Freie Universität Berlin

Bachelorarbeit am Institut für Informatik der Freien Universität Berlin

Arbeitsgruppe Software Engineering

Content Analysis of Conclusions of Empirical Studies in Software Engineering

Lars Simon Spitzlay
Matrikelnummer: 4882344
lspitzlay@zedat.fu-berlin.de

Betreuer/in: Linus Ververs
Eingereicht bei: Prof. Dr. Lutz Prechelt

Berlin, January 3, 2023

Abstract

Software engineering is still relatively new to empirical research. As such, it is reasonable to assume that its quality is still somewhat lackluster. This thesis aims to evaluate the quality of recent empirical studies in software engineering by performing a content analysis on 58 papers taken from the ICSE 2021, focusing on their conclusions. Due to a lack of quality of the data gathered, the goals of this thesis were ultimately changed to analyzing the shortcomings of the project as well as providing hints to improve future similar research. This included an analysis of the data gathered during the content analysis to investigate what makes some sentences harder to code than others. Results show that the students that performed the coding part of the content analysis were often not abiding by the rules given and had a problem drawing a distinction between some specific codes. Reaching a sufficiently high inter-reviewer agreement to make based claims and statements about the data is not an easy task, especially when working with students as compared to a group of researchers who may already have some experience working together. Still, to maximize the potential, this thesis provides the following suggestions. Try to convey the rules of the coding part to the reviewers better by stating and perhaps restating them more clearly and discussing them. Give reviewers enough time to do their work, by either reducing the workload, giving more time, or focusing on certain sections from the beginning. Build the code system as concise and clear as possible and try to add a useful memo to each code to reduce ambiguity.

Empirische Forschung ist noch relativ neu in der Informatik, daher ist die Annahme, dass dessen Qualität noch etwas zu wünschen übriglässt, durchaus angemessen. Diese Arbeit versucht die Qualität kürzlich erschienener empirischer Studien in der Informatik mittels einer Inhaltsanalyse von 58 Publikationen, entnommen aus ICSE 2021, zu erörtern. Dabei soll das Hauptaugenmerk auf dem Fazit der Publikationen liegen. Durch mangelnde Datenqualität jedoch, wurden die Ziele dieser Arbeit letztendlich abgeändert. Die neuen Ziele beinhalten das Analysieren der Probleme und Fehler des Projektes, sowie das Bereitstellen von nützlichen Hinweisen für künftige ähnliche Arbeiten. Dafür wurden die bereits gesammelten Daten analysiert um zu verstehen was einige Sätze schwerer zu codieren macht als andere. Die Ergebnisse zeigen, dass die im Projekt involvierten Studenten oft nicht die gegebenen Regeln richtig befolgt haben und einige bestimmte Codes nicht gut auseinanderhalten konnten. Eine hohe Zustimmung über die Codes zwischen den Studenten zu erreichen ist nicht leicht, besonders da diese sich sicherlich noch stark von einer eingearbeiteten Gruppe von Wissenschaftlern unterscheiden. Dennoch präsentiert diese Arbeit folgende Hinweise, die dabei helfen können, die Datenqualität zu maximieren. Versuchen Sie die Regeln des Codier-Teils den Rezensenten näher zu bringen, indem diese klar formuliert und eventuell regelmäßig wiederholt und diskutiert werden. Zudem sollte man den Rezensenten genügend Zeit für ihre Arbeit geben, indem man die Arbeitsmenge reduziert, mehr Zeit bereitstellt, oder von Beginn sich nur auf bestimmte Kapitel der Publikationen beschränkt. Das Codesystem sollte außerdem so genau und klar wie möglich konzipiert und definiert sein, inklusive Code Memos, die die Codes besser voneinander abgrenzen.

Contents

1 Introduction 7

2 Related Work 8

3 Methods 10

4 Results 13

5 Discussion 17

6 Threats to Validity 19

7 Conclusions 20

A Appendix 23

1 Introduction

In comparison to other sciences, software engineering is relatively new to empirical studies since software engineering itself is one of the most recent sciences to exist, getting its name in 1968 [16]. The exact beginning of empirical research in software engineering is hard to pinpoint, but it is reasonable to assume that some early studies conducted in software engineering at least had some empirical aspects. The *IEEE Transactions on Software Engineering* and the *International Conference on Software Engineering*, both of which having started in 1975 and being two of the biggest publishers of software engineering papers, have since contained many empirical studies. Perhaps one of the biggest milestones of empirical software engineering though was achieved when the first volume of the international journal *Empirical Software Engineering* was released in 1996. This journal focuses heavily on empirical research in software engineering, often consisting solely of empirical studies and is still growing in the number of articles to this day.

Due to being relatively new, it may be a fair assumption to make that empirical studies in software engineering might still be lacking in quality compared to other sciences. A fair amount of meta-research has already been done about empirical studies in software engineering. This research most often focuses on the methods used, the effectiveness of the studies or on improvements to the approach. This thesis on the other hand aims to focus on an even higher level of abstraction, analyzing the conclusions of empirical studies in software engineering to determine their quality and relevance. An understanding of the shortcomings of empirical studies in software engineering can be beneficial to the community, as it may help future research avoid some common mistakes. This thesis also aims to give advice to future studies using open coding or code systems in general, providing some useful hints that have surfaced during the project.

The main data analysis methods used in this thesis are open coding and qualitative as well as quantitative content analysis. Open coding describes the method of generating and applying codes to parts of a text while reading it. These codes may consist of single words or small groups of words. The length of a part of text is variable as well, ranging from single words to sentences or even paragraphs. The content of the codes can vary based on the goal of the analysis, including for example concepts, keywords, or topics. During open coding the resulting code system can be constantly modified based on the findings of the author. Codes may be merged, expanded, or otherwise altered to better fit the data or goals. Once done, open coding should yield a code system that can be used for further analysis.
Content analysis is a qualitative-quantitative method used in empirical research. It aims to break down the input data into smaller pieces in order to either quantitatively or qualitatively analyze them. While the input data, mainly some sort of text, is always qualitative, the output data can be quantitative as well. The precondition for doing content analysis is to have a code system, which can either be taken from literature or previous studies or created, by using open coding for example. Once the code system is established, one searches for all occurrences of each code. Just as

in open coding, these marked passages can vary in length (for example words, sentences, paragraphs). When performing quantitative content analysis, one now counts the occurrences of each code. This whole process should be performed by at least two independent reviewers, in order to reduce the weight of biases and errors. Afterwards, the inter-reviewer agreement should be calculated and only coded passages with a high agreement score should be further considered, depending on the number of reviewers. This inter-reviewer agreement can be calculated in different ways. One of the easier ways, and the one i chose here, is to sum up the amount of codes per student that at least one other student agreed upon and divide it by the total number of codes used. For quantitative content analysis, one can now make statements based on the relative or absolute occurrences of certain codes, while for qualitative content analysis, one needs to further investigate the coded passages to reach a conclusion.

Unfortunately, I did not manage to provide any useful insights into the quality of empirical studies in software engineering due to a lack of data quality. Alas, I focused more on analyzing the shortcomings of our project as well as providing useful information for researchers or tutors choosing similar approaches and methods.

The rest of this thesis is structured as follows. Firstly, I will give an insight into related work. The following sections will be vaguely divided into two parts: the project this thesis is based on, and the work done after said project. The first one of these sections consist of a description of the methods used and the research process, including the main difficulties faced. Afterwards, the results are presented and discussed. Then, the threats to validity are discussed and finally the conclusions drawn are presented.

2 Related Work

A fair amount of studies have been conducted analyzing the quality of empirical studies in software engineering, pointing out strengths and weaknesses of work done thus far, as well as providing hints and roadmaps on how to properly conduct empirical research and write papers. The following works mainly relate to my original goal of analyzing conclusions to provide an insight into the quality of recent empirical studies in software engineering.

Perry *et al.* [12] for example, argue that "[t]oo many empirical studies study the obvious" [12, p. 5] and that there "are too many papers whose only selling point is that they have lots of data" [12, p. 5]. They also point out that "[s]ince many researchers are reluctant to draw conclusions from their data, it's easy to imagine that they aren't too happy to generalize them either" [12, p. 5] and follow up with giving some advice on how to create and conduct better empirical studies.

Zannier *et al.* [18] report, that „the quantity of empirical evaluations performed has increased over 29 years of ICSE proceedings" [18, p. 10]. But they conclude "that the soundness of empirical evaluations has not improved over 29 years of ICSE proceedings" [18, p. 10]. They argue that there is a "lack of improvement in self-confirmatory

studies" [18, p. 10] and "extremely little hypothesis specification" [18, p. 10].

Other literature review papers concentrate their research on specific parts of software engineering. Tonella *et al.* [14] for example focus on the state of empirical research in reverse engineering and provide "a roadmap for the future research in the field" [14, p. 1]. They "propose a tentative set of taxonomic criteria to support the positioning of each study in the wider context, and [...] describe a tentative framework, borrowed from the broader area of software engineering, which should guide researchers conducting new studies" and "try to sketch the future trends in reverse engineering by providing a research agenda, which [they] consider an essential step to move forward" [14, p. 17].

Bosu and MacDonell [1] on the other hand focus more on the quality of the data of empirical software engineering by "perform[ing] a targeted literature review of empirical software engineering studies" [1, p. 1]. They conclude that "data collection procedures and the identification of data quality issues [...] need to be given greater attention" [1, p. 6]. They also criticize that "a third of the studies reviewed employed private data sets" [1, p. 6] and end with saying that "[i]ncreasing the public availability of data sets that have also been enhanced through improved collection, cleaning and transformation procedures will lead to more reliable predictive models and thus improve software engineering practice" [1, p. 6].

Feldt and Magazinius [3] "perform a review of 43 papers published in the ESEM conference in 2009 and analyse the validity analysis they include" [3, p. 1] in order to "discuss what is working well and less well in validity analysis of empirical software engineering research and present recommendations on how to better support validity analysis in the future" [3, p. 1]. They reveal that "[m]ore than 20% of these papers contains no discussion of validity threats and the ones that do discuss on average only 5.44 threats" [3, p. 6] and further criticize that "[f]or only half of the discussed threats are any strategy to overcome or mitigate the threat discussed [...] [a]nd more than 25% of these mitigation strategies mentioned have not been used in the studies but are just discussed as future work" [3, p. 6]. Thus, they "propose that a common model for the process of conducting empirical research in software engineering is created and that a simpler terminology for validity threats and analysis is adapted to this model" [3, p. 6].

On a different note, Carver *et al.* [2] "provide some advice on how to carry out empirical studies with students" [2, p. 1], pointing out some aspects that, in my opinion, have already been done well in the course this thesis is based on. The main point that may have been done better in the course is "[g]ive realistic time estimates." [2, p. 10], as certain parts of the project were more time-consuming than anticipated.

3 Methods

During the summer semester of 2022, 10 students in the Freie Universitaet Berlin masters course Empirical Methods in Software Engineering were part of a project that created the input data used in this thesis. I was a part of this group of students as well.

The tutorials of said course consisted of several steps to investigate the original research topic of this thesis, trying to evaluate the quality of recent empirircal research in software engineering, using quantitative content analysis. Firstly, using open coding on a small subset of empirical studies taken from the *International Conference on Software Engineering* (ICSE) 2021, a code system was created by each student independently. This, as well as some of the next steps were done using MAXQDA 2022, a software used for data analysis. Afterwards, a unified code system was agreed upon, which was to be used on the rest of the empirical papers published in the same conference. This code system was built hierarchical, consisting of seven subtrees:

- **Further work –** consists of codes to be assigned to sentences where the authors talk about further work they propose or advice. Includes codes about the intended authors of the further work, as well as its importance and kind, relative to the paper at hand. Was mainly intended as an indicator for the amount of further work the authors proposed and whether they highlighted its importance, or if they were intending to do it themselves.

- **Generalization –** consists of codes that reflect a sentence where the authors generalize their results. Includes codes about the degree to which the authors generalize, its kind, the audience the results are relevant to as indicated by the authors, and an evaluation of the adequacy of the generalization, as judged by the reviewers. This subtree was especially important, since generalization should be an important part of the conclusion of a paper. The purpose of these codes was to evaluate the amount of generalization used in the conclusion, as well as how adequately it was used.

- **Meta -** consists of a code used to mark sentences that could not be assigned any other code, which was only relevant for the conclusion, since every sentences was to be assigned a code there. Also includes section titles used to mark whole sections in order to make further processing of the data easier, like identifying the sentences in the conclusion section.

- **Research motivation -** consists of codes about the motivation provided by the authors for the work they did. Due to the focus on conclusions, this subtree ended up having little relevance.

- **Research process -** consists of two codes, indicating sentences where the authors claim to be the first to do something or sentences where the authors simply repeat their research process in the conclusion. Mainly intended to identify the amount of sentences where the authors repeat something from previous sections that does not belong into the conclusion.

- **Risk -** consists of codes to be applied to sentences that state risks of the paper. Includes codes that identify a stated risk as addressed or untreated, the risks' magnitude as described by the authors and whether the sentence simply states a risk or talks about its impact. The relevance of this subtree is questionable due to the focus on conclusions and the fact that one would usually not talk about risks in the conclusion.

- **Validity -** consists of codes about the threats to validity mentioned by the authors. Includes codes that indicate the kind of validity the authors talk about (internal, external, conclusion or construct validity), whether the authors merely state threats, talk about an improvement to one, or claim one to have no effect on the validity of their work. Also includes codes that reflect the evaluation of the threats impact, as seen by the reviewer. Intended as a measure of the reflection of the authors as to the validity of their paper and the actions taken to improve the validity. Usually handled in a separate section of the paper, thus seldom included in the conclusion and ultimately of little use when focusing solely on the conclusion.

Note: The full code system can be found in table IV in the appendix section.

Once the code system was established, students were tasked with identifying empirical papers of the ICSE 2021, separating them from others that were, for example, proposing new software or methods. This produced a list of 58 papers, which was then divided between all students, so that each paper would be reviewed by at least three students, resulting in each student reviewing 18 papers. The students were instructed to only code full sentences and to focus their attention on the conclusion, while also identifying relevant passages in other sections of the papers, especially the abstract, introduction, results, and discussion sections. For the conclusion, it was mandatory to assign each sentence with at least one code. Also, due to the hierarchical structure of the code system, the students were instructed to always use a code of each subtree of a code cluster. So, if for example one would identify a sentence as talking about further work, one needed to either use the code "advice further work" or "propose further work", and additionally one of the codes of the subtrees "further work assignment", "further work importance" and "further work kind" respectively, resulting in at least four codes for that particular sentence.

The students were given a total of three weeks to review the 18 papers they were assigned. Afterwards, they were divided into three groups, each focusing on a part of the quantitative content analysis we aimed to perform. The first group was to work on data validation and cleansing, which consisted of extracting the data from MAXQDA 2022 and improving its quality through different means, like correcting some smaller mistakes made by students. This group also looked at the inter-reviewer agreement, trying to identify potential outliers in the students and codes and reporting on the data. The second group was tasked with document and conclusion clustering, trying to find clusters of papers or conclusions sharing some common trait or structure based on the codes used. Lastly, the third group was to look at code correlation, identifying codes that often appear together or codes that almost only appear in certain sections,

etc. In the very end, students were also tasked with writing a final report, focusing on the data analysis as well as the project as a whole, possibly pointing out flaws in the approach, as well as lessons learned etc.

After the end of the semester, I continued to work on the quantitative content analysis. This involved first and foremost improving the data quality. I tried several different approaches to improve data quality:

- Reviewing and improving the script provided by the data validation group to match the passages coded by different reviewers more accurately. This involved using a sequence matcher, a tool that measures the likeness of, for example, two sentences, to better correct or accept some common mistakes made by students, like coding a few characters too little or too much.

- Only considering the conclusion since we had stricter rules while coding conclusions. In other sections, some reviewers might simply have not coded anything, which would decrease the inter-reviewer agreement. Also, the conclusions were our focus in the first place.

- Merging some codes together into more general and less ambiguous or complex codes. This involved, for example, merging "for all generalization" and "limited generalization" into "mention generalization". A lot of coders seemed to disagree on some of these codes and used different ones on the same passage.

After exhausting all options to improve data quality, I realized that we would not be able to accomplish an inter-reviewer agreement high enough to make based claims about the quality of the studies investigated or any other topic we originally wanted to report on. Thus, my work after this realization focused more on secondary goals:

- Drawing lessons from analyzing the data and providing suggestions to future studies on similar topics. My intense work with the data, as well as the work I put in during the semester already gave me a good understanding of some problematic topics and approaches that could have been done better to ensure an improved quality of the data. Further investigation proved valuable as well.

- Get an understanding of what makes some sentences harder to code than others. This was done by first importing all sentences with zero and 100 percent inter-reviewer agreement respectively back into MAXQDA 2022.

After importing the sentences, I started to try and find any patterns within the sentences to explain why they were apparently easier or less ambiguously to code. Looking at codes with an inter-reviewer agreement of 100 percent, I noticed a few patterns emerge in some specific codes, so I started to build up a new, small code system, using a sort of open coding, to try and pinpoint these patterns. Afterwards, I did the same with the sentences with no agreement, also trying to find the lack of patterns I identified before, in sentences with the same code but an inter-reviewer agreement of 100 percent.

4 Results

Due to a lack in quality of the data obtained from the students during the semester, the results of said data is mostly useless. Alas, I can only report on results of drawn about the data quality itself, focusing on the inter-reviewer agreement. The inter-reviewer agreement started at about 38 percent. My initial improvements to the script provided by the data validation group only improved the agreement by about three percent, though further improvements to the script were made later as well. Focusing on the conclusion and discarding all other code improved the agreement by some more percent points, up to about 46 percent. Several more improvements to the script slightly increased the agreement again, to 48 percent and merging the codes, along with some final improvements brought the score up to its final value of just over 51 percent.
All further statements here refer to the data after all data quality improvement steps mentioned in the execution section.

TABLE I. Inter-reviewer agreement by codes - Top 10

Code	**Total times used**	**Agreement in %**
state first of a kind	49	86
own further work	94	83
anyone further work	104	69
subsequent further work	112	64
repeat research process	365	61
no classification of importance further work	90	61
mention generalization	296	54
promising further work	74	53
universal statement generalization	200	52
unclassified risk	9	44

While some codes achieved a rather high inter-reviewer agreement, others could not be agreed upon at all. As can be seen in Table I, the code "state first of a kind" reached the highest inter-reviewer agreement score of 86 percent, followed by three codes concerning further work, "own further work", "anyone further work" and "subsequent further work". On the other hand, of those codes used at least once in the conclusions, there were ten codes that the students did not agree on a single time, although most of them were rarely used anyway.

As can be seen in Fig. 1, the inter-reviewer agreement varied across papers, although exactly half showed an agreement of 25 percent to 40 percent with one paper having an inter-reviewer agreement of zero and one topping off at 72.7 percent.
Fig. 2 shows that the total number of codes used varied between the students, as well as the number of agreements. The relative agreement scores per reviewer however, measured as number of agreements divided by total number of codes used, vary far less. The top coder achieved a score of 63.5 percent and all others ranged from 46.3 to 53.5 percent, averaging at about 51.1 percent.

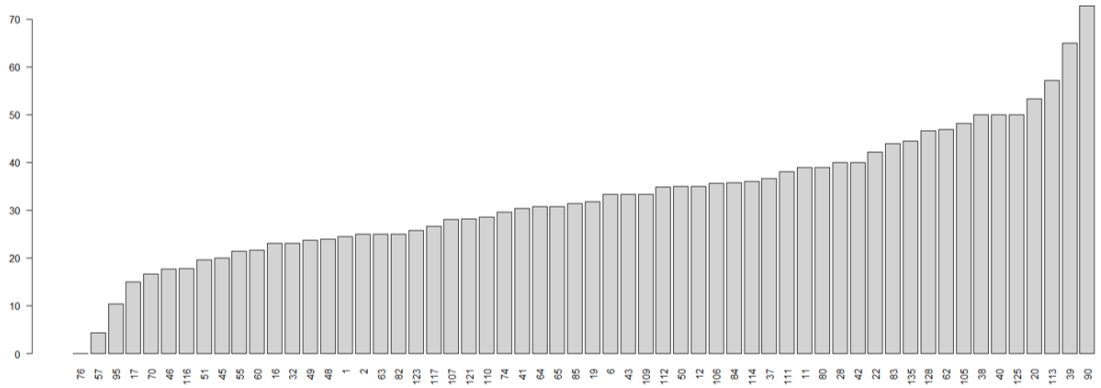

Figure 1. Inter-reviewer agreement in percent per paper. Papers are identified by IDs on the x-axis while the y-axis represents the inter-reviewer agreement in percent.

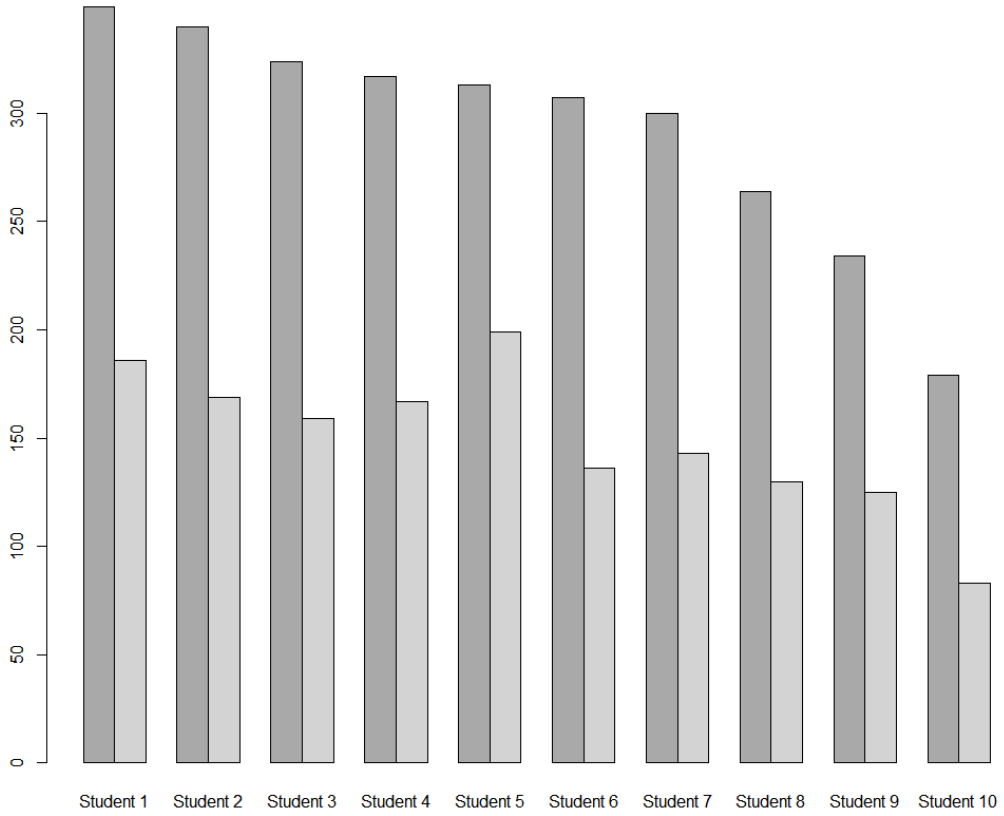

Figure 2. Total number of codes and agreed-upon codes per reviewer. The x-axis represents the different students (anonymized) and the y-axis the number of codes. The dark grey column represents the total number of codes given by a student and the light grey column the ones that at least one other student agreed upon.

Table II shows that the codes most used were “no code applicable”, “repeat research process” and “mention generalization”, having been used a total of 373, 365 and 296 times respectively. Most of the codes talking about further work and risks were not used as often however with some not having been used at all.
Table III shows that the code “mention generalization” was used across the most papers by students at 94.92 percent, meaning only 3 papers had no reviewer identifying any sentence as talking about generalization. The code “repeat research process“ was used in 53 out of 58 papers as well. Meanwhile, the codes “major risk” and “unimportant further work” were only used in 1.69 percent of papers, which translates to exactly one paper. Also, all codes regarding or containing the word “risk” have been used at most in 6 papers, with “minor risk” not having been used at all.
The average amount of words and characters used in sentences with codes the students could not agree on did not differ significantly from those with a 100 percent inter-reviewer agreement.

TABLE II. Total number of occurences per code - Top 10

Code	**Total number of occurences**
no code applicable	373
repeat research process	365
mention generalization	296
universal statement generalization	200
adequate generalization	167
relevant result without specification	134
no generalization	132
mention further work	127
relevant result for practitioner	117
subsequent further work	112

TABLE III. Number of papers per code in percent - Top 10

Code	**Included in percent of papers**
mention generalization	94.92
repeat research process	91.53
no code applicable	84.75
adequate generalization	83.05
universal statement generalization	83.05
relevant result without specification	74.58
no generalization	69.49
relevant result for practitioner	67.80
mention further work	57.63
speculative generalization	54.24

Comparing the sentences with an inter-reviewer agreement of zero and 100 percent respectively, no significant difference in the lengths of the sentences could be

seen. Sentences with no agreement averaged about 160 characters and 24 words, while sentences where all reviewers agreed on the code averaged 154 characters and 23 words.
Coding the 179 sentences with an inter-reviewer agreement of 100 percent lead to three main codes which I assigned to some of the sentences.

1. In total, there were 37 sentences which included a keyword of the code they were assigned. For example, some sentences contained the words “practitioner” or “researcher” and were thus coded as “relevant results for practitioner” and “relevant results for research” respectively. Also, all sentences coded with “state first of a kind” with an inter-reviewer agreement of 100 percent had the word “first” in them.

    - "Our work benefit both researchers and practitioners." [6, p. 10]
    - "To our best knowledge, this study is the first research work on software developer onboarding tasks and strategies." [6, p. 10]

2. Additionally, 20 out of 23 sentences coded with “own further work” and an inter-reviewer agreement of 100 percent contained some phrase akin to “we intend”, “we plan” or “we will”, talking about the researchers themselves.

    - "First we intend to study the effect of having our prototype in use by engineers at Frequentis." [8, p. 11]

3. Furthermore, in seven of the ten sentences coded with “universal statement generalization” and an inter-reviewer agreement of 100 percent, some kind of hard wording is used, like “there is”, “[they] are” or “results show”.

    - "Test developers are forced to apply workarounds to over-come the current limitations of testing frameworks." [7, p. 11]

On the other hand, there were 1,333 sentences that had an inter-reviewer agreement of 0 percent, for which I managed to identify three main codes as well.

1. In a total of 68 sentences, I managed to pinpoint the problem to students not abiding by our set rules properly, meaning they used a code of a subgroup without also using a code of the other subtrees in the cluster. For example, the code “promising further work” was used, without using one of the “mention further work” codes, or “universal statement generalization” was used, without using one of the “mention generalization” codes. I do not claim to have found all occurrences, so I believe there to be many more.

2. There also were 21 cases where students disagreed on whether a sentence was a generalization or just repeating research process.

    - "By investigating the latest versions of 360 open-source applications using Google and AWS ML Cloud APIs, we have found 8 types of common API misuses that cause functionality, performance, and service cost problems." [15, p. 11]

3. Lastly, there were 14 cases where students disagreed on whether the sentence was talking about generalization or further work.

    - "Understanding team productivity is a new avenue for research and an important topic as developers are now more distributed during the pandemic and likely will continue to be so after it." [10, p. 11]

I was not able to find any more general codes that could be applied to a significant amount of the sentences.

5 Discussion

As previously mentioned, the data I obtained from the students was lacking in terms of data quality. More precisely, the final inter-reviewer agreement across all papers was 51 percent, including all data quality improvements I made. This means that the reviewers were unable to agree on almost half of the codes they assigned to sentences in the conclusion. Since there were only two codes with an inter-reviewer agreement of over 80 percent, which would be sufficiently high to make real statements about, a focus on such codes would not yield much either. Also, no single paper managed to achieve an inter-reviewer agreement of 80 percent or more, so there were none to focus on here at all. Further investigation showed no significant differences in the quality of the coding by different reviewers either, so excluding some outlier here was not possible.

That said, a few timid statements can be made about the data gathered. The fact that the most used code was “no code applicable” may be an indicator for a lackluster code system, as it would be preferrable to use this code as little as often and cover as many kinds of sentences as possible. Although this could have also been due to students failing to properly identify the right code to use. Generalization was identified in all but three conclusions in the papers, which is to be expected, considering a significant part of the conclusion should be generalizing one’s results to a bigger scope. On the other hand, “repeat research process” being used in the conclusions of all but five papers could be seen as an indicator for poor quality of the papers, as repeating the research process should not necessarily be a part of the conclusion, although our lack of data quality does not allow a definitive statement on that topic.

Analyzing the differences between sentences with an inter-reviewer agreement of 0 percent and 100 percent respectively yielded some promising leads. Firstly, I managed to identify some patterns that might shed some light on what makes some sentences or codes easier to code than others. As one would expect, finding a keyword of a code inside of a sentence makes that sentence more likely to be coded unanimously by the different reviewers. Sentences which contain certain words that very clearly and directly relate to specific code increase the inter-reviewer agreement as well. For example, using “we intend” etc. implies “own further work” directly, as well as using hard words like “there is” might imply a “universal statement generalization”.
More vague language on the other hand, coupled with the fact that the students are

German and perhaps do not understand English as well as their mother's tongue, perhaps lead to the students agreeing on codes less often. Furthermore, students did not always follow the rules that were given properly. This is shown by the fact that a lot of students failed to use the other subtrees when using a code in a cluster. Also, some students used code-nodes in the tree that had leaves beneath them, something that should not have been done.

Other reasons I uncovered for the low inter-reviewer agreement were more specific to the codes themselves. Students seem to disagree rather often on whether a sentence included a generalization or was simply repeating the research process, as well as on whether a sentence was generalizing or talking about further work. This may stem from a differing understanding of what generalization really is and perhaps to a certain degree on the language barrier again. A bit of ambiguity may well lead to students interpreting a sentence differently as well.

On a more general note, achieving a high inter-reviewer agreement with a group of students who do not know each other for the most part and have most likely never worked with each other is very difficult to begin with. None of the students having done open coding or content analysis before is definitely another factor contributing to a low inter-reviewer agreement.
That being said, some improvements could be made to ensure similar projects may yield better results in the future.

- Making sure the students understood the rules to coding clearly. This can perhaps be done by restating the rules every week for example and writing them down clearly and by discussing them with the students directly.
- Making sure that students have enough time for the work they are given. Reducing the workload by either decreasing the number of papers per student or focusing only on the conclusion from the get-go can help them make fewer mistakes along the way. Perhaps, introducing some data validation steps during the coding can also help the students better understand the traps and difficulties of coding properly.
- Refine the code system to be more concise and clearer. Less ambiguity in the code system should lead to the students agreeing on codes more often. Although some codes had memos, the students seemed to disagree on how some of them should be used, thus additional and clearer memos might help students better understand the codes. Also, consider merging some codes together whose distinction may be a little arbitrary or simply not important to the goal of the project.

Difficulties faced

During the semester, many obstacles were encountered by all students, including learning to code properly as well as a lack of time while coding 18 papers within three weeks.

My most prominent difficulty was the poor quality of data obtained from the coding during the semester. Although several improvements were tried, none were sufficient, so I ended up readjusting my goals to match this problem I encountered. Another difficulty I encountered afterwards was the highly exploratory nature of my analysis after this change in objectives. It was hard to find any good leads on how to progress with the data generated, so I mainly had to rely on my intuition and hard work. Finding patterns in the sentences with an inter-reviewer agreement of zero and 100 percent respectively was especially difficult. The way I finally managed to get promising results was to focus more on specific codes rather than try to find universal patterns that make a sentence complex. Afterwards, I managed to identify a few patterns in specific codes.

6 Threats to Validity

The validity of the work done in this thesis is prone to many threats. The internal validity of the first part of this project was already accepted to be poor in respect to the original goals of the thesis, as the students' agreement on codes was too low to draw any meaningful conclusions from. Yet, since the data gathered was then used for further analysis, validity concerns should still be further addressed.

Internal validity

The initial data was gathered from students in a non-controlled environment over several weeks, alas its quality is questionable. Students are known to often work in changing environments and under variable and suboptimal circumstances, such as stress or too little sleep, two factors I can personally vouch for. This is reinforced by the fact that the students in the project agreed on the workload being too big. This could have led to the students varying in their choices of codes given. Some students may have also worked together from time to time, which would defeat the purpose of having separate reviewers. This is very unlikely though, as the agreement between students is poor across the board.
The internal validity of the work following the course is to be investigated as well. Since I primarily worked alone after the course and failed to find proper hints in literature as to how to progress, my methods were not properly grounded. Thus, the quality of the data gathered, the results obtained, and the conclusions drawn, is questionable. I argue however, that since the conclusions drawn are rather timid and mostly consist of suggestions to future research, this threat to the internal validity is neglectable.

External validity

Since the preliminary data was gathered by students who likely have never done content analysis before, generalizing conclusions drawn to a broader audience may be unreasonable. Alas, my suggestions are catered mainly to similar projects working with students. Still, I think it is reasonable to assume that other researchers may face

similar challenges during their studies and may stand to benefit from my work as well. Another threat to the external validity of this thesis is the fact that articles were only drawn from one conference. But as the focus of this thesis is recent and future empirical studies in software engineering, I argue that this threat is minimal.

Conclusion validity

The conclusions drawn in this thesis are mainly suggestions for further research, that, as I would argue, are reasonably easy and intuitive to agree with. No great counterintuitive discoveries were made, or conclusions drawn, thus, threats to the conclusion validity are minimal.

7 Conclusions

Due to software engineering being relatively new to empirical research, it is reasonable to assume there are still some mistakes being made by most researchers. Thus, the original goal of this thesis was to evaluate the quality of recent empirical research in software engineering, potentially highlighting shortcomings the community can improve on. This goal was not accomplished; therefore, I suggest other researchers to repeat my work while learning from the mistakes made. This thesis provides some suggestions and hints for conducting better content analysis, especially when working with students or researchers new to this method. These suggestions focus on making the rules of the coding as well as the code system itself more concise and clearer, as well as minimizing mistakes made by the reviewers. Further work into the quality of empirical research in software engineering is crucial to improve future empirical studies conducted by the community, increasing their impact and value.

References

[1] M. F. Bosu and S. G. MacDonell, "Data quality in empirical software engineering: A targeted review," in *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE '13)*, New York, NY, USA: Association for Computing Machinery, 2013, pp. 171–176. DOI: 10.1145/2460999.2461024.

[2] J. Carver, L. Jaccheri, S. Morasca, and F. Shull, "Issues in using students in empirical studies in software engineering education," in *Proceedings. 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry (IEEE Cat. No.03EX717)*, IEEE Comput. Soc, 2003, pp. 239–249. DOI: 10.1109/metric.2003.1232471.

[3] R. Feldt and A. Magazinius, "Validity threats in empirical software engineering research-an initial survey.," in *Seke*, 2010, pp. 374–379.

[4] J. Gläser and G. Laudel, "Life with and without coding: Two methods for early-stage data analysis in qualitative research aiming at causal explanations," in *Life With and Without Coding: Two Methods for Early-Stage Data Analysis in Qualitative Research Aiming at Causal Explanations. Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*, 2. Forum: Qualitative Social Research, 2013, vol. 14. DOI: 10.17169/fqs-14.2.1886.

[5] A. Höfer and W. Tichy, "Status of empirical research in software engineering," in *Empirical Software Engineering Issues. Critical Assessment and Future Directions*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 10–19. DOI: 10.1007/978-3-540-71301-2_3.

[6] A. Ju, H. Sajnani, S. Kelly, and K. Herzig, "A case study of onboarding in software teams: Tasks and strategies," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, IEEE, 2021, pp. 613–623.

[7] D. J. Kim, N. Tsantalis, T.-H. Chen, and J. Yang, "Studying test annotation maintenance in the wild," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, IEEE, 2021, pp. 62–73.

[8] C. Mayr-Dorn, M. Vierhauser, S. Bichler, *et al.*, "Supporting quality assurance with automated process-centric quality constraints checking," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, IEEE, 2021, pp. 1298–1310.

[9] P. Mayring, "Qualitative content analysis," in *International Encyclopedia of Education(Fourth Edition)*, 2. Elsevier, 2000, vol. 1, pp. 314–322. DOI: 10.1016/b978-0-12-818630-5.11031-0.

[10] C. Miller, P. Rodeghero, M.-A. Storey, D. Ford, and T. Zimmermann, ""' how was your weekend?" software development teams working from home during covid-19," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, IEEE, 2021, pp. 624–636.

[11] J. Molléri, E. Mendes, K. Petersen, and M. Felderer, "Determining a core view of research quality in empirical software engineering," *Computer Standards and Interfaces*, vol. 84, p. 103 688, 2019. DOI: 10.1016/j.csi.2022.103688.

[12] D. Perry, A. Porter, and L. Votta, "Empirical studies of software engineering: A roadmap," in *Proceedings of the conference on The future of Software engineering - ICSE '00*, New York, NY, USA: ACM Press, 2000, pp. 345–355. DOI: 10.1145/336512.336586.

[13] C. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 557–572, Aug. 1999. DOI: 10.1109/32.799955.

[14] P. Tonella, M. Torchiano, B. Du Bois, and T. Systä, "Empirical studies in reverse engineering: State of the art and future trends," *Empirical Software Engineering*, vol. 12, no. 5, pp. 551–571, Mar. 14, 2007. DOI: 10.1007/s10664-007-9037-5.

[15] C. Wan, S. Liu, H. Hoffmann, M. Maire, and S. Lu, "Are machine learning cloud apis used correctly?" In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, IEEE, 2021, pp. 125–137.

[16] N. Wirth, "A brief history of software engineering," *IEEE Annals of the History of Computing*, vol. 30, no. 3, pp. 32–39, 2008.

[17] C. Wohlin, "Writing for synthesis of evidence in empirical software engineering," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '14*, vol. 46, New York, NY, USA, Article: ACM Press, 2014, pp. 1–4. DOI: 10.1145/2652524.2652559.

[18] C. Zannier, G. Melnik, and F. Maurer, "On the success of empirical studies in the international conference on software engineering," in *Proceedings of the 28th international conference on Software engineering*, New York, NY, USA: ACM, May 28, 2006, pp. 341–350. DOI: 10.1145/1134285.1134333.

A Appendix

TABLE IV. Full code system

Parent code	Code
	further work
further work	advice further work
further work	propose further work
further work	further work assignment
further work assignment	someone further work
further work assignment	anyone further work
further work assignment	own further work
further work	further work importance
further work importance	unimportant further work
further work importance	promising further work
further work importance	no classification of importance further work
further work importance	necessary further work
further work	further work kind
further work kind	distant further work
further work kind	other kind further work
further work kind	subsequent further work
further work kind	neighboring further work
further work kind	repeat further work
	generalization
generalization	generalization kind
generalization kind	universal statement generalization
generalization kind	recommendation generalization
generalization kind	speculative generalization
generalization	generalization degree
generalization degree	no generalization
generalization degree	for all generalization
generalization degree	limited generalization
generalization	generalization relevance evaluation by authors
generalization relevance evaluation by authors	not relevant result
generalization relevance evaluation by authors	relevant result for other group
generalization relevance evaluation by authors	unclear relevance
generalization relevance evaluation by authors	relevant result for practitioner
generalization relevance evaluation by authors	relevant result without specification
generalization relevance evaluation by authors	relevant result for research
generalization	generalization classification by reviewer
generalization classification by reviewer	too little generalization
generalization classification by reviewer	too strong generalization
generalization classification by reviewer	adequate generalization
	meta
meta	no code applicable

meta	section titles
section titles	discussion and implications
section titles	experiment results
section titles	conclusions and outlook
section titles	background and motivation
section titles	thread to validity
section titles	implications and future work
section titles	limitations
section titles	results discussion
section titles	further work
section titles	threats to validity
section titles	conclusions
section titles	conclusion
section titles	introduction
section titles	abstract
section titles	discussion
section titles	results
section titles	conclusion and future work
	research motivation
research motivation	building on past work motivation
research motivation	lack of research motivation
research motivation	other motivation
research motivation	practical need motivation
	research process
research process	repeat research process
research process	state first of a kind
	risk
risk	risk actions
risk actions	addressed risk
risk actions	untreated risk
risk	risk classification by authors
risk classification by authors	minor risk
risk classification by authors	unclassified risk
risk classification by authors	major risk
risk	risk discussion
risk discussion	state risk
risk discussion	state risk impact
	validity
validity	validity kind
validity kind	construct validity
validity kind	external validity
validity kind	internal validity
validity kind	conclusion validity
validity	validity discussion
validity discussion	state validity improvement

validity discussion	state validity limitations
validity discussion	state validity no effect
validity	validity classification by reviewer
validity classification by reviewer	minor impact classification
validity classification by reviewer	major impact classification
validity classification by reviewer	no impact classification