtig — 9 = eher unwichtig)

Die Anforderung ist so formuliert, dass sie nur in einer Weise interpretiert werden kann. Die Anforderung ist einfach und leicht verständlich formuliert.

Eindeutig	1 = eher wichtig
Implementierungsunabhängig	2
Konsistent	3
Notwendig	4
Realisierbar	5
Rückverfolgbar	6
Singular	7
Verifizierbar	8
Vollständig	9 = eher unwichtig

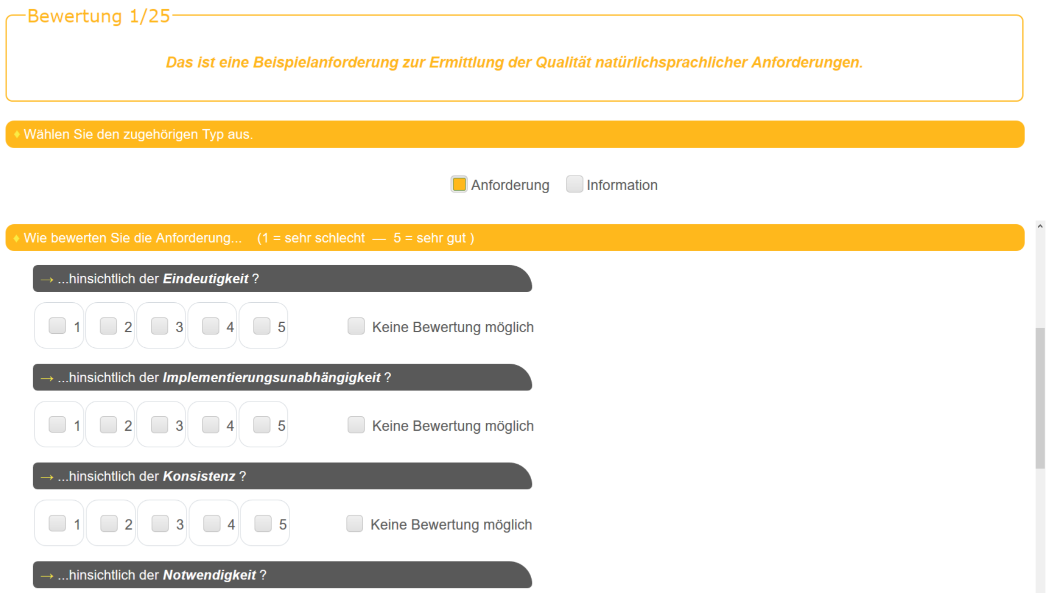
Fig. 3.3.: Prioritization of characteristics from our requirements quality model.

We do not evaluate or analyze these results due to the issues previously discussed in section 2.2.4 (e.g. that priorities of the characteristics may vary depending on the role of the person in a development project). With the prioritization, we rather want to ensure that the experts deal with the descriptions of the quality characteristics before the assessment of the requirements begins, thus reducing the factor of different interpretations of the characteristics.

After this task, the experts begin to assess randomly selected requirements from the data set. Figure 3.4 shows the graphical interface for the assessment, which presents the current requirement to be assessed in the upper text box. In the first step, the expert assesses the type and selects between "Requirement" and "Information". Although we have only selected objects from our data set that are assigned to the type "Requirement", we consider the expert's estimation at this step. We reveal in the result analysis later in section 3.2.2, that not all requirements are assessed as such.

The expert can display the definition of a requirement translated into German, which is based on the Institute of Electrical and Electronics Engineers (IEEE 610, 1990)[6]. If a requirement does not correspond to this definition, the expert can select the type "Information", which will not enable the assessment of the quality characteristics. If "Requirement" is selected, the expert can assess the nine characteristics of the requirement.

[6]A requirement is:
(1) a condition or capability needed by a user to solve a problem or achieve an objective;
(2) a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document;
(3) a documented representation of a condition or capability of the previous two arguments.


Bewertung 1/25

Das ist eine Beispielanforderung zur Ermittlung der Qualität natürlichsprachlicher Anforderungen.

Wählen Sie den zugehörigen Typ aus.

[✓ Anforderung] Information

Wie bewerten Sie die Anforderung... (1 = sehr schlecht — 5 = sehr gut)

→ ...hinsichtlich der ***Eindeutigkeit*** ?

1 2 3 4 5 Keine Bewertung möglich

→ ...hinsichtlich der ***Implementierungsunabhängigkeit*** ?

1 2 3 4 5 Keine Bewertung möglich

→ ...hinsichtlich der ***Konsistenz*** ?

1 2 3 4 5 Keine Bewertung möglich

→ ...hinsichtlich der ***Notwendigkeit*** ?

Fig. 3.4.: Expert assessment of requirements according to our quality model.

The assessment is the "process of assigning labels (typically numbers) to an attribute of an object or action in such a way that the characteristics of the attribute are mirrored in the characteristics of the labels" (Rosenberg, 2008, p. 156). However, there is no agreement on what number of alternatives (in our case it is the number of "scores") represents an optimum. Many of the assessment scales are defined by five or seven response options (Bearden et al., 1993; Peter, 1979; Shaw et al., 1967). Some contributions claim that the results of the answers improve with the number of alternatives. However, this increase becomes significantly smaller when this number exceeds five (Aiken, 1983; Brown et al., 1991; D. McCallum et al., 1988; Weng, 2004). "In practice, the number of alternatives most frequently found [...] is five" (Lozano et al., 2008, p. 1). These results are also confirmed by one of the most influential articles on human perception: Miller (1956) describes that humans are able to distinguish seven (plus or minus two) different items. Scales that have more than seven alternatives can therefore only provide little additional information.

Despite the extensive research contributions, "the issue of the optimal number of response categories in rating scales is still unresolved" (Preston & Colman, 2000, p. 2) and "common practice varies widely" (Krosnick & Presser, 2010, p. 268). O'Muircheartaigh et al. (2000) generally propose to use an odd number of scores for assessment scales. This enables a neutral assessment and allows for evasion of uncertain judgments (Malhotra, 2006; Tsang, 2012). Additionally, Stevens (1946) provides an overview of four generic types that can be applied for assessment scales in scientific measurements: nominal, ordinal, interval, and ratio scales. "Ordinal scales are those that measure rank-ordered data" (Bhattacherjee, 2012, p. 46).

As a result, we decided to use an ordinal five-point scale for the assessment of the characteristics. On the one hand, it provides a suitable range of assessment alternatives and on the other hand, it is the number with the highest "ease of use" for the participants (Jones, 1968). Today, five-point scales are widely used, e.g. in online shopping portals and hotel ratings. Therefore, it is already advisable to present a scale that is familiar to people. In detail, we apply a numeric assessment scale where only "end categories are labelled and are known as self-anchoring rating scales" (Saunders et al., 2012, p. 438). We define the minimum value (1; "very bad") and the maximum value (5; "very good") as end categories and present these labels when the expert assesses the characteristics.

If an expert is not able to assess a quality characteristic at all, the option "No assessment possible" ("NA"; translated from German for "Keine Bewertung möglich") is also available. Thereby, experts are shown "that it is acceptable to say they have no [sufficient] information with which to answer a question" (Krosnick & Presser, 2010, p. 282). Saunders et al. (2012) also suggest adding such a category and place it slightly beside the scale. Therefore, we integrate the option of not having to assess a characteristic (see figure 3.4). A possible disadvantage could be that such assessments increase as a session progresses, "at which point motivation [...] is presumably waning" (Krosnick & Presser, 2010, p. 284). We take this into account when we analyze the results in section 3.2.2. In the following, exemplary German requirements are shown, taking confidentiality of the content into account. These examples should give the reader a sense of the task the experts are confronted with during the assessment.

Exemplary software requirement 1

Die Funktion muss die beiden Heckleuchten eines LKW klassifizieren.

Exemplary software requirement 2

Es muss für Einfahrten die Anzahl der Fehlklassifikationen kleiner als 1 sein.

Exemplary software requirement 3

Die automatische Kalibrierung muss den Gierwinkel der Kamera ermitteln. Dieser muss durch das System automatisch korrigiert werden.

3.2.2 Assessment results

Our research aims to collect reliable data through the requirements assessment by experts. This is also the reason for using methods and techniques from survey research. The assessment tool collects information on how the quality of the requirements is perceived by an expert based on defined characteristics. The collection of assessments implies the consideration of quality issues. Therefore, we developed different strategies to overcome potential problems. Besides, it is not only relevant to know whether an expert can assess a requirement. Rather, it is important to know for which characteristics the expert has difficulties in assessing the requirements. The following section contains the presentation and the result analysis from the assessment sessions. We provide an overview of general results and take a look at the distribution of assessment scores per characteristic.

Between April and October 2018, 1,000 software requirements were assessed twice by 113 experts (15 female and 98 male[7]) from the automotive industry. Each of the experts had a background in software engineering in areas of automotive development for comfort and safety functions. The experts had an average of 6.5 years (median of five years) of professional experience in requirements engineering. There is a huge spectrum that ranges from beginners to domain experts with more than 35 years. Figure 3.5 illustrates the distribution. Almost 60 percent of the experts have between four and seven years of professional experience. Mostly, the experts have five years which indicates a solid knowledge of requirements engineering.

We carried out the assessment sessions only in person and held the sessions in a meeting room with a maximum of six participants. This ensured a requirement assessment without disturbances or distractions. The sessions were conducted several times at three different locations (Stuttgart, Munich, and Ingolstadt). For each location, attention was paid to ensuring that the same conditions (e.g. meeting room, time) were available for the experts. A session took about two hours so that an expert needed on average five minutes to assess the characteristics of a requirement. Therefore, it was a time-consuming task resulting in more than 170 hours of effort in total.

Each of the 1,000 requirements in our data set was independently assessed by two different experts according to the quality characteristics. Every assessment was stored individually so that the data set finally consists of 2,000 entries of assessed requirements. Beyond that, a single expert had assessed a requirement for the

[7]We want to note that we have taken both genders into account, as far as availability permits, for the quality assessment. In the following, we do not make any distinction between the expert gender.

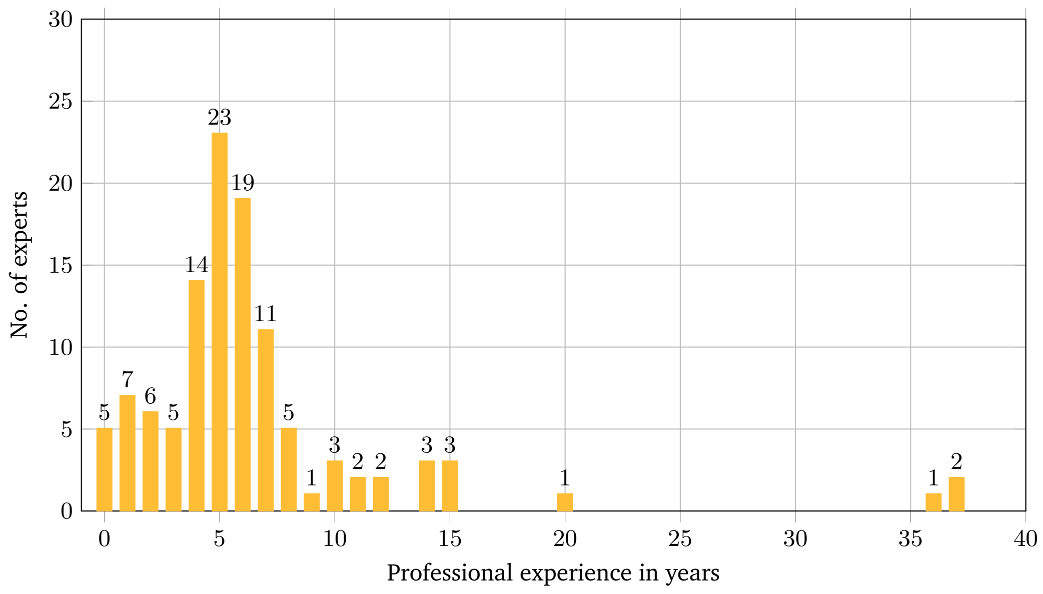


Fig. 3.5.: Number of experts in regard to professional experience in years.

second time within an assessment session: one of the first five requirements, that the expert had already assessed, was randomly chosen and displayed again for assessment as one of the last five requirements of a session. We use these two assessments of a single requirement by the same expert for the agreement analysis in chapter 3.3.4. Of 113 experts, 76 experts have completed the assessment session. 37 experts were not able to assess the full set of 25 requirements due to various reasons (e.g. short-term deadlines, project topics, customer calls).

In the following, we further reduce the data set of the assessed requirements. First, we consider only assessments where the expert has completed the session. We can then ensure that the approach to assessing the requirements has been carried out equally for all experts. This in turn means that there could be only one assessment for a requirement in our data set afterward. For this reason, and this describes our second step, we additionally reduce by requirements that only contain one assessment by an expert. These two steps reduce our data set from 1,000 to 674 requirements that are assessed by two different experts.

We only considered objects attributed to the type "Requirement" during the selection of the data set. However, some objects were classified as type "Information": for 97 requirements, "Information" was selected in the assessment by two different experts. Besides, 159 requirements were classified as "Information" only by one expert. These results indicate that experts could have difficulties in identifying requirements.

The experts realized that an assessment of the characteristics was not required if the type "Information" was selected. This can sometimes lead to the fact that "Information" was selected more frequently when expert motivation was reduced to avoid a more complex assessment. We considered this issue during the development of the tool. Therefore, we saved the position for each assessment in a session. In table 3.3, we see the number of requirements assessed as "Information" depending on the position in a session. The positions are summarized in five ranges. Accordingly, 86 requirements within the first five assessments were classified as "Information". A decreasing motivation of the experts cannot be recognized by an increasing selection of the type "Information". The results, however, have an impact on our data set in that we only take into account those assessments of the experts who agreed on the type "Requirement". This leads to a reduction to 418 assessed requirements.

Tab. 3.3.: Number of selections of the type "Information".

Position	No. of "Information"	Percentage
1 – 5	86	25%
6 – 10	72	20%
11 – 15	72	20%
16 – 20	77	22%
21 – 25	46	13%

In the following, we present the distribution of the assessments for the characteristics. Table 3.4 and figure 3.6 illustrate how often a score was selected. For example, for the characteristic *Singular (C6)*[8], the majority of all requirements were given the score "5" (43 percent). Also, the experts were mostly able to assess this characteristic, as the option "NA" (="No assessment possible") was rarely selected.

Tab. 3.4.: Frequency distribution of the selected scores for each characteristic.

Score	C1	C2	C3	C4	C5	C6	C7	C8	C9
1	12%	3%	4%	6%	4%	10%	3%	8%	6%
2	21%	11%	7%	16%	9%	14%	8%	14%	8%
3	23%	16%	16%	26%	18%	13%	15%	19%	18%
4	23%	35%	30%	25%	28%	19%	28%	28%	27%
5	17%	24%	33%	24%	28%	43%	17%	30%	39%
NA	4%	11%	10%	3%	13%	1%	29%	<1%	2%

If an expert cannot assess a characteristic for a requirement, the option "NA" is possible to select. Therefore, the number of "NA"s indicates challenges during the assessment. The characteristic *Consistent (C2)*, for example, could not be assessed for 11 percent of the requirements. The experts also seem to have difficulties in assessing

[8]In tables, the nine quality characteristics are abbreviated as follows: C1 = Complete, C2 = Consistent, C3 = Feasible, C4 = Implementation Free, C5 = Necessary, C6 = Singular, C7 = Traceable, C8 = Unambiguous, C9 = Verifiable.

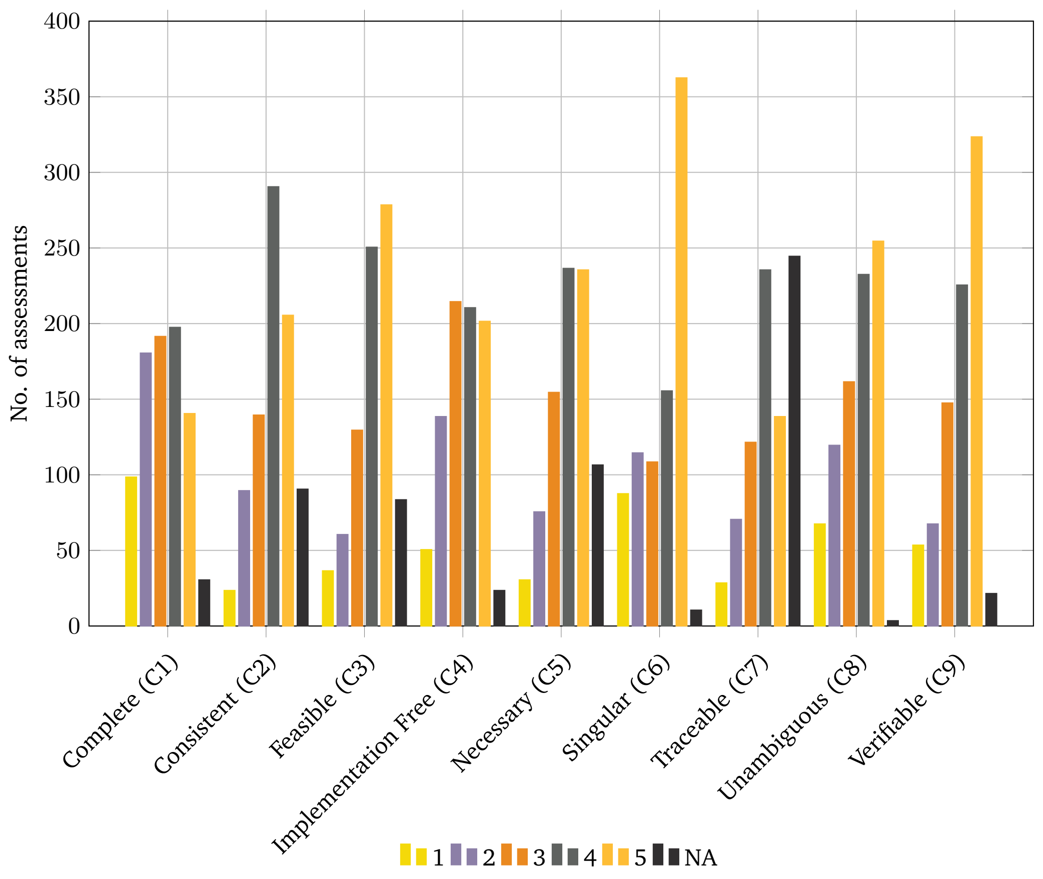


Fig. 3.6.: Number of scores for each characteristic.

Feasible (C3) (10 percent) and *Necessary (C5)* (13 percent). The characteristic *Traceable (C7)*, however, seems to present a major challenge in the assessment: 29 percent of the requirements could not be assessed by the experts. On the other hand, for *Unambiguous (C8)* (<1 percent) and *Singular (C6)* (1 percent) the experts are able to assess almost every requirement. For *Complete (C1)* (4 percent), *Implementation Free (C4)* (3 percent) and *Verifiable (C9)* (2 percent), an assessment seems to be less sophisticated as well compared to *Traceable (C7)*.

In addition to the number of "NA"s, other findings can be drawn from the frequency distribution of the scores for each characteristic. It is remarkable that for a few of the characteristics (e.g. *Complete (C1)*, *Consistent (C2)*, and *Necessary (C5)*) a kind of a normal distribution of the scores can be assumed, whereas for other characteristics the selection of one score clearly predominates (e.g. *Singular (C6)*).

In the next section, we identify possible challenges that experts have encountered during the assessment process and describe issues regarding the collection of reliable assessment data.

3.2.3 Assessment issues

The characteristic *Unambiguous* is well considered in the literature within the research area of ambiguity detection. The experts seem to recognize ambiguity and are therefore able to conduct an assessment. It is also understandable why some of the characteristics are hardly considered in research. Approaches in the literature especially for *Feasible* and *Necessary* are barely available. Experts seem to have challenges to assess these characteristics of a single requirement when other contextual information is not available.

The survey design also indicates reasons why the experts are not able to assess the characteristics equally well. In the first step, the experts are confronted with the characteristics (and the descriptions) that they have to prioritize. This task is only for the experts to deal with the characteristics in advance of the assessments. In the further course, however, we do not give specific examples of how to assess the characteristics of a requirement. Due to the different experience levels of the experts, a requirement can therefore be assessed more or less well.

Besides, the descriptions of the characteristics in the ISO 29148 standard already allow for interpretation and have a certain degree of fuzziness. Although all characteristics refer to an individual requirement, *Traceable*, for example, describes that a requirement should be "upwards and downwards traceable", which would require additional context information for an assessment. However, the experts were able to assess 71 percent of the requirements in terms of this characteristic. The analysis in section 3.3 shows us whether these results are also reliable.

On the other hand, the experts were able to assess almost every requirement with regard to the characteristics *Singular* and *Unambiguous*. *Singular* describes itself by the fact that only one requirement is specified and no conjunctions are used. *Unambiguous* applies to a requirement if it can only be interpreted in one way and if it is specified simple and easy to understood. Thus, experts can identify multiple requirements and are also able to appreciate the complexity of how a requirement is specified. These observations are intuitive and are not based on statistical analyses. We provide a more thorough analysis in the next section.

In most cases, the choice of experts has a considerable influence on the quality of the results. A typical automotive development cycle lasts an average of four years. It is then relevant that the majority of the experts have at least this amount of experience. This is important because the expert has then passed through all the steps of a development process at least once from a requirements engineering perspective. In our sample, only every fifth expert had less professional experience. Even though less experience could then eventually lead to less reliable results, we have taken such assessments into account. Thus, in the trade-off between reliability and quantity of labeled data, we tend to focus on the latter.

As we are aware of existing bias, we followed different measures to overcome these. At the beginning of a session, each expert received the same level of information. We also emphasized that the results are treated anonymously so that no conclusions can be drawn about an individual person. Every session was conducted using the same tool and survey approach. We also emphasized the importance of face-to-face sessions, which leads to a high response rate and a conscientious assessment, especially when the topic is considered interesting and relevant for the experts' project work. By conducting the sessions in person and by securing access to the tool, we were also able to ensure who was assessing the requirements, giving us control over the data we receive. This was complemented by the fact that the sessions were not held at the workplace, but in a separate meeting room with other experts. As we had up to six experts simultaneously assess requirements within one session, the experts were able to cope with the task at different speeds. This could sometimes lead to other experts feeling under pressure and therefore assessing requirements less conscientiously. We are aware of it, but consider this a low risk.

With the results of the sessions and under consideration of the issues, it is possible to determine the strength of the agreement for the assessments. In the following section, we present the definition and the determination of agreement in the context of our research.

3.3 Agreement on assessments

In this section, we focus on the agreement of the assessments[9]. We consider agreement from two perspectives: intra-rater agreement and inter-rater agreement. First, we provide an introduction to both measures. Then, we describe how agreement is determined in general and focus on obtaining the values for each perspective.

"The key to reliability is the agreement observed among independent [experts]" (Hayes & Krippendorff, 2007, p. 78). The ability to reliably assess requirements is reflected in the degree of agreement between the assessments. The results from this section, i.e. whether experts are able to reliably assess quality characteristics of a natural language requirement, answer our second research question.

[9]In the literature, different terms are used for the analysis of agreement. We use terms that refer to our research topic (e.g. experts, scores, characteristics) to provide a better reference.

"When relying on human [experts], researchers must worry about the quality of the data" (Hayes & Krippendorff, 2007, p. 78). In data collection techniques, in which assessments are obtained from experts, the following need to be taken into account: the ability of the participants to answer questions and the existing interest of the participants for the topic (Bhattacherjee, 2012). The avoidance of ambiguous alternatives in the scale and the clear and simple formulation of questions help to improve the results. Although we have considered these issues, this does not completely exclude unreliable assessments by experts. For that reason, we ask to what extent the experts can assess the characteristics of the requirements in agreement.

The determination of agreement requires multiple assessments or interpretations of the same data. Thus, we gathered multiple assessments of the same requirement. This enables us to determine whether the experts assess a requirement consistently based on quality characteristics. In the following, we define both types of agreement—intra-rater agreement and inter-rater agreement—in relation to the quality assessment of natural language requirements.

3.3.1 Intra-rater agreement

For the determination of the intra-rater agreement, we have the same expert assess a requirement for the second time within one session (see section 3.2.2). Highly similar or identical assessments indicate a high degree of intra-individual objectivity and the consistent assessment of a single expert (Herz, 2010). Figure 3.7 illustrates the concept. The expert "Exp1" assesses the requirement "ReqA" a second time within a session. With these two assessments it is possible to determine the intra-rater agreement which we define in the context of our research as follows:

Intra-rater agreement

The intra-rater agreement determines the degree of agreement of repeated requirement assessments by a single expert.

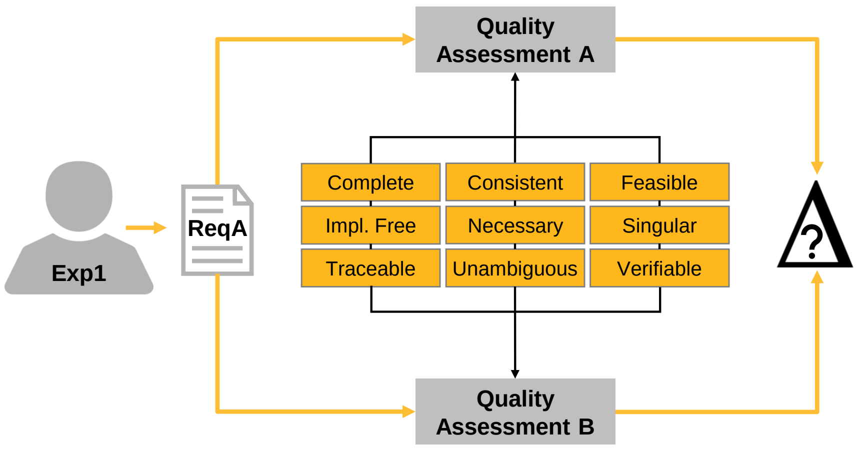


Fig. 3.7.: For the determination of the intra-rater agreement, the expert "Exp1" assesses the requirement "ReqA" a second time within an assessment session.

3.3.2 Inter-rater agreement

The inter-rater agreement measures the degree of agreement when more than one person assesses the same data. For the determination of the inter-rater agreement in our research, we calculate the degree of agreement by two different experts. The concept is shown in figure 3.8, in which two experts—"Exp1" and "Exp2"—assess the same requirement "ReqB". The inter-rater agreement is a direct measure of inter-individual objectivity which we define in our research as follows:

Inter-rater agreement

The inter-rater agreement determines the degree of agreement of requirement assessments by two different experts.

3.3.3 Determination of agreement

Usually, a contingency table is used to present assessment results (Bortz & Döring, 2007; Landis & Koch, 1975; Rost, 2004). Table 3.5 illustrates such a table with exemplary values for the assessment of a quality characteristic by two different experts. The table shows the number of the experts' assessments ("Exp1" and "Exp2") along the ordinal five-point scale that we have defined for the requirement assessment. In a contingency table, the numbers indicate how often assessments occur according to the intersection of the two scores. In our example, "Exp1" assessed

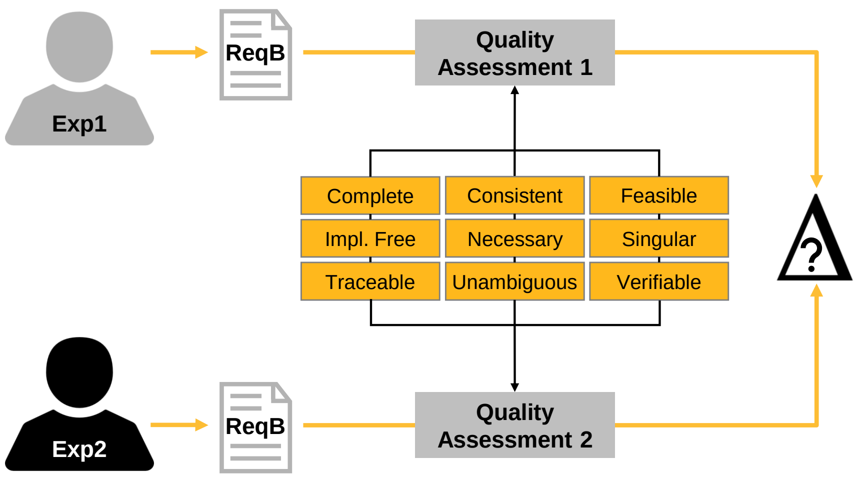


Fig. 3.8.: For the determination of the inter-rater agreement, two experts ("Exp1" and "Exp2") assess the same requirement "ReqB".

20 requirements with the score "2". For 14 of these requirements, "Exp2" selected the score "2" as well. However, "Exp2" also assessed three of these requirements with a score of "1". The row sums describe how often "Exp1" used each score to assess the requirements. The column sums show that analogously for "Exp2".

Tab. 3.5.: Contingency table with exemplary values for the number of scores.

	Exp2					
Exp1	1	2	3	4	5	Total Exp1
1	14	3	1	2	0	20
2	3	14	1	0	2	20
3	1	2	14	1	2	20
4	0	1	3	14	2	20
5	2	0	1	3	14	20
Total Exp2	20	20	20	20	20	100

The determination of agreement in its simplest form is described by the proportion of observed agreements to the total number of assessments. It is defined as "the percentage of judgments on which the two analysts agree when coding the same data independently" (Scott, 1955, p. 323). In our example from table 3.5, this is represented by the main diagonal of the contingency table with a result of 0.70. We simply count the values where the experts agree (i.e. 70) and divide this result by the total number of assessments (i.e. 100). A major drawback, however, is that this approach does not take into account random agreements. Therefore, the literature suggests different alternatives to quantify the extent of agreement among experts (Gwet, 2015).

We follow Gwet (2014), who describes a process to determine an appropriate coefficient for a different number of experts and types of scales (ordinal, interval, and ratio). As we defined an ordinal scale and the whole assessment of the characteristics includes more than two experts, we have the choice of five alternatives from which we have chosen the coefficient alpha (α_k) by Krippendorff (2011). "Krippendorff's alpha is a reliability coefficient developed to measure the agreement among observers, coders, judges, raters, or measuring instruments drawing distinctions among typically unstructured phenomena or assign computable values to them" (Krippendorff, 2011, p. 1).

Krippendorff's alpha can be used regardless of the sample size and number of experts, and can also handle missing data and different metrics or levels of measurement (Hayes & Krippendorff, 2007). It is also described as a generalization of several known reliability coefficients. We refrain from describing the coefficient using coincidence tables and difference functions as listed in Hayes and Krippendorff (2007). Instead, for the definition, we use notations provided by Gwet (2014) and Gwet (2015).

In the following, with n' we describe the number of requirements assessed with a particular score x_k (score "1" to score "5") by two different experts or by one expert twice. The total number of requirements that could also include the assessment "NA" is designated by n. With q we describe the number of different scores (i.e. 5) an expert can choose from to assess a requirement. The number of experts who assessed requirement i with a particular score x_k is denoted by r_{ik}. Further, r_i describes the number of experts assessments for requirement i with $\bar{r}$ as the average value of all r_i's. We also consider $\epsilon_n = 1/(n'\bar{r})$. Accordingly, the coefficient α_k (Krippendorff's alpha) is defined as:

$$\alpha_k = \frac{(p_o - p_e)}{(1 - p_e)}, \text{ where} \tag{3.1}$$

$$p_o = (1 - \epsilon_n)\, p'_o + \epsilon_n \tag{3.2}$$

$$p'_o = \frac{1}{n'} \sum_{i=1}^{n'} \sum_{k=1}^{q} \frac{r_{ik}(r^*_{ik} - 1)}{\bar{r}(r_i - 1)} \tag{3.3}$$

$$p_e = \sum_{k,l}^{n'} w_{kl} \pi_k \pi_l \tag{3.4}$$

Besides, r^*_{ik} and π_k are used to calculate the observed agreement p_o and the expected agreement p_e:

$$r^*_{ik} = \sum_{l=1}^{q} w_{kl} r_{il} \tag{3.5}$$

$$\pi_k = \frac{1}{n'} \sum_{i=1}^{n'} \frac{r_{ik}}{\bar{r}} \tag{3.6}$$

The observed agreement is generally described with p_o, i.e. the empirical probability of agreement. It is often found in different forms when determining agreement, "but on its own it does not yield values" (Artstein & Poesio, 2008, p. 558). One of the main reasons is the lack of correction in case of an unequal distribution of the scores across the requirements. An example from Di Eugenio and Glass (2004) illustrates this phenomenon, which we adapt to our research topic: Assuming that a data set contains 90 percent of objects from the type "Requirement" and 10 percent of objects from the type "Information". The goal is to have the correct classification of the type by experts. By chance alone, we expect in this example an 81 percent classification as "Requirement" (0.90 x 0.90) and a 1 percent classification as "Information" (0.10 x 0.10). Hence, there is an 82 percent agreement between experts. Although this value could appear acceptable at first glance, an observable agreement that is below 82 percent would be worse than a random assessment. Therefore, it is important to correct the observed agreement by an expected agreement.

The expected agreement is represented by p_e and denotes the random assessment of the experts. It "quantifies how often you would normally expect 2 randomly selected observers to agree if the scoring is performed randomly according to the observed classification probabilities" (Gwet, 2015, p. 6). Thus, "[observed agreement, p_o] is an observational probability of agreement and [expected agreement, p_e] is a hypothetical expected probability of agreement under an appropriate set of baseline constraints" (Landis & Koch, 1977, p. 163).

Besides the consideration of random agreement, some of the typical coefficients do not distinguish deviations in the experts' assessments. This seems to make sense with dichotomous alternatives. However, if a scale has more than two alternatives, different assessments by the experts may be close to each other. A measure should therefore also take into account when the assessments of two experts differ only slightly from each other. If the scale alternatives are available in the ordinal form, as is the case in our research, discrepancies in the assessments can be weighted differently.

It is generally suggested to consider a weighted approach, as "weighted coefficients [are] more appropriate for many annotation tasks" (Artstein & Poesio, 2008, p. 576). Assessments that are increasingly farther apart from each other are increasingly weighted less. This makes it possible to assess discrepancies in the assessments to the score distances. For our example in table 3.5, a disagreement between score "5" and score "4" would be weighted differently than a disagreement between score "5" and score "2".

In our research, for each pair of scores (k, l) a weight is assigned which can take a value between 0 and 1. This value decreases with increasing disagreement between the two scores. For example, if the two experts assess a requirement with the same score $(k = l)$, full agreement is reached. The weight w_{kk} is then assigned the highest value of 1. If the two experts assess a requirement with different scores $(k \neq l)$, then a weight less than 1 is assigned to this pair and describes the fraction of full agreement. For any two scores k and l a weighting function w_{kl} is defined. For the ordinal scale in our research, Krippendorff (2011) suggests a weighting scheme. Gwet (2015) offers a simplified representation of this scheme, which we present in the following:

$$w_{kl} = \begin{cases} 1, & if k = l \\ 1 - \frac{\#\{(i,j),\, min(k,l) \leq i < j \leq max(k,l)\}}{w_{max}}, & if k \neq l \end{cases} \tag{3.7}$$

Gwet (2015) notes that $\#\{(i,j), min(k,l) \leq i < j \leq max(k,l)\}$ describes the number of pairs (i, j) with $i < j$, that can be created with numbers between $min(k, l)$ and $max(k, l)$. Further, w_{max} is defined as maximum value over all values of k and l.

With the definition of the weighting function, we are now finally able to calculate values for Krippendorff's alpha (α_k). The result of such a calculation ranges from 0 to 1, where 0 represents perfect disagreement and 1 represents perfect agreement. Regarding the question of when a value for α_k is sufficient, Krippendorff provides suggestions on how to interpret the results. Thus, values for α_k equal to or higher than .667 is the lowest conceivable limit to draw tentative conclusions (Krippendorff, 2004).

In general, a low level of agreement among experts can have various reasons. Among other things, there may be incongruities that relate to the requirements to be assessed. We realize that the descriptions of the characteristics on which the quality of natural language requirements is based are quite vague. We discussed vagueness as a research issue in chapter 2.2.4 and already stated that the descriptions allow a certain degree of interpretation.

Therefore, we would like to draw attention to a probable discrepancy in the assessments, which can be explained by two factors: first, different interpretations of the descriptions for the characteristics by the experts, which can lead to a varying assessment; second, identical or very similar interpretations of the descriptions for the characteristics, but different professional experiences of the experts, which can lead to a varying assessment. We are aware of these issues and address them in the next section.

Besides, the deviation from the "perfect agreement", whether by examination of the assessments between two experts or of a single expert, is considered and determined in the following section as well.

3.3.4 Agreement analysis

With the foundations from the previous chapter, we are able to determine agreement in the context of our research. Our premise is that if the experts statistically agree on the quality of natural language requirements, we are able to implement an automated solution. On the other hand, if the experts are not able to assess requirements in agreement, an automated approach is difficult to implement.

We use two approaches to determine the agreement of the expert assessments: intra-rater agreement and inter-rater agreement. Thus, we can evaluate a consistent assessment of a single expert and the agreement between two experts. In the following section, we first analyze the results for the intra-rater agreement. Afterward, we focus on the determination and analysis of the inter-rater agreement. For both measures, we use Krippendorff's alpha (α_k).

In general, we can expect the results for intra-rater agreement to be higher than for inter-rater agreement. This is because if one expert is not "reliable", then two different experts with different professional experience may be even less reliable. However, this is then not only a question of the experience of the experts but could rather refer to whether these characteristics—based on the descriptions from ISO 29148—are assessable at all for a single requirement.

Determination of intra-rater agreement.

In total, 76 experts have completed an assessment session and therefore certainly assessed one requirement a second time. Two of the experts classified an object as "Information" that they had previously classified as "Requirement" in the first assessment. These two assessments are not considered in the following. We are aware that this already describes a certain degree of disagreement. Each of the remaining requirements (i.e. 74 requirements) was repeatedly assessed by an expert based on the nine characteristics, resulting in a total of 666 assessment pairs. One pair includes both assessments of a single expert for a characteristic of a requirement.

We first analyze the consistent use of the option "NA" ("No assessment possible"). In 76 of the 666 assessment pairs, "NA" is present. Of these, 35 pairs have a consistent assessment of "NA". For 26 pairs, "NA" was chosen first, but in the second assessment, a score on the five-point scale was selected by the expert. There are 15 pairs where the reverse case is true. This shows that experts tend to choose a score more often when reassessing a requirement. We can therefore conclude that experts are consistent in about half of all cases where they cannot assess a requirement. Besides, experts tend to assess a requirement on the five-point scale, even if "NA" was previously selected.

For the further determination of the intra-rater agreement, we use the coefficient α_k. We reduce the data set by the assessment pairs where the expert has selected at least one "NA" since we are only considering the discrepancy between the numerical scores[10]. The results from the determination of α_k for the intra-rater agreement are shown in table 3.6. We also present the results for the observed agreement p_o.

Tab. 3.6.: Observed agreement (p_o) and Krippendorff's alpha (α_k) for the determination of intra-rater agreement.

	C1	C2	C3	C4	C5	C6	C7	C8	C9
p_o	.582	.563	.608	.614	.573	.681	.527	.604	.671
α_k	.669	.672	.691	.707	.768	.775	.673	.776	.714

The results differ only slightly between the characteristics. The values for the observed agreement p_o vary between .527 for *Traceable (C7)* and .681 for *Singular (C6)*. This indicates that more than half of the assessments of a single expert are identical on the five-point scale.

The results for α_k varies between .669 for *Complete (C1)* and .776 for *Unambiguous (C8)*. For the characteristics *Necessary (C5)*, *Singular (C6)*, and *Unambiguous (C8)*,

[10]Furthermore, the discrepancy cannot be reasonably weighted for an "NA" assessment.

an expert is particularly capable of repeatedly assigning very nearby scores. For the characteristics *Feasible (C3)*, *Implementation Free (C4)*, and *Verifiable (C9)* there are larger discrepancies. The results for *Complete (C1)*, *Consistent (C2)*, and *Traceable (C7)* imply that a consistent assessment of these characteristics is somewhat more challenging. According to Krippendorff (2004), we can draw tentative conclusions for all characteristics as the results are equal to or higher than .667. Thus, a single expert sufficiently "agrees"[11] on all characteristics in our research. However, for some of these, e.g. *Complete (C1)*, *Consistent (C2)*, and *Traceable (C7)*, the results are close to the required minimum value stated by Krippendorff.

Overall, two points can be identified from the results. First, an expert repeatedly chooses the same score at least in every second assessment. For *Singular (C6)* and *Verifiable (C9)*, more than two out of three requirements are assessed identically. As the expert assesses on a five-point scale, one has to consider how far the scores differ in the two assessments. Therefore, the second point to note is that an expert agrees on a sufficient level to draw conclusions and reassess also close to the score of the first assessment. We can conclude that a single expert can reliably assess the characteristics of a natural language requirement based on our quality model.

Determination of inter-rater agreement.

The determination of the inter-rater agreement follows a similar process. In section 3.2.2, we examine the data set and discover that some of the requirements are classified as "Information" by the experts. The analysis reduces our data set to 418 requirements based on the assessment of two different experts. This results in a total of 3,762 assessment pairs for the nine characteristics. In a first step, we examine again whether the option "NA" was chosen consistently among the experts.

In our data set, we identify the selection of at least one "NA" in 540 assessment pairs. Out of this, both experts agree on "NA" in 70 assessment pairs. For the remaining pairs, a score on the five-point scale was selected by the other expert. We describe in section 3.2.2 that the use of "NA" can indicate challenges during the assessment. We asked experts with different professional backgrounds for the assessment sessions, which means that some requirements could presumably be assessed more frequently by implicit knowledge from professional experience.

In general, it is also less challenging for the expert to assess a requirement as "NA" than to give a reliable score. For that reason, we also examine when "NA" was used

[11]The word "agree" in this context is representative for the consistent assessment of the characteristics by a single expert.

in the assessment session. In section 3.2.2, we mentioned that our tool adds the position of each assessment in a session. Table 3.7 shows when and how often a requirement is assessed as "NA". The number is to be interpreted as an aggregated frequency for all characteristics. The results allow us to determine whether the more a session progresses, the less the experts are motivated to assess a requirement.

Tab. 3.7.: Number of "NA"s related to the position in the assessment sessions.

Position	No. of "NA"s	Percentage
1 – 5	148	24%
6 – 10	139	23%
11 – 15	153	25%
16 – 20	98	16%
21 – 25	72	12%

The analysis reveals that the frequency of "NA" does not increase with the progress of an assessment session. Rather, it can be seen that experts select "NA" less frequently towards the end of an assessment session than at the beginning.

In the further course, we focus on the discrepancies in the assessments for the scores. For the determination of the inter-rater agreement, we use Krippendorff's alpha (α_x) again and reduce the data set by the assessment pairs containing at least one "NA". We extend the analysis by two additional steps: first, we analyze whether experts agree on the extreme values; second, we analyze whether the experts tend to agree in the requirement assessment.

The first step is analyzed by considering only the extreme values of the five-point scale. If one expert has chosen a score "1" or "5", the corresponding assessment pair is considered for a separate determination of the Krippendorff's alpha and described with the coefficient α_x. For the second step of our extended analysis, we determine whether the experts tend to assess equally. For this purpose, we transform the five-point scale: the scores "1" and "2" are grouped into the first class; the second class contains the score "3"; the third class includes the scores "4" and "5". Again we use Krippendorff's alpha to determine the values and introduce the coefficient α_y. The results from the calculation of α_x and α_y are considered for a better interpretation of the assessed requirements. However, only α_k is used to decide whether experts can reliably assess natural language requirements based on our quality model. We present results for the observed agreement p_o and Krippendorff's alpha (α_k, α_x, α_y) for each characteristic in table 3.8.

The observed agreement p_o is approximately at the same level for all characteristics. The range is from .302 for *Complete (C1)* to .401 for *Singular (C6)*. This means that the experts assess every third requirement identically. The values of α_k, α_x, and α_y

Tab. 3.8.: Observed agreement (p_o) and Krippendorff's alpha ($\alpha_k, \alpha_x, \alpha_y$) for the determination of inter-rater agreement.

	C1	C2	C3	C4	C5	C6	C7	C8	C9
p_o	.302	.354	.357	.333	.329	.401	.318	.342	.361
α_k	.331	.185	.387	.336	.213	.698	.144	.673	.321
α_x	.333	.046	.063	.039	.032	.421	.024	.410	.056
α_y	.283	.079	.291	.356	.133	.696	.097	.674	.258

differ more between the characteristics. The results can be divided into three areas, which we present below.

The first area[12] considers alpha values with $\alpha_k \leq .222$ and includes *Consistent (C2)*, *Necessary (C5)*, and *Traceable (C7)*. For these characteristics, the experts' assessments diverge comparatively more. In addition, low values can be determined for α_x and α_y. This indicates that there is no agreement on the assessment of extreme values and that the two experts also do not generally tend to agree. A closer look at the description from ISO 29148 reveals the challenges that arise. The characteristic *Consistent (C2)* is described as free of conflicts with other requirements. *Necessary (C5)* is defined as an essential capability that cannot be fulfilled by other capabilities. Finally, *Traceable (C7)* describes that a requirement is traceable upwards and downwards. From the descriptions, it can be deduced that additional contextual requirements are necessary for a reliable assessment. A requirement can be assessed in terms of consistency if all additional requirements are known. The same applies to the necessity of a requirement. For traceability, on the other hand, further requirements are needed that are related to the requirement being assessed. This relationship can be present in higher-level or lower-level requirements in the whole development project.

In the second area, characteristics with $.222 < \alpha_k \leq .444$ can be grouped. This includes *Complete (C1)*, *Feasible (C3)*, *Implementation Free (C4)*, and *Verifiable (C9)*. The description for the characteristic *Complete (C1)* indicates that "the stated requirement needs no further amplification because it is measurable and sufficiently describes the capability and characteristics to meet the stakeholder's need" (ISO 29148, 2011, p. 11). A requirement is *Feasible (C3)* if it is technically achievable and does not require major technological advances. To assess *Implementation Free (C4)*, it is important to determine whether the requirement avoids placing unnecessary constraints and states what is required (and not how to implement). The characteristic *Verifiable (C9)* describes whether the "requirement has the means to prove that the system satisfies the specified requirement" (ISO 29148, 2011, p.

[12]Since the lowest conceivable limit to draw tentative conclusions has the value .667 according to Krippendorff (2004), we have divided this value into thirds for the limits of the areas.

11). These characteristics have in common that an assessment may depend on the professional experience of the expert. In our assessment sessions, we have asked experts with different years of experience. However, whether a requirement is complete, feasible, implementation free, and verifiable may become more reliable to assess with increasing years of professional experience. It is also noticeable that the values for α_x are similarly low as for the characteristics from the first area, whereas significantly higher values can be identified for α_y. This means that the experts have rather a common tendency in the assessment than for the characteristics of the previous area, but extreme values are still hardly assessed in agreement. An exception is the characteristic *Complete (C1)*, for which a significantly higher value for α_x can be determined compared to the characteristics of the first and second area. Consequently, experts are more likely to agree on extreme values for the completeness of a requirement.

The third area involves the characteristics *Singular (C6)* and *Unambiguous (C8)*, for which results for α_k can be identified at a sufficient level of agreement to draw conclusions (according to Krippendorff (2004) this relates to $.667 \leq \alpha_k$). Besides, the comparatively highest values can be determined for α_x and α_y. Experts are therefore not only able to assess with sufficient agreement along the five-point scale but there seems to be agreement on the general tendency in the assessment as well. There is also comparatively high agreement when the experts assess the characteristics of the requirements with the extreme values "1" or "5". In ISO 29148, the characteristic *Singular (C6)* is described when only one requirement is contained in a statement and no conjunctions are used. *Unambiguous (C8)* applies to a requirement that is simply specified and can only be interpreted in one way. These descriptions only refer to the content of a single requirement. This means that the requirements can be reliably assessed for the characteristics independently of a broader context and even without extensive professional experience. The hint that conjunctions should not be used in a requirement also contributes to the general ability to assess the characteristic *Singular (C6)*.

The results show that for a large part of the characteristics no reliable assessment of the quality characteristics of natural language requirements can be performed by experts. Only for two characteristics, it appears that the experts agree on a sufficient level. We conclude that experts agree on and can reliably assess whether a requirement is singular and unambiguous. We consider these results to be sufficient to implement an automated approach that is capable of assessing the quality characteristics *Singular* and *Unambiguous* as an expert would. However, the identified deviation in the assessment shows that even for these two characteristics a superior performance of an automated approach cannot be achieved.

3.4 Conclusion

In our second research question, we focused on whether experts can reliably assess natural language requirements based on the characteristics of our quality model. In particular, we focused on the agreement of the assessments, since this is an important prerequisite for the performance of an automated approach. By proving reliable results from the assessment sessions with experts, we can use the labeled data as input for a supervised learning approach to create classification models. A classifier then enables a quality assessment of requirements that were previously unknown to the model and assesses them as if an expert would do it. Thus, a reliable data basis ensures the accuracy of the results and the performance of the automated classification.

In this chapter, we described the assessment of requirements quality by experts. First, we presented the process for data selection based on software requirements from the automotive industry. This resulted in a data set of 14,256 unique German natural language requirements. Of these, we have imported 1,000 requirements into our assessment tool, which allowed an expert assessment of requirements based on our quality model. The general results revealed that experts are not always able to assess the characteristics of a requirement. Especially for the characteristic *Traceable*, every third requirement could not be assessed by the experts. On the other hand, almost every requirement could be assessed with regard to the characteristics *Singular* and *Unambiguous*.

Further, we examined agreement from two perspectives. Intra-rater agreement determined the "agreement" of a single expert by the repeated assessment of the same requirement. With the inter-rater agreement, we considered the agreement of two different experts. We determined the strength of agreement by using the alpha coefficient (α_k) from Krippendorff (2011). We identified that a single expert assessed the requirements with varying degrees of agreement, but at a sufficient level. This led us to conclude that the assessments of the quality characteristics of a single expert are reliable.

The determination of inter-rater agreement revealed insights into the assessment of the characteristics of different experts. We identified three areas. The first area included characteristics for which the information of additional relevant requirements would be helpful for assessing. Accordingly, values for α_k revealed a low agreement level. The second area described characteristics whose assessment could presumably depend on the professional experience of the experts. As we have asked experts with different years of professional experience, the results for these characteristics still described an insufficient, albeit higher, level of agreement compared to the first area. The third area contained characteristics for which values for α_k equal to or higher than .667 could be determined. The results from the latter area allowed us to draw tentative conclusions (Krippendorff, 2004).

The results showed that experts cannot reliably assess all quality characteristics of requirements. The strength of agreement differed with regard to the characteristics. Therefore, we answered our second research question, whether experts can reliably assess the quality characteristics of natural language requirements, for *Singular* and *Unambiguous* only. For the others, no sufficient level of agreement could be determined, so that for these characteristics no reliable database is available for the implementation of an automated approach.

We paid attention to ensure the same conditions for the experts and tried to reduce external factors that might influence the expert's ability to assess. However, we also identified several threats to the validity of our assessment sessions. The major threat came from the actual assessment process since this task is dependent on human judges. For the validity of the results, characteristics of the experts such as experience, expertise, and expectations influenced how requirements are assessed. Besides, there was a further threat from the anonymous nature of our assessment sessions. This prevented us from validating the results in detail with the experts afterward.

An important factor that influenced the ability to assess requirements by characteristics was the domain and context knowledge of the experts. The assessments were performed by experts with different project experiences. Existing knowledge of functions or systems related to the requirement could therefore influence the assessment of requirements that were known to one expert but not to another due to a lack of domain knowledge. Thus, the experts' experience plays a role in assessing quality and identifying quality defects. This factor is also mentioned in similar articles dealing with the assessment of requirements quality (Femmer et al., 2017). We have mitigated this risk by selecting mainly practitioners with several years of experience for the assessment sessions.

In some contributions, guidelines are provided for the assessment or annotation of data. These guidelines specify how the requirements are to be assessed (Femmer et al., 2017; Ormandjieva et al., 2007). In our assessment sessions, we presented guidelines to the experts in the form of descriptions of the characteristics. The descriptions provided general information about what needs to be considered in a requirement to fulfill the characteristic. However, they varied in length and precision and we did not give an explicit indication of how the requirements should be assessed afterward. The descriptions were based on the international standard ISO 29148 and were provided to the experts shortly before the start of the assessments. Since the descriptions already allow for interpretability and contain fuzziness, the reliable assessment of the requirement was quite a challenge.

Besides, an identical assessment of a requirement by two experts does not mean that the assessment is correct (Krippendorff, 2004). If a single requirement contains a large number of sentences that are additionally linked by conjunctions, this suggests that this requirement probably does not fulfill the characteristic *Singular* according to the description from ISO 29148. If, for example, both experts assessed such a requirement with the score "5" for *Singular*, this contributed positively to the inter-rater agreement. Whether the requirement was then actually *Singular* in the sense of ISO 29148 remains questionable.

With the results of this chapter, we can call the data set of the assessed requirements for the characteristics *Singular* and *Unambiguous* reliable. In the next chapter, we answer the question of whether machine learning algorithms can be used to accurately assess these two quality characteristics of a natural language requirement.

4 Automated assessment of requirements quality

*The goal is to turn data into information,
and information into insight.*

— **Carly Fiorina**
Former CEO of Hewlett-Packard

In the previous chapter, we examined whether experts are able to assess the quality characteristics of natural language requirements defined by ISO 29148. The results reveal differences regarding the agreement of experts. Consequently, only for the characteristics *Singular* and *Unambiguous*, experts agree on a sufficient level.

The following chapter describes the approach to assess these two characteristics in an automated way by applying techniques from natural language processing and algorithms from machine learning. More precisely, a supervised machine learning approach is presented that uses assessed requirements as labeled data for the training process. With the results from this chapter, we are able to provide an answer to our first research question, whether machine learning algorithms can be used to accurately assess (two of the) quality characteristics of natural language requirements.

The chapter is divided into four sections. First, we present methodological foundations for natural language processing, machine learning, and text mining. The second section focuses on the implementation of the automated approach. We describe the preprocessing of our data set and present the analysis and definition of features that can be derived from requirement text. In addition, we apply different machine learning algorithms and identify one that can be used most successfully for the assessment of natural language requirements. As a result of this section, we obtain trained models for both characteristics *Singular* and *Unambiguous*. In the third section, we focus on evaluating and optimizing the trained models and determine the feature importance. This provides information about which of the features contributes most to the automated quality assessment of a requirement in our research. Finally, the conclusion of the chapter can be found in the last section.

4.1 Methodological foundations

Several related contributions describe static rules for the quality assessment of a requirement. We introduced this type as rule-based systems in chapter 2.2.2. In our research, we aim at an automated classification of the quality characteristics, which is a typical task known from the area of machine learning.

The requirements in our data set are available in text form. Therefore, algorithms from the field of machine learning cannot be applied directly. The "unstructured text" of a requirement must therefore first be transformed into a "structured form". Methods that perform this transformation can be found in the field of natural language processing (NLP). This enables the extraction of numerical information from texts, which can then be used as input for a machine learning algorithm. "Text will be processed and transformed into a numerical representation" (Weiss et al., 2015, p. 1). Consequently, by applying NLP methods, we can transform unstructured requirement text into a set of numerical characteristics (also called features) that can be used and processed by machine learning algorithms.

For a common understanding of the methodological approaches and foundations, we provide a brief overview in the following section. We describe how natural language can be processed and which steps are applied in the preprocessing of textual data. We enhance the overview with a short explanation of feature engineering. Besides, relevant methods from machine learning are presented. The combination of NLP with methods from machine learning is widely known as text mining, which we describe in the last part of this section.

4.1.1 Natural language processing

Textual data cannot be directly analyzed by an algorithm as "text is usually a collection of unstructured documents with no special requirements for composing the documents" (Weiss et al., 2015, p. 2). However, computers can explore a large number of documents and extract relevant information from individual words or paragraphs. There is a huge amount of literature on the analysis of text data. Fundamental work in the field has been done by Manning and Schütze (1999) who present foundations of statistical NLP and helpful tools.

The extraction of information from a text can be described as a restricted form of full natural language understanding (Hotho et al., 2005). Analytical methods are used to process the amounts of data in natural language—based on words and structures—and to uncover patterns in unstructured text. NLP is intended to enable computers to better understand natural language by advanced linguistic analysis. Thus, it can be described as the machine processing of natural language (Kodratoff, 1999). The "ultimate goal of NLP is to mathematically model the understanding and generation of human language" (Johnson et al., 2015, p. 174).

For the extraction of information, the text needs to be transformed into a numerical representation. The transformation can be divided into two steps: preprocessing of text and feature engineering. Depending on which properties of a text are to be used for the representation, it initially requires different preprocessing tasks. Afterward, features can be extracted based on the preprocessed text. In the following, we present both steps.

Preprocessing.

The tasks allocated to the preprocessing of text typically include, among others, tokenization, filtering, and stemming (Uysal & Gunal, 2014; Weiss et al., 2015). In addition, we consider part-of-speech tagging as part of the preprocessing and provide a short description of these tasks in the following.

- **Tokenization.** The first step usually involves the tokenization of a text and the "split into a stream of words by removing all punctuation marks and by replacing tabs and other non-text characters by single white spaces" (Hotho et al., 2005, p. 6). Tokenization describes the decomposition and segmentation of text into its words and enables the processing and analysis of each word (Webster & Kit, 1992). The Natural Language Toolkit ($NLTK$)[1] provides a tokenizer function that we apply for our data set of assessed requirements (Bird et al., 2009). As an example, the German natural language requirement "Das Fahrzeug muss deutsche Verkehrszeichen erkennen" turns into ['Das', 'Fahrzeug', 'muss', 'deutsche', 'Verkehrszeichen', 'erkennen'] after the tokenization step. The result is an array with tokens consisting of text elements.
- **Filtering.** Specific elements of a text can be removed by filtering. A well-known approach is called "stop word removal". It removes words that have little or no information content and are not discriminatory or specific for differentiation of requirements (Silva & Ribeiro, 2003). This includes words such as auxiliary verbs, conjunctions, and articles (Ikonomakis et al., 2005). A

[1] $NLTK$ is a sophisticated open-source toolkit for the programming language Python to process and analyze natural language.

stop word removal can also take into account words that are very common or very rare in the documents (Frakes & Baeza-Yates, 1992). For the requirement "Das Fahrzeug muss deutsche Verkehrszeichen erkennen", a stop word removal results in "Fahrzeug deutsche Verkehrszeichen erkennen". Other filtering approaches are described by "number removal" and "punctuation removal".

- **Stemming.** The process of tracing words back to their word stem or root form is called stemming (Porter, 1980). It is described as "actions to remove meaningless differences between words" (Arellano et al., 2015, p. 232), where several forms of the same word are grouped into a single word. Usually, stemming is done after the tokenization step. For a better understanding, we apply the stemming process directly to our requirement "Das Fahrzeug muss deutsche Verkehrszeichen erkennen", that results in "das fahrzeug muss deutsch verkehrszeich erkenn"[2].

- **Part-of-speech (POS) tagging.** The annotation of words is realized by a part-of-speech tagger, that identifies grammatical types of each word in a text (e.g. adjective, verb, noun, determiner). For demonstration purposes, we apply the Stanford POS tagger of Toutanova et al. (2003), which is known as a statistical probabilistic tagger. For our exemplary requirement "Das Fahrzeug muss deutsche Verkehrszeichen erkennen" we get the result: [determiner, noun, verb, adjective, noun, verb]. Each tag represents a part of speech or a lexical category. For the implementation in section 4.2, we use built-in functions from $spaCy$[3] for this preprocessing step (spaCy, 2020).

The described tasks create the basis for further processing. In general, the words of a requirement data set are described as vocabulary that can be considered as features. Therefore, features are regarded as characteristics or distinct properties of a text, which can contain significant information for a machine learning algorithm. "Selecting relevant features and deciding how to encode them for a learning method can have an enormous impact on the learning method's ability to extract a good model" (Bird et al., 2009, p. 224). In the following, we describe how features for textual data can be extracted after the preprocessing tasks are conducted and present different approaches.

[2]We use the Snowball Stemmer from $NLTK$, that provides an improved approach based on the algorithms developed by Porter (1980). It also converts all uppercase letters to lowercase letters.

[3]$spaCy$ is a Python package and provides an extensive set of functions for NLP.

Feature engineering.

Feature engineering describes the process of identifying and generating features from a text and enables the transformation into a numerical representation (Mitkov, 2004; Zheng & Casari, 2018). This process creates a vector consisting of features that are necessary for an algorithm to learn and is therefore called "vectorizing". The result of the transformation is a vector space representation of text (Ikonomakis et al., 2005; Jurafsky & Martin, 2014; Manning & Schütze, 1999). "In language processing, the vectors [...] are derived from textual data, in order to reflect various linguistic properties of the text" (Goldberg & Hirst, 2017, p. 65).

An approach that uses vectorization is called "bag-of-words" (Jurafsky & Martin, 2014; Zhang et al., 2010). A bag-of-words simply represents a requirement as individual words, for which two approaches exist. First, a binary vector is created, with a value of "1" indicating the presence of a word in a requirement and a value of "0" indicating that the requirement does not contain it. In the second approach, the frequency of occurring words is represented. As words are "chosen as the basic representational unit" (Joachims, 2002, p. 33), it counts the presence and detects the absence of a word in a requirement. Thus, it generates features from the text, where each feature is an attribute that can be measured (Søgaard, 2013). The approach is generally described as the most fundamental model of text (Weiss et al., 2015).

Because of the simplicity of the bag-of-words representation, it is often used for feature generation (Joachims, 2002; Nassirtoussi et al., 2014). Even though a bag-of-words does not take into account the position, order, and the context of words in a text, remarkable results can be achieved in some applications (De Vries et al., 2018; Purda & Skillicorn, 2015). Table 4.1 presents part of the bag-of-words for the requirement "Das Fahrzeug muss deutsche Verkehrszeichen erkennen".[4]

Tab. 4.1.: Bag-of-words with a binary vector for a natural language requirement.

Object ID	Das	Fahrzeug	muss	deutsche	Verkehrszeichen	(...)
ID_01	1	1	1	1	1	(...)

A more sophisticated method can be used to weight features according to their relative frequency distribution. The procedure, originally proposed by Salton and Buckley (1988), is called "term frequency-inverse document frequency" (tf-idf): "Tf-idf is a weighting of the importance of a term to a document in a corpus" (Huq et al., 2018, p. 107). As a result, words that are often found in natural language

[4]For better illustration, preprocessing steps from the previous section have been omitted in this example.

receive a lower weighting in comparison. In our research, we apply tf-idf for the implementation in section 4.2, as we expect more from the consideration of a weighted term than the simple reflection of the presence or absence of words from a bag-of-words approach (Ramos, 2003).

In addition to the presented approaches, feature engineering can also include the definition and implementation of individual features. These include features that are specific and relevant to the particular use case. In our research, we additionally describe two types of features that are common in related research. One type counts the occurrence of certain characteristics of the text (e.g. number of sentences). The other type focuses on whether a feature is present. This can be used, for example, to identify whether a requirement is written in passive language. A detailed description of these features can be found in section 4.2.1.

In summary, with different preprocessing tasks and feature engineering, a simple vector space representation can be realized, which is exemplarily illustrated in figure 4.1. In general, "Stemming" and "Filtering" can also be carried out optionally, depending on the desired type of representation.

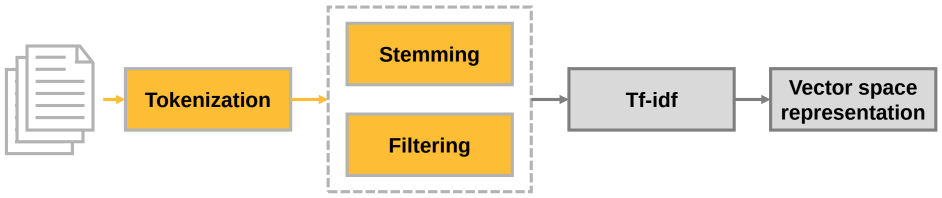


Fig. 4.1.: Preprocessing tasks (tokenization, stemming, and filtering) and feature engineering (tf-idf) for a vector space representation of textual data.

In this section, we introduced several tasks for preprocessing text and described different approaches for feature engineering. The resulting numerical representation of textual data enables the next step. In the following section, we look at the possibilities of using machine learning methods and particularly consider approaches that use labeled data.

4.1.2 Machine learning

Generally, a machine learning process trains an algorithm that tries to learn the logic of complex decisions. Thus, algorithms from machine learning are used to automate tasks that have previously been performed by a human. For our research, a complex decision refers to the quality assessment of a natural language requirement according to the characteristics *Singular* and *Unambiguous*. In the following, we shortly describe relevant terms and focus especially on methods for solving a classification problem.

Machine learning can be located at the intersection of statistics and computer science (Mitchell, 2006). In the field of statistics, machine learning is often seen as the application and implementation of statistical learning (James et al., 2013). For computer science, machine learning aims to enable computers to act without being explicitly programmed and describes the development of efficient algorithms that can solve various types of problems.

In our research, we apply algorithms from supervised learning. Such algorithms "learn" from labeled data by using input and output data. This enables an automated classification of unknown (i.e. unlabeled) data. For the automated assessment of requirements quality, we have input data by various features and output data by the expert assessment of the quality characteristics *Singular* and *Unambiguous* for each requirement. Therefore, we are able to follow supervised machine learning.

Classification and regression.

Supervised machine learning is divided into classification and regression problems. Both approaches have in common that they use labeled data sets. The distinction between a classification and a regression can be defined, in simple terms, by the data type of the output variable. A typical regression problem is the calculation of a purchase price for a house based on known factors (e.g. size, location, age). Another example is the determination of income by factors such as age and education. As a result of a regression problem, the output variable can assume a continuous value. In a classification problem, the process of determining a class is described. The value of the output variable is derived from a discrete set of values. As in our research, we are not aiming to predict continuous values, we define the five points of the ordinal scale from the manual assessment of requirements quality as discrete classes.

Aggarwal and Zhai (2012b) describe the problem of classification in terms of a training set in which each element is labeled with a class value. Thus, a classification approach generally requires labeled data and the division of the data set into a training set and a test set. The training process creates a classification model that uses the training data. The test set is used to measure the performance of a model. A model with sufficient performance can then be used to predict class values for elements that previously had no label.

Various approaches allow the data set to be divided into a training set and a test set. These typically include the percentage split and cross-validation. With a percentage split, a defined percentage part (usually between 10 and 40 percent) of the data set is put aside as a test set. The remaining part serves for training a machine learning model. The test set is then used for the evaluation. In cross-validation, the data set is randomly divided into a number of "folds" of equal size. One fold serves as a validation set, the others as the training set. "This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining k-1 folds" (James et al., 2013, p.181). The process continues until each fold is defined once as a validation set. Both approaches follow the principle of separating the data set into a test (or validation) set and a training set.

Performance evaluation.

Supervised learning describes the employment of an algorithm to transform a mapping function from input variables (x) to output variables (y). The goal of supervised learning is to enable the most accurate approximation of this function. It is particularly important to consider that for new and unknown input data (x_n), the corresponding output variable (y_n) can be determined with a certain performance. The performance of a trained model is assessed by using the test set. Typically, the output values from the test set (true labels)—these are the labels from the expert assessments in our research—are compared with the output values of the classification (predicted labels).

A confusion matrix illustrates these results and is shown in figure 4.2. "TP" refers to "True Positive" and describes the number of correctly classified labels from the positive class. "TN" means "True Negative" and describes it equally for the negative class. "FP" stands for "False Positive" and is the number of data incorrectly classified for the positive class. "FN" is short for "False Negative" and describes the number of data incorrectly classified for the negative class.

The proportion of correctly classified data to the total number of classified data is called "accuracy" and provides an initial performance measure for the model. TP and TN are added and divided by the total number of classified data.

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{4.1}$$

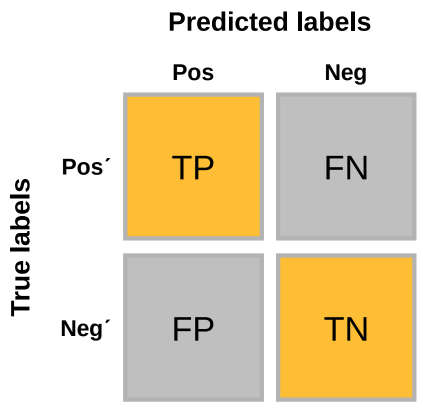


Fig. 4.2.: Confusion matrix with true (yellow) and false (grey) classifications.

It is not unusual, however, that in classification tasks the classes to be determined are not evenly distributed (referred to as imbalanced classes). Therefore, the result for "accuracy" is not always meaningful (Kuhn & Johnson, 2013). For this reason, when determining the performance of a supervised learning model, additional statistical measures are used. Mostly, the performance is evaluated in terms of "precision", "recall" and "F_β-score" (Chawla et al., 2002; Powers, 2011). We briefly describe these measures in the following.

"Precision" describes the ratio of TP (i.e. the number of correctly classified labels of the positive class) to the total number of classified positive (TP + FP). The result can take values between 0 and 1. A high value indicates a solid classification of the positive class.

$$precision = \frac{TP}{TP + FP} \tag{4.2}$$

The ratio of TP to the total number of all true positive labels (TP + FN) is described by the measure "recall". This measure can also take values between 0 and 1. A high value implies that positive labels were mostly correctly classified by the model.

$$recall = \frac{TP}{TP + FN} \tag{4.3}$$

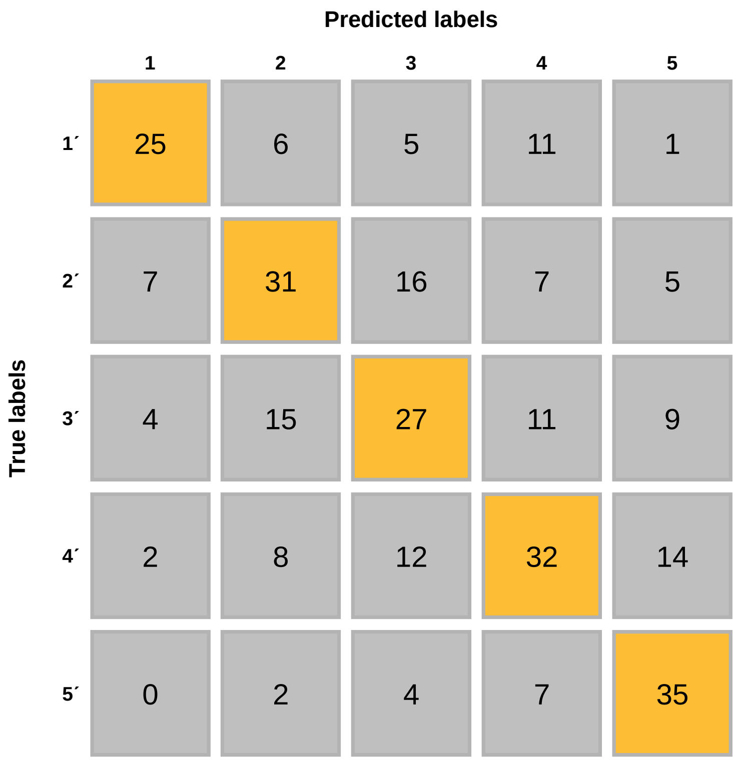
Finally, for the performance determination of a model, the F_β-score can be used to enable a weighted compromise between "precision" and "recall" (Van Rijsbergen, 1979). The result can take values between 0 and 1 as well.

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{4.4}$$

Since "precision" and "recall" each concentrate on different views, the F_β-score offers a good approach to consider both aspects (Powers, 2011). The parameter β enables to weight the two measures differently in the overall consideration. For $\beta<1$, the focus is on "precision", whereas for $\beta>1$, the focus is on "recall". Depending on which measure is more relevant for the performance evaluation of a classification model, the parameter can be adjusted. Often, the value 1 is used for β, so it is possible to calculate "the harmonic mean between precision and recall" (Jimenez et al., 2009, p. 566).

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4.5}$$

For a multi-class classification problem, the determination of the performance measures is different from a binary classification problem. In figure 4.3, we show a confusion matrix for a five-class classification problem with exemplary values.


Predicted labels
True labels

	1	2	3	4	5
1´	25	6	5	11	1
2´	7	31	16	7	5
3´	4	15	27	11	9
4´	2	8	12	32	14
5´	0	2	4	7	35

Fig. 4.3.: Exemplary confusion matrix for a five-class classification problem.

In the case of a multi-class classification problem, the values for TP, FP, FN, and TN are determined for each class. One class is considered a positive class, while the other classes are treated as negative classes. We demonstrate the determination of the values by using class "1" as an example in the following.

For class "1", TP_1 is the number of correctly classified labels by the model. In the confusion matrix, the value "25" can be determined. FP for class "1" (FP_1) is the number of labels that the model has incorrectly assigned to the class "1". This value can be found in the sum of the remaining values of the first column (7+4+2+0=13). Similarly, the value for FN_1 is derived from the results that belong to class "1" but have been assigned by the model to other classes. This corresponds to the following values in the first row: 6+5+11+1=23. Finally, in simple terms, TN_1 contains the sum of the remaining values in the confusion matrix (equal to 235).

Accuracy for class "1" ($accuracy_1$) results from the sum of TP_1 and TN_1. This sum is divided by the sum of TP_1, FP_1, FN_1, and TN_1. Thus, the result for $accuarcy_1$ is .8784 for the exemplary values in figure 4.3. The accuracy for the multi-class classification problem ($accuracy_{multi}$ = .8027) is then determined by the average accuracy of the classes (Sokolova & Lapalme, 2009).

$$accuracy_{multi} = \frac{\sum_{i=1}^{m} \frac{TP_i+TN_i}{TP_i+FP_i+FN_i+TN_i}}{m} \tag{4.6}$$

For the determination of the performance measures "precision", "recall", and "F$_1$-score", different approaches exist in a multi-class problem (Özgür et al., 2005; Y. Yang & Liu, 1999). Sokolova and Lapalme (2009) describe two alternatives as "macro-averaging" and "micro-averaging". With macro-averaging, the performance measures are determined by the average of the performance measures of the individual classes (analogous to the calculation of $accuracy_{multi}$). For example, precision for class "1" is calculated as TP_1 divided by ($TP_1 + FP_1$) and results in the value .6579, whereas for recall the value .5208 can be obtained. For the determination of the F$_1$-score, we follow the recommendations of Lipton et al. (2014) and Opitz and Burst (2019) and first calculate the F$_1$-score per class. Then the average of the F$_1$-score of all classes is determined. With the values for each class we can calculate "macro precision", "macro recall" and "macro F$_1$-score".

$$macro\ precision = \frac{\sum_{i=1}^{m} \frac{TP_i}{TP_i+FP_i}}{m} \tag{4.7}$$

$$macro\ recall = \frac{\sum_{i=1}^{m} \frac{TP_i}{TP_i+FN_i}}{m} \tag{4.8}$$

$$macro\ F_1 = \frac{\sum_{i=1}^{m} F_{1\,i}}{m} \tag{4.9}$$

Micro-averaging, on the other hand, considers the overall numbers of TP, FP, FN, and TN and calculates the performance measures of a classifier over the total sum of the values (Wang & Chiang, 2007). The F_1-score is determined by calculating the harmonic mean between "micro precision" and "micro recall".

$$micro\ precision = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m} (TP_i + FP_i)} \tag{4.10}$$

$$micro\ recall = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m} (TP_i + FN_i)} \tag{4.11}$$

$$micro\ F_1 = 2 \cdot \frac{micro\ precision \cdot micro\ recall}{micro\ precision + micro\ recall} \tag{4.12}$$

With macro-averaging, all classes are treated equally, since the performance measures are calculated individually for each class. On the other hand, micro-averaging is considered as an average in all classes. In general, the choice of measures to assess the performance of a classifier depends on various factors and is individual for each application. A recommendation as to which metric to choose is therefore not possible to provide. For that reason, the performance measure often considers the "cost of misclassification". Related to our research, we want to consider the following issues in equal parts, since these are equally important for the results of an automated quality assessment of natural language requirements: The percentage of correctly classified labels per class and the percentage of correctly classified labels across all classes. Therefore, a good measure for our application gives equal weight to recall and precision and also considers equal weight for all classes. We identify this measure in the *macro* F_1-score.

Classification algorithms.

For a classification problem, several algorithms from machine learning exist, some of which have been shown to work better for text classification tasks. Several research papers present the use of "Support Vector Machines" (SVM) to analyze and classify text (Bloehdorn & Moschitti, 2007; Cristianini & Shawe-Taylor, 2000; Ma et al., 2011; Marseguerra, 2014; Muller et al., 2001). In particular, Joachims (1998) explores and identifies the benefits of SVM for text categorization. On the other hand, some researchers tend to use "Naïve Bayesian Classification" as the most popular technique for text classification (Casamayor et al., 2010). Aggarwal and Zhai (2012b) and Miner et al. (2012) mention algorithms like Decision Trees, Support Vector Machines, and Bayes classifier, among others, which are frequently applied for text classification. Thus, we briefly introduce the algorithms which we also use in the implementation in section 4.2.

"A considerable amount of emphasis has been placed on [...] SVM classifiers, being particularly suited to the characteristics of text data" (Aggarwal & Zhai, 2012b, p. 213). The idea of an SVM is the differentiation between classes by an optimal hyperplane based on a set of feature vectors in the vector space. The data points closest to the hyperplane are called support vectors. SVMs are vector-based and therefore require features that are encoded numerically within the vectors. When classifying unlabeled data, the feature vector determines on which side of the hyperplane the data has to be assigned.

Bayes classifiers are algorithms that have been proven for text classification problems as well. They are based on the Bayes theorem and use a probabilistic model of text for the estimation of the probability that a document is in one of the given classes. A commonly used implementation is called Naïve Bayes (Lewis & Ringuette, 1994; A. McCallum & Nigam, 1998). It is a classifier that is based on the assumption of independent features. Even though in most cases this assumption does not apply, the use of the Naïve Bayes algorithm shows remarkable results.

Besides, there is a category of tree-based approaches (Quinlan, 1993; Quinlan, 1986). As an example, a Decision Tree describes an algorithm that uses a "divide-and-conquer" strategy and subdivides data by applying rules based on text features to make a decision (Quinlan, 1996). It is "a hierarchical decomposition of the (training) data space, in which a predicate or a condition on the attribute value is used in order to divide the data space hierarchically" (Aggarwal & Zhai, 2012b, p. 176). In general, tree-based methods are less susceptible to unadjusted data and are only slightly influenced by outliers.

The division or splitting of data in a tree is performed recursively until a node contains a minimum number of data points, or until a node describes only one class. The former could result in the predicted class of a node being determined by the most common class of the node (also called majority vote). The second describes a node for which no further subdivision of the data is necessary as this part is uniquely assigned to a class. These nodes are called "leaf nodes" (Quinlan, 1993). Other nodes are called "decision nodes", which result in a branch and subtree for each possible outcome of a decision (Quinlan, 1993).

A tree is built iteratively, starting with the root node. The aim is to divide the training set into homogeneous areas in which a class is clearly described by the features. In the case of text data, the features for the decision nodes are usually based on terms in the text (Aggarwal & Zhai, 2012b). This can include the features resulting from the application of tf-idf. In our research, a decision in a node can also be based on other features that are described later in section 4.2.1. The split is decided by determining different parameters. One approach describes the determination of "Gini impurity", which calculates the probability from a randomly chosen element to be classified wrong. The Gini impurity is defined as follows.

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2 \tag{4.13}$$

The ratio of class k samples among all training samples in node i is defined by $p_{i,k}$. The following calculation is used to determine the Gini impurity of the first node in the case of five quality classes:

$$G_1 = 1 - p_{1,1}^2 - p_{1,2}^2 - p_{1,3}^2 - p_{1,4}^2 - p_{1,5}^2 \tag{4.14}$$

In this example, $p_{1,1}$ is the probability of choosing samples from class 1 among all the training samples in the first node.

"The decision tree splitting criterion is based on choosing the attribute with the lowest Gini impurity index of the split" (Mather & Tso, 2016, p. 188). The determination of the optimal decision boundary of a node is conducted by testing all possible boundaries. When the value 0 occurs at a node for the Gini impurity, it only contains elements of one class. A further subdivision is then not necessary, as additional information gain is not possible.

Different implementations of the Decision Tree classifier exist. Often these are based on the C4.5 taxonomy of Quinlan (1986), which allows the use of discrete and continuous data sets (Sathyadevan & Nair, 2015). CART (Classification and Regression Trees) follows a similar approach, but only binary trees can be created, while C4.5 also allows non-binary decisions. In section 4.2, we use the implementation of the Decision Tree classifier from the Python library $scikit-learn$, which is mainly based on the CART algorithm.

More complicated methods often promise significant advantages over the simple model of a Decision Tree, especially with regard to predicting power. These include so-called ensemble classifiers, where the classification is achieved by a combination of classifiers and a voting mechanism. In general, this type of classifiers leads to much more robust results compared to a single classifier and is also frequently used for text classification problems (Aggarwal & Zhai, 2012b; Bao & Ishii, 2002; Bi et al., 2004).

The Random Forest algorithm describes the simplest form of an ensemble classifier. It applies algorithms of the same type and is therefore called a bagging classifier (Breiman, 1996). In general, bagging algorithms create different models by training the same classifier on different subsets of the training set. "The idea in bagging is to pick bootstrap samples (samples with replacement) from the underlying collection, and train the classifiers in these samples. The classification results from these different samples are then combined together in order to yield the final result" (Aggarwal & Zhai, 2012b, p. 211). Bagging algorithms can be used with any type of classifier, although they are often associated with Decision Trees.

Thus, Random Forest is a bagging algorithm that uses an ensemble of several random Decision Trees (Breiman, 2001; Hastie et al., 2009). The randomness is expressed in two ways: first, each Decision Tree is based on a random sample of the training set; second, at each node of a Decision Tree, a random subset of all available features is selected to determine the optimal split. Every Decision Tree creates an output value. The majority vote of these values results in the output value of a Random Forest algorithm and usually provides an increased accuracy compared to a single Decision Tree.

In general, a tree-based approach can be implemented in such a way that the algorithm does not stop until all data is uniquely assigned to a class. This describes a Gini impurity of 0, where further splitting is not necessary and reasonable. At a certain decision depth, however, a tree can tend to refer too much to the training set. The model is then able to classify the data in the training set but has difficulties with unknown data in the future. This issue is called "overfitting" and is a general problem when training machine learning models.

A possible approach to identify overfitting may be the application of k-fold cross-validation, as described in section 4.1.2. "Probably the simplest and most widely used method for estimating prediction error is cross-validation" (Hastie et al., 2009, p. 241). As the data set is divided into k mutually exclusive subsets with approximately the same size and the process is performed iteratively for the number of k folds, the output shows a scattering of the results. This scattering provides an indication of possible overfitting of a trained model and could help to ensure that the model is generalizable to unknown data. It is well known, that bagging methods are well suited and "designed to reduce the model overfitting error which arises during the learning process" (Aggarwal & Zhai, 2012b, p. 211). We refer to this issue in the implementation in section 4.2.

4.1.3 Text mining

The previous section describes three types of algorithms from supervised learning that follow different approaches for classification problems. Common to all supervised learning approaches is that training data, labeled data, and a numerical representation are required.

In figure 4.4, we show the learning process and the classification process[5]. In the learning process, we apply methods from natural language processing to generate a numerical representation of text by various features. Together with labels, the representation can be used as input to train a model. In the classification process, unknown (i.e. unlabeled) data is transformed into a numerical representation that is based on the same features. The classifier can then be used to classify this data. In our research, labels (i.e. assessment scores) can be determined for the quality characteristics of a natural language requirement that have not been assessed by experts.

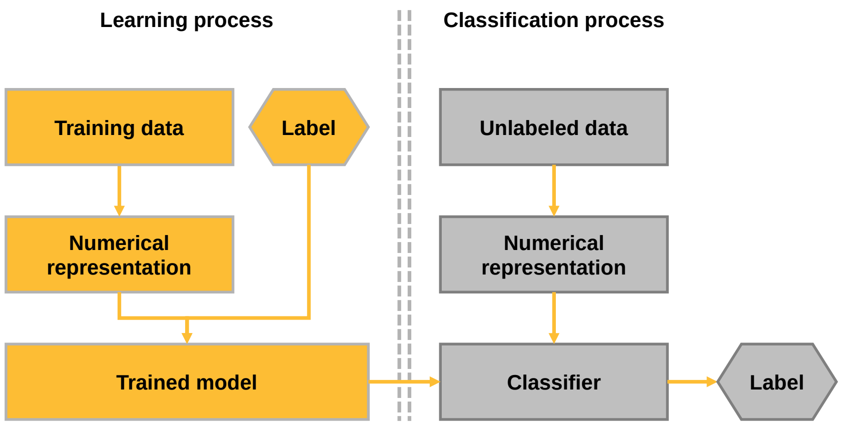


Fig. 4.4.: Learning and classification process for text mining problems.

The combination of natural language processing and machine learning techniques is generally referred to as "Text Mining". Text mining is increasingly used in companies due to the increase of information and data in an unstructured form. "One of the main themes supporting text mining is the transformation of text into numerical data, so although the initial presentation is different, at some intermediate stage, the data move into a classical data-mining encoding. The unstructured data become structured" (Weiss et al., 2015, p. 3).

The activities assigned to text mining are manifold and result in various definitions of the term (Mehler & Wolff, 2005). In general, it is referred to as the automated

[5]For simplicity, we have skipped the usual implementation of determining model performance with test data in this figure.

processing and analysis of text (Fleuren & Alkema, 2015) and is defined as "the discovery by computer of new, previously unknown information, by automatically extracting and relating information from different written resources" (Hearst, 2003). Felden (2006) details this view and describes text mining as an approach to determine knowledge in text documents by machine processes that use classification approaches, among other things. This is where our research for the automated quality assessment of natural language requirements is oriented.

Miner et al. (2012) present seven practice areas related to text mining: "Information Extraction", "Natural Language Processing", "Concept Extraction", "Web Mining", "Information Retrieval", "Document Clustering", and "Document Classification". The practice areas overlap with six adjacent research areas: "AI and Machine Learning", "Statistics", "Computational Linguistics", "Library and Information Sciences", "Databases", and "Data Mining". We refrain from describing all application areas and research fields in detail here. As our focus is on the linguistic structure of the requirements, which can be determined using features, we apply techniques from the field of natural language processing. Thus, we consider text mining in our research as the application of the practice area "Natural Language Processing" in combination with algorithms from the research area "AI and Machine Learning" for the automated assessment of requirements quality.

4.2 Implementation

From the conclusion of chapter 3, we are able to call the data set of the assessed requirements for the characteristics *Singular* and *Unambiguous* reliable. Building on the methodological foundations from the previous section, we describe the implementation of the automated assessment of requirements quality based on this data set in the following.

First, we present the preprocessing tasks for the data set and show the results of the feature engineering for the natural language requirements. With the numerical representation at hand, we are then able to apply several machine learning algorithms. We present configurations, with which we test these algorithms and that involve the usage of different feature sets and sampling techniques. The results enable us to identify a classification algorithm and to train a model that performs the most accurate quality assessment of requirements for each of the characteristics *Singular* and *Unambiguous*.

4.2.1 Natural language processing

For preprocessing the data set, we use the programming language Python and apply functions from different libraries, which we refer to in the following. We also describe an individual preprocessing for the creation of two distinct feature sets.

Tf-idf features (feature set 1).

For the extraction of tf-idf features, first tokenization is required to separate the requirement text into its elements. We use a tokenizer function provided by $NLTK$. Afterward, we apply several filtering processes to delete content in the requirement text that have no information content and are not necessary for further steps. This includes stop word removal, number removal, and punctuation removal. For stop word removal, we consider a predefined list from $NLTK$ consisting of 229 German stop words. Furthermore, in the context of the tf-idf features, we estimate the presence or frequency of numbers and punctuation marks as text elements without information content. Thus, we remove numbers and punctuation marks in the requirements for the extraction of our tf-idf features. The final step comprises stemming, i.e. reducing the words of the requirement text to their root form. We apply the "snowball stemmer" provided by $NLTK$, which is based on the widely used and established algorithm developed by Porter (1980)[6].

In a separate comprehensive study, we used all possible combinations of the presented preprocessing tasks and applied them for the algorithm evaluation. However, no improvement in performance could be observed. Therefore, we take into account all the preprocessing tasks presented in table 4.2.

Tab. 4.2.: Preprocessing tasks for the extraction of tf-idf features.

Preprocessing task	**Description**
Tokenization	Separation of requirement text into tokens of single words.
Stop word removal	Removing stop words from a requirement with a stop word list.
Number removal	Removing numbers in the requirement text.
Punctuation removal	Removing punctuation marks in the requirement text.
Stemming	Reduction of a word to its stem by using "snowball stemmer".

After the preprocessing tasks, we apply the function "TfidfVectorizer" from the Python library $scikit-learn$. The result of the function describes a set of tf-idf features consisting of unigrams and bigrams of the requirements text. We define this set as feature set 1.

[6]The "snowball stemmer" also returns all words in a lower case.

Quantity and binary features (feature set 2).

The extraction of tf-idf features from the requirements text is a first step. Various publications also indicate that an extended set of appropriate features can significantly improve the results of a machine learning algorithm (Bird et al., 2009; Johnson et al., 2015; Nassirtoussi et al., 2014). "The extent that a specification is fundamental is subjective based on the assessor's knowledge of the application domain, whereas certain phrases and sentence structures that indicate a complicated statement can be objectively identified" (Wilson et al., 1997, p. 3). For that reason, we build on additional features that have already been proposed and described in the literature.

Bird et al. (2009) state that a decent performance of an algorithm can be obtained "by using a fairly simple and obvious set of features" (Bird et al., 2009, p. 224). Based on the findings from our literature review in section 2.2.2, we can aggregate such features for the quality assessment of natural language requirements. We briefly refer to the most relevant publications before we describe our extended feature set in detail.

The derivation of features from requirements texts is manifold and considers different linguistic areas. "Requirements analysis is concerned with the question if textual [...] requirements are clearly and concisely formulated. Accordingly, features are used that identify weak words, subjunctives, or passive voice, which are all indicators for ambiguous language" (Fromm et al., 2019, p. 3). Parra et al. (2015) propose "metrics" based on a framework that describes features for measuring quality in natural language requirements. The framework is built upon aggregated indicators from different sources and presents the most commonly used features for the quality assessment of requirements (Génova et al., 2013).

Wilson et al. (1997) describe nine categories of "quality indicators" (imperatives, continuances, directives, options, weak phrases, size, text structure, specification depth, readability statistics) and propose specific approaches for measurement. We focus on such categories that can be applied to individual requirements and include them in our feature list. Wiegers (2003) identifies six best practices to be considered when specifying requirements. From some of them, we can derive features for our research. Regarding the best practice "Use active voice", for example, we develop a detection of passive voice in a requirement. For some best practices, on the other hand, features cannot be uniquely identified (e.g. "Write complete sentences that have proper spelling and grammar").

Besides, Soeken et al. (2014) define a set of ten rules extracted from several publications (Alexander & Stevens, 2002; Wilson et al., 1997). From some of these rules, we are able to derive features. Others, however, are not defined in a way that proper features can be identified (e.g. "R10. Define verifiable criteria"). Some contributions deal with requirement language criteria[7] extracted from ISO 29148 and use POS tagging and dictionaries to identify these criteria in a requirement (Femmer et al., 2017). We are able to derive additional text features. In addition to the research contributions presented, we also have access to several best practices from major companies in the automotive industry. This enables us to identify additional features from industrial practice.

The extracted features from the literature and best practices may expand a numerical representation based on tf-idf and can also be of a very different nature. Thus, we aggregate the extracted features into two categories.

1. **Quantity features.** Features of the first category focus on counting textual characteristics (e.g. words or sentences) of a requirement and partly require a POS tagging or specific dictionaries. Some of the features are also combined for a statistical readability measure.

2. **Binary features.** Features of the second category identify the presence of defined characteristics of a requirement (e.g. template structure or passive voice). If a characteristic is present, the value 1 is assigned to the feature, otherwise the value 0.

An overview of the quantity and binary features is presented in table 4.3. We also describe each feature in the following.

Tab. 4.3.: Overview of quantity features and binary features.

Quantity features	**Binary features**
No. of words (NOW)	Impl. obligations (IMO)
No. of sentences (NOS)	Options (OPT)
Avg. no. of words per sentence (AWS)	Template structure (TES)
Avg. no. of syllables per word (ASW)	Passive voice (PAV)
Avg. no. of punctuation marks per sentence (APS)	
No. of conjunctions (NOC)	
No. of verbs (NOV)	
No. of pronouns (NOP)	
No. of implementation terms (NOI)	
No. of weak words (NWW)	
Readability index (REI)	

[7]Requirement language criteria from ISO 29148 describe terms that should be avoided. This includes, among others, superlatives, subjective language, and vague pronouns.

Quantity features.

- **Number of words (NOW).** We use the simplest way to measure the size of a requirement and count the number of words (Fantechi et al., 2003; Génova et al., 2013). "Words are defined as non-whitespace strings enclosed by whitespace characters" (Joachims, 2002, p. 33). For this purpose, we use a tokenizer function available in Python, that transfers the words of a text into a list, of which we count the number of elements.

- **Number of sentences (NOS).** Besides the number of words, we also count the number of sentences in a requirement with a built-in function from $NLTK$[8]. The feature gives us a further indication of the size and structure of a requirement (Parra et al., 2018).

- **Average number of words per sentence (AWS).** Based on the two previous features, we count the average number of words per sentence and divide the number of words (NOW) by the number of sentences (NOS). With this feature proposed by Fantechi et al. (2003), we can determine the average sentence length for a requirement.

- **Average number of syllables per word (ASW).** Texts in a technical context can contain longer and more complex words (Kasser et al., 2006). By counting the average number of syllables per word, we can analyze whether it affects the quality of a requirement. For the determination, we implement a function that returns the number of syllables per word by counting vowels (e.g. a, e, i, o, u) and diphthongs (e.g. au, ei, eu).

- **Average number of punctuation marks per sentence (APS).**[9] Génova et al. (2013) suggest counting the number of punctuation marks in a requirement. "[A]dequate punctuation is essential to sentence understandability, and punctuation excess makes the text more difficult to read, maybe pointing out that the sentence must be split in two or more sentences" (Génova et al., 2013, p. 30). We implement a function that determines the average number of punctuation marks per sentence in a requirement. The POS tagger from $spaCy$ identifies not only the typical elements of a text (e.g. noun, verb) but also assigns a tag for punctuation marks. We first count the related tags and then divide the result by the number of sentences in a requirement.

[8]The function from NLTK has been proven through manual analysis of different samples to provide the most reliable results compared to functions from other Python libraries.

[9]For the extraction of tf-idf features, we removed punctuation marks. For this feature, we count punctuation marks in the relation to the number of sentences of a requirement, following the suggestion of some researchers.

- **Number of conjunctions (NOC).** Several publications address the use of conjunctions in requirements and their potential impact on quality (Ghosh et al., 2016; Huertas & Juárez-Ramírez, 2013; H. Yang et al., 2012). Parra et al. (2015) indicate that a high number of connective terms may result in multiple needs being specified in a requirement. Kasser et al. (2006) also claim that the occurrence of words such as "and" ("und") and "or" ("oder") influences whether an individual requirement can be clearly identified. Conjunction terms are also referred to as "multiplicity-revealing words" that affect the quality characteristics of a requirement (Berry et al., 2006; Fabbrini et al., 2001). We consider this feature in our research and implement a function that counts how many German conjunctions (e.g. "als", "bis", "da", "dadurch", "damit", "nachdem", "obwohl", "obgleich", "oder", "und", "während", "weil") exist in a requirement.

- **Number of verbs (NOV).** Several best practices from the automotive industry claim, that the number of verbs provides information about how many needs are contained in a requirement. Thus, we determine the number of verbs in a requirement as a feature. We refer to the POS tag for a verb and implement a function that counts these tags in a requirement.

- **Number of pronouns (NOP).** A pronoun is usually used as a substitute for a noun to prevent repetition in a sentence. However, in the context of natural language requirements, it can be an indicator of ambiguity (Kasser et al., 2006). "Even with a grammatically impeccable usage, [pronouns] increase the risk of imprecisions and ambiguities in texts of technical character" (Génova et al., 2013, p. 31). Thus, Fabbrini et al. (2001), Berry et al. (2006) and Génova et al. (2013) suggest determining the number of pronouns. In the specification of German requirements, personal pronouns (including "es") and demonstrative pronouns (including "diese") may occur. We determine the number of pronouns by counting the corresponding tag from the POS tagger.

- **Number of implementation terms (NOI).** A requirement should describe a necessity and not express a solution—specify "what" has to be implemented and not "how" (Parra et al., 2015). For this reason, Génova et al. (2013) present "design terms", which give an indication of words in a requirement when a solution is specified instead of a necessity. We aggregate the German translated words (e.g. "Methode", "Parameter", "Datenbank", "Anwendung", "Programm") into a dictionary to determine whether design or technology-related terms exist and count the number of occurrences in a requirement.

- **Number of weak words (NWW).** The impact of "weak words" on the quality of requirements is discussed in different research contributions (Kovitz, 1999; Krisch, 2014; Krisch & Houdek, 2015) and "can indeed be a problem in requirements documents" (Krisch & Houdek, 2015, p. 349). Weak words are also called vague terms and indicate that a text is most likely imprecise (Krisch, 2014; Wiegers, 2003). Looking at the following example of a software requirement "The vehicle must brake in time where necessary", it can be seen that the words "in time" and "necessary" allow a broader interpretation. With the aim of a well-specified requirement, it is important to avoid such words. We have compiled an extensive list of weak words from different sources and stored them in a dictionary. As a primary source, we have obtained several lists from three major automotive companies. The lists are not domain-specific but are subject to general guidelines to avoid weak words. Also, we included input from researchers, such as Krisch and Houdek (2015), in which the authors provide an internal list of weak words from the German automotive company Daimler AG. Our dictionary is supplemented by weak words presented in several publications[10]. Wilson et al. (1997) describe terms, "that are apt to cause uncertainty and leave room for multiple interpretations" (Wilson et al., 1997, p. 4). In Génova et al. (2013), weak words are defined as "imprecise terms" and are summarized by the kind of imprecision that characterizes them (e.g. "enough", "sufficient", "approximately", "several", "to be defined", "adaptable", "extensible", "easy", "familiar", "safe"). Parra et al. (2015) consider "incompleteness expressions" (e.g. "etc.", "not limited to") which affect the clear scope of a requirement. Fabbrini et al. (2001) propose several weak words according to different categories (vagueness, subjectivity, and optionality). Femmer et al. (2017) suggest a list of terms to determine so-called "requirement smells" (e.g. "best", "most", "highest"). The analysis and aggregation of different sources result in a dictionary of 961 weak words[11]. Tokenization allows us to identify whether weak words from our dictionary occur in a requirement which we then count.

[10]In the literature, some of the suggested weak words overlap with words that are also assigned to other features. Therefore, we first ensured that these words are not assigned to other features of our research before we included them in our weak word dictionary.

[11]To give the reader an idea of the words contained, we present a small extract of the weak word dictionary of German terms in the following: ähnlich, alternativ, angemessen, annähernd, anscheinend, augenscheinlich, ausnahmsweise, ausreichend, bestmöglich, circa, denkbar, eventuell, gebräuchlich, geeignet, gewohnt, größtenteils, halbwegs, höchstwahrscheinlich, irgendwelche, manchmal, meistens, möglicherweise, offensichtlich, optimal, regelmäßig, typischerweise, vergleichbar, wahnsinnig, winzig, zufriedenstellend.

- **Readability index (REI).**[12] Some researchers propose the calculation of a readability index to measure the difficulty in reading (and understanding) a requirement (Génova et al., 2013; Wilson et al., 1997). "The underlying idea is that a text is more readable when average sentences and words are shorter" (Génova et al., 2013, p. 30). Mostly, the "Flesch Reading Ease" index is applied that evaluates the text in a range between 0 and 100 (Flesch, 1948). The index is initially designed for English text. Thus, we apply the German variant of the index which is called the "Amstad Understandability Index (AUI)" (Amstad, 1987, cited in Vor der Brück and Leveling, 2007). The index determines the readability of a text by the following formula:

  $$AUI = 180 - (AWS) - (58.5 \cdot (ASW))$$

  The lower the resulting index, the higher the difficulty is to read the text. A higher value indicates that text is easier to understand. We use the result from the calculation of AUI as the feature value.

Binary features.

- **Implementation obligations (IMO).** Modal verbs are used in the specification of requirements as auxiliary verbs that can express the necessity for the implementation. In the German language a total of six modal verbs are mentioned: "dürfen", "können", "mögen", "müssen", "sollen" und "wollen" (Diewald, 2012). However, the different use of modal verbs in a requirement does not make it clear whether a requirement must be implemented, leading to the ambiguity between an obligation and an option (Krisch et al., 2016). In the specification of English requirements, the word "shall" is usually referred to as a binding verb (Génova et al., 2013; Kasser et al., 2006). This corresponds to the German modal verb "sollen", but it is considered less strict than "müssen" in a requirement (Femmer et al., 2017). Thus, the use of the German modal verb "müssen" indicates a binding requirement. We describe a function that checks the presence of different forms of the modal verb "müssen". Besides, we also consider different forms of the modal verb combination "dürfen nicht/kein" in a requirement that is used for the specification of a prohibition of a characteristic or functionality—e.g. "Das Auto darf nicht brennen" (Rupp, 2014). The feature is assigned the value "1" if the modal verb "müssen" or the combination

[12]Since the readability index is determined using two quantity features (AWS and ASW), we have assigned it to the same category. We are aware that the index does not count a quantity, but a ratio.

"dürfen nicht/kein" occurs in a requirement. Otherwise, the feature is assigned the value "0".

- **Options (OPT).** In addition to the modal verbs assigned to the previous feature (IMO), we also determine the "weak verbs" of a requirement (Fabbrini et al., 2001). They are described as "the category of words that give the developer latitude in satisfying the specification statements that contain them" (Wilson et al., 1997, p. 4). These words leave room for interpretation by the recipient when implementing the requirements. According to some of the best practices from the automotive industry, the remaining modal verbs[13] such as "dürfen", "können" and "sollen" should not be included in a requirement. An exception is the already mentioned word combination "dürfen nicht/kein". We implement a function that checks the presence of the words and returns the value "1" in case of detection. If these modal verbs do not occur, the feature is assigned the value "0".

- **Template structure (TES).** Parra et al. (2015) suggest that requirements should correspond to a given grammatical structure. Also, best practices from the automotive industry prescribe a structure for the specification of a requirement. For our research, we adopt a common requirement specification template defined by several automotive companies. It describes that the requirement should start with a determiner and a subject, followed by a strict modal verb (see feature "Implementation obligations (IMO)"). Afterward, objects and parameters can optionally follow before another verb must complete the requirement. As an example, the following requirement fulfills the given structure: "Das Fahrzeug muss deutsche Verkehrszeichen erkennen". First, we identify a determiner ("Das"), followed by a subject ("Fahrzeug") and a strict modal verb ("muss"). The two words in the requirement ("deutsche" and "Verkehrszeichen") are considered optional in the template structure. The requirement then concludes with a further verb ("erkennen"). On the other hand, the following requirement does not correspond to the template structure: "Das ist keine Anforderung". To detect the structure and required elements of the sequence, we use the results from POS tagging of the requirements and a simple regular expression. If the sequence is detected in at least one sentence of a requirement, the value "1" is assigned. Otherwise, the value is "0".

- **Passive voice (PAV).** There is much discussion about the use of the passive voice in requirements (Rupp & Goetz, 2000; Wiegers, 2003). Génova et al.

[13]An analysis of our data set reveals that the modal verbs "mögen" and "wollen" do not occur in the natural language requirements.

(2013) point out that the use of passive voice "tend to leave implicit the verbal subject, leading to a certain degree of imprecision" (Génova et al., 2013, p. 32). When using passive voice, the actor of a requirement is usually missing. For this reason, Rupp (2014) suggests formulating each requirement in the active voice to also ensure simple direct sentences (Soeken et al., 2014). Some online tools[14] offer a passive voice detection by determining the syntax and a specific verb form. In our research, we implement a function that detects passive voice in each sentence of a requirement. With the POS tagging we are able to identify two constructions: first, the past participle of a verb and a form of the German word "werden" (e.g. "wird", "wurde"); second, a construction consisting of a form of the German words "sein + zu + *infinitive verb form*" (Donaldson, 2007; Eroms, 1992). When we detect at least one of these constructions in a requirement, we identify a passive sentence. For a requirement that consists of multiple sentences, we assign a value of "1" if at least one sentence is in the passive voice. If no passive voice is detected, the feature is assigned the value "0".

We define the quantity and binary features as feature set 2. For the extraction of this set, we performed separate preprocessing tasks of the requirement text independently from the previous extraction of tf-idf features. Our goal in selecting these features is to strive for a variety of text features. Features that include other information besides the textual part of a requirement (e.g. requirement versions, hierarchical structure, number of links to other requirements) are not considered in this thesis.

Sampling and scaling of data.

In addition to suitable features, a machine learning algorithm can only provide good results if the data set is balanced (Chawla, 2005). When a classifier receives imbalanced data during training, a bias in favor of the majority class may be present as there could be not enough data to analyze the minority classes.

We have already revealed in section 3.2.2 that the assessments for the characteristics are not always equally distributed. For *Singular* and *Unambiguous*, we therefore determine the frequency distribution for the assessments, illustrated in figure 4.5 and figure 4.6.

[14]Krisch and Houdek (2015) mention the online providers "Congree" (www.congree.com) and "Languagetool" (www.languagetool.org).

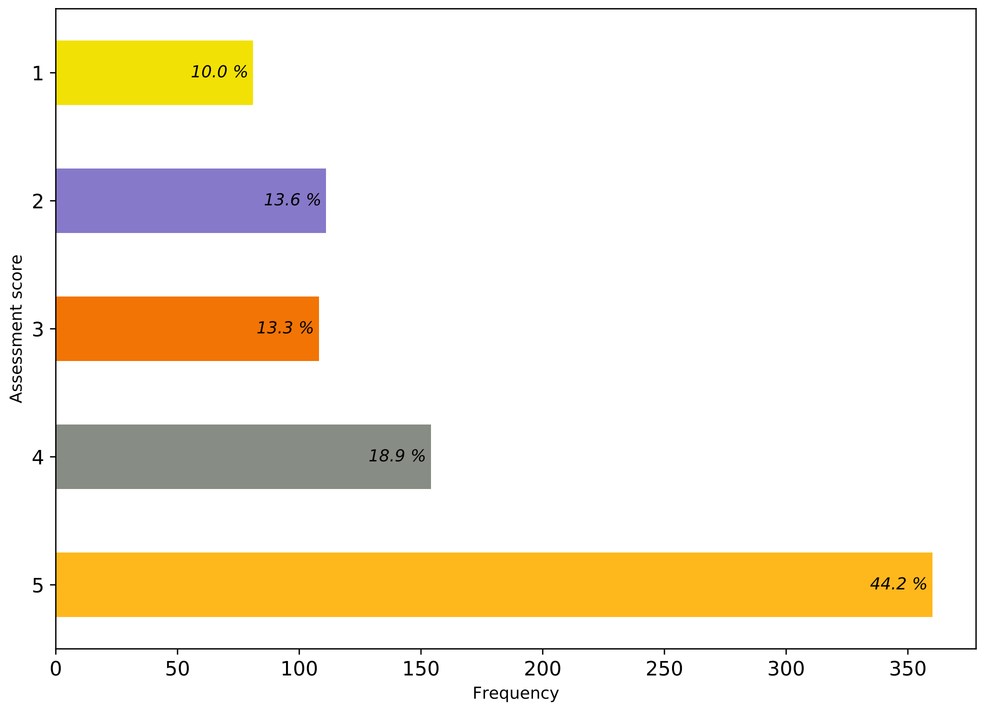


Fig. 4.5.: Frequency distribution of assessment scores for *Singular*.

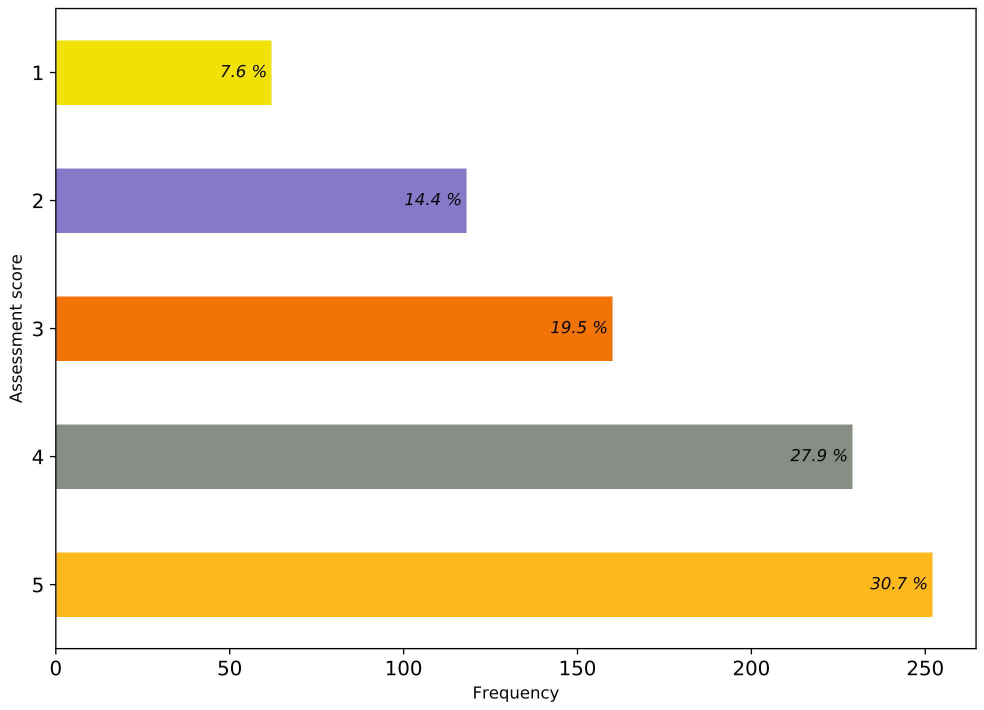


Fig. 4.6.: Frequency distribution of assessment scores for *Unambiguous*.

In general, the number of class instances in an imbalanced data set differs significantly between each other. Figure 4.5 reveals that for *Singular*, more than 44 percent of the requirements are assessed with the highest value "5". Similar observations can also be drawn for *Unambiguous*. Thus, the data set for the characteristics *Singular* and *Unambiguous* can be seen as imbalanced. Since machine learning algorithms require balanced classes to achieve proper performance, we present three sampling techniques that ensure this in a data set: undersampling, oversampling, and SMOTE (Chawla et al., 2002).

Undersampling describes a procedure that randomly reduces the number of class instances. The class with the lowest number of samples defines the value for all other classes and training data is randomly removed until all classes have the same size (Drummond & Holte, 2003; Pelayo & Dick, 2007). However, "the major drawback of random undersampling is that this method can discard potentially useful data that could be important for the induction process" (A. Fernández et al., 2018, p. 83).

With oversampling, minority classes randomly duplicate training data until the number of instances is identical in all classes. Thus, additional instances are created so that all classes have the same size. It has to be considered, that "random oversampling may increase the likelihood of occuring overfitting, since it makes exact copies of the minority class examples" (A. Fernández et al., 2018, p. 83).

The third technique is called SMOTE (Synthetic Minority Over-Sampling Technique). It creates synthetic training data in the minority classes so that the amount of data is the same in all classes of the training set (Chawla et al., 2002). Simply explained, SMOTE finds the n-nearest neighbors for each sample of the minority classes. Then, by drawing a line between the neighbors, it creates random points on these lines. To avoid sampling effects in our research, we generally apply these techniques only for the training set[15].

Besides, most of the quantity features have varying scales. This is apparent when we look at the features "NOW" ("Number of words"; range from 6 to 84) and "NOS" ("Number of sentences"; range from 1 to 4). For some machine learning algorithms, which rely on distances between data points for their prediction (as is the case for an SVM), features with varying scales pose a problem. Therefore, we normalize the scales for all quantity features[16]. For the feature scaling, we use the function "MinMaxScaler" provided by $scikit-learn$ that scales the values of a quantity feature to a range between 0 and 1.

[15]For a reliable evaluation, the test set always remains in its original distribution.

[16]Because of the binary values, it is not necessary to normalize the binary features. In addition, it is not recommended to normalize tf-idf features.

4.2.2 Machine learning

The results from the previous section enable us to create a numerical representation of our data set based on tf-idf features (feature set 1) and on quantity and binary features (feature set 2). With this representation, we aim to automatically determine the score for the characteristics *Singular* and *Unambiguous* for unlabeled requirements.

For the automated assessment, we refer to the scale from the assessment sessions. Therefore, we consider the points of the scale as discrete classes for which an automated approach should be implemented. This is a problem that can be found in the field of machine learning and can be handled by applying multi-class classification methods.

There has been a lot of research on which algorithms are best suited for text classification problems. However, there is no algorithm that consistently gives better results. Consequently, we are not able to decide in advance which machine learning algorithm will perform best. The performance is also influenced by a variety of factors, such as the available data set, the number of features, and the number of different classes. Thus, it is recommended to compare the performance of different algorithms to select the one best suited for solving a classification problem.

In this section, we first identify a supervised machine learning algorithm that is most suitable for our multi-class classification problem. We implement classical algorithms of three commonly used classifiers that are known to perform well in text classification: Support Vector Machines, Bayes classifiers, and Tree-based classifiers. In addition to the classical algorithms, approaches from deep learning are also increasingly applied. For good performance, however, a large amount of training data is needed, which we do not (yet) have in our research. Therefore, we refrain from using deep learning methods to solve our classification problem. Furthermore, the use of neural networks such as a multilayer perceptron would have been a possible addition. However, we could not identify any purposeful or beneficial application of neural networks in related research on quality assessment of requirements, which is why we initially focused on the aforementioned classical algorithms in this research.

The development is mainly based on the programming language Python, which provides a large number of relevant libraries for handling text classification problems (Bird et al., 2009; Swamynathan, 2019). We use the Python library $scikit - learn$, which offers an established machine learning library for the implementation of the algorithms (Buitinck et al., 2013; Pedregosa et al., 2011). In table 4.4, we present the algorithms and the related implementations from $scikit - learn$.

Tab. 4.4.: Algorithms and their implementations.

Algorithm	**Implementation**
Bayes classifier	Multinomial Naïve Bayes (Kibriya et al., 2004) Complement Naïve Bayes (Rennie et al., 2003)
Support Vector Machines	Support Vector Classifier (Bishop, 2006) Linear SVC[17] (Bishop, 2006)
Tree-based classifier	Decision Tree (Breiman et al., 1984) Random Forest (Breiman, 2001)

Besides, we define three feature sets for the implementation of an algorithm: first, we use the tf-idf feature set (feature set 1); second, we apply the quantitative and binary feature set (feature set 2); third, we use both feature sets (feature set 3). These feature sets are applied with different sampling techniques, that are necessary to ensure balanced classes in the training set. In table 4.5, we present the various feature sets and sampling techniques.

Tab. 4.5.: Variants of feature sets and sampling techniques for algorithm selection.

Type	**Variant**
Feature set	1: Feature set 1 (tf-idf features) 2: Feature set 2 (binary and quantity features) 3: Feature set 3 (feature set 1 and 2)
Sampling	1: No sampling 2: Undersampling 3: Oversampling 4: SMOTE

With the results from this section, we can conclude the implementation. Besides applying different feature sets and sampling techniques, we also consider varying scales when using different implementations of the described algorithms. In the next section, we focus on the evaluation. We analyze and optimize the trained models and identify features that are relevant for the assessment of the two characteristics *Singular* and *Unambiguous*.

[17]Support Vector Classifier and Linear SVC are implementations of the same algorithm using different Python libraries. Since Linear SVC supports only a linear kernel, it tends to be faster and is able to scale better.

4.3 Evaluation

By applying the implementations from table 4.4 and by using the variants (and combinations) of feature sets and sampling techniques from table 4.5 of the previous section, we receive a total of 72 results for each of the quality characteristics *Singular* and *Unambiguous*.

We use the standard configuration of the implementations from $scikit - learn$ and apply cross-validation with three iterations. An important point is to strictly separate training data to avoid data leakage from the test set to the training set. Therefore, we apply sampling techniques only on the training set (in each iteration of the cross-validation). Table 4.6 lists the best ten results sorted by *macro* F_1-score of the characteristic *Singular*. Besides, we add the best result where only feature set 1 is considered. Table 4.7 shows the results for the characteristic *Unambiguous*. We also present the values for the standard deviation of the *macro* F_1-score.

For *Singular*, we identify the Random Forest as the algorithm that provides the highest *macro* F_1-score. The use of feature set 3 and the application of oversampling lead to the best performance. The standard deviation (STD) of .0080 also indicates that the results are only slightly scattered and that the algorithm is the most stable. We can also recognize that the decision between feature set 2 and feature set 3 has only a minor impact on the results. When using feature set 1 exclusively, the results are predominantly in the lower range.

Tab. 4.6.: Results for *Singular* in descending order by *macro* F_1-score (Mean).

Algorithm	Feature set	Sampling	macro F_1 (Mean)	macro F_1 (STD)
Random Forest	**Feature set 3**	**Oversampling**	**.4379**	**.0080**
SVC	Feature set 3	Oversampling	.4375	.0317
SVC	Feature set 2	SMOTE	.4368	.0167
Random Forest	Feature set 2	No sampling	.4358	.0286
SVC	Feature set 3	SMOTE	.4310	.0521
Linear SVC	Feature set 3	SMOTE	.4302	.0363
Random Forest	Feature set 3	Undersampling	.4291	.0224
SVC	Feature set 2	Oversampling	.4277	.0184
Linear SVC	Feature set 2	SMOTE	.4248	.0211
Linear SVC	Feature set 2	Undersampling	.4245	.0279
...	...	...	...	...
Random Forest	Feature set 1	Oversampling	.3708	.0293

For *Unambiguous*, similar results can be observed. A significant difference between the use of feature set 2 or feature set 3 is not discernible. Besides, results in the lower range can be identified when using only feature set 1. In the table 4.7, the SVC algorithm shows a slightly better *macro* F_1-score (.3085), but the standard deviation is much higher (.0538) compared to the second-best result obtained by a Random Forest algorithm (*macro* F_1-score of .3054; standard deviation of .0046). Therefore, we choose the Random Forest algorithm for the characteristic *Unambiguous* as well.

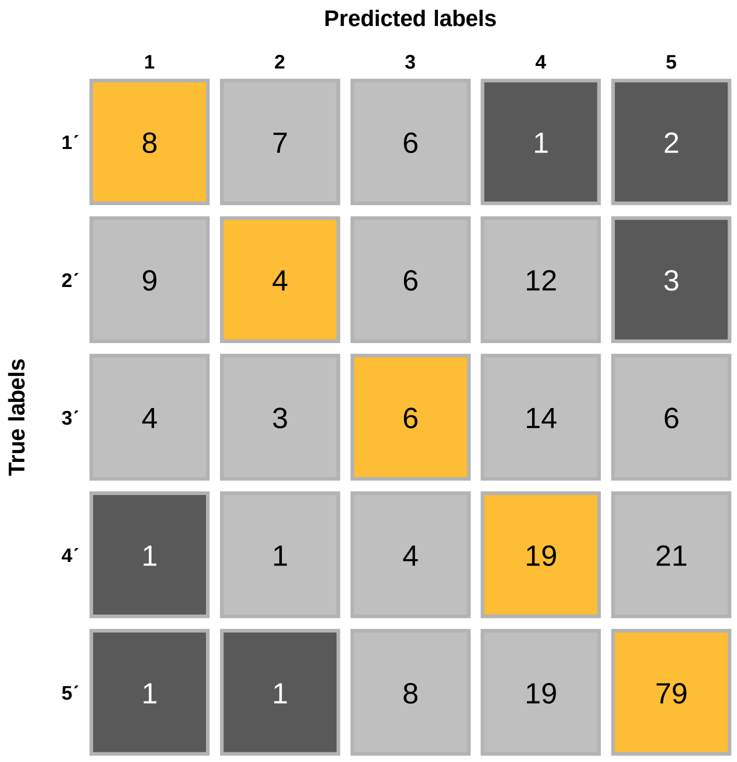
Tab. 4.7.: Results for *Unambiguous* in descending order by *macro* F_1-score (Mean).

Algorithm	Feature set	Sampling	macro F_1 (Mean)	macro F_1 (STD)
SVC	Feature set 3	Undersampling	.3085	.0538
Random Forest	**Feature set 3**	**Oversampling**	**.3054**	**.0046**
Random Forest	Feature set 2	No sampling	.3036	.0356
SVC	Feature set 3	No sampling	.3008	.0173
Random Forest	Feature set 2	Undersampling	.3001	.0176
MultinomialNB	Feature set 2	SMOTE	.2989	.0255
SVC	Feature set 3	SMOTE	.2984	.0171
Random Forest	Feature set 3	Undersampling	.2969	.0051
Linear SVC	Feature set 2	SMOTE	.2961	.0515
Linear SVC	Feature set 3	Undersampling	.2940	.0220
...	...	...	...	...
Linear SVC	Feature set 1	Undersampling	.2608	.0241

In general, it can be stated for both characteristics, on the one hand, that the sampling technique makes hardly any difference. On the other hand, the worst values predominantly use feature set 1. Therefore, the consideration of additional features besides those of a tf-idf approach is of significant importance for the performance of an algorithm. We emphasize here again the five-class classification problem. Thus, results that are only one score off to the true labels are evaluated in the same way as results that have a larger difference from the true labels. We discuss this point later in section 4.3.1.

Certainly, there exist additional preprocessing tasks and implementations of supplemental algorithms to provide better performance. Based on the results of the analysis so far, however, we decide to use the Random Forest algorithm for our research in the further course. When applying the Random Forest algorithm to our data set, we receive a machine learning model. In general, this is called an instantiation of a machine learning algorithm (Carbonell, 1990). In the further evaluation, we take a closer look at the resulting models of the best performing algorithm and analyze the results of the classification based on Random Forest. We focus on the hyper-parameters of the algorithm, which can be specified and optimized for the training of a model. Besides, we apply an approach to identify features that have a high impact on the classification task. This helps us to determine whether the features suggested by the literature are helpful for our research. Finally, the results enable us to answer our first research question, whether machine learning algorithms can be used to accurately assess (two of) the quality characteristics of natural language software requirements.

First insights into the results of a trained model are provided by a confusion matrix that presents the discrepancies between predicted and true labels. In figure 4.7, we exemplarily illustrate the confusion matrix for one of the folds from the cross-validation in section 4.2.2 for the characteristic *Singular*[18]. It can be recognized that the data set has imbalanced classes as a major part of the true labels is located in class "5". Correct classifications of the model can be found on the diagonal of the confusion matrix from the upper left to the lower right. Misclassifications are represented by the remaining values.


Predicted labels
True labels

	1	2	3	4	5
1´	8	7	6	1	2
2´	9	4	6	12	3
3´	4	3	6	14	6
4´	1	1	4	19	21
5´	1	1	8	19	79

Fig. 4.7.: Exemplary confusion matrix for *Singular*.

The trained model can easily recognize and assign class "5". The matrix also reveals that the outer corners, i.e. the areas where the highest delta between the true and predicted labels occurs, have mostly small values. The outer corners are

[18]The confusion matrix is only intended as an example to show how results of a model can be additionally analyzed. We do not consider it useful to show all folds of each quality characteristic.

highlighted black in figure 4.7. Such outliers are relatively rare as the number of these misclassifications ranges between 1 and 3. We can identify further misclassified requirements, which are especially apparent in classes "2" and "3". It can be stated that the classifier assigns the true labels in these two classes to other classes to a large extent. Such misclassifications can also be seen for true labels in class "1" and "4", but to a lower extent.

From the exemplary confusion matrix for *Singular* it can be derived that the classifier seems to have difficulties with the assignment of these classes. This can have different causes, which may also have to do with the selection of the hyper-parameter of the algorithm. For this reason, in the following, we focus on the analysis and optimization of the algorithm-specific parameters that can be defined for model training.

4.3.1 Model optimization

The performance of a classification algorithm depends on several different factors, such as the number of features or the number of different classes. Besides, each algorithm has individual parameters, which influence the performance as well.

In general, a machine learning model includes two types of parameters. The model parameters and the hyper-parameters. The model parameters are determined during the model training and are also sometimes called "fitted parameters" (Figueroa et al., 2012; Varghese et al., 2020). On the other hand, hyper-parameters are configurable parameters that require tuning to obtain a model with optimal performance. Each algorithm contains specific hyper-parameters that affect the performance of a model and that can be individually adapted to the application scenario. For a Random Forest algorithm, which we identified as the best performing algorithm for the characteristics *Singular* and *Unambiguous*, there are hyper-parameters such as the number of trees and the maximum depth of the trees.

For the determination of optimal hyper-parameter values, a method called grid search can be applied. For this purpose, a grid with possible values of the relevant hyper-parameters is defined. These values are then tested against each other. Often cross-validation is used, which enables to ensure the stability of a model created with the parameters to be validated.

A complementary procedure is described under the term nested cross-validation. In particular, the focus is on tuning hyper-parameters and on training the model-specific parameters (Statnikov et al., 2005). The procedure considers two nested loops. First, the inner loop is responsible for tuning the hyper-parameters, while the outer loop trains the model parameters. In each loop, cross-validation is performed. Regarding sampling techniques, that we additionally apply at this step, Weihs et al. (2013) recommend that "both the inner and the outer resampling" should be considered.

The Python library $scikit - learn$ provides a function that allows a combined grid search and cross-validation. The result is an exhaustive search for the optimal and best performing parameter combination for the Random Forest algorithm on our data set (Dietterich, 1998). For the extraction of the results, we additionally apply a nested cross-validation function presented in Müller and Guido (2016).

Tab. 4.8.: Hyper-parameters and grid values for Random Forest algorithm.

Hyper-parameter	Grid values
RF-trees	(10,50,100,250,500)
RF-depth	(5,10,20,50,100)
RF-features	(50,100,250,500)
RF-samples	(2,3,5,10)

For our research, we select the grid values presented in table 4.8 for the hyper-parameter search of a Random Forest algorithm. With "RF-trees" we define the number of Decision Trees used for the Random Forest algorithm. The parameter "RF-depth" specifies the maximum number of decisions that can occur in a tree. As default value, the decisions in the trees are executed until each leaf node has uniquely assigned samples for a class. This corresponds to a Gini impurity value of "0" (see section 4.1.2).

With the parameter "RF-features", we define the number of features to be considered in a split within a Decision Tree. The parameter "RF-samples" describes the minimum number of training samples required for splitting and serves as a stopping criterion for the trees. If the number of samples for a node is less than a defined value, no further splitting occurs and the node is considered a leaf node. A small value can lead to a strong refinement of the tree and can facilitate model overfitting. On the other hand, a high value can lead to leaf nodes describing samples from multiple classes.

Figure 4.8 visualizes the splitting process of a Decision Tree from a Random Forest algorithm. Starting with the first decision over a depth of five, the leaves of a single Decision Tree are determined. In this example, we use only the features from the feature set 2 and select the minimum number of samples for splitting.

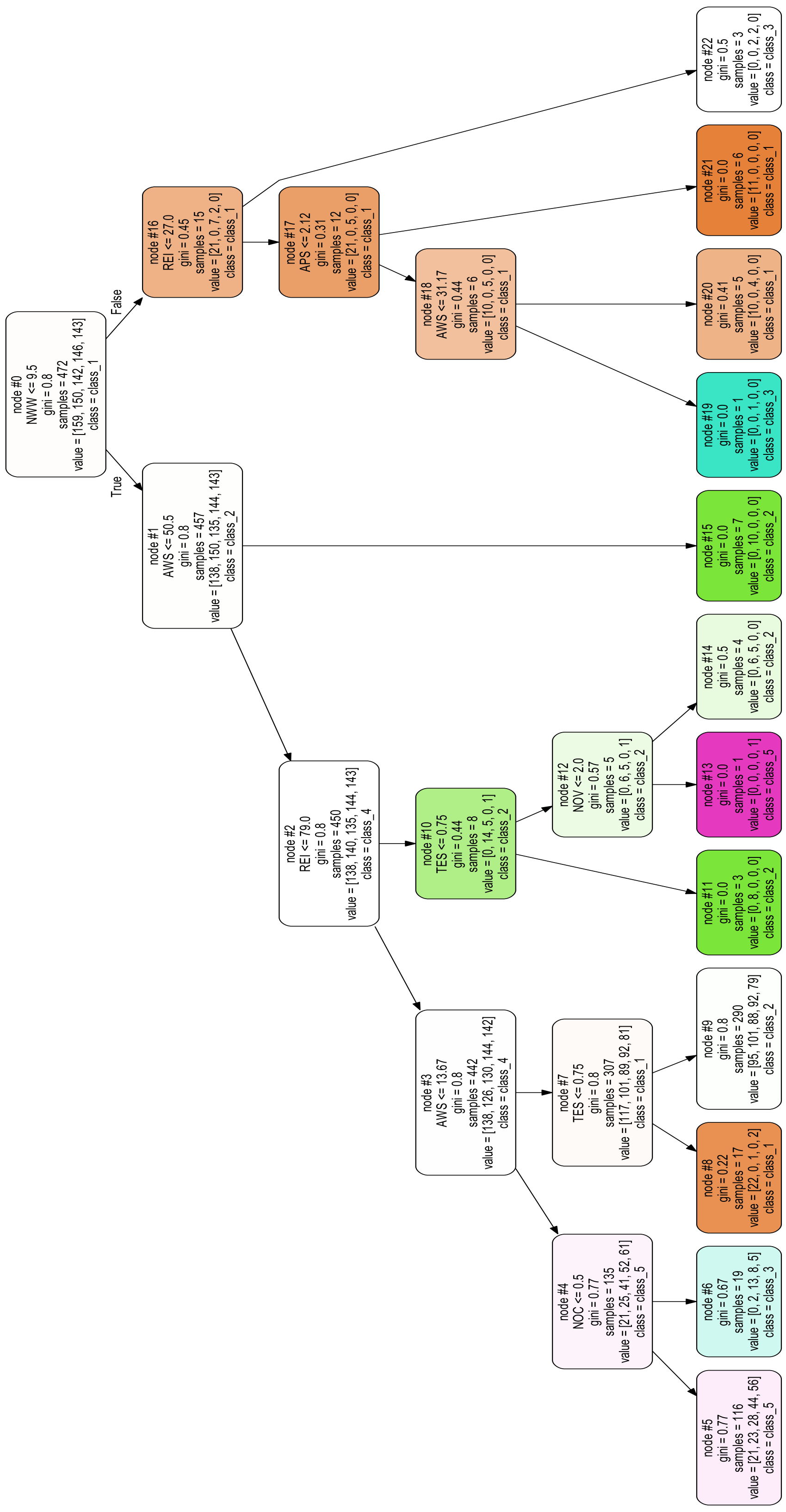


Fig. 4.8.: Visualization of the splitting process in a Decision Tree.

For the parameter grid, we have chosen a broad and reasonable range of values, resulting in a total of 400 different combinations. We iterate through all parameter combinations on the inner folds and validate the optimal parameters on the outer folds. This enables us to receive an estimation of the model error.

In the tables 4.9 and 4.10, the results from the nested cross-validation are listed for *Singular* and *Unambiguous*. We identify similar *macro* F_1-scores in each fold. For *Singular*, we obtain values between .3566 and .4067, with a mean of .3843. The standard deviation is .0238. For *Unambiguous*, we achieve values between .3239 and .3829, with a mean of .3468 and a standard deviation of .0243. These results indicate that there is no tendency towards overfitting since the values for each characteristic are not widely scattered.

Tab. 4.9.: Results from nested cross-validation for *Singular*.

Outer loop	**Grid values**				**macro F$_1$**
	RF-trees	*RF-depth*	*RF-features*	*RF-samples*	
Fold 1	50	100	250	5	.4067
Fold 2	10	100	100	5	.3566
Fold 3	50	50	250	2	.3937
Fold 4	10	50	100	2	.3611
Fold 5	50	100	50	3	.4033

Tab. 4.10.: Results from nested cross-validation for *Unambiguous*.

Outer loop	**Grid values**				**macro F$_1$**
	RF-trees	*RF-depth*	*RF-features*	*RF-samples*	
Fold 1	10	100	500	10	.3239
Fold 2	10	50	250	10	.3324
Fold 3	10	50	250	5	.3345
Fold 4	10	20	100	10	.3603
Fold 5	50	5	100	5	.3829

For better interpretation, we compare the results in a further step with two baselines. The first baseline describes a random guess of the scores for a quality characteristic. All scores are estimated randomly with equal probability (p=0.2). Opitz and Burst (2019) note that this form of a so-called 'dummy'-classifier, in which classes are predicted uniformly at random, is often used as a baseline by researchers. The trained classifiers provide an improvement of 107.73 percent for *Singular* and 82.53 percent for *Unambiguous* over this baseline. The second baseline describes another random guess classifier but with knowledge about the distribution of scores. As a result, the scores are estimated proportionally according to the distribution. We can observe that our best models are able to improve the results by 92.15 percent for *Singular* and 73.40 percent for *Unambiguous*. The results are presented in table 4.11 and show the feasibility of automated quality assessment of natural language requirements for both *Singular* and *Unambiguous*. Thus, we can conclude statistically that automated assessment on unlabeled requirements is well feasible.

Tab. 4.11.: Comparison of different baselines with the trained classifiers for *Singular* and *Unambiguous*.

Characteristic	Baseline	macro F_1	Increase
Singular	Trained classifier (Random Forest)	.3843	-
Singular	Random guess (no knowledge about the distribution)	.1850	+107.73%
Singular	Random guess (knowledge about the distribution)	.2000	+92.15%
Unambiguous	Trained classifier (Random Forest)	.3468	-
Unambiguous	Random guess (no knowledge about the distribution)	.1900	+82.53%
Unambiguous	Random guess (knowledge about the distribution)	.2000	+73.40%

Based on these findings, we can identify the models including their specific hyper-parameters for the quality characteristics *Singular* and *Unambiguous*. Again, we use cross-validation, which we apply to the entire data set in the following.

In table 4.12, we present the best ten results from the cross-validation for the characteristic *Singular* sorted by the *macro* F_1-score. In addition, we display the accuracy value. The best performing model achieves a *macro* F_1-score of .4845 with the following hyper-parameters:

- *RF-trees* = **500**
- *RF-depth* = **5**
- *RF-features* = **250**
- *RF-samples* = **5**

In table 4.13, we present the results for *macro* F_1-score and accuracy for the characteristic *Unambiguous*. The best performing model achieves a *macro* F_1-score of .3785 with the following hyper-parameters:

- *RF-trees* = **100**
- *RF-depth* = **5**
- *RF-features* = **50**
- *RF-samples* = **10**

For the characteristic *Singular*, the values for the *macro* F_1-score are slightly higher compared to the values of the outer folds of the nested cross-validation. For *Unambiguous*, similar values can be identified compared to the nested cross-validation. The results reveal again that there is no tendency for overfitting in the models. On

Tab. 4.12.: Results from cross-validation for *Singular* in descending order by *macro* F_1-score (Mean).

Grid values				Accuracy (Mean)	macro F_1 (Mean)
RF-trees	*RF-depth*	*RF-features*	*RF-samples*		
500	**5**	**250**	**5**	**.5012**	**.4845**
500	10	100	10	.4988	.4844
100	10	100	10	.5012	.4824
250	5	250	2	.4988	.4819
250	5	250	5	.4988	.4787
100	5	500	10	.4889	.4785
50	5	500	5	.4902	.4783
250	10	100	5	.4938	.4781
500	10	100	3	.4926	.4780
50	10	100	5	.4902	.4766

Tab. 4.13.: Results from cross-validation for *Unambiguous* in descending order by *macro* F_1-score (Mean).

Grid values				Accuracy (Mean)	macro F_1 (Mean)
RF-trees	*RF-depth*	*RF-features*	*RF-samples*		
100	**5**	**50**	**10**	**.3873**	**.3785**
500	5	50	3	.3922	.3752
50	5	100	3	.3812	.3727
100	5	50	2	.3824	.3710
250	5	50	5	.3837	.3709
10	5	500	2	.3727	.3704
250	5	100	10	.3812	.3699
500	5	50	5	.3910	.3691
50	10	50	2	.3800	.3691
50	5	100	10	.3849	.3684

the one hand, however, one would expect that the values for the *macro* F_1-score to be somewhat higher since more data is available for training the models. On the other hand, we consider a five-class classification problem in our research. Due to the complexity of the classification task, the available data, and features, it might not be able to achieve higher values for these characteristics. Therefore, we perform further analysis on the results in the following, which provides us with more detailed insights into the models and performance.

In a classification problem, misclassification is the discrepancy between the true label and the predicted label. The degree of a wrong classification is usually not taken into account. If the trained model classifies a value next to the true label, it is treated as a false classification. However, the same applies if the classified label has four values next to the true label. In the case of multi-class classification problems with different ordinal classes, the exact classification may be much less frequent. "Therefore, the prediction accuracy should be judged with respect to the number of classes presented in the classification problem. A relatively smaller value [...] may indicate a reasonably good performance when the number of classes is large, and vice versa" (Sharda & Delen, 2006, p. 248).

For that reason, we extend the current analysis with an approach that is particularly applicable to multi-class classification problems (Sharda & Delen, 2006). Huang et al. (2004) refer to this approach as "within-1-class", which considers classifications next to the true label as correct. Consequently, the classes that are considered correct for classification are those that lie within one class based on the true label. Only classifications that are outside the neighboring classes of the true label are handled as false classifications. The approach is illustrated in figure 4.9 and the cells highlighted in yellow show a correct classification. For example, for the true label in class "3", the predicted classes "2", "3", and "4" are considered as correct classifications. On the other hand, the cells highlighted in grey are misclassifications.

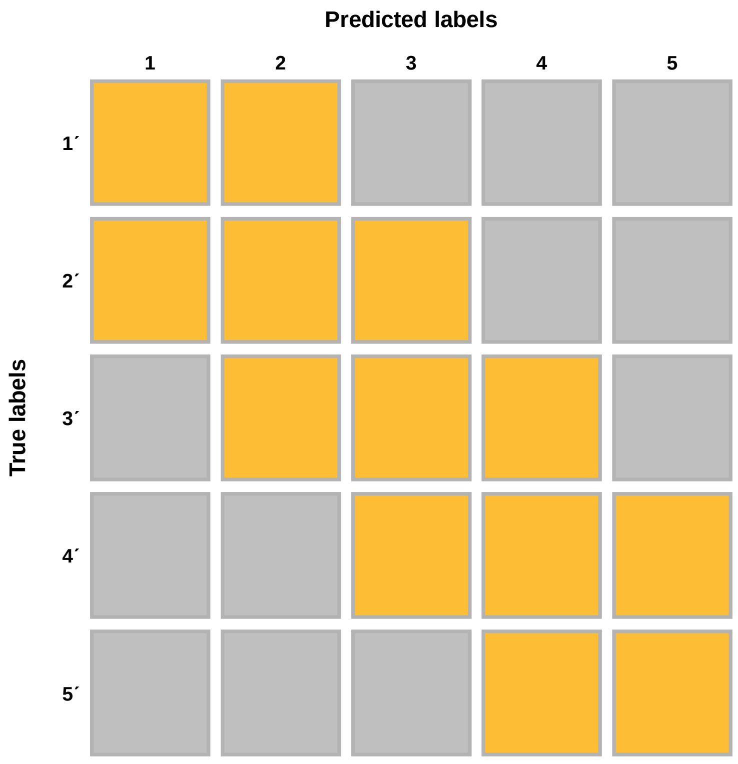


Fig. 4.9.: Correct classifications (yellow) in the within-1-class approach.

Table 4.14 shows the results for the *macro* F_1-score and accuracy of the within-1-class approach for *Singular* and *Unambiguous*. We also present the original results.

Tab. 4.14.: Results from within-1-class approach for *Singular* and *Unambiguous*.

Characteristic	Original		Within-1-class	
	Acc. (Mean)	macro F_1 (Mean)	Acc. (Mean)	macro F_1 (Mean)
Singular	.5012	.4845	.7905	.6383
Unambiguous	.3873	.3785	.7503	.6265

For *Singular*, an increase can be recognized for both measures. The *macro F_1*-score shows a result of .6383 and the accuracy increases from .5012 to .7905. For the characteristic *Unambiguous*, the values for accuracy and *macro F_1*-score are almost doubled. Again, we compare these results with the previously defined baselines. The results are shown in table 4.15.

Tab. 4.15.: Comparison of different baselines with the trained classifiers for *Singular* and *Unambiguous* for within-1-class approach.

Characteristic	Baseline (Within-1-class)	macro F_1	Increase
Singular	Trained classifier (Random Forest)	.6383	-
Singular	Random guess (no knowledge about the distribution)	.4107	+55.42%
Singular	Random guess (knowledge about the distribution)	.4231	+50.86%
Unambiguous	Trained classifier (Random Forest)	.6265	-
Unambiguous	Random guess (no knowledge about the distribution)	.4263	+46.96%
Unambiguous	Random guess (knowledge about the distribution)	.4435	+41.26%

A random guess without knowledge about the score distribution achieves a *macro F_1*-score of .4107 for *Singular* and a *macro F_1*-score of .4263 for *Unambiguous*. With the within-1-class approach for our trained classifiers, values are achieved that are 55.42 percent above a random guess for *Singular* and 46.96 percent above a random guess for *Unambiguous*. The second baseline describes a random guess with knowledge of the score distribution. In this case, as well, improvements of 50.86 percent for *Singular* and 41.26 percent for *Unambiguous* can be stated with our trained classifier. These results once again demonstrate the feasibility of our approach for automated assessment of the characteristics *Singular* and *Unambiguous* of a natural language requirement.

The results also indicate that for our five-class classification problem, the trained classifiers predict close to the true labels. The models for the quality characteristics *Singular* and *Unambiguous* can be used for a prediction of the classes, even if there is uncertainty left. This also results from the fact, which we already observed in the previous chapter, that even the experts do not always agree when assessing the requirements. Consequently, a supervised classification model can only ever be as good as expert assessments. Based on the obtained results, we can answer our first research question and confirm that machine learning algorithms can be used to accurately assess (two of the) quality characteristics—*Singular* and *Unambiguous*—of a natural language software requirement.

We are now able to further analyze and determine which of the features contribute most to the classification. For that purpose, in the following section, we describe the procedure for determining the importance of the features and present the results for *Singular* and *Unambiguous*.

4.3.2 Feature importance

"Often, we desire to quantify the strength of the relationship between the predictors and the outcome. Ranking predictors in this manner can be very useful when sifting through large amounts of data" (Kuhn & Johnson, 2013, p. 463). A decisive advantage, especially in the context of our research, is the interpretability of the results for tree-based approaches. When using the Random Forest algorithm for predicting the score of a characteristic, we can apply a function from the Python library $scikit - learn$ called "feature importances". This function allows us to view all the features used in our model and identify the most important ones.

Feature importance by definition is the increase in the model error if we remove the information received from the feature. In other words, it determines which features (i.e. number of sentences (NOS), number of words (NOW)) are more important for the classifier to make an accurate prediction on the requirement assessment. Using this method, we can visualize which features have a greater effect on the model's ability to make accurate predictions. As well as identifying features that have a less significant effect on the classification outcome if these are used solely. In general, features that are used at the beginning to divide a decision tree have a greater effect on the classification.

The function "feature importances" returns a list of the importance values which correspond to the features. These values range from 0 to 1 and can be interpreted as the percentage of how much the model's performance decreases without these features. The higher the number, the greater the effect is. The classifier determines these values using Gini impurity. As described in section 4.1.2, Gini impurity measures the probability of an incorrectly classified prediction based on some features. It computes how much each feature contributes to decreasing the impurity for each tree within the forest, then averages the impurity over all trees.

While the primary goal is to accurately predict the score of a given requirement, identifying the important features can be a useful tool in providing reasoning behind the classifier's decision to assess the characteristics of a requirement as one of the five classes. Determining the feature importance helps us to identify which features from the feature sets are relevant for each characteristic. Therefore, we perform the feature importance analysis for the characteristics *Singular* and *Unambiguous* and rank the top five results by the importance values in table 4.16.

Tab. 4.16.: Top five features according to importance values for *Singular* and *Unambiguous*.

Rank	**Singular**	**Unambiguous**
1	NOW	NOW
2	NOC	NWW
3	NOV	NOC
4	AWS	AWS
5	NOS	NOV

Various findings can be derived from the results of feature importance. The quantity features from the feature set 2 have the highest importance values for both characteristics. For the characteristic *Singular*, the number of words ("NOW"), the number of conjunctions ("NOC"), the number of verbs ("NOV"), the average number of words per sentence ("AWS"), and the number of sentences ("NOS") appear in the ranking of the top five features. We do not identify binary or tf-idf features as most important. A similar ranking can be observed for the characteristic *Unambiguous*, in which only quantity features are identified as well. The number of words ("NOW"), the number of weak words ("NWW"), the number of conjunctions ("NOC"), and the average number of words per sentence ("AWS") affect the unambiguity of a requirement, as well as the number of verbs ("NOV"). The feature "NOW" appears on the first rank for both quality characteristics. Thus, the number of words has a comparable stronger influence on the general assessment of requirements quality.

Referring to the description of the two characteristics in ISO 29148, the following points can be identified. For the characteristic *Singular*, a requirement includes only one requirement with no use of conjunctions. Consequently, the number of words, conjunctions, and verbs indicate that there is more than one requirement. Besides, the number of sentences, as well as the average number of words per sentence have an influence on this characteristic. A similar result can be seen for *Unambiguous*. In particular, it becomes clear that the number of weak words is a relevant feature for the automated quality assessment of the characteristic *Unambiguous*. It can be deduced that weak words influence the ability to interpret requirements in only one way.

In general, the determination of the feature importance reveals which of the features from our feature sets are important for the particular characteristics. Relevant features are based on the quantity features from the feature set 2. Furthermore, binary features and tf-idf features do not appear in the ranking of the top five feature importance values.

In this section, we focused on the evaluation of our trained models. In the beginning, we presented a confusion matrix to exemplarily demonstrate how to get insights into the results of a model and how to identify misclassifications. We further described the optimization of the models by hyper-parameter tuning. Besides, we applied an approach that also provides additional insights into the degree of misclassifications in multi-class classification problems. In the following section, we conclude the chapter 4 and discuss open issues.

4.4 Conclusion

In this chapter, we provided an answer to our first research question, whether machine learning algorithms can be used to accurately assess the quality characteristics of natural language software requirements. For this purpose, we described our approach for the automated quality assessment.

With the help of extensive feature engineering and the use of different algorithms, we proved the feasibility of an automated quality assessment of requirements based on the two characteristics *Singular* and *Unambiguous*. We achieved a *macro* F_1-score of .4845 for *Singular* and a *macro* F_1-score of .3785 for *Unambiguous*.

Regarding the results, we must take into account the following issue. In our research, we are confronted with a multi-class classification problem in which a classifier uses labeled data from experts who do not always agree on the assessment of the quality characteristics (see section 3.3.4). However, in comparison with two different baselines (random guess with/without knowledge about the distribution), our models outperformed. Extended approaches also revealed that the classifier results are close to the true labels. These results confirm the feasibility of automated quality assessment of natural language requirements.

In a further analysis, we evaluated, whether the features describe the quality characteristics sufficiently. The results reveal that the feature importance ranking for both characteristics is very similar. Besides, some of the features did not contribute to the classification models. Using the passive voice, for example, did not affect the characteristics *Singular* or *Unabmiguous*. Although it is claimed, that an actor is crucial for the implementation of the requirement, we cannot conclude that the use of passive voice causes problems. An assumption that should be investigated in further research efforts is that the missing actor of a requirement can be determined or assumed by the context of a requirement and/or by expert knowledge. This does not mean that the use of passive voice causes no problems in general. In the context of our research, however, it does not seem to be as harmful as generally assumed. Besides, an exclusive assignment of measurable indicators to *Singular* or *Unambiguous* can not be confirmed.

Our approach to automated quality assessment of natural language requirements surely has some limitations. The performance of our model and the ability to automatically determine the quality of natural language requirements were highly dependent on the defined features and thus on the numerical representation of the text. For the definition of the features, we concentrated on standard techniques in the field of natural language processing on the one hand, while at the same time considered additional features proposed in research papers and best practices. With the defined features, we covered relevant indicators to determine the quality of requirements. However, it must be taken into account that not all possible aspects were included. Rather, domain- or language-specific features can play an important role as well in determining the quality of requirements. Also, we focused primarily on textual features in our research, neglecting non-textual features. In extended research, these features could capture and analyze meta information of the requirement, such as the number of linked requirements, the frequency of requirement changes, and the structural information of the specification. Furthermore, semantic analyses can also improve the performance of our models. A semantic analysis goes beyond the structural elements of a requirement and examines the meaning of the text. For example, determining sentiment in terms of its polarity describes a simple form of semantic analysis. More complex analyses consider the different meanings of words in the context of other words.

Further limitations include that we only applied simple preprocessing tasks and did not consider additional machine learning algorithms, such as neural networks. As we only had a small amount of data available for training the models, no more insights could be drawn from the elaborate assessments. With our current research, we aim at a first approach, which is why some of these aspects are not considered at this point. For future research, however, the consideration of these aspects is essential.

The managerial implications are apparent. With the application of an automated quality assessment, requirements engineers can specifically identify requirements that do not satisfy the quality characteristics *Singular* and *Unambiguous*. In current industrial practice, such requirements quality reviews are mostly performed by time-intensive manual work. A manual review or manual assessment of the quality characteristics of a requirement is typically the most accurate approach. However, such a procedure is also the most expensive. The costs of an automated assessment are practically non-existent. Thus, we consider the results to be appropriate for a task that is usually performed manually and requires a huge investment of time.

The results of this chapter provide evidence that an automated assessment of the quality characteristics *Singular* and *Unambiguous* by using a supervised learning approach is possible. We identified different aspects for the further optimization of our approach. Nevertheless, the presented results can already be used to provide an initial automated assessment for requirements engineers in industrial practice. It reduces the time-intense effort that would be required by a comprehensive manual review of the increasing number of requirements in software development projects.

Conclusion 5

In this thesis, we model and assess the quality of natural language requirements through the identification and definition of characteristics. We analyze the feasibility to automatically and accurately assess quality characteristics and present an approach based on textual indicators. The main output of our research is the successful development of this approach for the assessment of two quality characteristics by using natural language processing and supervised machine learning. Requirements from industrial practice and experts' input from the automotive industry ensure the practical applicability of our results.

In the final chapter of this thesis, we summarize the relevant aspects and results of our research and present limitations and implications. Finally, we provide an outlook on future research efforts.

5.1 Summary

The motivation for the automated assessment of requirements quality is based on the premise that the development of software is becoming an increasingly important economic factor, especially for the automotive industry. In the purchasing process of a new vehicle, for example, customers are increasingly looking at the services and software functions a car provides. This trend leads to additional challenges in the development of automotive software. In our research, we identify three major challenges. First, an increase in legal demands and safety requirements. Second, a general increase in the complexity of the developed software. Third, distributed development approaches with different partners. These challenges are directly related to the requirements of a project.

Nowadays, requirements in a software development project are primarily used to communicate with relevant stakeholders and are largely specified in natural language. Despite the availability of numerous approaches with a higher degree of formalization, hardly any of them are applied in practice. The general comprehensibility and the easy applicability of natural language play a decisive role. However, natural language has three problems. It is often imprecise, incomplete, and ambiguous, which makes reliable communication based on requirements difficult. Due to the increasing challenges in the development of software, we identify a necessity for quality assessment of natural language requirements as a kind of quality assurance. Traditional approaches are described in the form of requirement reviews, which are performed manually by engineers. With the increasing number of requirements, new, reliable, and automated approaches are necessary.

Our research provides an approach for the automated quality assessment of natural language requirements. To enable an objective assessment, requirements should be interpreted within a widely accepted standard framework. We can then expect that our results are more useful for researchers investigating similar studies. We identify the international standard ISO/IEC/IEEE 29148:2011, which describes nine quality characteristics of a requirement: *Complete*, *Consistent*, *Feasible*, *Implementation Free*, *Necessary*, *Singular*, *Traceable*, *Unambiguous*, and *Verifiable*. Besides, we introduce our requirements quality model based on these characteristics.

For the implementation of an automated approach, we encounter techniques and methods from natural language processing and machine learning. We decide on a supervised machine learning approach based on labeled data. In several assessment sessions, we let experts from the automotive industry assess 1,000 real requirements from automotive development projects. Each assessment is based on the characteristics of our quality model. We analyze the expert assessments and take reliability into account from two points of view. First, we evaluate the reliable assessment of a single expert. For that reason, we have an expert re-assess a requirement that he already assessed before and compare these results. The intra-rater agreement determines the degree of agreement. Second, we have each requirement assessed by two different experts. With the inter-rater agreement, we can determine the degree of agreement by two different experts.

We calculate Krippendorff's alpha (α_k) for the determination of reliability. The values for the intra-rater agreement vary between .669 and .776 for the characteristics, which indicates a sufficient agreement and confirms the reliable assessment of a single expert for all quality characteristics. For the determination of the inter-rater agreement, the results vary to a larger extent between the characteristics. We identify three groups of characteristics. For *Consistent*, *Necessary*, and *Traceable*, a low agreement can be determined ($\alpha_k \leq .222$). The second group contains characteristics with a slightly better level of agreement. These include the quality characteristics *Complete*, *Feasible*, *Implementation Free*, and *Verifiable* ($.222 < \alpha_k \leq .444$). Finally, the characteristics *Singular* and *Unambiguous* belong to the third group, for which values for $\alpha_k \geq .667$ are identified. We can conclude that not all characteristics from the ISO 29148 standard can be assessed equally. In particular, quality characteristics for which context-relevant requirements would be helpful can not be assessed with the same reliability as characteristics that focus only on the single requirement—like *Singular* or *Unambiguous*.

Based on the results, we decide to continue with the characteristics from the third group for which, according to Krippendorff, the lowest conceivable limit is reached to draw tentative conclusions. The results also answer our second research question, whether experts are able to reliably assess the quality characteristics of a natural language requirement. We can confirm that this is possible only for *Singular* and *Unambiguous* in the context of our research.

With the labeled data at hand, we can apply supervised machine learning. We experiment with different feature sets, sampling techniques, hyper-parameters, and algorithms and are able to identify the best-performing classification models for *Singular* and *Unambiguous*. We obtain a *macro F_1-score* of .4845 for *Singular* and a *macro F_1-score* of .3785 for *Unambiguous*. The results of the trained classifiers are compared with different baselines and indicate that our models perform significantly better than a random guess (+92.15 percent for *Singular*; +73.40 percent for *Unambiguous*). We can then answer our first research question, that machine learning algorithms can be used to accurately assess (two of) the quality characteristics of natural language requirements.

We demonstrate the effectiveness and feasibility of our approach for the quality assessment of natural language requirements. Further validation in industrial practice is currently planned with several companies to improve the practical suitability. In general, quantitative measurement represents one of the foundations of modern empirical science. It is important that numerical measures are not applied as a matter of principle, but are used with care and wisdom. For the assessment of requirements quality, this means that human judgment is still required and can only be supported by objective measurements. Complete substitution of the human is not possible, however. Nevertheless, we are convinced that the automated approach for assessing requirements quality can provide valuable information for improving the characteristics.

We have diversified our research across different development projects and automotive companies. However, we consider the presented results as the first step towards the continuous quality assessment of requirements in software development projects. In the next section, we, therefore, discuss the limitations of our research.

5.2 Limitations

Our research is subject to several limitations. In each chapter of the thesis, we discuss the specific limitations for the manual assessment (see section 3.4) and for the automated assessment of requirements quality (see section 4.4). Besides, we identify the general limitations of our research, which we describe in the following section.

The generalizability of research should always be a primary goal. The available data in our research originates exclusively from the automotive industry. This is advantageous because the data tends to be coherent and is therefore suitable for statistical analysis. Our approach for the assessment of requirements quality, on the other hand, is based on the international standard ISO 29148 and general machine learning algorithms. Similarly, the presented features are not necessarily automotive-specific. Only some of the features that originate from the best practices of the automotive industry or are based on weak word lists of the major companies could be specific to the automotive industry. Consequently, there are clear reasons to believe that the results provide a basis for general applicability. Mostly, approaches and solutions from the software engineering body of knowledge for the automotive industry can be adopted from other domains. However, the verification of the automated assessment of requirements quality for domains other than the automotive industry has not yet been performed.

Also, we focused on German language requirements only and initially neglected other languages. A transfer especially to English language requirements seems reasonable and will be considered in future research efforts. Another limitation relates to the textual indicators as we cannot exclude that we have considered all relevant features for the quality assessment of requirements. Rather, we identified features based on similar research contributions and best practices from the industry. Besides, we did not use semantic aspects in the models so far but focus primarily on the extraction of syntactic features of a requirement text. The additional extraction of entities, for example, as well as their relations from text, can help uncover more meaningful semantic information compared to a simple tf-idf approach (Aggarwal & Zhai, 2012a).

One of the major limitations of our research, however, is that only for two quality characteristics an automated assessment can be realized. For the remaining characteristics, no sufficient agreement in the assessment between the experts could be determined. This could have two reasons: First, the selection of the experts, who were not able to assess the requirements due to lack of competence and experience;

Second, the procedure of how the assessment was performed. We did not provide the experts with any guidelines but simply used the descriptions of the quality characteristics from the standard ISO 29148. However, the quality characteristics are not described in such a way that an expert could reliably assess to what degree the quality is fulfilled in the individual case. As a result, an algorithm is neither able to perform an automated quality assessment. Some authors have hypothesized that the quality characteristics could be assessed based on textual indicators. This hypothesis is partly verified in our research and worked well only for *Singular* and *Unambiguous*. For the remaining characteristics, however, it did not. The reasons for this are manifold, but limit the current research and the automated assessment of requirements quality to the two characteristics.

Besides, as our research includes assessments that have been performed by humans, the (agreed) assessments may therefore also be subject to human errors. The main risk in validating the methodology lies in the errors in the assessment of requirements by the experts. The classifier can learn an incorrect assessment and repeat the error when assessing new requirements. Although we are confident that our procedures for assessing requirements and selecting labeled data have helped to reduce this threat, it is still apparent. Future research efforts may further investigate this type of reliability for manual assessments.

There are also general limitations to the automated assessments. This is especially the case for the classifier results, which do not provide a perfect assessment of the requirements for the characteristics *Singular* and *Unambiguous*. We can demonstrate with our research that we enable a significant improvement compared to different baselines (random guess with/without knowledge about the distribution). The results are, however, far from a perfect classifier that would have a *macro* F_1-score of 1. The reason for this could be that we have not yet exploited the full potential of natural language processing and machine learning. Among other things, we use simple preprocessing tasks and focus only on classical algorithms.

Finally, requirements engineering is a diverse field. Thus, one of the main limitations is that our results do not cover all domains and ways of specifying requirements. To mitigate this limitation to a certain degree, we focused on the specification technique that is used mostly today in industrial practice. Requirements stated in other ways than natural language are not considered and should be addressed in future research efforts. The presented limitations critically examine the current state of the thesis, but also open up future research activities.

5.3 Implications

In general, the assessment of requirements itself is not limited to specific quality assurance tasks and therefore does not depend on a specific process model. Rather, the use of a quality tool and its integration into quality assurance depends on how the relevant specifics of this context are answered. We follow the view of Femmer et al. (2017) and support the application of automated quality assessment both in the process of constructive and analytical quality assurance. The former describes that the automated quality assessment can be used by requirements engineers to increase overall quality perception and to eliminate problems. For analytical quality assurance, quality assessment can be used as a preparation to optimize potentially cost-intensive quality assurance tasks of external reviewers. Examples of such tasks include the Fagan inspection (Fagan, 2002), where several different reviewers examine the requirements. The implications and full application of our research for these two quality assurance areas must first be validated in future research efforts.

Nowadays, the determination of requirements quality is conducted primarily in the context of requirement reviews, which are performed manually by engineers. However, our quality assessment can not (and should not) replace the manual review by an expert. Rather, it serves as a preliminary step and offers a helpful approach to support quality assurance in requirements engineering. It should be considered as a supplement to a review so that the expert can focus exclusively on the essential elements of a requirement without being distracted by language deficits.

Thus, the practical implications of our research are apparent. Our approach allows us to ensure the quality characteristics *Singular* and *Unambiguous* of requirements at an early stage of development and to reduce or eliminate the negative effects of poorly specified requirements. Engineers can examine their requirements and identify potential defects. In the current state of research, though, the scope is limited. Future research helps to improve this state from different points of view and to determine the effects of quality assurance measures for requirements in a software development project.

5.4 Future research

With the current research results, we are able to automatically assess two of the quality characteristics of natural language requirements. We apply machine learning techniques and natural language processing to a problem in the field of requirements engineering. In general, we do not see our approach as a final solution, but rather as proof that quality assessment is fundamentally possible in that way. It is necessary that future research efforts focus on the automated assessment of further characteristics, the performance of the models, and complement approaches for optimization.

The consideration of further approaches for the vector representation of requirements is one possibility for optimization. As an alternative to the use of the presented feature sets, word embedding methods should be considered, such as static word embedding and contextualized word embedding. For the former, two approaches exist, among others. GloVe (Pennington et al., 2014) and word2vec (Mikolov et al., 2013) represent words of a vocabulary with dense low-dimensional vectors. The meaning of the words is learned by using the local context. The order of the words is not considered, the focus is only on the presence (or absence) of a word in the environment of another word. Promising and revolutionary for NLP problems is the possibility of converting an arbitrary word into a meaningful vector representation. In contextualized word embedding, vectors are also formed for each word but additionally conditioned on their context. This form of context integration has improved the state of the art in NLP challenges such as sentiment analysis and document classification. Approaches like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) contribute significantly to this and should be considered in future research efforts.

A further important aspect is, in particular, to investigate how the remaining quality characteristics can also be assessed in an automated way. This could include the need to gather additional labeled data. One way to increase the reliability of the assessments is to have a third expert assess the quality characteristics for each requirement. This might allow a majority vote to be identified and used for an alternative determination of reliability, allowing additional quality characteristics to be integrated into the automated assessment.

The additional amount of data available for model training is a factor that has become commonplace in improving the performance of machine learning tasks. With our labeled data set, we can demonstrate the general applicability of an automated approach to assess the quality characteristics *Singular* and *Unambiguous* of natural language requirements. An increase of these data by additional expert assessments would allow a possible optimization of the models and could also enable the use of further machine learning algorithms. Deep learning, for example, relies on artificial neural networks, where a hierarchical approach is used to train the networks. The learning with training examples remains the same but the definition of suitable feature representation is not necessary. Through deep learning and the hierarchical process, an implicit representation of the data is learned and represented over several layers. In addition, recent research also enables the interpretation of neural networks through so-called ablation studies. In these studies, certain parts of a neural network are removed in order to better understand the behavior of it. This in turn enables interpretable machine learning.

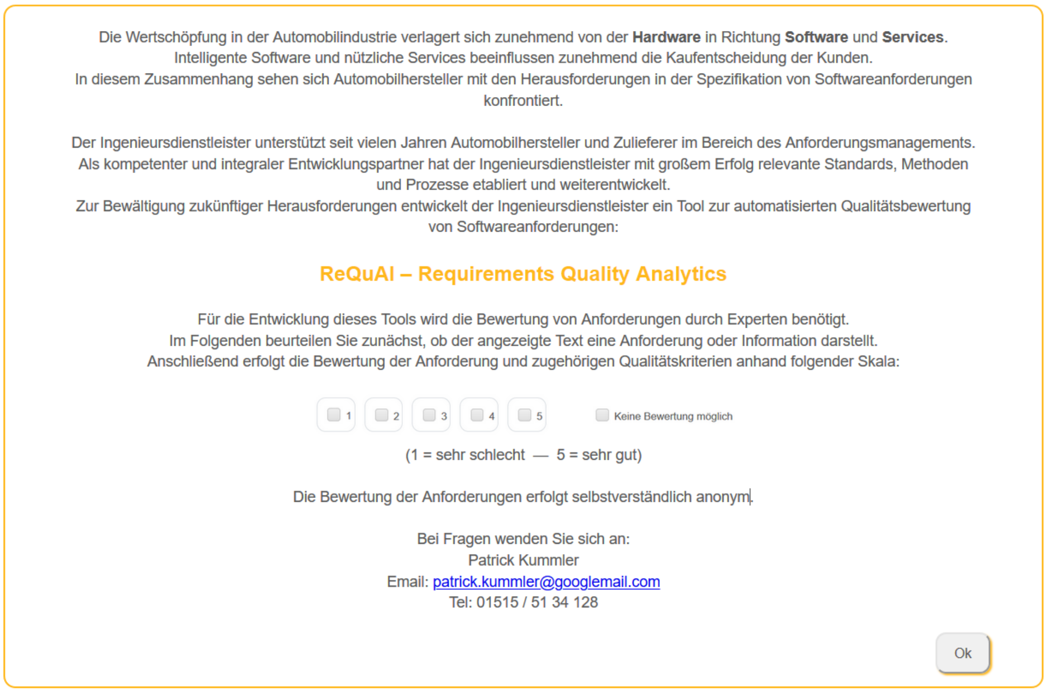
In general, we provide a comprehensive view of requirements quality based on nine quality characteristics. However, every single quality characteristic is a field of its own, which enables a variety of future research activities. The identification of specific textual indicators to determine quality, in particular, is an exciting field of research. Besides, the interpretability of the classifier results is an important task. Most engineers in industrial practice are aware that the requirements do mostly not correspond to the quality characteristics. Therefore, it would be important to automatically recommend individual actions to increase the quality of a requirement. In our research, we have established the first step by determining the feature importance values for two of the characteristics. These results can be used as a basis to gain initial insights. Future research should further focus on interpretable machine learning.

The most important aspect for future research, however, is not to involve additional or more experienced experts to increase the reliability of the assessments. Also, enlarging the requirement data set, training other algorithms, or optimizing parameters do not solve the core problem. Rather, the most important aspect is to investigate how the quality characteristics of a natural language requirement can be better assessed from the perspective of the experts and the algorithms. Starting points for this can be the creation of detailed guidelines according to the defined quality characteristics, which mitigate the fuzziness of the existing descriptions. In addition, consideration should be given to extending the approach from the level of individual requirements to the entire requirements document. The most promising aspect, though, seems to be the implementation of deeper linguistic approaches that could allow for an automated assessment of additional characteristics of natural language requirements. A proper grammatical and deep analysis of the text should then identify quite precisely the quality of a requirement based on its characteristics.

Finally, the integration of our approach into requirements management tools (like IBM Rational DOORS, PTC Integrity, or codebeamer) is helpful to increase acceptance. Thereby, requirements can be analyzed immediately after the creation. The requirements engineer then receives hints about the quality characteristics and can decide which adjustments should be made. The consequences of a complete integration should be analyzed in detail. In particular, we need to understand whether the assessment enables requirements engineers to improve the quality of their requirements. These issues must be carefully evaluated in future research efforts and analyzed in practice. Then the main purpose of our research, to support quality assurance in requirements engineering, can further be realized.

Appendix A

A.1 Assessment sessions



Die Wertschöpfung in der Automobilindustrie verlagert sich zunehmend von der **Hardware** in Richtung **Software** und **Services**.
Intelligente Software und nützliche Services beeinflussen zunehmend die Kaufentscheidung der Kunden.
In diesem Zusammenhang sehen sich Automobilhersteller mit den Herausforderungen in der Spezifikation von Softwareanforderungen konfrontiert.

Der Ingenieursdienstleister unterstützt seit vielen Jahren Automobilhersteller und Zulieferer im Bereich des Anforderungsmanagements.
Als kompetenter und integraler Entwicklungspartner hat der Ingenieursdienstleister mit großem Erfolg relevante Standards, Methoden und Prozesse etabliert und weiterentwickelt.
Zur Bewältigung zukünftiger Herausforderungen entwickelt der Ingenieursdienstleister ein Tool zur automatisierten Qualitätsbewertung von Softwareanforderungen:

ReQuAI – Requirements Quality Analytics

Für die Entwicklung dieses Tools wird die Bewertung von Anforderungen durch Experten benötigt.
Im Folgenden beurteilen Sie zunächst, ob der angezeigte Text eine Anforderung oder Information darstellt.
Anschließend erfolgt die Bewertung der Anforderung und zugehörigen Qualitätskriterien anhand folgender Skala:

(1 = sehr schlecht — 5 = sehr gut)

Die Bewertung der Anforderungen erfolgt selbstverständlich anonym.

Bei Fragen wenden Sie sich an:
Patrick Kummler
Email: patrick.kummler@googlemail.com
Tel: 01515 / 51 34 128

Fig. A.1.: Introductory information for the assessment sessions with experts.

References

Abernethy, K., Kelly, J., Sobel, A., Kiper, J. D., & Powell, J. (2000). Technology transfer issues for formal methods of software specification. In *Proceedings of the Thirteenth Conference on Software Engineering Education and Training* (pp. 23–31).

Aggarwal, C. C., & Zhai, C. (2012a). An introduction to text mining. In *Mining Text Data* (pp. 1–10). Springer.

Aggarwal, C. C., & Zhai, C. (2012b). A survey of text classification algorithms. In *Mining Text Data* (pp. 163–222). Springer.

Aiken, L. R. (1983). Number of response categories and statistics on a teacher rating scale. *Educational and Psychological Measurement*, *43*(2), 397–401.

Alexander, I., & Stevens, R. (2002). *Writing better requirements*. Pearson Education.

Alshazly, A. A., Elfatatry, A. M., & Abougabal, M. S. (2014). Detecting defects in software requirements specification. *Alexandria Engineering Journal*, *53*(3), 513–527.

Ambler, S. W. (2004). *The object primer: Agile model-driven development with UML 2.0*. Cambridge University Press.

Ambriola, V., & Gervasi, V. (1997). Processing natural language requirements. In *Proceedings of the 12th IEEE International Conference on Automated Software Engineering* (pp. 36–45).

Amstad, T. (1987). *Wie verständlich sind unsere Zeitungen?* (PhD thesis). Universität Zürich.

Anda, B., & Sjøberg, D. I. (2002). Towards an inspection technique for use case models. In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering* (pp. 127–134).

Andreessen, M. (2011, August 20). *Why software is eating the world*. Retrieved May 5, 2021, from https://www.wsj.com/articles/SB10001424053111903480904%20576512250915629460

Antinyan, V. (2020). Revealing the complexity of automotive software. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 1525–1528).

Antinyan, V., & Staron, M. (2017a). Proactive reviews of textual requirements. In *Proceedings of the IEEE 24th International Conference on Software Analysis, Evolution and Reengineering* (pp. 541–545).

Antinyan, V., & Staron, M. (2017b). Rendex: A method for automated reviews of textual requirements. *Journal of Systems and Software*, *131*, 63–77.

Arellano, A., Zontek-Carney, E., & Austin, M. A. (2015). Frameworks for natural language processing of textual requirements. *International Journal on Advances in Systems and Measurements*, *8*, 230–240.

Artstein, R., & Poesio, M. (2008). Inter-coder agreement for computational linguistics. *Computational Linguistics*, *34*(4), 555–596.

Balzert, H. (2010). *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. Springer-Verlag.

Balzert, H. (2008). *Software-Management: Lehrbuch der Software-Technik*. Spektrum Akademischer Verlag.

Bao, Y., & Ishii, N. (2002). Combining multiple k-nearest neighbor classifiers for text classification by reducts. In *Proceedings of the International Conference on Discovery Science* (pp. 340–347).

Bearden, W. O., Netemeyer, R. G., & Mobley, M. F. (1993). *Handbook of Marketing Scales: Multi-Item Measures for Marketing and Consumer Behavior Research*. Sage Publications.

Belfo, F. (2012). People, organizational and technological dimensions of software requirements specification. *Procedia Technology*, *5*, 310–318.

Benington, H. D. (1983). Production of large computer programs. *Annals of the History of Computing*, *5*(4), 350–361.

Berman, S. J., & Bell, R. (2011). Digital transformation - Creating new business models where digital meets physical. *IBM Institute for Business Value*, 1–17.

Berry, D. M. (2007). Ambiguity in natural language requirements documents. In *Monterey Workshop* (pp. 1–7).

Berry, D. M., Bucchiarone, A., Gnesi, S., Lami, G., & Trentanni, G. (2006). A new quality model for natural language requirements specifications. In *Proceedings of the International Workshop on Requirements Engineering: Foundation for Software Quality* (pp. 1–12).

Berry, D. M., Kamsties, E., & Krieger, M. M. (2003). *From contract drafting to software specification: Linguistic sources of ambiguity* (tech. rep.). School of Computer Science, University of Waterloo.

Bhatia, J., Sharma, R., Biswas, K. K., & Ghaisas, S. (2013). Using grammatical knowledge patterns for structuring requirements specifications. In *Proceedings of the 3rd International Workshop on Requirements Patterns (RePa)* (pp. 31–34).

Bhattacherjee, A. (2012). *Social Science Research: Principles, Methods, and Practices* (2nd ed.).

Bi, Y., Bell, D., Wang, H., Guo, G., & Greer, K. (2004). Combining multiple classifiers using Dempster's rule of combination for text categorization. In *Proceedings* *of the International Conference on Modeling Decisions for Artificial Intelligence* (pp. 127–138).

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bloehdorn, S., & Moschitti, A. (2007). Combined syntactic and semantic kernels for text classification. In *European Conference on Information Retrieval* (pp. 307–318).

Boehm, B. W. (1984). Software engineering economics. *IEEE transactions on Software Engineering*, (1), 4–21.

Borgmann, M.-M. (2019, February 28). *BMW Group and Daimler AG to jointly develop next-generation technologies for automated driving*. Retrieved May 5, 2021, from https://www.press.bmwgroup.com/global/article/detail/T0292550EN/bmw-group-and-daimler-ag-to-jointly-develop-next-generation-technologies-for-automated-driving?language=en

Bortz, J., & Döring, N. (2007). *Forschungsmethoden und Evaluation für Human-und Sozialwissenschaftler*. Springer-Verlag.

Bowen, J. P., & Hinchey, M. G. (2006). Ten commandments of formal methods... ten years later. *Computer*, *39*(1), 40–48.

Boyd, S., Zowghi, D., & Farroukh, A. (2005). Measuring the expressiveness of a constrained natural language: An empirical study. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05)* (pp. 339–349).

Braude, E. J. (2000). *Software engineering: An object-oriented perspective*. John Wiley & Sons, Inc.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and Regression Trees*. CRC press.

Brennen, S., & Kreiss, D. (2014). Digitalization and digitization. *Culture Digitally*, *8*.

Brooks, F., & Kugler, H. J. (1987). No silver bullet - Essence and accidents of software engineering. *IEEE computer*, *20*(4), 10–19.

Brown, G., Widing, R. E., & Coulter, R. L. (1991). Customer evaluation of retail salespeople utilizing the SOCO scale: A replication, extension, and application. *Journal of the Academy of Marketing Science*, *19*(4), 347–351.

Broy, M. (2006). Challenges in automotive software engineering. In *Proceedings of the 28th International Conference on Software Engineering* (pp. 33–42).

Bucchiarone, A., Gnesi, S., & Pierini, P. (2005). Quality analysis of NL requirements: An industrial case study. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05)* (pp. 390–394).

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., & Grobler, J. (2013). API design for machine learning software: Experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*.

Burkacky, O., Deichmann, J., Doll, G., & Knochenhauer, C. (2018). Rethinking car software and electronics architecture. *McKinsey & Company*.

Carbonell, J. G. (1990). *Machine Learning: Paradigms and Methods*. Elsevier North-Holland, Inc.

Carew, D., Exton, C., & Buckley, J. (2005). An empirical investigation of the comprehensibility of requirements specifications. In *Proceedings of the International Symposium on Empirical Software Engineering, 2005.* (p. 10).

Carson, R. S. (2001). Keeping the focus during requirements analysis. In *Proceedings of the 11th International Symposium of the International Council on Systems Engineering (INCOSE)* (pp. 1–8).

Casamayor, A., Godoy, D., & Campo, M. (2010). Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. *Information and Software Technology*, *52*(4), 436–445.

Cerpa, N., & Verner, J. M. (2009). Why did your project fail? *Communications of the ACM*, *52*(12), 130–134.

Charette, R. N. (2009, February 1). *This Car Runs on Code*. Retrieved May 5, 2021, from https://spectrum.ieee.org/transportation/systems/this-car-runs-on-code

Chawla, N. V. (2005). *Data Mining and Knowledge Discovery Handbook* (O. Maimon & L. Rokach, Eds.). Springer US.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Chillarege, R., Kao, W.-L., & Condit, R. G. (1991). Defect Type and Its Impact on the Growth Curve. In *Proceedings of the 13th International Conference on Software Engineering* (pp. 246–255).

Chitkara, R., Ballhaus, W., Kliem, B., Berings, S., & Weiss, B. (2013). Spotlight on automotive: PwC semiconductor report. *PwC Technology Institute*.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*(1), 37–46.

Collin, J. (2015). Digitalization and dualistic IT. *IT Leadership in Transition: The Impact of Digitalization on Finnish Organizations*, 29–34.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. New York, Cambridge University Press.

Crosby, P. B. (1979). *Quality is free: The art of making quality certain* (Vol. 94). McGraw-Hill New York.

Curtis, B., Krasner, H., & Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM*, *31*(11), 1268–1287.

Davis, A. (1993). *Software Requirements: Objects, Functions and States* (2nd ed.). Prentice Hall.

Davis, A., Overmyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., Kincaid, G., Ledeboer, G., Reynolds, P., Sitaram, P., Ta, A., & Theofanos, M. (1993). Identifying and measuring quality in a software requirements specification. In *Proceedings of the First International Software Metrics Symposium* (pp. 141–152).

De Almeida Ferreira, D., & Da Silva, A. R. (2012). Formally specifying requirements with RSL-IL. In *Proceedings of the Eighth International Conference on the Quality of Information and Communications Technology* (pp. 217–220).

De Marneffe, M.-C., MacCartney, B., & Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of International Conference on Language Resources and Evaluation (LREC)* (pp. 449–454).

De Vries, E., Schoonvelde, M., & Schumacher, G. (2018). No longer lost in translation: Evidence that Google Translate works for comparative bag-of-words text applications. *Political Analysis*, *26*(4), 417–430.

de Buhr, J. (2015). Joint-Venture-Fantasie – Freund oder Feind? Daimler-Chef Dieter Zetsche über Google und Apple als Herausforderer sowie die Neuerfindung des Automobils. *DUB Unternehmer*, 1–6.

Denger, C., Berry, D. M., & Kamsties, E. (2003). Higher quality requirements specifications through natural language patterns. In *Proceedings of the 2003 Symposium on Security and Privacy* (pp. 80–90).

Desjardins, J. (2017, February 8). *How Many Millions of Lines of Code Does It Take?* Retrieved May 5, 2021, from https://www.visualcapitalist.com/millions-lines-of-code/

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Di Eugenio, B., & Glass, M. (2004). The kappa statistic: A second look. *Computational Linguistics*, *30*(1), 95–101.

Diess, H. (2020, January 16). *Autokonzern im Wandel*. Retrieved May 5, 2021, from https://www.handelsblatt.com/unternehmen/industrie/autokonzern-im - wandel - vw - chef - diess - wenn - wir - in - unserem - jetzigen - tempo - weitermachen-wird-es-sehr-eng/25441126.html

Diess, H. (2019, January 24). *The car will become a software product*. Retrieved May 5, 2021, from https://www.linkedin.com/pulse/car-become-software-product-herbert-diess

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, *10*(7), 1895–1923.

Diewald, G. (2012). *Die Modalverben im Deutschen: Grammatikalisierung und Polyfunktionalität* (Vol. 208). Walter de Gruyter.

Donaldson, B. (2007). *German: An Essential Grammar*. Taylor & Francis.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 194–202). Elsevier.

Drucker, P. (2014). *Innovation and Entrepreneurship*. Taylor & Francis.

Drummond, C., & Holte, R. C. (2003). C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. In *Workshop on Learning from Imbalanced Datasets* (pp. 1–8).

Duncan, J., Lulla, S., & Marshall, A. (2015). *Driving digital destiny - Digital Reinvention in automotive*. Retrieved May 5, 2021, from https://www.ibm.com/downloads/cas/6PNR9QXO

Ebert, C. (2014). *Systematisches Requirements Engineering: Anforderungen ermitteln, dokumentieren, analysieren und verwalten* (5th ed.). dpunkt. verlag.

Ebert, C. (2019). *Systematisches Requirements Engineering: Anforderungen ermitteln, dokumentieren, analysieren und verwalten* (6th ed.). dpunkt. verlag.

Ebert, C., & Favaro, J. (2017). Automotive Software. *IEEE Software*, *34*(3), 33–39.

Ebert, C., & Lederer, D. (2007). Dem Kostendruck begegnen-Effizienz nachhaltig steigern. *Automobil Elektronik*, 46–48.

Ebert, C., & Lederer, D. (2012, November 30). *Evolution und Trends in der E/E* [Der weg zum schnellen, wirksamen und trotzdem flexiblen agieren]. Retrieved May 5, 2021, from https://www.all-electronics.de/evolution-und-trends-in-der-ee/

Eroms, H.-W. (1992). Das deutsche Passiv in historischer Sicht. *Deutsche Syntax: Ansichten und Aussichten*, 225–249.

Fabbrini, F., Fusani, M., Gnesi, S., & Lami, G. (2000). Quality evaluation of software requirement specifications. In *Proceedings of the Software and Internet Quality Week 2000 Conference* (pp. 1–18).

Fabbrini, F., Fusani, M., Gnesi, S., & Lami, G. (2001). The linguistic approach to the natural language requirements quality: Benefit of the use of an automatic tool. In *Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop* (pp. 97–105).

Fagan, M. (2002). Design and code inspections to reduce errors in program development. In *Software Pioneers* (pp. 575–607). Springer.

Fantechi, A., Gnesi, S., Lami, G., & Maccari, A. (2003). Applications of linguistic techniques for use case analysis. *Requirements Engineering*, *8*(3), 161–170.

Felden, C. (2006). Extraktion, Qualitätssicherung und Klassifikation unstrukturierter Daten. *HMD-Praxis der Wirtschaftsinformatik*, *247*, 54–62.

Femmer, H. (2013). Reviewing Natural Language Requirements with Requirements Smells. In *Proceedings of 11th International Doctoral Symposium on Empirical Software Engineering* (pp. 1–8).

Femmer, H., Fernández, D. M., Wagner, S., & Eder, S. (2017). Rapid quality assurance with requirements smells. *Journal of Systems and Software*, *123*, 190–213.

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). *Learning from Imbalanced Data Sets*. Springer International.

Fernández, D. (2018). Supporting Requirements-Engineering Research That Industry Needs: The NaPiRE Initiative. *IEEE Software*, (1), 112–116.

Figueroa, R. L., Zeng-Treitler, Q., Kandula, S., & Ngo, L. H. (2012). Predicting sample size required for classification performance. *BMC Medical Informatics and Decision Making*, *12*(1), 1–10.

Flesch, R. (1948). A new readability yardstick. *Journal of Applied Psychology*, *32*(3), 221–233.

Fleuren, W. W. M., & Alkema, W. (2015). Application of text mining in the biomedical domain. *Methods*, *74*, 97–106.

Fowler, M., Beck, K., Brant, J., Opdyke, W., & Roberts, D. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley.

Frakes, W. B., & Baeza-Yates, R. (1992). *Information Retrieval: Data Structures and Algorithms*. Prentice Hall.

Fromm, H., Wambsganss, T., & Söllner, M. (2019). Towards a Taxonomy of Text Mining Features. *Twenty-Seventh European Conference on Information Systems (ECIS2019)*, 1–12.

Fuchs, N., & Schwitter, R. (1995). Specifying logic programs in controlled natural language. In *Workshop on Computational Logic for Natural Language Processing* (pp. 1–16).

Gallego, E., Chalé-Góngora, H.-G., Llorens, J., Fuentes, J., Álvarez, J., Génova, G., & Fraga, A. (2016). Requirements quality analysis: A successful case study in the industry. In *Proceedings of the International Conference on Complex Systems Design & Management* (pp. 187–201).

Gause, D., & Weinberg, G. (1989). *Exploring Requirements: Quality before Design* (Vol. 7). Dorset House New York.

Gawich, M., Badr, A., Hegazy, A., & Ismail, H. (2012). A methodology for ontology building. *International Journal of Computer Applications*, *56*(2), 39–45.

Génova, G., Fuentes, J. M., Llorens, J., Hurtado, O., & Moreno, V. (2013). A framework to measure and improve the quality of textual requirements. *Requirements Engineering*, *18*(1), 25–41.

Ghosh, S., Elenius, D., Li, W., Lincoln, P., Shankar, N., & Steiner, W. (2016). ARSENAL: Automatic requirements specification extraction from natural language. In *NASA Formal Methods Symposium* (pp. 41–46).

Glass, R. (1998). *Software Runaways*. Prentice Hall.

Glass, R. (2002). Sorting out software complexity. *Communications of the ACM*, *45*(11), 19–21.

Glass, R. (2006). The Standish report: Does it really describe a software crisis? *Communications of the ACM*, *49*(8), 15–16.

Glass, R., & Becker, P. (2003). *Facts and Fallacies of Software Engineering*. Addison-Wesley.

Glinz, M., & Wieringa, R. (2007). Stakeholders in requirements engineering. *IEEE Software*, *24*(2), 18–20.

Goldberg, Y., & Hirst, G. (2017). *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers.

Goldin, L., & Berry, D. M. (1997). AbstFinder, a prototype natural language text abstraction finder for use in requirements elicitation. *Automated Software Engineering*, *4*(4), 375–412.

Goodwin, T. (2015, March 4). *The Battle Is For The Customer Interface* (T. Crunch, Ed.). Retrieved May 5, 2021, from https://techcrunch.com/2015/03/03/in-the-age-of-disintermediation-the-battle-is-all-for-the-customer-interface/

Gross, A., & Doerr, J. (2012). What you need is what you get!: The vision of view-based requirements specifications. In *Proceedings of the 20th IEEE International Requirements Engineering Conference (RE)* (pp. 171–180).

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, *43*(5-6), 907–928.

Gwet, K. L. (2014). *Handbook of Inter-Rater Reliability: The Definitive Guide to Measuring The Extent of Agreement Among Raters* (4th ed.). Advanced Analytics.

Gwet, K. L. (2015). On Krippendorff's Alpha Coefficient [Revised on october 05, 2015].

Haist, F., & Fromm, H. (1991). *Qualität im Unternehmen: Prinzipien-Methoden-Techniken* (2nd ed.). Hanser.

Hakuli, S., & Krug, M. (2015). Virtuelle Integration. In *Handbuch Fahrerassistenzsysteme* (pp. 125–138). Springer.

Hall, A. (1997). What's the use of requirements engineering? In *Proceedings of ISRE'97: 3rd IEEE International Symposium on Requirements Engineering* (pp. 2–3).

Hanelt, A., Piccinini, E., Gregory, R. W., Hildebrandt, B., & Kolbe, L. M. (2015). Digital transformation of primarily physical industries-Exploring the impact of digital trends on business models of automobile manufacturers. In *Wirtschaftsinformatik* (pp. 1313–1327).

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Hayes, A. F., & Krippendorff, K. (2007). Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures*, *1*(1), 77–89.

Hearst, M. (2003, October 17). *What Is Text Mining?* Retrieved May 5, 2021, from http://people.ischool.berkeley.edu/~hearst/text-mining.html

Heck, P., & Parviainen, P. (2008). Experiences on analysis of requirements quality. In *Proceedings of the Third International Conference on Software Engineering Advances* (pp. 367–372).

Heinrich, L. J., Riedl, R., & Stelzer, D. (2014). *Informationsmanagement: Grundlagen, Aufgaben, Methoden*. Walter de Gruyter.

Herz, M. (2010). Exploring consumers' brand image perceptions with collages: Implications on Data Collection, Data Analysis and Mixed Method Approaches. In *Aktuelle Beiträge zur Markenforschung* (pp. 121–143). Springer.

Hickey, A. M., & Davis, A. M. (2003). Elicitation technique selection: How do experts do it? In *Proceedings of the 11th IEEE International Requirements Engineering Conference* (pp. 169–178).

Hinchey, M., & Coyle, L. (2012). *Conquering Complexity*. Springer Science & Business Media.

Holtmann, J., Meyer, J., & von Detten, M. (2011). Automatic validation and correction of formalized, textual requirements. In *Proceedings of the IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops* (pp. 486–495).

Hooks, I. (2001). Managing requirements. *NJIT Requirements Engineering Handout*, 1–8.

Hooks, I. (1994). Writing good requirements. In *INCOSE International Symposium* (pp. 1247–1253).

Hotho, A., Nürnberger, A., & Paaß, G. (2005). A brief survey of text mining. In *LDV Forum* (pp. 1–37).

Hu, H., Zhang, L., & Ye, C. (2010). Semantic-based requirements analysis and verification. In *Proceedings of the International Conference on Electronics and Information Engineering* (pp. 241–246).

Huang, Z., Chen, H., Hsu, C.-J., Chen, W.-H., & Wu, S. (2004). Credit rating analysis with support vector machines and neural networks: A market comparative study. *Decision Support Systems*, *37*(4), 543–558.

Huertas, C., & Juárez-Ramírez, R. (2013). Towards assessing the quality of functional requirements using English/Spanish controlled languages and context free grammar. In *Proceedings of the Third International Conference on Digital Information and Communication Technology and its Applications (DICTAP 2013)* (pp. 234–241).

Huq, K. T., Mollah, A. S., & Sajal, M. S. H. (2018). Comparative study of feature engineering techniques for disease prediction. In *Proceedings of the International Conference on Big Data, Cloud and Applications* (pp. 105–117).

Hussain, A., Mkpojiogu, E. O. C., & Mohmad Kamal, F. (2016). The role of requirements in the success or failure of software projects. In *International Review of Management and Marketing* (pp. 306–311).

Hussain, I., Ormandjieva, O., & Kosseim, L. (2007). Automatic quality assessment of SRS text by means of a decision-tree-based text classifier. In *Proceedings of the Seventh International Conference on Quality Software (QSIC 2007)* (pp. 209–218).

IABG. (1993, February 1). *V-Model - Lifecycle Process Model* (tech. rep.). IABG Information Technology.

Ibanez, M., & Rempp, H. (1996). *European user survey analysis* (tech. rep. TR95104). European Software Institute.

IEEE 1233. (1998). *Guide for Developing System Requirements Specifications* (tech. rep.). Institute of Electrical and Electronics Engineers.

IEEE 610. (1990). *Standard Glossary of Software Engineering Terminology* (tech. rep.). Institute of Electrical and Electronics Engineers.

IEEE 830. (1984). *Guide for Software Requirements Specifications* (tech. rep.). ANSI/IEEE.

IEEE 830. (1993). *Recommended Practice for Software Requirements Specifications* (tech. rep.). Institute of Electrical and Electronics Engineers.

IEEE 830. (1998). *Recommended Practice for Software Requirements Specifications* (tech. rep.). Institute of Electrical and Electronics Engineers.

Ikonomakis, M., Kotsiantis, S., & Tampakas, V. (2005). Text classification using machine learning techniques. *WSEAS Transactions on Computers*, *4*(8), 966–974.

Ilieva, M., & Ormandjieva, O. (2005). Automatic transition of natural language software requirements specification into formal presentation. In *Proceedings of the International Conference on Application of Natural Language to Information Systems* (pp. 392–397).

ISO 25010. (2010). *System and Software Quality Models* (tech. rep.). International Standard.

ISO 26262. (2011). *Road Vehicles - Functional Safety* (tech. rep.). International Standard.

ISO 29148. (2011). *Systems and Software Engineering - Requirements Engineering* (tech. rep.). International Standard.

ISO 9000. (2005). *Quality Management Systems - Fundamentals and Vocabulary* (tech. rep.). International Standard.

Jacobs, S. (1999). Introducing measurable quality requirements: A case study. In *Proceedings of the IEEE International Symposium on Requirements Engineering* (pp. 172–179).

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer New York.

Jimenez, S., Becerra, C., Gelbukh, A., & Gonzalez, F. (2009). Generalized mongue-elkan method for approximate text string comparison. In *Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics* (pp. 559–570).

Joachims, T. (2002). *Learning to Classify Text Using Support Vector Machines* (Vol. 668). Springer Science & Business Media.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning* (pp. 137–142).

Johnson, S. L., Safadi, H., & Faraj, S. (2015). The emergence of online community leadership. *Information Systems Research*, *26*(1), 165–187.

Jones, R. R. (1968). Differences in response consistency and subjects' preferences for three personality inventory response formats. In *Proceedings of the 76th Annual Convention of the American Psychological Association* (pp. 247–248).

Jørgensen, M., & Moløkken-Østvold, K. (2006). How large are software cost overruns? A review of the 1994 CHAOS report. *Information and Software Technology*, *48*(4), 297–301.

Juárez-Ramirez, R., Gómez-Ruelas, M., Gutiérrez, A. A., & Negrete, P. (2011). Towards improving user interfaces: A proposal for integrating functionality and usability since early phases. In *Proceedings of the International Conference on Uncertainty Reasoning and Knowledge Engineering* (pp. 119–123).

Jurafsky, D., & Martin, J. H. (2014). *Speech and Language Processing* (2nd ed.). Pearson Education.

Juran, J. M., Gryna, F. M., & Bingham, R. S. (1974). *Quality Control Handbook* (Vol. 3). McGraw-Hill New York.

Kaiya, H., & Saeki, M. (2006). Using domain ontology as domain knowledge for requirements elicitation. In *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)* (pp. 189–198).

Kamalrudin, M., Hosking, J., & Grundy, J. (2011). Improving requirements quality using essential use case interaction patterns. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE)* (pp. 531–540).

Kamata, M. I., & Tamai, T. (2007). How does requirements quality relate to project success or failure? In *15th IEEE International Requirements Engineering Conference (RE 2007)* (pp. 69–78).

Kamsties, E., Berry, D. M., Paech, B., Kamsties, E., Berry, D. M., & Paech, B. (2001). Detecting ambiguities in requirements documents using inspections. In *Proceedings of the First Workshop on Inspection in Software Engineering (WISE'01)* (pp. 68–80).

Kamsties, E., & Peach, B. (2000). Taming ambiguity in natural language requirements. In *Proceedings of the Thirteenth International Conference on Systems and Software Engineering and their Applications* (pp. 1–8).

Kane, G. C., Palmer, D., Nguyen-Phillips, A., Kiron, D., & Buckley, N. (2017). Achieving digital maturity. *MIT Sloan Management Review*, *59*(1).

Kassab, M., Neill, C., & Laplante, P. (2014). State of practice in requirements engineering: Contemporary data. *Innovations in Systems and Software Engineering*, *10*(4), 235–241.

Kasser, J., & Schermerhorn, R. (1994). Determining metrics for systems engineering. In *INCOSE International Symposium* (pp. 740–745).

Kasser, J., Scott, W., Tran, X.-L., & Nesterov, S. (2006). A proposed research programme for determining a metric for a good requirement. In *Proceedings of the Conference on Systems Engineering Research* (pp. 1–10).

Kibriya, A. M., Frank, E., Pfahringer, B., & Holmes, G. (2004). Multinomial naive bayes for text categorization revisited. In *Proceedings of the Australasian Joint Conference on Artificial Intelligence* (pp. 488–499).

Kiritani, K., & Ohashi, M. (2015). The success or failure of the requirements definition and study of the causation of the quantity of trust existence between stakeholders. *Procedia Computer Science*, *64*, 153–160.

Kitchenham, B., & Pfleeger, S. L. (2002). Principles of survey research: Part 2: Designing a survey. *ACM SIGSOFT Software Engineering Notes*, *27*(1), 18–20.

Kitchenham, B., & Pfleeger, S. L. (1996). Software quality: The elusive target. *IEEE Software*, *13*(1), 12–21.

Kiyavitskaya, N., Zeni, N., Mich, L., & Berry, D. M. (2008). Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. *Requirements Engineering*, *13*(3), 207–239.

Klein, J.-P., & Ferres, M. (2019, July 31). *Decoding the Autonomous Driving Landscape - Software will indeed eat the (automotive) world* (Firstmile, Ed.). Retrieved May 5, 2021, from https://medium.com/@firstmilevc/avlandscape-8a21491f1f54

Knauss, E., El Boustani, C., & Flohr, T. (2009). Investigating the impact of software requirements specification quality on project success. In *International Conference on Product-Focused Software Process Improvement* (pp. 28–42).

Kocerka, J., Krześlak, M., & Gałuszka, A. (2018). Analysing quality of textual requirements using natural language processing: A literature review. In *Proceedings of the 23rd International Conference on Methods & Models in Automation & Robotics (MMAR)* (pp. 876–880).

Kodratoff, Y. (1999). Knowledge discovery in texts: A definition, and applications. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems* (pp. 16–29).

Korosec, K. (2019, August 29). *Porsche expands on-demand subscription plans to four more cities* (TechCrunch, Ed.). Retrieved May 5, 2021, from https://techcrunch.com/2019/08/29/porsche-expands-on-demand-subscription-plans-to-four-more-cities/

Kotonya, G., & Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques*. Wiley Publishing.

Kovitz, B. L. (1999). *Practical Software Requirements: A Manual of Content and Style*. Manning Publications Co.

Krippendorff, K. (2004). *Content Analysis: An Introduction to Its Methodology* (2nd ed.). Sage.

Krippendorff, K. (2011). Computing Krippendorff's alpha-reliability, 1–10.

Krisch, J. (2014). Weak-Words und ihre Auswirkung auf die Qualität von Anforderungsdokumenten. *Softwaretechniktrends*, *34*, 14–15.

Krisch, J., Dick, M., Jauch, R., & Heid, U. (2016). A lexical resource for the identification of "weak words" in German specification documents. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)* (pp. 2846–2850).

Krisch, J., & Houdek, F. (2015). The myth of bad passive voice and weak words: An empirical investigation in the automotive industry. In *Proceedings of the IEEE 23rd International Requirements Engineering Conference (RE)* (pp. 344–351).

Krogstie, J., Lindland, O. I., & Sindre, G. (1995). Towards a deeper understanding of quality in requirements engineering. In *Proceedings of the International Conference on Advanced Information Systems Engineering* (pp. 82–95).

Krosnick, J. A., & Presser, S. (2010). Question and questionnaire design. In P. V. Marsden & J. D. Wright (Eds.), *Handbook of Survey Research* (pp. 263–313). Emerald Group Publishing.

Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. Springer Science & Business Media.

Kummler, P. (2017). Towards requirements analytics: A research agenda to model and evaluate the quality of unstructured requirements specifications. In *International Conference on Exploring Service Science* (pp. 197–209).

Kummler, P., Vernisse, L., & Fromm, H. (2018). How good are my requirements? A new perspective on the quality measurement of textual requirements. In *Proceedings of the 11th International Conference on the Quality of Information and Communications Technology (QUATIC)* (pp. 156–159).

Lan, H., Zhang, C., & Li, H. (2008). An open design methodology for automotive electrical/electronic system based on quantum platform. *Advances in Engineering Software*, *39*(6), 526–534.

Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 159–174.

Landis, J. R., & Koch, G. G. (1975). A review of statistical methods in the analysis of data arising from observer reliability studies (Part II). *Statistica Neerlandica*, *29*(4), 151–161.

Lehner, F. (1993). Quality control in software documentation based on measurement of text comprehension and text comprehensibility. *Information Processing & Management*, *29*(5), 551–568.

Lewis, D. D., & Ringuette, M. (1994). A comparison of two learning algorithms for text categorization. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval* (pp. 81–93).

Lindquist, C. (2005). Fixing the requirements mess. *CIO Magazine*, *2005*, 52–60.

Lipton, Z. C., Elkan, C., & Naryanaswamy, B. (2014). Optimal thresholding of classifiers to maximize F1 measure. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 225–239).

Liu, H., Hussain, F., Tan, C. L., & Dash, M. (2002). Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, *6*(4), 393–423.

Lozano, L. M., García-Cueto, E., & Muñiz, J. (2008). Effect of the number of response categories on the reliability and validity of rating scales. *Methodology*, *4*(2), 73–79.

Ma, H.-B., Geng, Y.-B., & Qiu, J.-R. (2011). Analysis of three methods for web-based opinion mining. In *Proceedings of the International Conference on Machine Learning and Cybernetics* (pp. 915–919).

MacDuffie, J. P., & Fujimoto, T. (2010). Why dinosaurs will keep ruling the auto industry. *Harvard Business Review*, *88*(6), 23–25.

Macías, B., & Pulman, S. G. (1995). *Natural-language processing and requirements specifications* (tech. rep. UCAM-CL-TR-373). University of Cambridge, Computer Laboratory.

Malhotra, N. K. (2006). Questionnaire design and scale development. *The Handbook of Marketing Research: Uses, Misuses, and Future Advances*, 83–94.

Mann, K., West, M., & Wilson, N. (2017, August 17). *Hype Cycle for Application Development*. Gartner Research. Retrieved May 5, 2021, from https://www.gartner.com/doc/3779164/hype-cycle-application-development

Manning, C. D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press.

Marasco, J. (2007, June 26). *What Is the Cost of a Requirement Error?* Retrieved May 5, 2021, from https://www.stickyminds.com/article/what-cost-requirement-error

Marsden, P. V., & Wright, J. D. (2010). *Handbook of Survey Research*. Emerald Group Publishing.

Marseguerra, M. (2014). Early detection of gradual concept drifts by text categorization and support vector machine techniques: The TRIO algorithm. *Reliability Engineering & System Safety*, *129*, 1–9.

Mather, P. M., & Tso, B. (2016). *Classification Methods for Remotely Sensed Data* (2nd ed.). CRC Press.

McCallum, A., & Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization* (pp. 41–48).

McCallum, D., Keith, B. R., & Wiebe, D. J. (1988). Comparison of response formats for mulitdimensional health locus of control scales: Six levels versus two levels. *Journal of Personality Assessment*, *52*(4), 732–736.

McConnell, S. (1996). *Rapid Development: Taming Wild Software Schedules*. Pearson Education.

Mehler, A., & Wolff, C. (2005). Einleitung: Perspektiven und Positionen des Text Mining. In *LDV-Forum* (pp. 1–18).

Meissner, F., Shirokinskiy, K., & Alexander, M. (2020, January 1). *Computer on wheels: Disruption in automotive electronics and semiconductors* (tech. rep.). Roland Berger.

Mich, L., Franch, M., & Novi Inverardi, P. (2004). Market research for requirements analysis using linguistic tools. *Requirements Engineering*, *9*(1), 40–56.

Mich, L., & Garigliano, R. (2000). Ambiguity measures in requirement engineering. In *Proceedings of the International Conference on Software Theory and Practice* (pp. 39–48).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, *63*(2), 81–97.

Miner, G., Elder, J., Fast, A., Hill, T., Nisbet, R., & Delen, D. (2012). *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications*. Academic Press.

Mitchell, T. M. (2006). The Discipline of Machine Learning. Carnegie Mellon University, School of Computer Science.

Mitkov, R. (2004). *The Oxford Handbook of Computational Linguistics*. OUP Oxford.

Moffett, J. (1999). Requirements and policies. In *Position Paper for Policy Workshop*.

Moravek, M. (2020, March 19). *Unlocking vehicle data and creating value for both the vehicle manufacturer and driver*. Retrieved May 5, 2021, from https://medium.com/engineering-data-economies/unlocking-vehicle-data-and-creating-value-for-both-the-vehicle-manufacturer-and-driver-cfd2d91121fb

Müller, A. C., & Guido, S. (2016). *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media.

Muller, K.-R., Mika, S., Ratsch, G., Tsuda, K., & Scholkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, *12*(2), 181–201.

Naeem, A., Aslam, Z., & Shah, M. A. (2019). Analyzing quality of software requirements: A comparison study on nlp tools. In *Proceedings of the 25th International Conference on Automation and Computing (ICAC)* (pp. 1–6).

Nassirtoussi, A. K., Aghabozorgi, S., Wah, T. Y., & Ngo, D. C. L. (2014). Text mining for market prediction: A systematic review. *Expert Systems with Applications*, *41*(16), 7653–7670.

Nikora, A., Hayes, J., & Holbrook, E. (2010). Experiments in automated identification of ambiguous natural-language requirements. In *Proceedings of 21st IEEE International Symposium on Software Reliability Engineering* (pp. 1–10).

O'Muircheartaigh, C., Krosnick, J. A., & Helic, A. (2000). Middle alternatives, acquiescence, and the quality of questionnaire data, 1–48.

Opitz, J., & Burst, S. (2019). Macro f1 and macro f1. *arXiv preprint arXiv:1911.03347*.

Ormandjieva, O., Hussain, I., & Kosseim, L. (2007). Toward a text classification system for the quality assessment of software requirements written in natural language. In *Proceedings of the Fourth International Workshop on Software Quality Assurance (SOQUA 2007)* (pp. 39–45).

Özgür, A., Özgür, L., & Güngör, T. (2005). Text categorization with class-based and corpus-based keyword selection. In *International Symposium on Computer and Information Sciences* (pp. 606–615).

Palvia, P., Leary, D., Mao, E., Midha, V., Pinjani, P., & Salam, A. F. (2004). Research methodologies in MIS: An update. *The Communications of the Association for Information Systems*, *14*(1), 526–542.

Parasuraman, A., Zeithaml, V. A., & Berry, L. L. (1988). SERVQUAL: A multiple-item scale for measuring consumer perceptions of service quality. *Journal of Retailing*, *64*(1), 12–40.

Parra, E., de la Vara, J. L., & Alonso, L. (2018). Analysis of requirements quality evolution. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings* (pp. 199–200).

Parra, E., Dimou, C., Llorens, J., Moreno, V., & Fraga, A. (2015). A methodology for the classification of quality of requirements using machine learning techniques. *Information and Software Technology*, *67*, 180–195.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., & Dubourg, V. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pekar, V., Felderer, M., & Breu, R. (2014). Improvement methods for software requirement specifications: A mapping study. In *Proceedings of the 9th International Conference on the Quality of Information and Communications Technology* (pp. 242–245).

Pelayo, L., & Dick, S. (2007). Applying novel resampling strategies to software defect prediction. In *Annual Meeting of the North American Fuzzy Information Processing Society* (pp. 69–72).

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543).

Peter, J. P. (1979). Reliability: A review of psychometric basics and recent marketing practices. *Journal of Marketing Research*, *16*(1), 6–17.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pirsig, R. M. (1999). *Zen and the art of motorcycle maintenance: An inquiry into values*. Random House.

Pohl, K. (2008). *Requirements Engineering: Grundlagen, Prinzipien, Techniken*. dpunkt-Verlag.

Pohl, K., & Rupp, C. (2015). *Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level*. dpunkt-Verlag.

Polpinij, J. (2009). An ontology-based text processing approach for simplifying ambiguity of requirement specifications. In *Proceedings of the IEEE Asia-Pacific Services Computing Conference (APSCC)* (pp. 219–226).

Porter, M. F. (1980). An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, *14*(3), 130–137.

Powers, D. M. (2011). Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, *2*(1), 37–63.

Press Release. (2019, October 31). *Groupe PSA and FCA plan to join forces to build a world leader for a new era in sustainable mobility*. Retrieved June 24, 2020, from https://media.groupe-psa.com/en/groupe-psa-and-fca-plan-join-forces-build-world-leader-new-era-sustainable-mobility

Preston, C. C., & Colman, A. M. (2000). Optimal number of response categories in rating scales: Reliability, validity, discriminating power, and respondent preferences. *Acta Psychologica*, *104*(1), 1–15.

Purda, L., & Skillicorn, D. (2015). Accounting variables, deception, and a bag of words: Assessing the tools of fraud detection. *Contemporary Accounting Research*, *32*(3), 1193–1223.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, *1*(1), 81–106.

Quinlan, J. R. (1996). Learning Decision Tree Classifiers. *ACM Computing Surveys (CSUR)*, *28*(1), 71–72.

Ramos, J. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning* (pp. 133–142).

Reidel, M. (2016, March 16). *So will sich die BMW Group für die Zukunft rüsten*. Retrieved May 5, 2021, from https://www.horizont.net/marketing/nachrichten/Number-One-Next-So-ruestet-sich-die-BMW-Group-fuer-die-Zukunft-139338

Rennie, J. D., Shih, L., Teevan, J., & Karger, D. R. (2003). Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th International Conference on Machine Learning* (pp. 616–623).

Rolland, C., & Proix, C. (1992). A natural language approach for requirements engineering. In *Proceedings of the International Conference on Advanced Information Systems Engineering* (pp. 257–277).

Rosenberg, J. (2008). *Statistical Methods and Measurement*. Springer.

Rost, J. (2004). *Lehrbuch Testtheorie - Testkonstruktion*. Hans Huber.

Royce, W. W. (1987). Managing the development of large software systems: Concepts and techniques. In *Proceedings of the 9th International Conference on Software Engineering* (pp. 328–338).

RTI. (2002). *The Economic Impacts of Inadequate Infrastructure for Software Testing* (tech. rep.). RTI - Health, Social, and Economics Research.

Rupp, C. (2014). *Requirements-Engineering und-Management: Aus der Praxis von klassisch bis agil* (6th ed.). Carl Hanser Verlag GmbH Co KG.

Rupp, C., & Goetz, R. (2000). Linguistic methods of requirements-engineering (NLP). In *Proceedings of the European Software Process Improvement Conference (EuroSPI)* (pp. 7–11).

Ryan, K. (1993). The role of natural language in requirements engineering. In *Proceedings of the IEEE International Symposium on Requirements Engineering* (pp. 240–242).

Saavedra, R., Ballejos, L. C., & Ale, M. A. (2013). Software requirements quality evaluation: State of the art and research challenges. In *Proceedings of the 14th Argentine Symposium on Software Engineering* (pp. 240–257).

Sabriye, A., & Zainon, W. M. N. (2018). An approach for detecting syntax and syntactic ambiguity in software requirement specification. *Journal of Theoretical and Applied Information Technology*, *96*(8), 2275–2284.

Saito, S., Takeuchi, M., Hiraoka, M., Kitani, T., & Aoyama, M. (2013). Requirements clinic: Third party inspection methodology and practice for improving the quality of software requirements specifications. In *Proceedings of the 21st IEEE International Requirements Engineering Conference (RE)* (pp. 290–295).

Salger, F. (2013). Requirements reviews revisited: Residual challenges and open research questions. In *Proceedings of the 21st IEEE International Requirements Engineering Conference (RE)* (pp. 250–255).

Salger, F., Engels, G., & Hofmann, A. (2009). Inspection effectiveness for different quality attributes of software requirement specifications: An industrial case study. In *Proceedings of the ICSE Workshop on Software Quality* (pp. 15–21).

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523.

Sarkar, D. (2016). *Thermal Power Plant: Pre-Operational Activities*. Elsevier Science.

Sathyadevan, S., & Nair, R. R. (2015). Comparative analysis of decision tree algorithms: ID3, C4. 5 and random forest. In *Computational Intelligence in Data Mining* (pp. 549–562). Springer.

Saunders, M., Lewis, P., & Thornhill, A. (2012). *Research Methods for Business Students*. Pearson Education Limited.

Scheffczyk, J., Borghoff, U. M., Birk, A., & Siedersleben, J. (2005). Pragmatic consistency management in industrial requirements specifications. In *Proceedings of the Third IEEE International Conference on Software Engineering and Formal Methods (SEFM'05)* (pp. 272–281).

Schneider, F., & Berenbach, B. (2013). A literature survey on international standards for systems requirements engineering. *Procedia Computer Science*, *16*, 796–805.

Scott, W. A. (1955). Reliability of content analysis: The case of nominal scale coding. *Public Opinion Quarterly*, *19*(3), 321–325.

Seiberth, G. (2015). *Wie verändern digitale Plattformen die Automobilwirtschaft* (Accenture, Ed.). Retrieved May 5, 2021, from http://plattform-maerkte.de/wp-content/uploads/2015/10/Gabriel-Seiberth-Accenture.pdf

Seiniger, P., & Weitzel, A. (2015). Testverfahren für Verbraucherschutz und Gesetzgebung. In *Handbuch Fahrerassistenzsysteme* (pp. 167–182). Springer.

Shah, U. S., & Jinwala, D. C. (2015). Resolving ambiguities in natural language software requirements: A comprehensive survey. *ACM SIGSOFT Software Engineering Notes*, *40*(5), 1–7.

Sharda, R., & Delen, D. (2006). Predicting box-office success of motion pictures with neural networks. *Expert Systems with Applications*, *30*(2), 243–254.

Shaw, M. E., Hawley, G. G., & Wright, J. M. (1967). *Scales for the Measurement of Attitudes*. McGraw-Hill.

Sikora, E., Tenbergen, B., & Pohl, K. (2012). Industry needs and research directions in requirements engineering for embedded systems. *Requirements Engineering*, *17*(1), 57–78.

Silva, C., & Ribeiro, B. (2003). The importance of stop word removal on recall values in text categorization. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 1661–1666).

Simon, H. A. (1981). *The Sciences of the Artificial* (2nd ed.). Cambridge, MA.

Singh, S. (2015). *Critical reasons for crashes investigated in the national motor vehicle crash causation survey* (tech. rep.). National Highway Traffic Safety Administration.

Soeken, M., Abdessaied, N., Allahyari-Abhari, A., Buzo, A., Musat, L., Pelz, G., & Drechsler, R. (2014). Quality assessment for requirements based on natural language processing. In *Proceedings of the Forum on Specification and Design Languages* (pp. 1–4).

Søgaard, A. (2013). Semi-supervised learning and domain adaptation in natural language processing. *Synthesis Lectures on Human Language Technologies*, *6*(2), 1–103.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427–437.

Somers, J. (2017). The coming software apocalypse. *The Atlantic*, *26*.

spaCy. (2020). *Industrial-Strength Natural Language Processing*. Retrieved May 5, 2021, from https://spacy.io

Statnikov, A., Aliferis, C. F., Tsamardinos, I., Hardin, D., & Levy, S. (2005). A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics*, *21*(5), 631–643.

Stevens, S. S. (1946). On the Theory of Scales of Measurement. *Science New Series*, *103*(2684), 677–680.

Suma, V., & Gopalakrishnan Nair, T. R. (2009). Defect management strategies in software development (M. A. Strangio, Ed.). *InTech*, 379–404.

Swamynathan, M. (2019). *Mastering Machine Learning with Python in Six Steps: A Practical Implementation Guide to Predictive Data Analytics Using Python*. Apress.

Terzakis, J. (2013). The impact of requirements on software quality across three product generations. In *Proceedings of the 21st IEEE International Requirements Engineering Conference (RE)* (pp. 284–289).

The Standish Group. (2006). *The Chaos Report* (tech. rep.). Standish Group.

Tjong, S. F., Hallam, N., & Hartley, M. (2006). Improving the quality of natural language requirements specifications through natural language requirements patterns. In *Proceedings of the the Sixth IEEE International Conference on Computer and Information Technology (CIT'06)* (p. 199).

Tobin, A. (2019, July 14). *Ford And Volkswagen Enter Self-Driving Car Joint Venture*. Retrieved May 5, 2021, from https://www.forbes.com/sites/annatobin/2019/07/14/ford-and-volkswagen-enter-self-driving-car-joint-venture/#48c574347cc8

Toutanova, K., Klein, D., Manning, C., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology* (pp. 173–180).

Tsang, K. K. (2012). The use of midpoint on Likert Scale: The implications for educational research. *Hong Kong Teachers' Centre Journal*, *11*(1), 121–130.

Unterkalmsteiner, M., & Gorschek, T. (2017). Requirements quality assurance in industry: Why, what and how? In P. Grünbacher & A. Perini (Eds.), *Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality* (pp. 77–84). Springer.

Uysal, A. K., & Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management*, *50*(1), 104–112.

Van Lamsweerde, A. (2009). *Requirements Engineering: From System Goals to UML Models to Software Specifications* (Vol. 10). John Wiley & Sons.

Van Rijsbergen, C. J. (1979). *Information Retrieval* (2nd ed.). Butterworths.

Varghese, A., Agyeman-Badu, G., & Cawley, M. (2020). Deep learning in automated text classification: A case study using toxicological abstracts. *Environment Systems and Decisions*, *40*(4), 465–479.

Velupillai, S., Dalianis, H., Hassel, M., & Nilsson, G. H. (2009). Developing a standard for de-identifying electronic patient records written in Swedish: Precision, recall and f-measure in a manual and computerized annotation trial. *International Journal of Medical Informatics*, *78*(12), e19–e26.

Verma, K., & Kass, A. (2008). Requirements analysis tool: A tool for automatically analyzing software requirements documents. In *Proceedings of the International Semantic Web Conference* (pp. 751–763).

Volkswagen. (2020). *Over-the-air-Technik wird Elektroautos jung halten*. Retrieved June 20, 2020, from https://www.volkswagen.at/elektroauto/id-magazin/technologie/over-the-air-technik

Vor der Brück, T., & Leveling, J. (2007). Parameter learning for a readability checking tool. In *Proceedings of LWA 2007: Lernen - Wissen - Adaption* (pp. 149–153).

Wang, T.-Y., & Chiang, H.-M. (2007). Fuzzy support vector machine for multi-class text categorization. *Information Processing & Management*, *43*(4), 914–929.

Weber, M., & Weisbrod, J. (2002). Requirements engineering in automotive development-experiences and challenges. In *Proceedings of the IEEE Joint International Conference on Requirements Engineering* (pp. 331–340).

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. JSTOR.

Webster, J., & Kit, C. (1992). Tokenization as the initial phase in NLP. In *Proceedings of the 14th Conference on Computational Linguistics* (pp. 1106–1110).

Weihs, C., Mersmann, O., & Ligges, U. (2013). *Foundations of Statistical Algorithms: With References to R Packages*. CRC Press.

Weiss, S. M., Indurkhya, N., & Zhang, T. (2015). *Fundamentals of Predictive Text Mining* (2nd ed.). Springer London.

Weng, L.-J. (2004). Impact of the number of response categories and anchor labels on coefficient alpha and test-retest reliability. *Educational and Psychological Measurement*, *64*(6), 956–972.

Wiegers, K. (2003). *Software Requirements* (2nd ed.). Microsoft Press.

Wilson, W. M., Rosenberg, L. H., & Hyatt, L. E. (1997). Automated analysis of requirement specifications. In *Proceedings of the 19th International Conference on Software Engineering (ICSE)* (pp. 161–171). ACM Press, New York.

Wohlin, C., & Aurum, A. (2015). Towards a decision-making structure for selecting a research design in empirical software engineering. *Empirical Software Engineering*, *20*(6), 1427–1455.

Yang, H., De Roeck, A., Gervasi, V., Willis, A., & Nuseibeh, B. (2012). Speculative requirements: Automatic detection of uncertainty in natural language requirements. In *Proceedings of the 20th IEEE International Requirements Engineering Conference (RE)* (pp. 11–20).

Yang, H., Willis, A., De Roeck, A., & Nuseibeh, B. (2010). Automatic detection of nocuous coordination ambiguities in natural language requirements. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering* (pp. 53–62).

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 42–49).

Zhang, Y., Jin, R., & Zhou, Z.-H. (2010). Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics*, *1*(4), 43–52.

Zheng, A., & Casari, A. (2018). *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media.

List of Figures

List of Tables

Colophon

This thesis was typeset with $\LaTeX\,2_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at http://cleanthesis.der-ric.de/.