2017 IEEE 25th International Requirements Engineering Conference

Mining User Rationale from Software Reviews

Zijad Kurtanović
University of Hamburg
Hamburg, Germany
kurtanovic@informatik.uni-hamburg.de

Walid Maalej
University of Hamburg
Hamburg, Germany
maalej@informatik.uni-hamburg.de

***Abstract*—Rationale refers to the reasoning and justification behind human decisions, opinions, and beliefs. In software engineering, rationale management focuses on capturing design and requirements decisions and on organizing and reusing project knowledge. This paper takes a different view on rationale written by users in online reviews. We studied 32,414 reviews for 52 software applications in the Amazon Store. Through a grounded theory approach and peer content analysis, we investigated how users argue and justify their decisions, e.g. about upgrading, installing, or switching software applications. We also studied the occurrence frequency of rationale concepts such as issues encountered or alternatives considered in the reviews and found that assessment criteria like performance, compatibility, and usability represent the most pervasive concept. We then used the truth set of manually labeled review sentences to explore how accurately we can mine rationale concepts from the reviews. Support Vector Classifier, Naive Bayes, and Logistic Regression, trained on the review metadata, syntax tree of the review text, and influential terms, achieved a precision around 80% for predicting sentences with alternatives and decisions, with top recall values of 98%. On the review level, precision was up to 13% higher with recall values reaching 99%. We discuss the findings and the rationale importance for supporting deliberation in user communities and synthesizing the reviews for developers.**

***Index Terms*—App Analytics, Rationale, Review Mining**

I. Introduction

According to Merriam-Webster *rationale* is "the explanation of controlling principles of opinion, belief, practice, or phenomena, or an underlying reason.". Rationale management focuses on capturing and sharing the reasons and justifications behind decisions. Over the last decades, managing design and requirements rationale has been a major concern in software engineering [3], [6], [10]. Ideally, the rationale should be captured in requirements and design artifacts to document why certain project decisions were taken [7]. This includes the questions or issues encountered by designers and analysts, the alternatives explored to solve the issues, and the criteria to evaluate the alternatives [2]. Rationale is also often found in informal artifacts such as team conversations or sketches [12].

With the increasing popularity of social media, user forums, and app stores, software vendors started giving more attention to the input of users when making decisions about software design, development, and evolution [16]. This observation led to the main question behind this work: Is there a "user rationale" which can be valuable for software and requirements engineering and thus should be captured and managed? A quick browse through the software reviews on Amazon shows that users not only share their opinions about software but also report *issues* (e.g. "there's no driver for my multifunction laser printer/scanne Samsung CLX-3160FN"), provide insights about the *alternatives* considered with their evaluation *criteria* (e.g. "it does everything norton did and more – without weighting down your processor speeds"), or describe their conclusive *decisions* (e.g. "did you know that Quicken no longer offers phone support? For me that's a huge negative, and if I knew that I would not have purchased the product").

Systematically identifying and managing the rationale in users' debates or feedback might bring several benefits. First, capturing user rationale would make users' tacit knowledge about preferences and needs more transparent. This would help software teams base their decisions on an improved understanding of the users and their reasoning. Second, user rationale might reveal the alternatives considered (e.g. other products, configurations, workarounds) and the criteria used to evaluate these alternatives (e.g. usability, performance): a useful knowledge for analysts, designers, and testers to derive their own decision criteria, arguments, and even new requirements. Finally, user rationale might be considered a crowd-based extension of design and requirements documentation.

This paper takes a first step towards understanding users rationale by empirically studying the reviews for software products available on Amazon's software store (www.amazon.com). Our goal is to develop a grounded theory for user rationale and evaluate approaches to automatically mine such information from the reviews. Section II introduces the methodological framework underlying this empirical work.

The contribution of our work is threefold. First, we introduce a new grounded, user-data-driven perspective on rationale in requirements engineering. We *qualitatively* describe how users denote rationale (Section III) and *quantify* the occurrence of rationale and its concepts such as alternatives and criteria in user reviews (Section IV). Second, we discuss how rationale concepts can automatically be *detected* in user sentences with supervised machine learning (Section V). Third, we *compare the accuracy* of various machine learning approaches and configurations to detect reviews and sentences that include rationale concepts. We discuss the research results in Section VI and limitations in Section VII, sketch related work in Section VIII, and conclude the paper in Section IX.

II. Research Design

We first introduce the research questions and then describe the research methods and data used in this work.

2332-6441/17 $31.00 © 2017 IEEE
DOI 10.1109/RE.2017.86

IEEE computer society

A. Research Questions

The overall goal of the paper is to qualitatively and quantitatively explore user rationale in software reviews and investigate approaches to automatically mine rationale information. We focus on the following three research questions:

- RQ1 How do users denote rationale in online reviews, and which concepts (information types) do they include?
- RQ2 How is the frequency distribution of the various rationale concepts over the reviews?
- RQ3 How accurately can different configurations of machine learning techniques mine rationale concepts from the user reviews and sentences?

With RQ1, we aim to develop a theory of user rationale, which includes concepts users apply to argue and explain decisions (e.g. issues or assessment criteria). RQ2 aims to provide quantitative evidence about various concepts of user rationale and how they are combined and distributed. RQ3 aims to evaluate and compare different machine learning techniques to mine the rationale concepts from a large and heterogeneous user input.

B. Research Method

Figure 1 overviews our methodological framework, which includes four main phases described below.

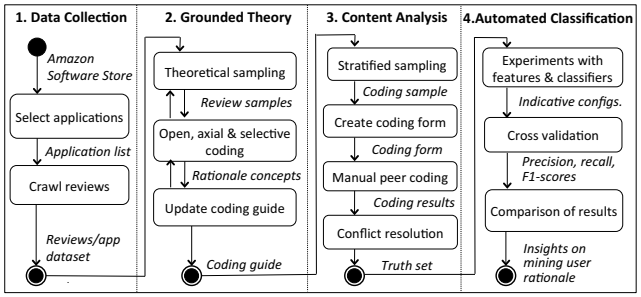


Fig. 1. Overview of our research methodology with four phases.

1) Data Collection: We first selected from the Amazon's software store the applications to study. We then crawled the online reviews for these applications to build the research data. In Amazon, users can rate products on a scale from one to five stars and write a text feedback. The user review consists of the title, the review text, the rating, and metadata including the name of the reviewer, whether the reviewer purchased the product, and the review submission date. Other users are allowed to comment on a review or vote for it to indicate its helpfulness. We manually compiled a list of popular applications with many reviews. We selected three to four applications from each software category in the store. The final list includes 52 applications from 17 categories (Table I).

Then we crawled the reviews for the selected applications from Amazon between November 2014 and May 2016, using the freely available crawler on GitHub[1]. Compared to mobile app stores such as Google Play or App Stores, Amazon has an older reviewing culture, longer and potentially more informative reviews. Table I summarizes our dataset.

TABLE I
OVERVIEW OF THE COLLECTED DATASET.

No.	Software Category	#Apps	#Reviews	#Sentences
1	Accounting & Finance	3	2,396	10,161
2	Antivirus & Security	3	11,752	33,762
3	Business & Office	3	2,224	9,585
4	Children's	3	1,817	6,427
5	Design & Illustration	3	1,030	4,531
6	Digital Software	3	543	3,394
7	Education & Reference	3	512	3,439
8	Games	3	1,872	12,990
9	Lifestyle & Hobbies	3	905	5,384
10	Music	3	440	2,206
11	Networking & Servers	3	497	2,704
12	Operating Systems	3	2,100	11,823
13	Photography	3	667	3,322
14	Programming & Web Development	4	435	2,242
15	Tax Preparation	3	2,330	10,119
16	Utilities	3	1,696	7,121
17	Video	3	1,198	6,185
	Total	**52**	**32,414**	**135,395**

2) Grounded Theory Approach: In the second phase, we manually analyzed a review sample following a Grounded Theory approach after Strauss and Corbin [27] to answer RQ1. Grounded theory is a systematic, qualitative method to develop a theory based on evidence in data. The goal and final output of this phase was the creation of a coding guide that describe the theory and that was used for the creation of the truth set during the content analysis. The coding guide can be downloaded from the project website[2]. The *Coding Guide* systematizes the coding task and reduces disagreements between peer-coders [17], [20]. It includes instructions on the coding process and the tasks, clear definitions of the concepts, detailed review examples and counter-examples, and additional hints as guidelines for coding. As Neuendorf [20] suggests, we conducted 8 coding dry runs and refined the guide from the feedback of the coders. We involved four coders in the dry runs, interviewed them, and discussed the disagreements.

We categorized, related, and refined the codes during the analysis in multiple iterations (open, axial, and selective coding). In each iteration, we used a theoretical sample of reviews and sentences including only data with potential new insights. During open coding, we focused on getting an improved understanding of the argumentation of users and related aspects they refer to. We continuously assigned codes to those aspects and made notes about our observations, particularly about special keywords and syntax. After each iteration, we refined the code definitions and extracted example sentences.

3) Manual Content Analysis: The third phase consists of the manual content analysis techniques as described by Neuendorf [20] and Maalej and Robillard [17], involving the systematic manual assessment of a document sample by human coders. The goal was to answer RQ2 and to create a truth set for answering RQ3.

[1]https://github.com/aesuli/Amazon-downloader, accessed on March 2016

[2]https://mast.informatik.uni-hamburg.de/app-review-analysis/

TABLE II
OVERVIEW OF THE CODING SAMPLE (USED FOR CONTENT ANALYSIS).

Star rating	#Reviews	#Sentences	#Sentences+titles
1 star	204	1643	1847
2 star	204	1434	1638
3 star	204	1169	1373
4 star	204	1223	1427
5 star	204	799	1003
Total	**1,020**	**6,268**	**7,288**

This phase included three main steps. First, we generated a stratified random sample of 1020 reviews, in proportion to the ratings and software categories. Table II shows the resulting totals of reviews and sentences for each star rating. As suggested by Maalej and Robillard [17], we did not adopt a uniform random sampling strategy, because reviews from categories with only a relatively small number of applications might be underrepresented or for certain ratings not represented at all.

In the second step, we created a coding form as an Excel document. We choose sentence as the basic information unit to be coded to enable a more fine-grained content analysis.

The third step consisted of peer-coding the reviews. Each of the seven coders received a coding form with the reviews and the coding guide. The coding form contained random coding assignments from the coding sample. For each of the seven coders, the coding assignments were peered and equally distributed to all remaining coders. Since one coder was unable to finish on time, the coding assignments of that coder were distributed to two of the remaining six coders. The coders were three master students of computer science, one Ph.D. student, one IT professional, and the first author of this paper. They all had a high command of English, had software development experience, and were trained for this task.

After the peer-coding, we merged the coding results and assessed their reliability, by calculating the coder agreement-rate and Cohen's kappa [5]. The coding results were merged into one coding form to resolve the disagreements. The resolution of disagreements was done by four coders (referred to as disagreement solvers) in two phases. Half of all disagreements were peer-solved by the first author and a second coder in several iterations. All disagreement solvers were involved in this step. The disagreement solvers improved their understanding of the rationale concepts, discussing and peer-solving the disagreements. The remaining half were split into three parts and recoded by one of the disagreement solvers.

4) Automated Classification: In the fourth and final phase of the study we assessed whether we can detect concepts of user rationale on the review and sentence level.

We used the Natural Language Toolkit NLTK[3] and the Stanford Parser[4] to analyze the reviews. In particular, we applied the tools for text preprocessing and feature extraction for concept classifiers. We then built supervised classifier, employing the machine learning library scikit[5]. We used our truth set for their training and evaluation.

[3]http://www.nltk.org/, accessed June 2017
[4]http://nlp.stanford.edu/software/lex-parser.shtml, accessed June 2017
[5]http://scikit-learn.org/, accessed March 2017

We experimented with different features: text, metadata, sentiment, and syntax features. Bag of words and word collocations such as bigrams and trigrams, are examples of textual features. Star rating and text length is an example of a metadata feature, while sentiments capture the positive, neutral, and negative emotions of a text. Examples of syntax features are: bag of part of speech (POS) tags, POS tag collocations, and text syntax tree height. We evaluated the prediction accuracy of classifiers using common metrics: precision, recall, and their harmonic mean known as the F-Measure (also called F1).

III. GROUNDED THEORY OF USER RATIONALE

Related codes that emerged during open coding were grouped into concepts (axial coding), until finally forming a theory of *User Rationale*. The following concepts are defined in the final coding guide: issue, alternative, criteria, decision, and justification. The necessity of the codes as defined in the coding guide was driven by their prevalence, their meaning, and how it is related to the study goal. For instance, if a code only occurs a few times in the open coding, it was merged to a related concept. If a code was discussed and found unrelated to rationales it was either removed or merged. We chose common, transparent names for the codes, for example, taking inspirations from the FURPS model for classifying software quality attributes for code names of the concept criteria. The coding guide also includes open options for "others" not yet identified concepts. In the following, we describe the rational concepts and give examples and counter-examples from the dataset (see coding guide for details).

a) Issue: The code for this concept is assigned to a sentence, if it reports a *concrete* issue or problem with the software. The problem can denote a bug in the software, an issue with a feature, an issue with software's quality, an issue with the software products's auxiliary (e.g. product documentation, support, licensing, or driver issues), or an issue described by the user that can or should be solved. An example sentence for this concept is: "It does NOT work in a server/multi-user environment, strictly one user at a time ONLY!!". Another example sentence reporting a usability issue is: "Can't find any controls that I'm use to using and doing searches for instructions isn't much help". A counter example is: "I had problems with it at first, but it was worked out to my satisfaction", as this sentence lacks a concrete issue.

b) Alternative: This concept consists of three codes that can be assigned to a sentence, if it reports a concrete alternative option described by a user. The code *alternative software* is assigned when the user mentions an alternative software (not version) in the review, such as: "Was using Defender Pro for 5 years before switching to Webroot". The code *alternative version* applies when an alternative edition, version, or release of the reviewed software application is mentioned. An example sentence is: "But I've heard good things about Corel Paintshop, and Paintshop Pro X6 Ultimate is certainly a more powerful program than I've been using". The code *alternative feature* is assigned when the user mentions a feature of another software, an improved version of a feature, a missing or

requested feature, or when two or more alternative features of the reviewed software are compared. A counter example is: "I read the reviews on cheaper items and decided to stay away from them" as this sentence refer to unclear alternatives.

c) Criteria: This concept consists of four codes that can be assigned to a sentence, if it reports a concrete assessment criterion reported by a user. Typically, such sentences report assessments of non-functional requirements. The code *usability* refers to sentences that address human factors related to the usage, user interface layout, attractiveness, aesthetics, consistency, learnability (how easy it is to learn using the software), integrated help, documentation, training material, or similar criterion related to usability of the software or its auxiliaries. Example sentences with this code are "The interface is more web interface-like, which will take me a little getting used to" and "The number of special effects is also impressive comparing X3 to X7".

The code *reliability* is assigned to sentences that address accuracy, frequency and severity of the failure, recoverability (e.g. after crashes), stability, availability (e.g. of a server), safety, or a similar criterion related to the reliability of the software. A candidate sentence for this code is "This product is reliable, it automatically updates regularly and best of all it is free!" or "Payee names change when transactions are downloaded, to names that are totally incorrect and inappropriate". The code *performance* refers to sentences that address speed, efficiency, throughput, response time, recovery time, start-up time, resource consumption (power, memory, battery, cache, etc.), capacity, scalability, or a similar criterion related to the performance of the software. Am example is: "The time between turning on the computer and the login screen pulling up is so much faster than Windows 7".

Finally, *supportability* refers to sentences that address serviceability, adaptability, maintainability, flexibility (modifiability, configurability, adaptability, extensibility, modularity), localizability (e.g. language, currency), accessibility, reusability (compatibility, interoperability, portability), or similar criterion related to supportability of the software. For instance: "Even if you are a power user, it has plenty of advanced system features and customizable settings for you to use".

d) Decision: This concept consists of four codes that can be assigned to a sentence reporting a single, conclusive, and actionable decision that is considered, taken, or planned.

The code *acquire software* applies to sentences that report a purchase, download, install, or similar decision on software acquisition. An example sentence is "I was somewhat skeptical about this particular photography editing software, but got it on the recommendation of a friend". The code *update software* applies to sentences that report an update, upgrade, renewal, or similar software renewal decision, as in example "My only quibble is that when I updated X6 I had to click through several screens to actually download the update". The code *relinquish software* refers to sentences that report software abandoning, returning, definite uninstalling, or a similar decision. As the excerpt shows: "I contacted tech support, ... before giving up and using another backup program". The code *switch software* is used for sentences that report decisions on changing a software. For instance: "Since I've upgraded to Windows 7, I've been looking for a replacement for GoBack".

e) Justification: The code for this concept is assigned to a sentence, if it contains a justification or acts as a justification for another sentence. That is, a justification can be part of a compound sentence where the statement it refers to is made, or can relate to a statement contained in another sentence. A justification either explains the reasons behind a statement or acts as an argument.

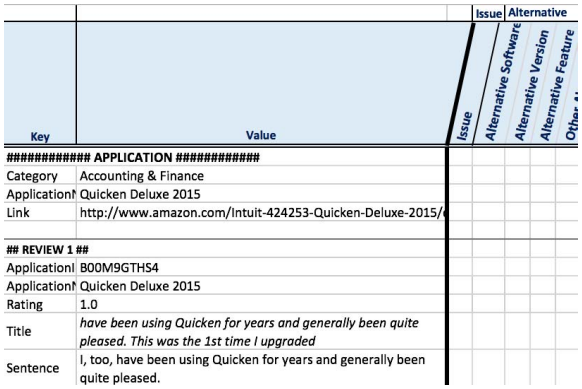
An explanation or argument can be explicitly related to a statement by certain lexemes such as connectives or syntactic constructions, whereas implicit relations can be detected only on the basis of a knowledge base and inference [24]. Explicitly related arguments are separated typically by a subordinate conjunction (e.g. because, until), prepositions (e.g. after, in), or just a comma. The following example illustrates an argument involved in an implicit (a) relation: "If you aren't an IT person or a nerd, don't purchase this software (statement). It's like going from basic math to calculus overnight (argument)".

Words, phrases, and clauses are combined using coordination and subordination. Coordination combines short independent clauses into single sentences, while subordination transforms independent clauses into dependent clauses. A subordinate conjunction emphasizes the importance of the independent clause and acts as a transition between two thoughts in a sentence. Since humans write unstructured text and typically adhere to the common linguistic rules, we can expect to find arguments in compound sentences. For instance, a sentence that has one independent clause (i.e. the statement), and one dependent clause (i.e. the justification). We state three review sentences with different types of clauses: a purpose, reason, and result clause.

A purpose clause expresses the purpose for an action in the independent clause, commonly introduced with the to-infinitive clause or with 'so that', 'in order that', 'for the purpose', 'in order to', and many other ways. An example is "But there is more than enough reason to upgrade to X6 Ultimate *in order to* get the Perfectly Clear and Face Filter 3 plug-ins". This sentence gives support for upgrading a product because of new explicitly mentioned features.

Reason clauses contain answers to why-questions. They give an explanation for a statement in the independent clause. This is typically introduced with conjunctions 'because', 'as', or 'since', e.g: "I am updating again *because* it does not work with 8.1". 'Because' is the typical argumentation marker that is not ambiguous in meaning like 'since'. The word 'since' can also be used as preposition or adverb.

Finally, result clauses indicate the result of an action or a situation. They are introduced by conjunctions such as 'so', 'so ... that', 'such ... that', or as/with a result, e.g. : "Capturing and editing video challenges the capabilities of any computer, *so* you should assess your computer's capabilities and this software's requirements before making a purchase decision".



		Issue	Alternative			
Key	Value	Issue	Alternative Software	Alternative Version	Alternative Feature	Other
########## APPLICATION ##########						
Category	Accounting & Finance					
ApplicationN	Quicken Deluxe 2015					
Link	http://www.amazon.com/Intuit-424253-Quicken-Deluxe-2015/					
## REVIEW 1 ##						
ApplicationI	B00M9GTHS4					
ApplicationN	Quicken Deluxe 2015					
Rating	1.0					
Title	*have been using Quicken for years and generally been quite pleased. This was the 1st time I upgraded*					
Sentence	I, too, have been using Quicken for years and generally been quite pleased.					

Fig. 2. A screenshot partly showing the coding form.

IV. User Rationale Frequencies

Each coder received the coding guide and the coding form containing all reviews from the coding sample. The coders assessed each title and sentence of the review, indicating whether it contains an issue, alternative, criteria, decision, or justification. Figure 2 shows a screenshot of the coding form. The first two columns named *Key* and *Value* denote a review item (i.e. the row) such as title, rating, or sentence. The next 16 columns represent one of the 16 user rationale codes. Columns representing the same user rationale concept adjoin each other. A coder could assign an x to any of the columns to denote that a sentence or title reports the code. The columns *Done* and *Comment* are used to mark the coding of a sentence as completed or add comments.

The reviews were grouped by the application, i.e. all reviews of an application that are part of the coding assignment appear in a consecutive order. The link to each application is at the beginning of a group. It allowed a coder to read application's description on Amazon before starting to code the reviews. The application's name is repeated for each review. Multiple codes for a title or a sentence of a review were allowed. Coders were able to stop and resume their coding tasks at any time.

After the peer-codings were completed, we merged all coding results and assessed the inter-coder agreement. Coders were considered to agree on a code on the sentence level, if both peers agreed on that code for that sentence. On the review level, coders were considered to agree on a code, if both peers agreed at least once on that code in that review. The inter-coder percent agreement on the sentence level ranged from 83% - 99% (average: 91%), while the Cohen's $kappa$ ranged from 0.17 - 0.55 (average: 0.36). On the review level, the inter-coder agreement ranged from 65% - 94% (average: 80%), while the Cohen's $kappa$ ranged from 0.27 - 0.66 (average: 0.43). The $kappa$ values are statistically significant with $p < 0.01$.

The resulting conflict-free truth sets contained only coding assignments where two of three coders agreed. From the sentence-level truth set, we derived the review-level truth set.

Overall 48% of sentences and 86% of reviews contain at least one user rationale concept. Justifications are reported by users in 39% of all reviews. Table III shows an overview of the rationale concepts' frequencies in our sample.

TABLE III
Truth set: on the sentence level and the review level.

	Issues	Alternatives	Criteria	Decisions	Justifications
Sentences with	1182	794	1981	570	785
Reviews with	484	390	726	387	396

The mosaic plot in Figure 3 gives an overview of concept collocations in reviews with rationale. The shading of the tiles highlights the sign and magnitude of the standardized Pearson residuals under the assumed model of independence. Issues, criteria, alternatives, and decisions tend to co-occur in reviews with a notable fraction of justifications ranging from 21 to 71%. It seems that, the more informative the reviews are (i.e. higher co-occurrence of issues, criteria, alternatives, and decision), the higher is the justification density. Criteria as the most pervasive concept, appear most often alone compared to other concepts (in 131 or 13% of all reviews, with a fraction of 21% justifications). In 123 reviews (or 12%) the concepts appear all together, with a considerable fraction of justifications (71%). Justifications appear in less than 2% of reviews without any other concept.

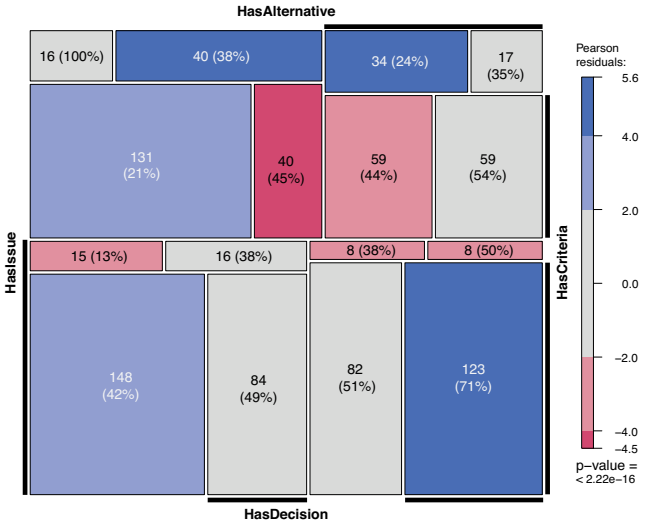


Fig. 3. Frequencies of co-occurrences of user rationale concepts. Parenthesis hold the justification ratio, black side-bars indicate a concept's presence.

We studied the distribution of the user rationale concepts with respect to 1-5 star ratings. Issues tend to appear more often in lower-rated reviews. The mean rating (2.2) for reviews that report issues was the lowest compared to other concepts ($p < 1e-06$, Student's t-test). This is in line with earlier studies [15], [22]. Also, decisions seem to occur more often in lower rated reviews, while criteria, alternatives, and justifications seem to appear more often in 3-star rated reviews.

We also studied the difference of reviews' verbosity (measured in word count) with respect to the reported rationale concepts. Reviews reporting alternatives seem to be more verbose than reviews reporting criteria. The mean verbosity of reviews including alternatives amounts to ∼171 words (median: 108), compared to ∼128 words (median: 74) for

TABLE IV
FEATURES USED TO PREDICT RATIONALE IN USER REVIEWS.

Feature	Description
Body n-grams	N-grams of words of the review body, for $n \in \{1, 2, 3\}$
Title n-grams	N-grams of words of the review title, for $n \in \{1, 2, 3\}$ (for review classification only)
Rating	Review rating
Length	Text length
Sentiment	Review body lexical sentiment score $\in \{-5, 5\}$
T-sentiment	Title sentiment (for review classification only)
POS n-grams	N-grams of part of speech (POS) tags on the word level, based on the Penn Treebank Corpus [19], $n \in \{1, 2, 3\}$
CP unigrams	Unigrams of part of speech (POS) tags on the clause and phrase level (CP), based on the Penn Treebank corpus [19]
Subtree count	Sentence syntax sub-trees count (review classifier: mean value)
Tree height	Sentence syntax tree height (review classifier: mean value)

reviews reporting criteria. Reviews reporting decisions seem to be more verbose (mean: ∼150, median: 97) compared to those reporting criteria (mean: ∼128, median: 74). Furthermore, reviews reporting justifications tend to be more verbose (mean: ∼163, median: 105) than reviews reporting criteria. The differences between these means are statistically significant (with $p < 0.05$, Student's t-test).

There were also statistically significant differences in the verbosity of sentences with respect to the concepts reported ($p < 0.05$, Student's t-test). Sentences that report issues, alternatives, criteria, and decisions tend to be 3 to 5 words shorter on average than those that report justifications (mean: ∼24 words, median: 20 words). Also, sentences reporting alternatives tend to be longer than those reporting decisions.

V. AUTOMATED CLASSIFICATION

We built two datasets that represent the truth set: a dataset for sentence classifier and a dataset for review classifier.

We evaluated one classifier for each concept: that is 5 sentence-level classifiers and 5 review-level classifiers. We used the popular classification algorithms Support Vector Classifier (SVC), Naive Bayes (NB), and Logistic Regression (LR). We conducted 10-fold cross validation [11], with an equal ratio of candidate and non-candidate items. The goal was to identify configurations that lead to accurate classifiers rather than on systematically evaluating all possible configurations.

Besides experimenting with part of speech (POS) tags – on the word, clause, and phrase level – based on the Penn Treebank Tagset [18], we also studied features to measure text sophistication such as text length, text syntax tree height, and count of clauses and phrases. The classifiers employed features listed in Table IV. The preprocessing steps were stopwords removal (abbreviated: -stops), punctuation removal (abbreviated: -punct), and lemmatization (abbreviated: +lemma).

A. Classifier Accuracy

We automatically generated a set of classifier configurations: combinations of compatible preprocessing steps and features that were used to train and test the classifiers. Besides excluding invalid configurations, we also excluded those containing only metadata features. Our nitial experiments showed that metadata is not sufficient to achieve a high classifier accuracy, similar to the earlier experiments of Maalej et al. [15].

TABLE V
MOST ACCURATE CLASSIFIERS TO MINE USER RATIONALE IN SENTENCES

R. concept	Prec.	Recall	F1	Configuration
Issue	**0.74**	0.70	0.71	NB: Body 1-grams, lemma, -stops, -punct, rating, sentiment, syn. subtree count, syn. tree height
	0.60	**0.93**	0.73	NB: Body 1-grams, POS 1-2grams, -punct
	0.70	0.86	**0.77**	LR: Body 1&2-grams, POS 1&2grams, -stops, -punct, rating, length, sentiment
Alternative	**0.80**	0.84	**0.82**	SVC: Body 1&2-grams, POS 1&2-grams, +lemma, -stops
	0.60	**0.98**	0.75	NB: Body 1&2-grams, POS 1&2-grams, +lemma, -stops, -punct, rating, length, sentiment, CP 1-grams, subt. count, t. height
	0.77	0.88	**0.82**	SVC: Body 1&2-grams, POS 1&2-gram, -stops, subt. count
Criteria	**0.76**	0.75	0.75	SVC: Body 1-grams, -stops, rating, CP 1-grams
	0.60	**0.96**	0.74	NB: Body 2-grams, POS 1&2-grams, +lemma, -stops, -punct, rating, length, sentiment, CP 1-grams, t. height
	0.71	0.84	**0.77**	SVC: Body 1-grams, POS 1-grams, rating, length, t. height
Decision	**0.80**	0.80	0.80	LR: Body 1-grams, -stops, rating, sentiment
	0.63	**0.98**	0.76	NB: Body 2-grams, length, subt. count
	0.76	0.93	**0.83**	SVC: Body 1&2-grams, +lemma, rating, subt. count
Justification	**0.69**	0.74	0.71	LR: Body 1-grams, POS 1-gram, sentiment
	0.60	**0.93**	0.73	NB: Body 1-grams, POS 1-grams, -punct, length, CP 1-grams, t. height
	0.66	0.86	**0.74**	SVC: Body 2-grams, sentiment, t. height

Prior to running a classifier, the list of classifier configurations was randomized. For each sentence classifier, we evaluated between 10,109 - 17,457 different configurations using cross-validation. For each review classifier, the number of cross-validated evaluations using different classifier configurations was between 9,895 - 20,433.

Table V and VI give an overview of classifier configurations leading to the top performance in mining user rationale in sentences and reviews respectively. We filtered all results having precision, recall, or the F1-score below 0.6.

For the sentence classifier, we achieved the highest values for precision (0.80) and F1-score (0.83) for classifying decisions. The sentence classifier for alternatives and decisions achieved overall the highest recall values (0.98), but overall lowest precision for classifying justifications. On the review level, the classifier for criteria achieved overall best precision and recall, reaching values for precision, recall, and F1-score of respectively 87%, 99%, and 81%. We achieved the best F1-score with the classifier for issues (0.82).

B. Feature Analysis

We used an ensemble of tree classifiers using the sentence and review dataset, to assess the feature importance for the review and sentence level classification. The classifiers were Adaptive Boost, Extra Tree, Gradient Boosting, and Random Forest. For each feature, we averaged the sum over all importance scores obtained by these classifiers. We then selected the top 10 most informative features.

1) Most informative features for Sentence Classifier: The most informative features for the concept Issue was rating, while the second most significant was length. The further

TABLE VI
MOST ACCURATE CLASSIFIERS TO MINE USER RATIONALE IN REVIEWS.

R. Concept	Prec.	Recall	F1	Configuration
Issue	**0.83**	0.61	0.70	NB: Body 1-gram, +lemma, -punct, rating, length, sentiment, t. height
	0.60	**0.98**	0.74	NB: Body 1-grams, -stops, -punct, length, CP 1-grams, subt. count, t. height
	0.78	0.86	**0.82**	SVC: B 1-grams, +lemma, -punct, rating, length, sentiment, CP 1-grams
Alternative	**0.81**	0.70	0.72	NB: Body 1-grams, -stops, rating, sentiment, subt. count, t. height
	0.62	**0.98**	0.74	SVC: Body 1&2grams, POS 1&2-grams, title, +lemma, -stops, subt. count, t. height
	0.74	0.91	**0.81**	NB: Body 1-grams, -stops, rating, length, sentiment, t-sentiment, t. height
Criteria	**0.87**	0.61	0.71	NB: POS 1&2-grams, T. 2-grams, length, sentiment , t. height
	0.60	**0.99**	0.74	NB: Body 2-grams, POS 1&2-grams, +lemma,-punct, length, sentiment
	0.73	0.92	**0.81**	SVC: Body 2-grams, +lemma, -punct, rating, sentiment
Decision	**0.80**	0.67	0.72	NB: Body 1-ngrams, -stops, rating, t-sentiment
	0.60	**0.96**	0.73	NB: Body 1-grams, -punct, rating, length, CP 1-grams, subt. count
	0.71	0.89	**0.78**	SVC: Body 1-grams, +lemma, length, t-sentiment, subt. count, t. height
Justification	**0.82**	0.60	0.70	NB: POS 1&2-grams, T. 1-grams, -punct, rating, sentiment
	0.61	**0.96**	0.74	SVC: Body 1&2grams, +lemma, -stops, sentiment, t. height
	0.71	0.87	**0.78**	SVC: Body 1-grams, -stops, length, sentiment, CP 1-grams, t. height

significant features were 2 POS unigrams, 2 POS bigrams, 2 POS trigrams and a word unigram. Rating and length were together almost three times more informative than the remaining features. The best scoring POS feature ranked overall third was the bigram RB-VB, i.e. adverb, and verb. An example excerpt of a sentence with this feature is "... *repeatedly attempting* to get my bank transactions loaded with only occasional random success". The best ranked POS trigram was MD-RB-VB, i.e. modal, adverb, and verb. This feature is found in "after using a couple of times it *won't let me proceed* until I register".

For the concept Alternative, POS unigrams were the most informative features, accounting for almost half of the overall top 10 significance score. The best scoring among them was the unigram NNP (proper singular noun), while the best scoring POS bigram was NNP-CD (i.e. singular noun and cardinal number). Among the best scoring word unigram feature were the word 'version' and 'versions'. Length was placed at the 10th rank of significance.

For the concept Criteria, length and sentiment were among the most significant features. They account for almost half of the overall score achieved by the top 10 informative features. The POS bigrams scored slightly better than the word unigrams. The best scoring bigrams were RB-JJ (i.e. adverb and adjective) and TO-PRP (i.e. to infinitive and personal pronoun). The most significant words were 'are', 'easy', and 'not'. The 10th placed feature was the POS trigram NNS-IN-DT (noun plural, preposition, and determiner).

The top 10 features for the concept Decision were word and POS unigrams, syntactic subtree count, and POS bigrams. Word and POS unigrams had a similar significance, word unigrams being slightly more important. VBD (i.e. verb past tense) was the most significant POS tag while the word 'bought' was the most significant word. The most significant bigram was PRP-VBD (i.e. personal pronoun and verb past tense). A sentence with this feature is "This is the thing that will make *me return* this in 55 more days". The subtree count feature is as informative as the POS bigrams.

Length was the best ranked feature for the concept Justification. Word unigrams were of similar importance, with the argumentation marker 'because' scoring best. The POS unigrams and bigrams were each approximately a third as informative as the word unigrams. The VBD (i.e. verb past tense) was the most informative POS unigram. The most informative bigram was NN-TO (i.e. noun and to infinite). The tree height feature was ranked 7th most informative feature.

2) Most informative features for Review Classifier: Rating was the most informative feature for the concept Issue. Among the top 10 best features were also length and sentiment. Rating alone was almost three times more informative than all top scoring words, and more than three times as informative as POS bigrams. The most informative word was 'not' in the review title. The best placed POS bigram was RB-VB (i.e. adverb and verb), that is present in "Syncing my online accounts just does *not work* correctly because it keeps asking me to enter a password even though it's saved".

For the concept Alternative, length was the most informative feature among the top 10. It scored more than all top ranked word unigrams, and almost two times more informative than the one POS unigram CD (i.e. cardinal number). The best ranked word was 'version', while the best ranked POS bigram was TO-NNP. The trigram VBG-IN-DT (i.e. verb in present participle, preposition, and determiner) was placed 10th. A review sentence with such a feature is given with "I started *playing with the* original Sims and bought Sims 2".

Length was the most informative feature for the concept Criteria. It was approximately three times more informative than the sentiment score. The four top ranked word unigrams had a similar significance as the top three ranked POS bigrams. They each had less than half of the significance of the feature length. The top words were 'are', 'not', 'easy', and 'user'. The best placed POS bigram was RB-JJ (i.e. adverb and adjective). The second best was PRP-VBD (i.e. personal pronoun and verb in the past tense). The third best places POS bigram was PRP-CC (i.e. personal pronoun and coordinating conjunction). An example review sentence having the feature RB-JJ is given with "For me, it is very dependable and easy to use", for the feature PRP-VBD with "For me, *it is* very dependable and easy to use". The top scoring POS unigram is JJ (i.e. adjective).

POS bigrams, followed by word unigrams, POS unigrams, rating, and POS trigrams were among the top 10 features for the concept Decision. The overall single best feature was PRP-VBD (i.e. personal pronoun and verb in the past tense). An example review sentence having this feature is "*We were* FORCED to upgrade to the newest version (Pro 2014) in order to continue using payroll & merchant services". Among the best placed words were the verbs 'bought' and 'upgrade'. VBD

(i.e. verb in the past tense) was the best ranked POS unigram and VBG-DT-NN (i.e. verb in present participle, determiner, and noun) the only POS trigram ranking among the top 10.

Length scored best for the concept Justification. It was slightly better than the top 5 words. The remaining top features were two POS bigrams, one trigram, and one CP unigram. The argumentation marker 'because' was ranked 2nd and was the best scoring word. Other top scoring words were 'bought', 'you', 'software', and 'since'. The best ranked POS bigrams were IN-NN (i.e. preposition and noun) and IN-PRP (i.e. preposition and personal pronoun), while the best placed POS trigram was TO-DT-NN (i.e. to, determiner, and noun). The CP unigram PP (i.e. prepositional phrase) was ranked 5th. Prepositional phrases act as adjectives or adverbs, as seen in the example review sentence "It takes several (up to 10) seconds to switch *between accounts or screens*".

VI. Discussion

Our findings stress the importance of user rationale and its relevance to software engineering. We found that approximately 39% of studied reviews contain justifications. User rationale concepts tend to co-occur in reviews with a notable fraction of justifications ranging from 21 to 71%. An illustration of a justified review sentence is "Delivered fast, but was unable to import from TAXACT to this software, so I returned it". The sentence reports a decision (relinquish software) and an argument (compatibility issue).

Both the manual and automated labeling of rationale in review have challenges and limitations. Since manual labeling is laborious, the classifiers can assist in reducing the workload. In case when false negatives are worse than false positives, high-recall classifiers might be used in mining user rationale to minimize the chance of missing relevant information. In contrary, high-precision classifiers might be used to collect any representative (i.e. true positive) relevant information. We think that our dataset and results are useful for researchers to study more fine-grained rationale concepts (e.g. performance criteria), their inter-dependencies, and to design and evaluate rationale analytics and summarization tools. We see our classification results as a first step towards such a tool. Since we studied whole reviews and single sentences, user rational concepts can be highlighted or searched for within the reviews.

We discuss the importance of user rationale for software engineering and the potentials of its mining and management in two scenarios: deliberation support for users and synthesis of reviews for developers and analysts.

A. Deliberation support for users

Large corpora of user comments, particularly user reviews on platforms such as the Amazon's software store, Google Play, uservoice.com provide an important resource for software vendors to investigate polarization and the broad spectrum of rationale and perspectives in the public discussion. At the same time, it challenges the users to quickly survey the emerging and growing number of user feedback and to easily provide feedback or get involved into an existing discussion.

User rationale might be employed to support user involvement by structuring the discussion/debate in existing reviews. A simple statistics might provide an overview of reported issues, criteria, alternatives, decisions, or users' justifications. The justification density can significantly indicate the usefulness of a review [4], [28]. User rationale might help users learn about the software and understand its complexity and the tradeoffs users might not have thought about. This might improve the application rating. Users might also get support to identify if a review should e.g. , include a criteria or justification and recommendations to extend the text (e.g. by auto-complete functionality). User rationale can also be useful for debates on user feedback platforms such as User Voice, for visualization of pro and contra user stances that highlight contrasting user perceptions on e.g. non-functional requirements (such as pro and contra usability stances). However, there are some general issues when involving users and the processing of their feedback. Users are not professionals, and so the usefulness of their reviews can strongly vary. They also do not always adhere to grammatical rules when they write, and can be biased in their review. This challenges the automated processing and making sense of their reviews.

B. Synthesis of software reviews for SE practitioners

User rationale can support developers and analysts in filtering reviews, make better decisions (e.g. during requirements prioritization), documentation, and communication.

User rationale can be used to mark and filter potentially low/highly informative reviews. Those for instance that have a low justification density might be filtered. Furthermore, rationale-backed reviews of special interest might be selected and explored, e.g. those that mention conclusive decisions on switching a software. This allows stakeholders to take best possible actions tailored to the reported justifications of users.

During requirements prioritization and negotiation, for example, an issue can be higher prioritized, when it is frequently mentioned as a reason for abandoning the software. A between-app analysis might support practitioners in finding out how their tool performs compared to others (e.g. "How does my application compare to the most mentioned software alternative?" and "What are the most important software features or issues provided by other apps that users are missing".) Alternatives give insight on how software features are used together and which compatibility requirements should be considered. Identification of duplicates or clustering and quantifying these concepts might support stakeholders in deciding on tradeoffs (e.g. alternative features). Furthermore, justifications of users can enrich existing documentation of requirements and design decisions. The broad spectrum of their justifications can also help to get insights and better understand their different perspectives (e.g. on usability). Our approach can also be applied to other types of inputs (e.g. social media comments).

Finally, user rationale classifier can improve communication among stakeholders. The arguments provided by users when negatively assessing the technical support might be reused when advocating for more resources. In the long term, we

aim to study user arguments and their pro/contra orientation towards reference topics.

VII. Threats to validity

As for every grounded approach and manual content analysis, the validity of our study might have been affected by the human assessment during coding. This is partly indicated by the moderate inter-coder agreement. Therefore and due to potential sampling biases we can not claim a completeness of the final codes. Instead, we aimed for a plausible theory that is empirically grounded.

We took several measures to mitigate these threats to validity. First, we created a detailed coding guide that describes the main coding tasks, the process, and definitions for all codes of the user rationale concepts, including the free option "other" for identifying new codes. The guide can be downloaded from the project website. Second, we included candidate and non-candidate examples for each code. Third, we refined the guide in eight iterations, with hired coders to reduce the volunteer bias. The human coders that were involved in the coding were trained prior to doing the coding. Fourth, we conducted peer-coding to increase their reliability. The final truth sets contained only codings on which at least two coders agreed.

We aimed to study the feasibility of identifying user rationale as well as the most informative features instead of obtaining complete results. Thus, we refrain from claiming the completeness of our classification results. We focused on rather simple machine learning features and were still able to achieve accurate results. This facilitates the applicability and reproducibility of our approach. We did not evaluate all possible feature combinations. Different parameterization of the used classifier algorithms (e.g, optimization parameter for the Support Vector Machine), as well as the application of statistical feature selection methods, might have lead to different results. Feature selection methods use scoring functions based on the statistical tests, such as Chi-squared [8]. They try to reduce the feature space by removing redundant and irrelevant features, while trying not to lose much information [9]. These methods can be used to improve performance on large high-dimensional datasets. The results might also be influenced by the ratio of candidate and non-candidate items used in the training and testing set. We chose the ratio after an initial investigation of the learning curves of the classifiers. To obtain more reliable results we employed a large number of experiments with cross-validation.

We can reasonably assume that only a subset of software users will make the effort of writing a review, including potentially fake reviews. This might limit the external validity of the results. However, our study was not designed to be generalizable nor representative to other software markets such as App Store. We see the results primary as indicative.

Despite that we crawled only a small fraction of applications compared to the overall more than 300,000 applications available on Amazon, we think that our results have a moderate degree of generalizability for popular applications on Amazon. First, our dataset included reviews from popular applications across all Amazon software categories and ratings. Second, we conducted statistical tests to check for the significance of the results excluding hazard influence.

VIII. Related Work

We discuss related work from three areas: rationale management, rationale mining, and argumentation mining.

Researchers have studied **rationale management** in software and requirements engineering for decades [3], [6], [10]. Lee [12] discusses design rationale from the human-computer-interaction perspective. The author identifies seven technical and business issues for design rationale systems such as cost-effective use, domain-knowledge generation, and integration. Bruegge and Dutoit [2] and Dutoit et al. [6] discuss design and requirements rationale and describe the activities of creating, maintaining, and accessing rationale models, and the issues on managing rationale with a focus on decision support and negotiation. Burge et al. [3] describe the capture and use of design rationale and how rationale can be used for decision making throughout the software life cycle. Dutoit et al. [6] overviews rationale management approaches and the current state-of-the-art. They conclude that formal, model-based rationale is hard to capture and maintain, which also motivates our goal to mine rationale concepts from the reviews.

Recently, researchers and tool vendors started suggesting approaches to automatically analyze, filter, and synthesize software artifacts into actionable advices for developers. They also started to propose approaches to automatically **mine rationale information** from existing documentation [25]. Liang et al. [13] propose an approach based on an issue, solution, and artifact layer-based design rationale modeling. Their approach focuses on automatically learning design rationale from patent documents. Rogers et al. [25] studied text mining and parsing techniques for identifying rationale from existing documents. Their initial results suggest, that the use of linguistic features such as modal auxiliaries improve classifier precision but significantly lower recall compared to text mining. In another study, Rogers et al. [26] focused on mining rationale from Chrome Bug Reports. We had a different perspective focusing on user reviews that can contain any type of information. Furthermore, we developed the rationale concepts based on a button up grounded approach and evidence in data.

Finally, **argumentation mining** from natural language text is a relatively new research area with the potential of many interesting applications [24] such as mining arguments in legal, political, or journalistic text. Argumentation mining aims to automatically identify argumentative structures within a document, e.g. the premises, conclusion, and whole arguments, as well as the relationships between arguments [14], [23]. A major challenge in this field is to define what counts as argumentation and what does not with regard to empirical data.

Palau et al. [23] proposed an approach to automatically detect premises and conclusions in legal texts. They employ n-grams, keywords, as well as linguistic features and metadata and report an F1-Score of around 70% for recognizing premises and conclusions. Wyner et al. [29] studied a corpus of

comments in an Internet forum about purchasing a camera, and various dialogical activities to examine, e.g. persuasion, negotiation, deliberation, and others. Boltui and Najder [1] studied reasoning in online discussions, focussing on identifying properties of comment-argument pairs. They employed different features such as entailment features, semantic text similarity features, and stance alignment features. Using support vector machine classifier they achieved F1-Scores between 70 and 80%. Works from argumentation mining (e.g. [21]) inspired ours to experiment with syntactical features for an improved classification of user justifications. While these works have a strong emphasis of conclusiveness and validation of argumentation, we rather focus on arguments diversity reported in user reviews as well as the argument contexts. Our work targets software engineering with a potential impact for stakeholders as users, developers, or analysts.

IX. Conclusion

In this paper we introduce a novel grounded theory of user rationale in software engineering. While design rationale focuses on design related decisions and the reasons why those decisions were made, the concepts encompassed by user rationale were developed around the justifications of users - on issues they encounter, alternatives, criteria of assessment, and their decisions. We qualitatively describe how and in which context users report rationale in software reviews and quantify how pervasive rationale concepts such as alternatives and criteria are found in the reviews. We found that issues, alternatives, criteria, and decisions tend to co-occur in reviews with a notable fraction of justifications ranging from 21 to 70% and observed that higher co-occurrence indicate a higher justification density. We also found that justifications appear rather in longer review sentences compared to other concepts.

In a series of experiments, we discuss how user rationale concepts can be automatically predicted with supervised machine learning techniques, using text, meta data, sentiments, and syntactic features. We evaluated three classifier algorithms (i.e. Naive Bayes, Support Vector Machine, and Logistic Regression) and different configurations to predict user rationale in reviews and sentences. The precision and recall for all five user rationale concepts were encouraging – ranging from reviews and sentences respectively between 80%-99% and 69%-98%. Our approach helps a) structure software users' debates and b) synthesize and filter rationale-backed reviews for certain stakeholders, as developers, analysts, or other users.

Acknowledgment

The authors thank the coders, particularly A. Alizadeh, J. Hennings, E. Kurtanović, D. Martens, and M. Ziaei for their help with the coding. This work is partly funded by the H2020 EU research project OPENREQ (ID 732463).

References

[1] F. Boltužić and J. Šnajder. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*, Baltimore, Maryland, 2014. Association for Computational Linguistics.

[2] B. Bruegge and A. A. Dutoit. *Object-Oriented Software Engineering; Conquering Complex and Changing Systems*. Prentice Hall, 1999.

[3] J. E. Burge, J. M. Carroll, R. McCall, and I. Mistrk. *Rationale-Based Software Engineering*. Springer, 2008.

[4] E. B. Charrada. Which one to read? factors influencing the usefulness of online reviews for re. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 46–52, Sept 2016.

[5] J. Cohen. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213, 1968.

[6] A. H. Dutoit, R. McCall, I. Mistrik, and B. Paech. *Rationale Management in Software Engineering*. Springer, Berlin, Heidelberg, 2006.

[7] A. H. Dutoit and B. Paech. Rationale-based use case specification. *Requirements Engineering*, 7(1):3–19, 2002.

[8] P. E. Greenwood and M. S. Nikulin. *A guide to chi-squared testing*, volume 280. John Wiley & Sons, 1996.

[9] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3, 2003.

[10] A. Jarczyk, P. Loffler, and F. Shipman. Design rationale for software engineering: a survey. In *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*.

[11] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, 1995.

[12] J. Lee. Design Rationale Systems: Understanding the Issues. *IEEE Expert: Intelligent Systems and Their Applications*, 12(3), May 1997.

[13] Y. Liang, Y. Liu, C. K. Kwong, and W. B. Lee. Learning the whys: Discovering design rationale using text miningan algorithm perspective. *Computer-Aided Design*, 44(10):916–930, 2012.

[14] M. Lippi and P. Torroni. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology*, 2016.

[15] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik. On the automatic classification of app reviews. *Requirements Engineering*, 2016.

[16] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe. Toward data-driven requirements engineering. *IEEE Software*, 33(1):48–54, 2016.

[17] W. Maalej and M. P. Robillard. Patterns of knowledge in api reference documentation. *IEEE Transactions on Software Engineering*, 39(9):1264–1282, 2013.

[18] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[19] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993.

[20] K. A. Neuendorf. *The Content Analysis Guidebook*. Sage Publications, Inc, Thousand Oaks, CA, 1st edition, 2001. Published: Paperback.

[21] H. V. Nguyen and D. J. Litman. Extracting argument and domain words for identifying argument components in texts. In *Proceedings of the Second Workshop on Argumentation Mining*.

[22] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. In *RE*, pages 125–134. IEEE Computer Society, 2013.

[23] R. M. Palau and M.-F. Moens. Argumentation Mining: The Detection, Classification and Structure of Arguments in Text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, ICAIL '09, pages 98–107, New York, NY, USA, 2009. ACM.

[24] A. Peldszus and M. Stede. From Argument Diagrams to Argumentation Mining in Texts: A Survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31, 2013.

[25] B. Rogers, J. Gung, Y. Qiao, and J. Burge. Exploring techniques for rationale extraction from existing documents. In *Software Engineering (ICSE), 2012 34th International Conference on*, June 2012.

[26] B. Rogers, Y. Qiao, J. Gung, T. Mathur, and J. E. Burge. Using Text Mining Techniques to Extract Rationale from Existing Documentation. In S. J. Gero and S. Hanna, editors, *Design Computing and Cognition '14*, pages 457–474. Springer, 2015.

[27] A. Strauss and J. Corbin. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE, 1998.

[28] L. M. Willemsen, P. C. Neijens, F. Bronner, and J. A. De Ridder. Highly recommended! the content characteristics and perceived usefulness of online consumer reviews. *Journal of Computer-Mediated Communication*, 17(1):19–38, 2011.

[29] A. Wyner, J. Schneider, K. Atkinson, and T. J. M. Bench-Capon. Semi-automated argumentative analysis of online product reviews. In B. Verheij, S. Szeider, and S. Woltran, editors, *COMMA*, volume 245 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2012.