Improving Technical Reviews: Metrics, Conditions for Quality, and Linguistic Considerations to Minimize Errors

Gabriella Tóth

***Abstract*—This paper presents a systematic methodology for evaluating the quality of technical reviews in software development, aiming to address the issue of ineffective reviews and their impact on product quality. The methodology, grounded in literature review, identifies key factors for high-quality reviews, including planning, reviewer commitment, and meaningful feedback. It emphasizes linguistic clarity and accuracy, drawing lessons from the Therac-25 case, where poor documentation contributed to fatal accidents. The paper analyzes specific linguistic issues like negative structures, passive voice, and terminology, highlighting their impact on comprehension. The methodology's effectiveness is validated through a Lean Six Sigma project in a large software development company, resulting in significant improvements. These include a 60% improvement in the Technical Review Quality KPI, a 29% reduction in customer-reported faults related to reviews, and the elimination of TL9000 non-conformities. This case study demonstrates the practical applicability of the framework and its potential for significant impact. The paper concludes by highlighting the importance of linguistic considerations in ensuring safer and more effective software products. Future research directions include extending the methodology to other types of artifacts and exploring textual analysis of reviewer feedback for deeper insights into the review process.**

***Index Terms*—Documentation, Grammar, Manuals, Quality management, Reviews, Writing**

I. INTRODUCTION

HAVING any kind of text or other artefacts checked by (at least) a second pair of eyes is a crucial step in the software development process. This does not happen differently in technical documents either. We can even say that for technical documents this is particularly critical as inaccuracies in content or unclear language and style can lead to misunderstandings, improper product use, system failures, or safety risks.

There are several terminologies used for the activity that include the observation and evaluation of software or documentation conducted by experts other than the author. In this paper I am using the term “technical review” in accordance with the terminology defined in [1]. According to this standard, a technical review is a “systematic evaluation of a software product by a team of qualified personnel that examines the suitability of the software product for its intended use and identifies discrepancies from specifications and standards. Technical reviews may also provide recommendations of alternatives and examination of various alternatives”. To clarify further, a “software product” is: “(A) A complete set of computer programs, procedures, and associated documentation and data. (B) One or more of the individual items in (A)”. Therefore, technical documents are software products, and they are checked in technical reviews, as opposed to “audits”, “inspections” or “walk-throughs”.

The concept of "error" permeates various domains, often used interchangeably with terms like "fault," "failure," "defect," or "mistake." For the purposes of this paper, an error is defined as any incorrect, misleading, ambiguous, inconsistent, outdated, or missing information in a technical document that can negatively impact user understanding, product use, or downstream processes.

There are several examples in the literature that highlight the problem of errors escaping the review or testing phase in product or system development. Some of these are in industries where such errors might lead to very severe consequences (severe injuries or even loss of life).

One example is the case of the Therac-25 software-controlled radiation therapy machine where software errors and poor documentation played a role in tragic accidents [2]. Although reviews were not specifically mentioned but at the time of the events there was a lack of code or documentation review practices. We can safely assume that with conducting technical reviews the most significant errors could have been eliminated and the machine operators would have had proper support for operating the machine.

Therefore, ensuring the quality of technical reviews is essential. Organizations frequently report that their review processes are ineffective, expressing concern over the recurrence of undetected errors that are ultimately present in the published documentation. These documents may serve both internal stakeholders (such as software developers or testers) and external audiences (including client organizations or end users of technical products). While acknowledging these concerns is important, identifying concrete strategies for improving the effectiveness of technical reviews is even more critical.

Submitted on 29.01.2025.

Gabriella Tóth is with the Doctoral School of Linguistics, University of Debrecen; also with the Department of English Linguistics, Institute of English and American Studies, University of Debrecen; and with Nokia Solutions and Networks, Budapest, Hungary. (e-mail: gabriella.toth.szakal@gmail.com).

DOI: 10.36244/ICJ.2025.3.8

Similarly to other data-driven analysis and improvements facts are needed first to understand what the problem is. As in [3], "Measurement is the first step that leads to control and eventually to improvement. If you can't measure something, you can't understand it. If you can't understand it, you can't control it. If you can't control it, you can't improve it."

"Lord Kelvin's statement that "one does not understand what one cannot measure" is at least as true for software engineering as it is for any other engineering disciplines." [4].

Despite their critical role technical reviews have rarely been systematically measured, analyzed, or improved. This paper addresses this significant gap by introducing a novel framework for evaluating and enhancing the quality of technical reviews, specifically in the context of technical documentation. Through a Lean Six Sigma project, this research demonstrates how a data-driven approach can transform review processes, substantially reduce errors, and improve compliance with industry standards. The implementation of a Technical Review Quality KPI, combined with mindset-shifting initiatives like gamification, led to measurable improvements: a 60% increase in the composite review quality score and a 29% reduction in customer-reported faults. These outcomes underscore the practical value of applying engineering rigor to a process often overlooked in systematic quality improvement efforts. The findings and methodologies presented here offer actionable insights and a reusable approach for organizations aiming to elevate the quality of their review processes.

II. Background and previous literature

One of the most renowned experts in information design, has extensively discussed the importance of technical reviews and quality assurance in documentation [5]. She argues that effective technical reviews should not only focus on the accuracy of content but also on the clarity and accessibility of language, ensuring documents are comprehensible to their target audience.

The following aspects are emphasized in [5] related to the quality of technical reviews: error detection rate and rate of missed errors, review coverage (the percentage of documents that have undergone formal review out of the total number of documents produced. It indicates how thoroughly the review process has been applied across a given documentation set), timeliness, reviewer expertise, consistency and compliance with standards and user feedback. However, while these factors underscore the multifaceted nature of review quality, [5] does not offer concrete guidance on how such elements should be systematically measured or operationalized. There is little elaboration on how metrics could be designed, collected, or applied in practice to support consistent evaluation across projects.

In addition to [5], several authors emphasized the importance of reviews for technical documents, similarly to inspections of software code that started already in the 1970s [6], [7].

There was a significant contribution to technical reviews in [8], particularly in the field of software engineering, with his development of the Fagan inspection process in the 1970s. His method was one of the first formalized, structured approaches for software review processes and focused on improving product quality and reducing defects through systematic reviews.

The concept of defect categorization for code reviews was first included in [8] to some extent, but no details were presented. While this foundational process is relevant as context, the present study does not aim to elaborate on formal inspection methodologies. Instead, it builds on their legacy by focusing specifically on the measurement of review quality in technical documentation, an area where structured metrics are still underdeveloped.

A work in 2009, [9], looked at the literature of code review error classification and concluded that before their work there had not been a complete system of code review defect categorization and the focus had been on defect counts instead of type of errors. They defined a taxonomy of defects identified during code reviews, distinguishing between functional defects - such as logic, interface, or timing issues - and evolvability defects, which affect maintainability, readability, and clarity.

Measuring the effectiveness of technical reviews already appeared in [8], both for code and document reviews, and there have been several studies on the topic but they mostly concentrated on Defect Detection Efficiency or Defect Density (e.g. [8], [10], [11]).

I will also refer to previous literature as part of the following sections regarding specific topics.

A. Errors Escaping the Technical Review Phase

Before we can even start thinking about possible improvements, we must look at the following facts:

1. How many errors escape from the technical review phase?
2. What is the quality of technical reviews?

Errors that escape the technical review phase can be found when one analyzes the errors that are reported by users. For each of the reported errors, it must be defined which phase of the development process the error should have been found. The technical review phase should be considered a suitable point for addressing this.

In the rest of the paper, I will not cover further details of error analysis and categorization but will focus on technical review quality.

B. Defining Review Quality

It is a relatively challenging task to have a universally agreed measurement of the quality of technical reviews. It might be impacted by the industry, technology, type of artefacts (technical documents, software code, project plans, test plans, etc.) or other factors.

I am using "effectiveness" instead of "quality" for the metrics quoted in earlier literature (Defect Detection Efficiency or Defect Density) as I consider these basic metrics to reflect the observable outcomes or measurable results of the review

process - what could be termed the "facts" of the review but not the quality of the review.

Defect Detection Efficiency is usually defined as the ratio of defects found during reviews and the total number of defects in the reviewed software product. Defect Density is usually measured per 1000 lines of code, reviewing hour or document page. In our experience, these do not show how well the review "performs" in reducing the errors escaping from the review phase. First, it is difficult to measure the total number of defects (are those the ones found later before delivery or after delivery, by the user?), also argued in [4] and second, the increase in defect density alone does not necessarily result in better reviews (it is also important to check what kind of defects are found). However, these metrics are useful for following up and later planning the needed efforts for reviews, which is an important aspect for Project Managers.

It was proposed in [4] that review time and duration are also considered in the measurement, as part of the cost: engineering time benefits and costs. This study did not give exact definitions but in the current context I would consider as following. Reviewer time is the total time invested by reviewers in planning, preparation, and conducting document reviews. Engineering time benefits are the estimated time savings in activities like reduced rework, fewer errors, or faster development cycles.

In [12], reviewer expertise was addressed through perspective-based reviews, where individuals evaluate the document from roles aligned with their specific competence—such as user, developer, or tester perspectives—to improve review effectiveness.

While the above studies discussed quality metrics for reviews, none of them proposed a systematic way of measurement that would include aspects other than defect density.

My objective in this paper is to show how such a measurement can be set up for documents that are created during software development.

C. Recent Developments in Automated and AI-Assisted Technical Reviews

In the past decade, technical documentation review practices have evolved significantly through the adoption of automation and documentation-as-code (DaC) workflows. Traditional peer review methods—where subject matter experts manually assess user guides or interface documents—are increasingly supplemented by CI/CD-integrated pipelines and version-controlled documentation. In this approach, documentation is written in lightweight markup languages (e.g., AsciiDoc), stored in Git, and reviewed via pull requests, similar to software code. In [13] the authors demonstrated a DaC-based pipeline for managing Interface Control Documents (ICDs), integrating CI tools that validate document structure, automate glossary generation, and enforce writing style rules through custom quality gates. Their work showcases how such pipelines can reduce documentation errors, ensure uniformity, and improve maintainability in complex engineering environments. These developments have improved traceability and efficiency, especially in agile and DevOps contexts. Review feedback is increasingly structured and tracked within issue systems, and changes can be linked directly to product development cycles. At the same time, Natural Language Processing (NLP) techniques are increasingly applied to support high-level documentation review tasks such as detecting vagueness, redundancy, or omissions in technical texts. A set of machine learning and deep learning models was developed and evaluated - including BERT - to automatically identify five types of "documentation smells" in API references, such as overly vague descriptions, overly technical detail, and fragmented or tangled content [14]. While these tools can significantly reduce reviewer workload and enhance consistency, the authors emphasize that they are best used in combination with human judgment, especially when domain-specific knowledge is required.

It is important to note that the present study does not focus on these recent technological developments. The analysis and findings reflect the current documentation review practices implemented within the specific organizational context in which the study was conducted. As such, the study concentrates on established manual review methods, as supported by existing literature. Nonetheless, the integration of automated and AI-assisted review processes represents an important and timely area for future research, particularly as these tools continue to evolve and gain adoption across technical domains.

III. CONDITIONS FOR A GOOD TECHNICAL REVIEW

It is important to explain the scenario for which the measurement is described. In this study, the term "technical documents" refer specifically to product-related documentation authored by technical writers—such as user guides, operating manuals, and reference documentation—that support key product milestones. The measurement framework focuses on technical reviews of such documents, typically organized and facilitated by technical writers, and conducted with input from subject matter experts.

There is no golden rule for defining what makes a technical review good or of expected quality. One approach is to select the key factors that need to be satisfied and measure the quality based on how much those conditions are fulfilled. Below is a list of conditions that proved to be the most relevant within the scope of this study. The conditions are grouped into pre-review and review conditions.

A. Pre-Review Conditions

Table 1 includes the criteria that are included in the pre-review requirements.

TABLE I
PRE-REVIEW REQUIREMENTS

#	Requirement
1	Technical reviews are part of the project plan, with clearly defined roles and responsibilities (like moderator, reviewer, approver)
2	Mandatory and optional review participants are selected
3	There are at least two mandatory reviewers
4	(At least) the mandatory reviewers have committed resources for participation

The prerequisite for the pre-review conditions is that the technical writing team has a plan for all documents that need to be reviewed for a milestone. They need to provide the expected schedule and the proposed review participants (including the moderator and approver) based on the scope and topic of the documents. The latter is important in defining the responsibility of technical writers, who must share it with the reviewers in the development and product management teams. The quality of technical reviews does not only depend on the reviewers; reviewers can only commit to and participate in reviews if they receive a proper plan.

We defined at least two mandatory reviewers (in addition to the author) as a pre-review requirement. There are different views on how many reviewers are needed, some look at it from the expertise point of view, some from effort/cost point of view. In [10] several studies were included that had mentioned 3-5 reviewers as optimal, while they also cite an experiment that concluded that reducing the number of reviewers to two may significantly reduce effort without increasing review time or reducing effectiveness.

In addition to providing a plan for reviewers to estimate the needed effort for the review and to reserve the time to be able to participate in the reviews, there are best practices that can enhance the participation rate of reviewers. It was discussed in [15] how positive framing can increase the effectiveness of technical reviews. She explains that "The technical review process often suffers from a mismatch of priorities and expectations within a cross-functional team". This often results in reviewers not considering participating in technical reviews important enough. She argues that document authors "need to reframe the technical review and "sell" the frame to engage reviewers".

There were many studies cited in [10] that emphasized the need to "advertise" reviews and show that reviews work and that errors are inevitable, so it is completely fine to search for and find errors.

While this is an important social aspect to planning technical reviews, priority should be given to the proper planning and execution of reviews based on institutionalized processes and quality systems.

B. Review Conditions

Table II includes the review requirements.

TABLE II
REVIEW REQUIREMENTS

#	Requirement
1	At least all mandatory reviewers participate [a] in the review process
2	Comments and findings are provided to the author in the given timeframe
3	At least all mandatory reviewers provide comments or findings [b]
4	At least one of the findings is related to the technical content of the document
5	The approver approves or rejects the document review based on the participants' findings
6	The author provides the needed corrections based on the findings

[a] Participation means that the reviewer either approves the document as such or provide comments or findings, out of which at least one is related to the technical content of the document.

[b] Providing comments or findings include both the listing of findings and explicitly stating that the document is approved as such.

C. Comments, Findings and Review Decisions

In the proposed framework, a review is considered of good quality if all mandatory reviewers provide comments or findings, and the approver makes a documented decision (either approval or rejection) based on the participants' input. This means that in some cases reviewers can fulfil their roles as reviewers by simply stating that the document is good as such, and it is approved by the reviewer without any other comment. This is completely fine as it is not a "silent" approval (when the reviewer simply does not provide any comment but does not state his approval for the document as such). The practice of "silent" approvals is not preferred as it remains ambiguous and gives no added value to the review.

However, accepting the document as such might also have some hidden risks of relying on other reviewers' opinions or "votes".

In [16] research was described in which they analyzed software code reviews and how the decision "votes" for accepting or rejecting a software patch of the previous reviewers might influence the vote of a reviewer. He describes several other factors (like relationship between reviewers, status, seniority etc.), claiming that such review dynamics might cause more error-prone code: "we find that the proportion of reviewers who provided a vote consistent with prior reviewers is significantly associated with the defect-proneness of a patch" [16]. Although the study was conducted for code reviews, the general statements about why code reviews are important and how code reviews are currently carried out using collaborative tools that make all reviewers' comments and votes visible to all other reviewers are all valid for technical document reviews as well.

However, influence of reviewer interaction dynamics on decision-making is not within the scope of this paper, but I wanted to show what other perspectives could also be considered when creating a review quality measurement concept.

D. Type of Comments as an Indicator for Review Quality

This condition may be considered one of the more debated factors influencing review quality. A common assumption is that comments addressing technical content carry greater value than those focused on language-related issues. However, this perspective is not universally valid. Language-related feedback can be equally important, particularly when it affects clarity, usability, and the accurate interpretation of information. It is also important to recognize that the relative importance of comment types may vary depending on the nature and purpose of the documentation under review. For reference-type documents for which the template is very controlled or there is little running text or graphics, this condition works well. On the other hand, for operating manuals, product descriptions or other non-referential documents, this condition is not one to be used.

Language-related comments can significantly influence the overall quality of a review. If aspects such as language, style, or layout are overlooked—or if reviewers notice issues but fail to provide feedback to the author—the intended message of technically accurate content may still be misinterpreted or overlooked by the user.

A detailed taxonomy for document issues based on a wide empirical study was defined in [17]. They listed five main categories (Completeness, Up-to-dateness, Usability, Tool-related, and Readability) and several sub-categories for issues collected from comments from various sources (i.e., emails, issues and pull requests of open-source projects, and Stack Overflow threads). Although these sources might considerably differ from the technical documents the current paper describes a framework for, the taxonomy in [17] supports the relevance of linguistic considerations and language issues.

Among the categories defined in [17], Readability—which includes subtypes such as too abstract, too technical, too verbose/noisy, and simple typos—is particularly relevant to the current paper's focus on review quality from a linguistic perspective. While Readability was not identified as the most frequent issue by the original authors, it is emphasized here due to its central role in evaluating the clarity and accessibility of reviewed technical content.

There was no further analysis on the exact issue types but the finding that "issues related to lack of clarity represented more than half (55%) of these problems" highlights the importance of including linguistic considerations into technical reviews.

IV. LINGUISTIC CONSIDERATIONS DURING TECHNICAL REVIEWS

In this section I will describe why the language related reviewer perspective during reviews can help with identifying significant problems before certain technical documents are published for users. As I elaborated in the previous section, linguistic considerations do not have the same priority for different documents. Documents that provide step-by-step instructions for operating or troubleshooting technical equipment must include the review of the language, while for example, product descriptions or parameter lists might not need such reviewing aspect.

There might be several linguistic structures where wrong language use can cause misunderstanding for users. I selected three structures that I consider critical for understanding technical documentation.

1. Negative structures may be wrongly used and thus, it becomes confusing whether something is stated or negated
2. The usage of passive and active voice substantially impacts the understanding of the text's intention
3. Incorrect use of terminology might cause the misunderstanding of the technical content

The earlier cited work, [2], summarized the findings of the tragic Therac-25 case, where part of the root causes was poor documentation. The study did not go into the details of the documentation problems, and the original machine documentation is not available for researchers. The only root causes mentioned related to documentation were vague error messages, insufficient instructions for what to do in situations when an error occurs. Based on this, I can only assume that the listed language issues might have played a part in this specific case.

A. Problems with Negative Structures

Structures with "does not", "cannot", "will not", "is not", "was/were not" are the most unambiguous, unless the contracted forms ("doesn't", "can't", "won't", "isn't", "wasn't", "weren't") are used. Contracted forms can be easily misinterpreted by non-native speakers of English or users less familiar with the language. To the best of my knowledge, there are currently no academic studies that directly investigate the impact of contracted forms on comprehension in written technical texts. However, numerous professional style guides and technical writing resources discourage the use of contractions, particularly in contexts aimed at international or non-native audiences (for example, [18]). The guide emphasizes that technical texts should prioritize clarity and minimize potential misunderstanding for global readers. This caution is well-founded, as many contracted forms are ambiguous. For instance: *"I'd"* could mean "*I would"* or "*I had*", "*Who's*" could mean "*who is*", "*who has*", or even be misinterpreted as "*who was*" depending on context.

An additional problem might arise when contracted (or even uncontracted) forms are used together with a negative word in the same sentence. Or, vice versa, if there is no negative structure in the sentence, however, there is another expression that has a negative meaning.

The below examples are samples from a technical text corpus used by the Sketch Engine online tool [19]. It is important to note that the publicly available corpora that the tool uses are final versions of texts, meaning that (ideally) they already went through some kind of review. However, the corpus might also include technical text not from product documents but from online guidelines or instructions that might not go through any reviews. Therefore, they will only show the structures

described, but not necessarily ones that are unclear or confusing.

Table III shows the function of the highlighted items.

TABLE III
EXAMPLES FOR PROBLEMS WITH NEGATIVE STRUCTURES

Example sentence	Type of negation
*"But it is **not a rare** feature in desktop CPUs anymore."*	DIRECT NEGATION + INDIRECT NEGATOR
*"Installed **despite** Kasperky AV **not** liking the set up file."*	CONSESSION + DIRECT NEGATION
*"The unidirectional level converter is in there, **but** the pins do **not** match."*	CONTRAST + DIRECT NEGATION
*"For instance, sending to anything **not** in the mozillamessaging.com domain will **fail**."*	DIRECT NEGATION + IMPLIED NEGATION
*"Our tests in Chicago **didn't** show great performance from 'nationwide' 5G."*	CONTRACTED DIRECT NEGATION
*"Very important: **Don't ever** try to call any method before the call of InitializeComponent method."*	CONTRACTED DIRECT NEGATION + INTENSIFIER + NEGATION-SENSITIVE DETERMINER
*"This is frequently caused by the chroot directory **not** being a mountpoint when the chroot is entered."*	DIRECT NEGATION
*"Make sure you have 50% or more battery life otherwise the update **won't** download and install."*	DIRECT NEGATION

Examples were manually selected from a concordance using the English Web 2021 (enTenTen21) public corpus, with filtering for topic=Technology & IT, word/lemma="not" and sorting based on GDEX, in sentence format. GDEX provides Good Dictionary Examples, identifying identify sentences which are easy to understand and illustrative enough [19].

The above examples show that having more than one DIRECT NEGATOR in a sentence makes it more complex and more difficult to understand it. The sentences themselves are not incorrect but would not be recommended to be used in English for Specific Purposes, in this case, for technical texts.

B. Problems with Active/Passive Voice

The incorrect usage of active vs. passive voice in technical texts might create unclarity and might result in serious consequences in situations when, for example, the text provides instructions for carrying out a specific task. Improper use of passive voice in technical documentation can reduce clarity, especially when the actor performing an action is not specified. This issue is particularly problematic in requirements documents, where missing agents can lead to ambiguity in system behavior descriptions. The author of [20] highlights that passive voice often signals omitted information in use case scenarios, as it obscures who is responsible for performing actions. This becomes a serious problem in early project stages, where lacking detail can hinder requirements analysis. This work supports the view that clarifying agency through explicit language—including avoiding unnecessary passive voice—contributes to more complete and interpretable documentation.

Table IV shows examples with the verb "install", which is a frequent action in technical documents like installation or operating manuals.

TABLE IV
EXAMPLES FOR PROBLEMS WITH ACTIVE/PASSIVE VOICE

Example sentence	Issue
*PICSRules: Specifies an interchange format for filtering preferences, so that preferences can be easily **installed** or sent to search engines.*	Who is responsible for the installation? End user or software system or an administrator?
*So P3P could be **installed** on major Server implementations like Apache, Jigsaw, Netscape-Server or Internet Information Server from Microsoft.*	Who is responsible for the installation? Server administrator or developers or automated tools?
*Once the real font has been successfully downloaded and temporarily **installed**, it replaces the temporary font, hopefully without the need to reflow.*	Who is responsible for the installation? A user or an automated system or a specific application?
*Magnetic flow meters can be **installed** in horizontal or vertical piping so long as the pipe remains full at the point of measurement.*	The "actor" is not specified, however, if the focus is on the flow meter's installation capabilities rather than on the installer, this may suffice for high-level overviews or specifications.
*After the extension is **installed**, you will find a new section has been **added** to the permissions window.*	Who is responsible for the installation? The user or is the extension pre-installed by an administrator or automated process? The sentence includes 2 passive structures. The second one ("*a new section has been added*") is less problematic because the focus is on the result (the new section in the permissions window), and the actor who added it is irrelevant to the reader's understanding of the outcome.

C. Problems with Terminology

The whole idea of terminology is to make sure that there is only one specific term used for a concept. If wrong terms are used in technical documents or if the terms are ambiguous (for example, because the abbreviation is not resolved, and the document does not include a glossary or index), the intention of the product specification or operating instructions might undermine the specificity of the technical text.

A thoroughly studied major incident included a basic terminological error. While the 1999 NASA Mars Climate Orbiter case was more of a software issue related to unit conversion, it is closely tied to documentation and communication errors between teams. NASA's Mars Climate Orbiter was lost due to a mismatch between metric and imperial units in the software documentation. One team used the imperial system while another used the metric system, and this was not clearly communicated in the software documentation. This ambiguity in the language and instructions led to a navigational error, causing the spacecraft to enter Mars' atmosphere at the wrong altitude, leading to its destruction [21].

In Table V there are additional examples of hypothetical scenarios developed to illustrate common themes in wrong terminology usage. These are not direct citations but inspired by text I came across during my work.

TABLE V
EXAMPLES FOR PROBLEMS WITH TERMINOLOGY

Issue type	Example	Issue
Unclarity about whether the two terms are the same or not	*"Refer to the firmware update guide for instructions."* **vs**. *"Download the latest driver update to ensure compatibility."*	Is the "firmware update" the same as the "driver update," or are they separate processes?
	"The configuration file is located in the /config directory." **vs**. *"The settings file can be found in the /config folder.*	Are the "configuration file" and "settings file" the same, or are they different?
Inconsistent use of terms	*"Press the **power button** to turn the device on."* **vs.** *"Hold the **on/off switch** to start the device."*	Mixing "power button" and "on/off switch" for the same component may confuse users about whether they are referring to different physical controls.
	*"Upload the document to the **cloud storage** service."* **vs**. *"Save the file to the **online repository**."*	Using "cloud storage service" and "online repository" inconsistently without specifying if they are the same or different systems can create unnecessary complexity.

D. Language-Focused Review Comments and Their Automation Potential

Language issues—such as ambiguity, excessive passive voice, or inconsistent terminology—pose significant risks in international and regulated documentation. Rule-based tools, including classical grammar checkers and language profilers, are effective at identifying superficial issues like punctuation, agreement errors, and overly complex sentences. For example, a systematic review of automated grammar checking tools notes these systems can flag syntax errors, missing articles, and punctuation problems, but often struggle with deeper stylistic nuances or domain-specific inconsistencies [22]. ML-based approaches, on the other hand, have begun to demonstrate greater sophistication. For instance, tools like GrammarTagger use transformer-based models to profile grammatical features across languages, while recent work in "documentation smells" detection employs pretrained models like BERT to identify vague phrasing, structural omissions, and incoherent logic in API documentation [14]. These AI-enhanced methods offer promise in highlighting subtle quality issues that rule-based tools typically miss. Nevertheless, human review remains essential to interpret context, validate technical relevance, and resolve ambiguities that automation alone cannot adequately address.

It should be emphasized, however, that the present study does not analyze or implement these automated approaches. The language-related observations and categorizations discussed in this chapter are based on manual review practices documented within the organizational setting of the current research. While this provides a grounded understanding of real-world review behavior, the exploration of automated or AI-assisted solutions for language-related feedback remains outside the present scope. Future research will seek to incorporate and assess such technologies more systematically.

V. DEFINING A REVIEW QUALITY MEASUREMENT FRAMEWORK

A. Designing a Meaningful Metric

After defining the criteria for good quality technical reviews, a usable and meaningful metric can be designed so that the As-Is and To-Be situations are identified.

This paper describes an option where:

- Each document is evaluated as compliant or non-compliant with the criteria
- The milestone- or project-based document sets are evaluated based on the document-level compliance

Tables VI and VII include illustrative examples of how the metric calculation is done.

TABLE VI
DOCUMENT-LEVEL COMPLIANCE SCORING

Criterion	Doc1	Doc2	Doc3
Number of mandatory reviewers defined and committed	4	2	1
Number of mandatory reviewers providing valid comments	3	2	1
At least 2 mandatory reviewers defined and committed before the start of the review: YES/NO	YES	YES	NO
Percentage of mandatory reviewers who gave valid comments during the reviews or in the mail review periods: %	75%	100%	100%
Were comments late? YES/NO fill in only if column F is <100%	NO	NO	NO
Overall compliance YES/NO	**NO**	**YES**	**NO**

In Table VII, you can see additional calculations on top of the overall score. It shows that once you have the figures, you can yourself define which additional factor you would like to focus on or bring attention to the development teams or management. In this case we used the number of reviewers and the timeliness of comments as these two were the fundamental issues in our assumptions.

TABLE VII
DOCUMENT SET LEVEL SCORING

DOCUMENT SET percentage of documents meeting all criteria for good quality technical reviews	**33%**
percentage of documents with only 1 defined and committed reviewer	33%
percentage of deviations due to late comments	0%

The examples show a scenario where the document set has a Technical Review Quality of 33%. Once you have the overall score, it is up to your project, team or organization to set a target score and to define actions for what should happen when the score goes below the target or if there is a negative trend over several months.

B. Automation of Data Collection

Similarly to any kind of metric, in ideal case, all data about the review quality conditions is automatically collected and analyzed. For most of them it is possible and advisable to find the right toolkit that can handle all aspects of reviews: a database of review items with metadata, the draft documents themselves, the comments, findings and decisions.

The illustrative example in Tables VI and VII is entered and calculated simply in a spreadsheet, with manual data entries. This is the easiest way to collect data but in case of complex products with even hundreds of artefacts to review spreadsheets are not an option. As the number of reviewed items and reviewers increases, manual data collection becomes time-consuming and error-prone, raising concerns about scalability and reliability.

Moreover, reliance on manual data entry can introduce bias, both in what data is recorded and how it is interpreted. Reviewers may inadvertently omit information, apply inconsistent labeling, or interpret review outcomes differently depending on their experience or focus. These inconsistencies can distort the assessment of review quality and undermine efforts to compare performance across projects or teams.

To mitigate these issues, future implementations of the framework should explore semi-automated or fully automated solutions that can extract review data directly from collaborative platforms such as Git repositories, issue trackers, or review tools. For example, metadata about review duration, number of reviewers, and comment volumes can be parsed automatically from pull requests or change logs. Additionally, Natural Language Processing (NLP) techniques could be integrated to categorize review comments (e.g., distinguishing between content-related and language-related feedback) and to identify recurring issues such as ambiguity, terminology misuse, or incomplete instructions. Such methods would significantly enhance data completeness, reduce manual effort, and enable consistent and scalable quality measurement across documentation projects.

C. Addressing Quality Deviations

Measuring the quality of technical reviews is only the first step: gathering data as opposed to having a “feeling” about how technical reviews go. When you have facts in detail, you can easily pinpoint concrete problems towards management or the stakeholders.

In the following sections, I will present possible actions you can take to improve the quality of the reviews, and eventually, to reduce the errors found by the user.

Our approach was to first make the defined **quality conditions** into **criteria lists** or **checklists**.

Fig. 1. Quality conditions made into criteria lists

As we earlier defined the most crucial factors that impact the quality and efficiency of technical reviews, the next step is to make sure that these are checked early enough to avoid deviations afterwards by intervening as quickly as possible.

Checklists are a good means that technical writers can use before and after reviews. Of course, having a checklist does not necessarily mean any change unless there is a clear agreement and guidelines for its usage. To further strengthen the integration of linguistic feedback into structured review practices, selected categories from the documentation issue taxonomy proposed in [17] - such as Readability and Completeness - could be incorporated directly into the checklist and quality conditions used during reviews. While these categories were not part of the original implementation, they align well with the current framework’s emphasis on clarity, accuracy, and consistency.

In our case a drastic action was that technical reviews cannot even start before the criteria on the entry checklist are fulfilled. Not starting the reviews on time might result in delays with the approval of the documents and eventually, delays in the documentation delivery and ultimately, delays in software deliveries.

Similarly, reviews cannot be closed before the review exit criteria are fulfilled. Again, this might result in delays with the projects, causing huge problems for the product delivery.

An important change this brought was in the mindset. It used to be a general understanding that problems with the reviews should be owned and managed by technical writers or the technical reviewers, depending on which organization you work in. It was very rare when both teams understood that this is their common responsibility. Technical writers cannot claim that everything is on the shoulder of the reviewers, and they depend on them, while reviewers cannot claim either that the reviews are the sole responsibility of the technical writer team, and they will attend and contribute only if they have time.

With these changes they both understand that it is a joint effort that will bring success or problems to both.

Another improvement action that can be taken is to build the technical review quality into the project completion criteria. As with any other milestone criterion that needs to be passed in order to be able to decide about milestone approval, technical review quality can be a checkpoint, and it provides an opportunity for the Quality Manager and Program Manager not to approve the milestone due to problems with review quality. This includes any delays as in most cases when there is a problem with review quality, the result is some kind of delay, and this is eventually shown in the milestone approval.

As one of the entry criteria is that all the defined mandatory reviewers are committed to taking part in the reviews, there must be a proper plan based on the input from the document authors (i.e. Technical Writers). In addition to the planning of the right mandatory reviewers and the timeframe for the review, effort estimation must be done per reviewer to make sure they will really have time allocated to review the assigned documents. Without such a good plan it is not realistic to expect a good-quality review.

VI. CASE STUDY

A. Background

The measurement concept and improvement actions described earlier in this paper were implemented in the documentation department of a large software development company as part of a Lean Six Sigma project. This initiative focused on addressing the consequences of issues with technical document reviews, which included:

1) Customer-reported faults caused by inadequate reviews. These refer to documentation errors reported by customers that should have been detected during the internal review process. Examples include ambiguous instructions, incorrect parameter names, outdated references, or mismatches between the documented behavior and the actual product.

2) Program milestone rejections or concessions linked to review problems. In several cases, software program milestones were delayed or conditionally accepted because documentation did not meet the required quality standards. Issues were considered critical enough to block or delay software release approvals until documentation was revised.

3) Non-conformities to TL9000 requirements due to review-related issues. These non-conformities were identified during quality audits and related specifically to deficiencies in the review process, such as missing review plans, lack of documented reviewer roles, or absence of traceability between review findings and corrective actions.

This case study outlines the structured approach taken, the execution of improvement actions, and the tangible results achieved. While certain aspects such as reviewer engagement were still assessed subjectively due to the early phase of gamification features, the initiative aimed to increase reviewer participation - measured informally through the number of active reviewers and volume of review comments.

B. Approach and Execution

1) Collection of Customer Faults Data

Data on customer-reported faults related to variations in technical review quality was gathered systematically, providing a foundation for identifying improvement areas.

2) Design and Implementation of a New KPI

A novel metric, the Technical Review Quality KPI, was designed and introduced. This KPI, along with a detailed methodology, enabled the department to measure and monitor review quality objectively.

3) Root Cause Analysis

A thorough Root Cause Analysis (RCA) was conducted to pinpoint recurring problems in the review process. This analysis identified gaps in reviewer engagement, entry and exit criteria, and adherence to best practices.

4) Education and Training

Dedicated training sessions were conducted for technical writers and expert review teams. These sessions focused on enhancing their understanding of the review process, setting clear expectations, and emphasizing their roles in achieving high-quality reviews.

5) Quality Checkpoints Implementation

Several review-related quality checkpoints were introduced, including:

- Entry and Exit Criteria Checklist: Ensuring that reviews met predefined standards at both the start and end of the process.
- Reviewer Commitment: Establishing accountability for reviewers in the process.
- Integration into R&D Definition of Done Criteria: Embedding review quality into the organization's standard development lifecycle.

6) Mindset and Behavior Change through Gamification

Recognizing the critical role of motivation in achieving sustainable improvement, a gamification project was introduced. Drawing inspiration from [10], a potential game was designed to reward participation and contributions to the success of technical reviews. This initiative aimed to shift the mindset of reviewers, fostering a sense of ownership and enthusiasm for the review process. The game has not yet been widely taken into use yet but in the first small-scale trial reviewers earned points for timely participation, adherence to review standards, and quality feedback, which could be exchanged for recognition within the organization. The gamification approach created a competitive yet collaborative environment, further enhancing the team's commitment to continuous improvement.

C. Novelty of the Approach

This project marked a significant departure from traditional practices. Previously, technical review quality had never been measured, analyzed, or systematically improved. The development of a quantifiable KPI and the comprehensive methodology provided a reusable framework for enhancing review quality. Additionally, the integration of gamification to drive mindset change represented an innovative way to ensure that improvements were not only procedural but also cultural. While the focus of this initiative was on technical documentation, the concept can be easily adapted to other review-intensive areas, such as software code reviews or product design assessments.

D. Results

The implementation of this framework yielded remarkable improvements in the documentation department's review process:

- 60% Improvement in Technical Review Quality: The KPI demonstrated a significant enhancement in the effectiveness of reviews.
- 29% Reduction in Customer Faults: The number of customer-reported faults attributed to issues with technical reviews dropped substantially.
- Zero TL9000 Non-Conformities: All review-related non-conformities to TL9000 requirements were eliminated.

• Improved Reviewer Engagement: The gamification initiative led to increased participation and a noticeable shift in reviewer behavior, with teams actively striving for better outcomes.

Due to confidentiality agreements within the organization where the study was conducted, exact quantitative data - including baseline values, detailed metric calculations, and project-specific documentation artifacts - could not be disclosed. The review processes evaluated were part of internal quality improvement initiatives linked to ongoing product development efforts, and the associated metrics were derived from proprietary review records and audit outcomes. As a result, while the case study outlines the structure and application of the measurement framework, the focus remains on illustrating the methodology and its practical relevance, rather than presenting fully open, reproducible datasets.

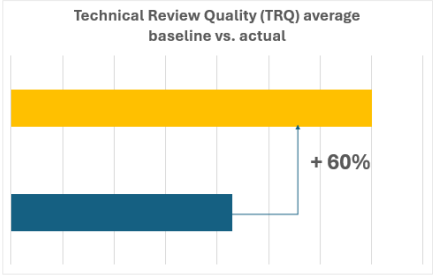

Fig. 2. Result: improvement in Technical Review Quality

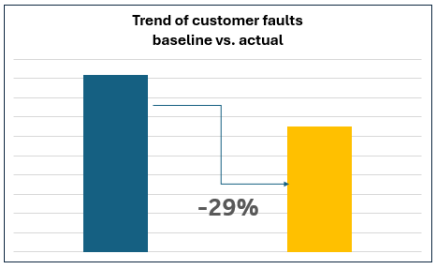

Fig. 3. Result: customer fault reduction

VII. CONCLUSION

The systematic measurement, analysis, and improvement of technical review quality in this project demonstrate the potential for significant impact on documentation processes. By addressing root causes, educating teams, embedding quality checkpoints into workflows, and fostering a mindset of continuous improvement through gamification, the initiative achieved sustainable and cultural changes. The methodology and outcomes serve as a model for other organizations seeking to enhance review processes in documentation or other domains.

To conclude, this study provides the following practical recommendations for professionals seeking to improve the quality of their review processes:

1. Measure review quality: Implement a systematic approach to measuring technical review quality, focusing on pre-review and review conditions. Establish clear metrics, such as a Technical Review Quality KPI, to track and monitor improvements effectively.

2. Prioritize linguistic factors: Pay attention to linguistic considerations during reviews, ensuring clarity, accuracy, and comprehensibility in technical documentation. Training and resources should emphasize the importance of precise language use to minimize errors.

3. Use checklists and automation: Employ checklists to standardize the review process, ensuring consistency and thoroughness. Consider automating data collection and analysis to streamline the identification of problem areas and track progress over time.

4. Foster collaboration: Promote a collaborative mindset between technical writers and reviewers by creating a culture of shared responsibility for review quality. Encourage open communication and teamwork, making the review process a constructive, cooperative effort.

5. Encourage and incentivize mindset changes: Leverage gamification or similar initiatives to boost motivation and engagement among reviewers. Reward participation, adherence to review standards, and quality contributions to foster a sense of ownership and enthusiasm for achieving high review standards.

Limitations and Future Research

The methodology and the implementation focus on technical reviews for documents created for users and therefore, it might not work the same way for other type of documents or technical documents in other industries or areas.

Potential future research will widen the scope of the methodology to other type of artefacts (software code, multimedia elements, internal specifications, etc.), technical documents in other STEM (Science, Technology, Engineering, Mathematics) fields. In addition, the impact of distinct groups of authors (software developers, engineering students), different review methodologies and the role of automation in review quality measurement will be studied.

While the proposed framework was developed with a primary focus on documentation authored by professional technical writers, many contemporary documentation practices - particularly in agile and DevOps environments - involve collaborative authoring with developers, engineers, or other subject matter experts. In such contexts, the nature and consistency of content, review expectations, and language quality may vary more widely. Future research should explore how the defined quality criteria perform across these more heterogeneous authoring scenarios.

This study did not extensively explore the use of automation or AI-assisted tools in documentation review, as its primary focus was on evaluating and improving existing manual review practices within a specific organizational context. While recent developments in natural language processing and review automation were acknowledged in earlier sections, their implementation was beyond the scope of the current work. Future research should investigate how such technologies, particularly AI-based comment analysis and automated quality checks, could be integrated into the proposed.

Another promising avenue for future research is the application of textual analysis methods to reviewer comments in technical documentation, as demonstrated in studies analyzing game reviews (e.g., [23]). The intent here is not to equate the subject matter of game reviews with that of technical documentation, but to illustrate how similar analytical methods can be adapted across domains. Techniques such as word frequency analysis, word associations, and sentiment analysis could provide deeper insights into the focus areas, concerns, and linguistic patterns in technical review feedback. By adapting these methods, it would be possible to identify recurring themes, better understand reviewer priorities, and even uncover implicit biases in the review process. Such an approach could inspire new strategies to further refine technical reviews and enhance their effectiveness.

Ultimately, this paper represents a significant step forward in closing the gap in understanding the effectiveness of technical reviews and establishing a systematic, structured methodology for measuring their quality. By integrating practical solutions, such as gamification to drive mindset change and actionable recommendations for practitioners, this study demonstrates the real-world impact of improving review processes. These principles not only deliver measurable benefits but also provide a foundation for organizations to extend this approach to other areas of quality management. Looking ahead, the insights and methodologies presented here offer a pathway for future research and innovation, helping industries align with evolving demands and set new standards for excellence in technical review quality.

REFERENCES

[1] IEEE Standard for Software Reviews. IEEE Std 1028-1997, pp.1–48, 1998. DOI: 10.1109/IEEESTD.1998.237324

[2] N. G. Leveson, C. S. Turner, “An investigation of the Therac-25 accidents,” *Computer*, vol. 26, no. 7, pp. 18–41, 1993. DOI: 10.1109/MC.1993.274940

[3] H. J. Harrington, “Measure and Control” in *Business Process Improvement: The Breakthrough Strategy for Total Quality, Productivity, and Competitiveness*, McGraw-Hill, 1991. ISBN: 978-0-07-026768-2

[4] B. Freimut, “A Measurement Framework for Software Inspections in the Quasar Context”, *IESE Report* No. 118.03/E, a publication by Fraunhofer IESE, 2003.

[5] JA. T. Hackos, "Conducting technical reviews" in *Management of Documentation Projects*, In Wiley Encyclopedia of Electrical and Electronics Engineering, J.G. Webster (Ed.), 1999.

[6] J. Zuchero, “Using inspections to improve the quality of product documentation and code”, *Technical Communication: Journal of the Society for Technical Communication*, 42(3), 426–435., 1995.

[7] M. E. Raven, “What kind of quality are we ensuring with document draft reviews? *Technical Communication*, 42(3), 399–408. 1995.

[8] M. Fagan, “Design and code inspections to reduce errors in program development”, *IBM Systems Journal*, vol. 15, no. 4, pp. 182–211., 1976.

[9] M. V. Mäntylä, C. Lassenius, “What types of defects are really discovered in code reviews?”, *IEEE Transactions on Software Engineering*, vol. 35, no. 3, pp. 430–448, May-June 2009. DOI: 10.1109/TSE.2008.71

[10] O. Laitenberger, J. M. DeBaud, “An encompassing life cycle centric survey of software inspection”, *Journal of Systems and Software*, 50(1), 5–31., 2000. DOI: 10.1016/S0164-1212(99)00073-4

[11] K. E.Wiegers, “Metrics” in *PeerReviews in Software: A Practical Guide*, Addison-Wesley, 2002.

[12] F. Shull, *et al*., “What we have learned about fighting defects” in *Proceedings of the 27th International Conference on Software Engineering (ICSE '02)*, 249–258., 2002. DOI: 10.1109/METRIC.2002.1011343

[13] H. Cadavid, *et al*., "Documentation-as-code for interface control document management in systems of systems: A technical action research study." in *European Conference on Software Architecture* (pp. 19–37). Cham: Springer International Publishing. 2022. DOI: 10.1007/978-3-031-16697-6_2

J. Y . Khan, *et al*., “Automatic detection of five api documentation smells: Practitioners’ perspectives” in *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 318–329, IEEE, 2021. DOI: 10.1109/SANER50967.2021.00037

J. Holdaway, “Technical reviews: framing the best of the best practices”, *IEEE International Professional Communication Conference*, Waikiki, HI, USA, pp. 1–13, 2009. DOI: 10.1109/IPCC.2009.5208713

[14] P. Thongtanunam, “Review dynamics and their impact on software quality”, *IEEE Transactions on Software Engineering*, Vol. 47, No. 12., 2021. DOI: 10.1109/TSE.2020.2964660

[15] E. Aghajani *et al*., “Software documentation issues unveiled”, *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, Montreal, QC, Canada, 2019, pp. 1199–1210, 2019. DOI: 10.1109/ICSE.2019.00122

[16] Progress Software. (n.d.). DevTools style guide. Retrieved June 10, 2025, from https://docs.telerik.com/style-guide/

[17] Sketch Engine [Online tool, academic use]. https://www.sketchengine.eu/

[18] L. Kof, “Treatment of passive voice and conjunctions in use case documents”, in *International Conference on Application of Natural Language to Information Systems*, pp. 181–192. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. DOI: 10.1007/978-3-540-73351-5_16

“Mars Climate Orbiter mishap investigation board phase report, 1999, https://llis.nasa.gov/llis_lib/pdf/1009464main1_0641-mr.pdf

[19] B. Raad, *et al*., “Exploring the Profound Impact of Artificial Intelligence Applications (Quillbot, Grammarly and ChatGPT) on English Academic Writing: A Systematic Review”, *International Journal of Integrative Research (IJIR)* 1.10 599-622., 2023.

[20] T. Guzsvinecz, and J. Szűcs, "Textual Analysis of Virtual Reality Game Reviews", *Infocommunications Journal, Joint Special Issue on Cognitive Infocommunications and Cognitive Aspects of Virtual Reality*, 2024, pp. 84–91, DOI: 10.36244/ICJ.2024.5.10

Gabriella Tóth is pursuing a PhD degree at the University of Debrecen, Institute of English and American Studies, Department of English Linguistics, Hungary. Her doctoral research is carried out under the Doctoral School of Linguistics at the same university. She holds an MSc in Information Science and an MA in English Linguistics from the University of Debrecen. During her doctoral studies, she conducted research in computational linguistics, corpus linguistics, and information retrieval at the Lister Hill National Center for Biomedical Communications, U.S. National Library of Medicine, Bethesda, MD, USA. She also has professional experience at Nokia Solutions and Networks in Hungary, where she worked as a technical writer and information designer, with a particular interest in technical reviews and their linguistic aspects. Her current research focuses on the linguistic analysis of technical texts and the pragmatics of technical communication, with a special emphasis on the formulation and interpretation of error messages.