Rationale in Development Chat Messages: An Exploratory Study

Rana Alkadhi*, Teodora Laţa*, Emitza Guzman† and Bernd Bruegge*

*Faculty of Informatics, Technische Universität München
Garching, Germany
{alkadhi, teodora, bruegge}@in.tum.de
†Department of Informatics, University of Zurich
Switzerland
guzman@ifi.uzh.ch

***Abstract*—Chat messages of development teams play an increasingly significant role in software development, having replaced emails in some cases. Chat messages contain information about discussed issues, considered alternatives and argumentation leading to the decisions made during software development. These elements, defined as *rationale*, are invaluable during software evolution for documenting and reusing development knowledge. Rationale is also essential for coping with changes and for effective maintenance of the software system. However, exploiting the rationale hidden in the chat messages is challenging due to the high volume of unstructured messages covering a wide range of topics. This work presents the results of an exploratory study examining the frequency of rationale in chat messages, the completeness of the available rationale and the potential of automatic techniques for rationale extraction. For this purpose, we apply content analysis and machine learning techniques on more than 8,700 chat messages from three software development projects. Our results show that chat messages are a rich source of rationale and that machine learning is a promising technique for detecting rationale and identifying different rationale elements.**

I. Introduction

Development teams make various decisions throughout the software lifecycle. They discuss current issues, propose alternative solutions, argue for and against these alternatives and collaboratively make decisions. All elements justifying the made decisions constitute *rationale*. The presence of rationale is invaluable during software evolution. For example, the availability of rationale improves the traceability, documentation and understandability of the system [16]. Moreover, documented and accessible rationale is essential for effective maintenance and for analyzing the impact of changes [3], [7]. However, different methods for explicit rationale capture, e.g., involving designers and developers in writing up their rationale in a formal schema, have been met with resistance. Possible explanations for their reluctance are the intrusiveness of these methods and time constraints [16], [17]. Nevertheless, although rationale is rarely captured in a structured, explicit form, it is embedded in different development artifacts and communication channels [5], [36]. To capture rationale from these artifacts, sources of rationale need to be identified and techniques for extracting rationale need to be developed [34].

One potential medium for rationale are chat messages exchanged between members of development teams. Chat messages are an integral part of the daily activities in software development [28]. Team members discuss a wide range of topics in their chat messages including social, managerial, organizational and development issues. Some of this information could be an invaluable source of rationale. However, to the best of our knowledge, the extent of rationale in chat messages from development teams has not yet been explored. This work is a first effort in this direction.

We report on an exploratory study to better understand the rationale present in chat messages, in regards to its *frequency*, *completeness* and the potential of machine learning techniques for *automatically extracting rationale* elements on two granularity levels. For the purpose of our study we manually analyzed 8,702 chat messages of three development teams using content analysis techniques [33] and used the manually labeled data to train and evaluate different machine learning techniques.

Our results provide quantitative evidence that chat messages from development teams are a valuable source for rationale. However, due to the high volume of chat messages and sparsity of the messages containing rationale, the manual extraction and classification of rationale is a tedious and time-consuming process. Our results show that machine learning techniques are promising for detecting rationale in chat messages with a recall ranging from 0.61 up to 0.88 and for filtering messages without rationale with a recall of up to 0.98.

The contribution of the study is threefold. First, we investigated the frequency and completeness of rationale present in chat messages by using content analysis techniques and descriptive statistics. Second, we studied the performance of machine learning techniques for detecting rationale on the manually analyzed data. Third, we analyzed the potential of machine learning techniques for classifying rationale into different elements, e.g., issues, alternatives and decisions.

These contributions shed light on the rationale knowledge contained in software developers' chat messages and provide insights that could aid future research in exploiting this tacit knowledge.

978-1-5386-1544-7/17 $31.00 © 2017 IEEE
DOI 10.1109/MSR.2017.43

TABLE I: Definitions of rationale elements (adapted from [3]).

Rationale element	Definition
Issue	Problem that needs discussion and negotiation to be solved. An issue typically can not be resolved algorithmically and does not have a single correct solution.
Alternative	Possible solution that could address the issue under consideration.
Pro-argument	Positive reason supporting an alternative.
Con-argument	Negative reason against an alternative.
Decision	The alternative selected to resolve an open issue.

II. Rationale Representation Models

Rationale captures the reasons behind decisions. It includes the justification behind decisions, other alternatives considered and the argumentation that led to the decision [26].

Since Kunz and Rittel [24] proposed to capture rationale as an issue model, many models have been proposed, such as IBIS (Issue Based Information System) [24], QOC (Question, Option and Criteria) [30], PHI (Procedural Hierarchy of Issues) [31] and DRL (Decision Representation Language) [25]. The rationale elements we analyze throughout this paper are based on IBIS for its conciseness and as it provides the basis for most of the subsequent issue models including DRL and QOC. Namely, we focus our analysis on five elements: issues, alternatives, pro-arguments, con-arguments and decisions. Table I lists the rationale elements used in this work and their definitions. The rationale elements definitions were adapted from Bruegge and Dutoit [3]. These rationale elements are basic elements shared between most of the rationale representation models.

III. Research Setting

In this section, we introduce our research questions, describe the followed research method and the data we used to perform our analysis.

A. Research Questions

The aim of our study is to evaluate chat messages of development teams as a potential source for rationale. For this purpose, we explored the rationale frequency in chat messages, the rationale completeness and the rationale automatic extraction potential from chat messages.

RQ1: Rationale frequency describes how often rationale appears in chat messages. This information gives a first insight of the worth of considering chat messages as a source of rationale. In particular, we answer the following question:

- What is the frequency of rationale in chat messages?

RQ2: Rationale completeness describes the quality of the recovered rationale from chat messages. Rationale is distributed across different development artifacts, e.g., bug reports [36] and design session transcripts [21], and all of these sources could be used together to capture a more complete rationale of the software system. *Rationale completeness* occurs when for each documented decision, all the rationale elements justifying the decision are documented [8]. Answering this question could help practitioners and researchers by providing insights on how to integrate the rationale extracted from chat messages with the rationale extracted from other sources. In particular, we answer the following question:

TABLE II: Overview of chat messages dataset.

Team	Chat messages before filtering	Chat messages after filtering
Team A	4,106	3,974
Team B	2,214	2,164
Team C	3,026	2,564
Total	**9,346**	**8,702**

- How complete are the rationale elements extracted from chat messages?

RQ3: Rationale automatic extraction describes the potential of applying automatic techniques to detect and extract rationale from chat messages. This information provides insights on the feasibility of applying techniques previously used for retrieving important information from other development artifacts to extract rationale from chat messages. In particular, we answer the following questions:

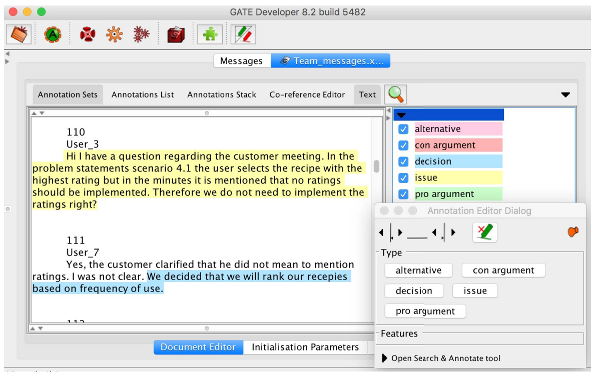
- *Binary classification:* Can rationale be accurately detected in chat messages by applying supervised machine learning techniques?
- *Fine-grained classification:* Can chat messages containing rationale be accurately classified into the different rationale elements: issues, alternatives, pro-arguments, con-arguments and decisions by applying supervised machine learning techniques?

B. Research Method

We employed content analysis techniques [33] to study the **rationale frequency** in chat messages and the **rationale completeness** of the existing rationale. Our analysis consisted of two consecutive steps. First, we identified the rationale elements present in the chat messages. Second, we evaluated the completeness of the identified rationale by linking semantically related rationale elements. Finally, to study the **rationale automation extraction** we applied machine learning techniques on the manually analyzed data and evaluated the classification performance of these techniques according to well established metrics. We explain each procedure in further detail in Sections IV, V, VI.

C. Research Data

We analyzed the chat messages of three development teams that were part of a multi-project capstone course at the Technical University of Munich in 2015 and 2016 [4], [22]. The course was designed to simulate industry settings [23].



110
User_3
Hi I have a question regarding the customer meeting. In the problem statements scenario 4.1 the user selects the recipe with the highest rating but in the minutes it is mentioned that no ratings should be implemented. Therefore we do not need to implement the ratings right?

111
User_7
Yes, the customer clarified that he did not mean to mention ratings. I was not clear. We decided that we will rank our recepies based on frequency of use.

Fig. 1: A screenshot of using GATE for the manual coding of chat messages.

During the course, teams developed mobile applications for industrial partners and dealt with incomplete and evolving requirements following an agile software methodology [23]. The participants used Atlassian HipChat for instant messaging[1], which supports diverse integrations to external services and bots, e.g., with Bamboo[2] to send notifications about build results and Standup Bot[3] to report and retrieve statuses for stand-up. Each team consisted of 8 to 9 students and a project leader for project management. When selecting the teams for the study, we considered only teams who communicated in English and wrote more than 2,000 messages. To focus our analysis on the messages written by members of development teams, messages that were automatically generated by one of the services or bots were filtered out. Table II shows the three selected teams and the number of chat messages before and after filtering automatically generated messages.

IV. Rationale Frequency

To explore the frequency of rationale in chat messages, we analyzed how often rationale appears in chat messages and the frequency of different rationale elements.

A. Procedure

We manually analyzed the chat messages in our dataset by applying content analysis techniques [33]. Two of the authors of this paper independently inspected the chat messages of the three teams in our dataset and identified the contained rationale. This process comprised three steps:

1) Developing the coding guide: The aim of this step was to systematize and minimize disagreements between the two coders. Since many representations have been proposed in literature to model rationale, it is important that the two coders share a unified understanding of the elements that constitute rationale. To this end, we developed a coding guide that includes clear definitions of the rationale elements and examples for each element[4]. Table I lists the rationale elements

[1]https://www.hipchat.com
[2]https://www.atlassian.com/software/bamboo
[3]http://botlab.hipch.at
[4]Available on: https://goo.gl/PAKLQU

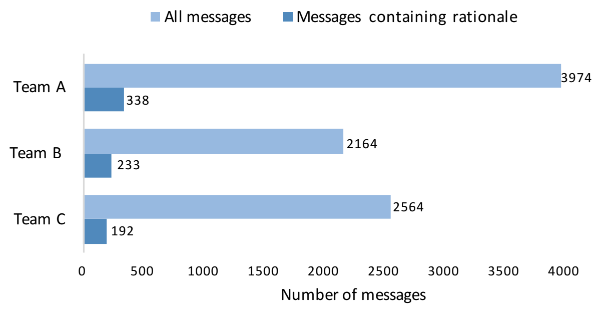


Fig. 2: Chat messages containing rationale per team.

used in this paper and their definitions. The coding guide was developed in an iterative process consisting of two trial iterations. In each iteration, the two coders independently identified rationale elements discussed in 1,000 chat messages. The coders disagreements were analyzed and the coding guide was modified to minimize disagreements in the next iterations.

2) Coding of chat messages: For the manual coding task, we used GATE (General Architecture for Text Engineering) [12], [13], a Java-based framework for a diverse set of natural language processing applications.

Figure 1 shows a screenshot of GATE as used by coders. The main window displays the list of chat messages to be coded. If a message contains rationale, the coder highlights the message part containing the rationale and specifies its type.

The coding unit, i.e., the highlighted part, can be one sentence, multiple sentences of a message or the complete message. We refer to the coded units as *text snippets*. For each snippet, coders specified whether it contains rationale and what type of rationale elements are present. A text snippet might contain multiple rationale elements. Coding text snippets allows for capturing the text containing rationale in the finest-grained manner since a message might contain additional irrelevant information.

The two coders independently coded each message in our dataset. The average time to code 8,702 messages was 13 hours per coder, highlighting the large effort required to manually extract rationale elements.

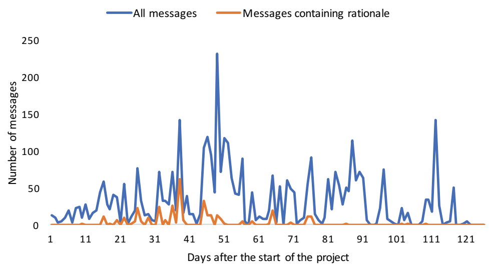
3) Disagreement reconciliation: Disagreements between the two coders included situations when the two coders identified different rationale elements in the same text-snippet or when only one coder coded a text-snippet as containing rationale. The disagreements where resolved through discussions between the two coders.

B. Results

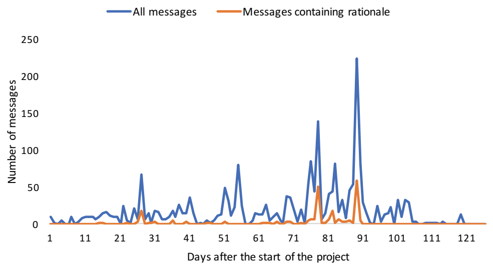
Figure 2 shows the numbers of chat messages identified as containing rationale by the coders. On average 9% of the team chat messages contain rationale. Although the number of chat messages containing rationale is not high, development teams discuss various elements of rationale in these messages that comprises valuable knowledge about the software system. Table III shows an overview of the

TABLE III: Frequency of rationale elements across messages containing rationale per team.

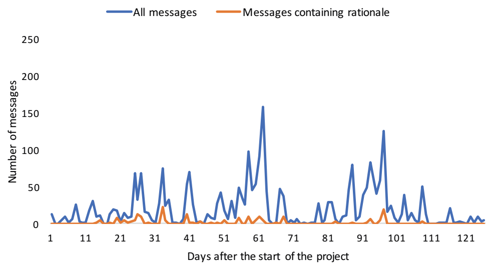
	Frequency				
Rationale element	**Team A**	**Team B**	**Team C**	**Total**	**Example**
Issue	25%	28%	17%	24%	*“Plus if this is implemented using segueways, what screen do you go back to when you click ’back’?”*
Alternative	45%	54%	57%	51%	*“What do u think of having a "start cooking" button? Clicking on the recipe name might not be the most intuitive? Thoughts?”*
Pro-argument	17%	26%	30%	23%	*“It’s better UX :) definitely”*
Con-argument	18%	17%	19%	18%	*“But still, I think it is too complicated for now to build it with tabs.”*
Decision	13%	7%	9%	10%	*“We decided that we will rank our recepies based on frequency of use.”*



(a) Team A



(b) Team B



(c) Team C

Fig. 3: Distribution of all messages and messages containing rationale over the duration of the project.

frequency of different rationale elements across messages and examples of coded rationale elements. Overall, we found that proposing alternatives to different issues is predominant in almost 51% of the messages containing rationale. The second most frequent rationale element is issue, present in 24% of the messages containing rationale. Pro-arguments were mentioned in 23% and con-argument in 18% of the messages containing rationale. These numbers confirm our observation (performed during the manual coding) that team members tend to argue for the alternative they proposed and their reasons to select the alternative (pro-arguments) more frequently than arguing against other alternatives (con-arguments). Lastly, decisions were identified in 10% of the messages containing rationale.

To explore how the passage of time and the number of messages influence the amount of rationale found in chat messages, we investigated the distribution of all chat messages and messages containing rationale over the duration of the project. Figure 3 visualizes the distribution of all messages and messages containing rationale over the project duration per team.

We found that different development teams had different rationale distribution in their chat messages. On the one hand, Teams A and B had a significant increase in the number of messages containing rationale at certain time points. One possible interpretation of this result is that discussions between developers might be more dense around particular milestones during the project, e.g., at the beginning of development sprints or before releasing to the customer. On the other hand, Team C had a steady distribution of chat messages containing rationale that spans the entire project duration. The Spearman’s correlation coefficients between the number of days passed since the start of the project and the number of messages containing rationale were 0.19, 0.11 and 0.25 for teams A, B and C respectively indicating a very weak correlation.

As shown in Figure 3, the increase in the number of chat messages is not always an indicator of an increase in the number of messages containing rationale. However, the Spearman’s correlation coefficient between the total number of chat messages and the number of messages containing rationale was 0.5 in all three teams indicating a moderate positive correlation.

V. Rationale Completeness

To explore the completeness of rationale in chat messages, we analyzed how many rationale elements were semantically related to other rationale elements also present in the chat messages.

A. Procedure

Rationale elements identified in different chat messages could be semantically related, as the rationale discussion could span multiple messages. For example, a developer might discuss a specific issue in a single message and other developers propose different alternatives to resolve the issue and argue for and against these alternatives in multiple short messages. Thus, it is important to identify the semantic relationships among these rationale elements in order to explore the completeness of the existing rationale. To achieve this, the same two independent coders who conducted the manual content analysis of the chat messages (detailed in Section IV) manually inspected the 752 messages containing rationale and identified the semantically related rationale elements contained in these messages. The disagreements were resolved through discussions between the two coders. After identifying the semantically related rationale elements, we answered the following questions:

- For each discussed issue, were alternative solutions proposed and was a decision made?
- For each selected alternative (made decision), were pro-arguments supporting its selection presented?

B. Results

Our results show that for 79% of the issues identified in the chat messages, alternatives were proposed in the chat messages to resolve the issues. In 48% of these issues, we were able to identify the decisions made, i.e., the selected alternatives, in the chat messages. However, in only 48% of the cases in which a decision was made, the pro-arguments in support of the selected alternative were present in the chat messages. Finally, we found that in 21% of the issues found in the chat messages, neither alternatives were suggested nor a decision has been made. One possible interpretation is that the team members discussed and resolved the open issues through face-to-face communication. For example, while discussing an issue one developer wrote, *"Probably best if we discuss this in the meeting"*. These results support our hypothesis that chat messages are not to be used as the only source for rationale but rather as one of many potential sources. The rationale extracted from chat messages could be integrated with rationale extracted from other sources for a more complete rationale, e.g., the decisions made to resolve some of the open issues extracted from chat messages might be available in the meeting minutes of the development team.

VI. Rationale Automatic Extraction

To explore the potential of automatic techniques for rationale extraction from chat messages, we performed two experiments. Each experiment classifies rationale, but at different levels of granularity. In the first experiment, we built a *binary classifier* that detects chat messages containing rationale and filters out messages without rationale. In the second experiment, we built a *fine-grained classifier* that classifies the messages containing rationale into the different rationale elements: issues, alternatives, pro-arguments, con-arguments and decisions (defined in Table I).

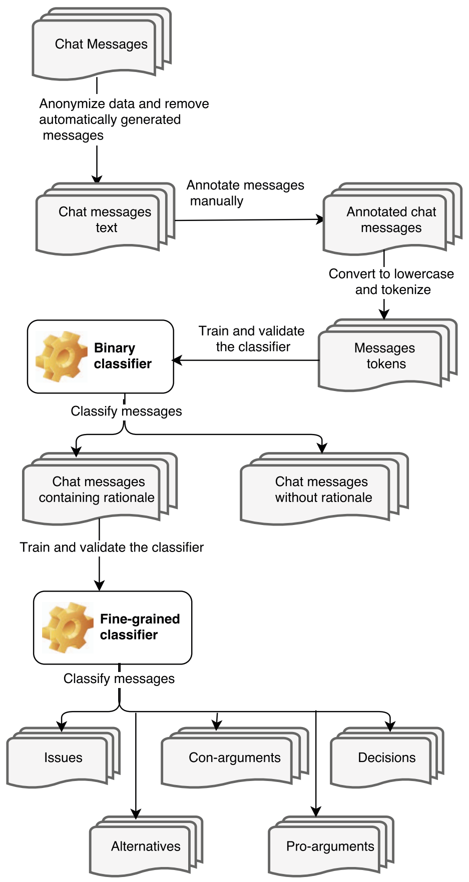

Chat Messages
Anonymize data and remove automatically generated messages
Chat messages text
Annotate messages manually
Annotated chat messages
Convert to lowercase and tokenize
Messages tokens
Train and validate the classifier
Binary classifier
Classify messages
Chat messages containing rationale
Chat messages without rationale
Train and validate the classifier
Fine-grained classifier
Classify messages
Issues
Con-arguments
Decisions
Alternatives
Pro-arguments

Fig. 4: Experiment setup.

For training and validating our classifiers, we used the manually annotated chat messages described in Section IV. Figure 4 shows the experiment setup. We detail each step in the following sections and describe the results.

A. Binary Classification

In the first experiment, we explored the potential of the automatic detection of chat messages containing rationale. We applied machine learning techniques to classify the 8,702 chat messages in our dataset into two classes: messages containing rationale and messages without rationale. Considering the large number of chat messages without rationale identified during the manual analysis, the binary classification could be used as a preceding step for a finer-grained classification to filter out message without rationale.

1) Experiment Setup: An important domain in machine learning is document classification, in which documents are assigned to one or more classes. Classifying chat messages into two classes, as in our experiment, is known as binary

TABLE IV: Binary classification results.

Validation technique	Balancing technique		Naive Bayes Multinomial			SVM		
			Precision	Recall	F1	Precision	Recall	F1
10-fold cross validation		Messages containing rationale	0.52	**0.62**	**0.57**	**0.60**	0.37	0.46
		Messages without rationale	**0.96**	0.95	0.95	0.94	**0.98**	**0.96**
	Under-sampling	Messages containing rationale	0.78	**0.87**	**0.82**	**0.83**	0.71	0.77
		Messages without rationale	**0.85**	0.75	0.80	0.75	**0.85**	**0.80**
	SMOTE + Under-sampling	Messages containing rationale	0.80	**0.88**	0.84	**0.85**	0.85	**0.85**
		Messages without rationale	**0.87**	0.78	0.82	0.85	**0.85**	**0.85**
Team cross validation		Messages containing rationale	0.43	**0.61**	**0.50**	**0.48**	0.28	0.35
		Messages without rationale	**0.96**	0.92	0.94	0.93	**0.97**	**0.95**

classification. We compared the performance of two learning algorithms, Naive Bayes Multinomial and Support Vector Machine (SVM) due to their popularity and good performance for text classification [11], [18], [39].

a) Preprocessing: Before training the classifiers, we preprocessed the message text by converting it into tokens and lowercase. Tokenization converts a stream of characters into a sequence of tokens. We used n-gram tokenizer with 1 and 3 as the minimum and maximum length. By applying an n-gram tokenizer we expected patterns of terms appearing together to be indicators of rationale presence in the chat messages, e.g., phrases like *"I would suggest"*, *"how about"* could be indicators of proposed alternatives and *"how do we"* could be an indicator of issues. We chose not to apply stopword removal (i.e., removing non-informative, common words) as we expected them to be representative of some rationale elements. For example, *which* and *how* might be indicators of issues, e.g., *"Which design pattern should we apply?"*, and *but* is commonly used before stating con-arguments against alternatives, e.g., *"but it sucks as UX"*.

b) Classification level: We performed our classification on the message level. The consideration of the classification on the message level was motivated by two factors. First, previous work on classifying development artifacts [1], [36] found that considering a sentence's neighbors (context) improved classification performance. Second, chat messages are of short length, which makes classification on the message level feasible.

c) Data balancing: Imbalanced datasets are emerging as an important issue in many machine learning applications [10]. Our dataset is imbalanced with an average of 9% messages containing rationale. Building a classifier from an imbalanced dataset can cause the classifier to be biased towards the majority class, i.e., the class with the greater number of instances, while ignoring the minority class [20]. Two popular techniques for handling class imbalance problem are under-sampling and SMOTE (Synthetic Minority Over-sampling Technique). Under-sampling [15] uses a subset of the majority class for training the classifier while SMOTE [9] applies oversampling on the minority class by generating synthetic examples. We compared between the application under-sampling and a combination of SMOTE and under-sampling as previous research has proved that classifiers achieve better performance when combining both sampling techniques [9].

d) Training and evaluation: Since we are applying supervised machine learning algorithms, we need to train our classifiers on the manually annotated chat messages (detailed in Section IV). For training and evaluating the classifiers, we applied 10-fold cross validation and team cross validation. In 10-fold cross validation, 9 folds are used for training the classifier and the remaining fold for validating its performance. The process is repeated 10 times rotating the training and testing folds. The evaluation is computed by calculating the average results among the 10 runs. We also applied team cross validation to better test the generalizability of the results, since different development teams tend to use different terminologies in their chat messages. In team cross validation, we trained the classifiers on two teams and predicted the classification of the remaining team. We repeated the process three times, i.e., a 3-fold cross validation, each time with a different team as a test set and we measured the average results.

We evaluated the classification accuracy using the standard metrics in machine learning: precision, recall and F-Measure (F1). They are calculated as follows:

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \quad Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{1}$$

Where TP_i is the number of messages that are correctly classified as being of type i, FP_i is the number of messages that are incorrectly classified as being of type i and FN_i is the number of messages that are incorrectly classified as not being of type i. The F-Measure is the harmonic mean of the precision and recall.

2) Experiment Results: Table IV gives an overview of the binary classification results. The numbers in bold represent the corresponding top values.

When applying 10-fold cross validation, both Naive Bayes Multinomial and SVM classifiers performed very well in classifying chat messages without rationale, achieving high values for both precision and recall (above 0.94). However, they performed less well when classifying messages containing rationale; where SVM had a better precision of 0.60 (compared to 0.52 from Naive Bayes Multinomial), while Naive Bayes Multinomial achieved a much higher recall of 0.62 (compared to 0.37 from SVM). One possible explanation of achieving less accuracy in classifying messages containing rationale is the sparsity of the messages containing rationale in the

TABLE V: Binary classification results examples.

Message	**Manual classification**	**Automatic classification**
"Well, to be honest you have 3 options. 1) Add new methods to persistency to deal with this new type of data type = redundant code, as it will be deleted. 2) Modify the load all like you mentioned. 3) Just not use persistency for now and, yea, notifications will appear again if the app is closed, but you can test the rest of the functionality."	Contain rationale	Contain rationale
"You can send it to me on HipChat :)"	No rationale	No rationale
"found two very very nice tutorials we could use for our graphical stuff (different charts, like pie chart, bar chart, history eg.) and one for scanning qr codes. (Y) i think they look very nice and clean... links"	No rationale	Contain rationale
"Do we need to support the iPhone4?"	Contain rationale	No rationale

results of our manual content analysis (Figure 2). Table V shows examples of binary classification results. Upon further inspection, we found that rationale discussions spanning over multiple messages were a common source of error. In these cases, it is important to consider the contextual information in the neighbor messages to identify the rationale contained in the message.

In the context of extracting rationale from chat messages (rationale was present in only 9% of the chat messages in our study), we argue that recall is more important than precision. We aim at recovering as much rationale from chat messages as possible, with the compromise of falsely predicting messages as containing rationale. Therefore, we can say that Naive Bayes Multinomial outperformed SVM in classifying messages containing rationale. In summary, both classifiers reported significantly better performance in classifying chat messages without rationale, with the highest achieved precision of 0.96 and recall of 0.98. We believe that a binary classifier could be applied as a preceding step for the fine-grained classifier to filter out messages without rationale, as shown in Figure 4.

Applying balancing techniques resulted in a significant increase in the classification performance of messages containing rationale. However, as expected, the performance of classifying the majority class, i.e., messages without rationale in our case, decreased. Applying SMOTE in combination with under-sampling achieved higher precision (0.87) and recall (0.88) for classifying messages containing rationale than applying under-sampling alone. These results suggest applying data balancing techniques on the training set to alleviate the classifier bias towards the majority class as a result of imbalanced training data. Consequently, increasing the amount of recovered rationale from chat messages.

When applying team cross validation, the overall classification performance of messages containing rationale decreased. While the Naive Bayes Multinomial classifier achieved an almost equal recall of 0.61 in both validation methods, there were slight decreases in the other accuracy measures. The results of classifying messages without rationale are comparable in both validation methods. These findings demonstrate that our results have a high degree of generalizability as we tested the classifier on unseen chat messages of a team different from the teams used for training the classifier and repeated this process three times while rotating the three development teams.

B. Fine-grained Classification

In the second experiment, we explored the performance of a fine-grained classifier that further classifies the 752 messages containing rationale into five different rationale elements: issue, alternative, pro-argument, con-argument, and decision (defined in Table I). The distribution of rationale elements among the messages containing rationale is shown in Table III.

1) Experiment Setup: When classifying messages containing rationale into different rationale elements, a chat message might contain more than one element. For example, a developer might propose an alternative and write the pro-argument supporting the alternative in the same message. In machine learning, classifying documents into one or more classes that are not mutually exclusive is referred to as multilabel classification. We applied two of the most popular techniques for multilabel classification, the binary relevance (BR) and the label powerset (LP) [42]. In the binary relevance method, a binary classifier is independently trained for each class and the final prediction for a message is determined by aggregating the classification results from all independent classifiers. The main drawback of the binary relevance method is the class independence assumption. The label powerset method (LP) takes into account the class correlation by considering each label combination as a single class.

We applied the same preprocessing steps on the message text as used on the binary classification. However, we did not apply data balancing techniques because the distribution of rationale elements among the messages containing rationale is more balanced than the data in the binary classification experiment.

For training and validating the classifiers, we applied both 10-fold cross validation and team cross validation used for the binary classification. In addition, we evaluated the classifiers performance using the same metrics as in the binary classification.

2) Experiment Results: Table VI summarizes the obtained classification results. The numbers in bold represent the corresponding top values.

As expected, the fine-grained classifier – classifying into five different rationale elements – performed less well than

TABLE VI: Fine-grained classification results.

		Binary Relevance						**Label Powerset**					
		Naive Bayes Multinomial			**SVM**			**Naive Bayes Multinomial**			**SVM**		
		Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
10-fold cross validation	Issue	0.43	**0.51**	0.47	0.51	0.47	0.49	0.43	**0.51**	0.47	**0.53**	0.47	**0.50**
	Alternative	**0.67**	0.67	**0.67**	0.65	0.58	0.61	0.66	**0.68**	**0.67**	0.62	**0.68**	0.65
	Pro-argument	0.43	**0.53**	**0.47**	0.43	0.38	0.40	**0.45**	0.37	0.40	0.41	0.32	0.36
	Con-argument	0.37	**0.44**	**0.40**	0.37	0.31	0.34	**0.39**	0.16	0.23	0.35	0.23	0.28
	Decision	0.26	**0.29**	**0.27**	**0.29**	0.21	0.24	0.23	0.12	0.15	0.21	0.14	0.17
Team cross validation	Issue	0.41	0.45	0.42	0.50	0.41	0.43	0.41	**0.50**	**0.45**	**0.51**	0.39	0.43
	Alternative	**0.70**	0.54	0.61	0.65	0.57	0.60	0.65	0.69	**0.66**	0.60	**0.70**	0.64
	Pro-argument	0.44	**0.49**	**0.45**	0.42	0.35	0.37	0.44	0.37	0.38	**0.46**	0.29	0.35
	Con-argument	0.28	**0.32**	0.30	**0.38**	0.28	**0.32**	0.29	0.12	0.17	0.35	0.19	0.24
	Decision	0.22	**0.32**	**0.25**	0.25	0.15	0.16	**0.26**	0.14	0.18	0.22	0.16	0.15

TABLE VII: Fine-grained classification results examples.

Message	**Manual classification**	**Automatic classification**
"@User_3 can we expand on how the app handles lack of internet in the beginning? is there any issue there?"	Issue	Issue
"Mornin guys. Can someone tell me, what steptype (cooking, measuring, mixing, chopping) preheating the oven is?:/I would make another category - preheating otherwise i would take cooking as the type. What do you guys suggest?"	Issue, alternative	Issue, alternative
"but it will look like our design and it is easier to implement than our first idea, as we will have one ingredient per line"	Pro-argument	Pro-argument
"Also it confuses me when the recipe says:"Warm up 50g of sugar and 400ml of cream in a pot."I guess there is one step needed for measuring the stuff and then for cooking them. I have to write in both steps the amount and the ingredients etc. right? That's how i have it in mind."	Issue, alternative	Issue, decision
"There should be only one Navigation Controller and the Rest are Views."	Decision	Alternative

the binary classifier. When applying 10-fold cross validation, no single classifier works best for all rationale elements. Nevertheless, the results obtained by applying the binary relevance method achieved higher accuracy than when applying the label powerset method in most of the cases. For predicting the different rationale elements, it might be more important to have higher recall values than those of precision to identify as many rationale elements contained in the messages as possible. When applying the binary relevance method, Naive Bayes Multinomial outperformed SVM with a higher recall for all rationale elements. We achieved the highest recall for predicting alternatives (0.67). Pro-arguments followed with a recall of 0.53, issues with a recall of 0.51 and con-arguments with a recall of 0.44. The accuracy of predicting decisions was the lowest with a recall of 0.29.

Table VII shows examples of fine-grained classification results. A possible interpretation of the obtained results is the sparseness of some rationale elements in the messages containing rationale (Table III). Chat messages are informal short messages and the rationale elements discussed in these messages are unstructured and intertwined. Distinguishing between different elements is a nontrivial and intensive task even for a human judgment. Upon further inspection of the results, we found that a possible interpretation of the poor accuracy in classifying decisions compared to other elements, is that it is not always obvious in the messages whether a decision has been made. And in many cases, the decisions were classified as alternatives by the classifier.

When applying team cross validation, the overall classification performance of rationale elements decreased. However, there were slight increases (0.03) in the recall of classifying alternatives and decisions. These results support the generalizability of our results.

We replicated the above described experiments on the sentence level. In both binary and fine-grained classification, the classification on the message level performed significantly better than the classification on the sentence level. Due to the space limitations, sentence level results are not reported.

VII. Discussion

In current agile software development methodologies, the development process is more dynamic and a working software is often given more importance over comprehensive documentation [2]. As a result, informal communication channels, such as chat messages, are seeing wide and rapid adoption by software development teams. These chat messages archive communication between developers and contain valuable information about the rationale behind made decisions. This study is a first step towards using chat messages as a source of rationale for the software system.

The results of our study show: (1) chat messages are a valuable source for rationale during software development, (2) chat messages should be used in combination with other development artifacts for capturing a complete rationale, (3) automated filtering of messages without rationale is possible with a high accuracy, and (4) classifying messages containing rationale into different rationale elements is a promising research direction. In the following we revisit our research questions and discuss possible future work directions.

Rationale frequency: Although a small percentage of chat messages contain rationale (9%), the manual content analysis results show that these messages contain valuable knowledge about the software system. In chat messages, team members actively engage in discussing issues, proposing alternatives, arguing for and against these alternatives and collaboratively making decisions. However, while the informality and unstructured nature of chat messages helps developers reveal their thoughts naturally, this poses a number of challenges for extracting rationale from chat messages. First, rationale could span multiple messages complicating its identification. Second, multiple elements of rationale could be discussed in a single message, and distinguishing between the different elements is a nontrivial task even for a human judgment.

Rationale completeness: In almost half of the identified issues (48%), chat messages contained a complete rationale of the alternatives considered, the selected alternative (decision) and the arguments supporting the made decision. However, for the remaining issues, the identified rationale from chat messages was incomplete. A possible explanation is that developers use different communication channels in addition to chat messages. For example, developers might continue some of the chat messages discussions in face-to-face meetings and decisions made to resolve the issues identified in chat messages might be documented in other development artifacts such as meeting minutes. This finding emphasizes the importance of linking related rationale elements extracted from different development artifacts for a more complete capturing of rationale. Future research needs to investigate methods and develop tools that systematically extract and aggregate rationale from its identified sources.

Rationale automatic extraction: Although the manual analysis of chat messages was a useful technique for our research, with the increasing number of chat messages and high volume of messages without rationale, manual analysis becomes infeasible. Therefore, we investigated the use of automated approaches for extracting rationale from chat messages. With the aim of recovering as much rationale from chat messages as possible, our results for detecting rationale in chat messages are encouraging, with a recall ranging from 0.61 to 0.88. The filtering of messages without rationale was possible with high precision and recall (both above 0.90). However, the results for classifying messages containing rationale into finer-grained rationale elements were less promising. Future work should focus on improving the classification performance by using a more balanced training set with a larger number of manually identified rationale elements. Considering additional features of the messages' text such as using linguistic features and taking into account the message context, i.e., neighbor messages, could improve the classification performance.

Future work: Our final goal is to extract rationale from different development artifacts (sources), link related rationale elements and structure rationale in a unified representation. Externalizing rationale knowledge can help in achieving a better understanding of the software system and thus supporting future changes. For example, recording explored alternatives could reveal that some proposed changes are inappropriate. Another possible use of rationale is supporting maintenance and design verifications by spotting conceptual and implementation errors [16]. Even though a completely automated approach for accurately extracting well-structured rationale is still partially unrealized, our preliminary results of the automatic detection and extraction of rationale from development chat messages are an encouraging step in this direction. Future work could investigate methods for identifying semantic links between messages and structuring rationale from the classified messages into a formal representation, i.e., the formalization of rationale from its capture [16].

VIII. Threats To Validity

During the manual analysis, the determination if a message contains rationale and what type of rationale elements are present is a subjective decision. To mitigate this risk, we created a coding guide with precise definitions and examples for different rationale elements. The guide was used by the coders during the coding task. Furthermore, each message in our dataset was coded by two people and the disagreements were discussed and resolved by the two coders.

Although the list of rationale elements used in our analysis are based on the well-known IBIS model [24], the list of elements could be incomplete and its descriptions simplified. This threat could lead to the capture of incomplete rationale. However, the used rationale elements are shared among most rationale representation models. Additionally, due to the exploratory nature of our study further extensions and replications are needed.

We believe that our results have a considerable level of generalizability in regards to other small, agile software projects. The students in the three development teams worked closely with industrial customers and dealt with incomplete and evolving requirements on innovative projects. Previous research found that subject's experience level might have more effect on the results than the experiment setting (academic or industry) [37]. In our study, the majority of the student participants described themselves as semi-professional developers and they reported having part-time jobs in the industry. Additionally, we conducted team cross validation to test the generalizability of our results regarding the automatic extraction of rationale. We encourage further replication of our study to examine if the results reported in this study also hold for different software development settings.

IX. Related Work

We focus the related work discussion in two areas: automatic extraction of rationale and mining developers' communication artifacts.

A. Automatic Extraction of Rationale

One method to overcome the rationale capturing *rationale reconstruction* [6]. In this method, the rationale is retrospectively created from development artifacts. To our best knowledge, this is the first work to explore the potential of automatic techniques to extract rationale from development team chat messages. However, there are several studies on automatic extraction of rationale from other development artifacts. Lopéz et al. [29] proposed an ontology-driven approach to extract knowledge units relevant for architecture rationale from plain-text documents. They argue rationale recovery from existing documents can be decomposed into three smaller problems: automatic extraction of rationale from these documents, formalization of the extracted rationale and manipulation of the formalized information for further reuse. Liang et al. [27] proposed an algorithm for discovering design rationale from patent documents. They captured rationale in a three layer model consisting of issues, design solutions and artifacts layers. Myers et al. [32] designed Rationale Construction Framework (RCF) that recorded designers interactions in a CAD tool and produced a rich history for detailed design.

Similar to our work, Rogers et al. [34], [36] applied machine learning techniques to extract rationale from bug reports. In their work, they investigated the use of ontology and linguistic features for training machine learning models to classify sentences containing rationale into decisions, alternatives and argumentation. In their recent work, Rogers [35] proposed a system that uses genetic algorithms to evaluate candidate feature sets for identifying rationale in two types of documents, bug reports and design sessions transcripts. Our work differs from theirs in that we focus on extracting rationale from developers' chat messages. This poses different challenges – as chat messages are short, informal and less structured than the previously analyzed documents.

B. Mining Developers' Communication Artifacts

Previous research found that development communication artifacts are a rich source for valuable information about the software system, its history and rationale [43], [44], as "developers reveal their thought processes most naturally when communicating with other software developers" [38].

Lin et al. [28] conducted an exploratory study to understand why developers use Slack, a team messaging platform, and how they benefit from it. Their analysis revealed that most developers use Slack for team-wide purposes including communication and collaboration with other team members. The findings of their study motivates the work presented in this paper.

Previous research has investigated mining Internet Relay Chat (IRC) logs from Open Source Software (OSS) projects. Shihab et al. [40], [41] analyzed the usage of developer IRC meetings channels by project developers and maintainers. In their work, they mined IRC meeting logs to investigate the meeting content, meeting participants, their contribution and communication styles. Chowdhury and Hindle [11] implemented machine learning approaches to filter out off-topic discussions in programming IRC channels by exploiting StackOverflow programming discussions and YouTube video comments. Yu et al. [45] investigated the use of synchronous (IRC) and asynchronous (mailing list) communication mechanisms in global software development projects. They observed that developers actively use both as complementary communication mechanisms. Our work differs from the previous work in that we focus our analysis of chat messages on capturing a specific type of knowledge about the software system, namely, rationale. Furthermore, we analyze chat messages in development settings more similar to commercial than open source software development.

Another stream of research focused on the automated processing of other communication artifacts. Guzman and Bruegge [19] described an approach to summarize collaboration artifacts, such as emails and wikis. Brunet et al. [5] used machine learning techniques for classifying design discussions in commits, issues and pull requests in open source projects. Similarly, Bacchelli et al. [1] and Di Sorbo et al. [14] proposed approaches to classify the content of developments' emails into different categories. Our work builds on these studies and highlights the need for automated techniques to support extraction and aggregation of rationale from developers' chat messages.

X. Conclusion

The rationale of the software system is a result of collaboration and negotiation between different stakeholders, as a result developers' discussions in chat messages are a valuable source for information about rationale.

In this paper, we report on an exploratory study that investigated rationale frequency, completeness and the automatic extraction of rationale in two granularity levels. We found that developers discuss various elements of rationale in these messages, which comprise valuable knowledge about the software system. However, due to the high volume of chat messages, automated approaches to extract rationale hidden in these chat messages are needed. The results of our experiments show that applying machine learning techniques can filter out messages without rationale with a precision and recall above 0.90 and that detecting messages containing rationale is possible with a recall ranging from 0.61 to 0.88. Furthermore, fine-grained classification of the detected rationale is a promising research direction. This work is a first step towards a more effective exploitation of rationale in chat messages.

Acknowledgments

We thank Jan Ole Johanßen and Nitesh Narayan for their valuable feedback. This work was partially supported by a PhD scholarship provided by King Saud University for Alkadhi.

References

[1] A. Bacchelli, T. Dal Sasso, M. D'Ambros, and M. Lanza. Content Classification of Development Emails. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE'12, pages 375–385, 2012.

[2] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. Manifesto for Agile Software Development, 2001.

[3] B. Bruegge and A. H. Dutoit. *Object-Oriented Software Engineering Using UML, Patterns, and Java*. Prentice Hall Press, 3rd edition, 2009.

[4] B. Bruegge, S. Krusche, and M. Wagner. Teaching Tornado: From Communication Models to Releases. In *Proceedings of the 8th Edition of the Educators' Symposium*, EduSymp '12, pages 5–12, 2012.

[5] J. Brunet, G. C. Murphy, R. Terra, J. Figueiredo, and D. Serey. Do Developers Discuss Design? In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR'14, pages 340–343, 2014.

[6] J. E. Burge. *Software Engineering Using RATionale*. Ph.D. thesis, Worcester Polytechnic Institute, 2005.

[7] J. E. Burge and D. C. Brown. Software Engineering Using RATionale. *Journal of Systems and Software*, 81(3):395–413, 2008.

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

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16(1):321–357, 2002.

[10] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: Special Issue on Learning from Imbalanced Data Sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6, 2004.

[11] S. A. Chowdhury and A. Hindle. Mining StackOverflow to Filter out Off-topic IRC Discussion. In *Proceedings of the 12th International Working Conference on Mining Software Repositories*, MSR'15, pages 422–425, 2015.

[12] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters. *Text Processing with GATE (Version 6)*. 2011.

[13] H. Cunningham, V. Tablan, A. Roberts, and K. Bontcheva. Getting More Out of Biomedical Documents with GATE's Full Lifecycle Open Source Text Analytics. *PLoS Comput Biol*, 9(2):e1002854, 2013.

[14] A. Di Sorbo, S. Panichella, C. A. Visaggio, M. Di Penta, G. Canfora, and H. Gall. DECA: Development Emails Content Analyzer. In *Proceedings of the 38th International Conference on Software Engineering Companion*, ICSE '16, pages 641–644, 2016.

[15] C. Drummond, R. C. Holte, et al. C4. 5, Class Imbalance, and Cost Sensitivity: Why Under-sampling Beats Over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, 2003.

[16] A. H. Dutoit, R. McCall, I. Mistrik, and B. Paech. *Rationale Management in Software Engineering*. Springer-Verlag, 2006.

[17] G. Fischer, A. C. Lemke, R. McCall, and A. I. Morch. Making Argumentation Serve Design. *Human-Computer Interaction*, 6(3):393–419, 1991.

[18] E. Frank and R. R. Bouckaert. Naive Bayes for Text Classification with Unbalanced Classes. In *Proceedings of the 10th European Conference on Principle and Practice of Knowledge Discovery in Databases*, PKDD'06, pages 503–510, 2006.

[19] E. Guzman and B. Bruegge. Towards emotional awareness in software development teams. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE'13, pages 671–674, 2013.

[20] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[21] T.-M. Hesse and B. Paech. Documenting Relations Between Requirements and Design Decisions: A Case Study on Design Session Transcripts. In *Proceedings of the 22nd International Working Conference on Requirements Engineering: Foundation for Software Quality - Volume 9619*, REFSQ'16, pages 188–204, 2016.

[22] S. Krusche and L. Alperowitz. Introduction of Continuous Delivery in Multi-customer Project Courses. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 335–343, 2014.

[23] S. Krusche, L. Alperowitz, B. Bruegge, and M. O. Wagner. Rugby: An Agile Process Model Based on Continuous Delivery. In *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*, RCoSE'14, pages 42–50, 2014.

[24] W. Kunz and H. Rittel. Issues as Elements of Information Systems. Working Paper 131, Institute of Urban and Regional Development, University of California, Berkeley, California, 1970.

[25] J. Lee. Artificial Intelligence at MIT Expanding Frontiers. chapter SIBYL: A Qualitative Decision Management System, pages 104–133. MIT Press, 1990.

[26] J. Lee. Design Rationale Systems: Understanding the Issues. *IEEE Expert*, 12(3):78–85, 1997.

[27] Y. Liang, Y. Liu, C. K. Kwong, and W. B. Lee. Learning the "Whys": Discovering Design Rationale Using Text Mining - An Algorithm Perspective. *Computer-Aided Design*, 44(10):916–930, 2012.

[28] B. Lin, A. Zagalsky, M.-A. Storey, and A. Serebrenik. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, CSCW Companion '16.

[29] C. López, V. Codocedo, H. Astudillo, and L. M. Cysneiros. Bridging the Gap Between Software Architecture Rationale Formalisms and Actual Architecture Documents: An Ontology-driven Approach. *Science of Computer Programming*, 77(1):66–80, 2012.

[30] A. MacLean, R. M. Young, V. M. E. Bellotti, and T. P. Moran. Questions, Options, and Criteria: Elements of Design Space Analysis. *Human-Computer Interaction*, 6(3):201–250, Sept. 1991.

[31] R. McCall. PHIBIS: Procedurally Hierarchical Issue-based Information Systems. In *Proceedings of the International Congress on Planning and Design Theory*, volume 44, 1987.

[32] K. L. Myers, N. B. Zumel, and P. Garcia. Automated Capture of Rationale for the Detailed Design Process. In *Proceedings of the Eleventh Conference on Innovative Applications of Artificial Intelligence*, AAAI'99, pages 876–883, 1999.

[33] K. A. Neuendorf. *The Content Analysis Guidebook*. Sage Publications, 2002.

[34] B. Rogers, J. Gung, Y. Qiao, and J. E. Burge. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE'12.

[35] B. Rogers, C. Justice, T. Mathur, and J. E. Burge. Generalizability of document features for identifying rationale. In *Proceedings of the 7th International Conference on Design Computing and Cognition*, DCC'16, pages 633–651, 2016.

[36] B. Rogers, Y. Qiao, J. Gung, T. Mathur, and J. E. Burge. Using Text Mining Techniques to Extract Rationale from Existing Documentation. In *Proceedings of the 6th International Conference on Design Computing and Cognition*, DCC'14, pages 457–474, 2014.

[37] I. Salman, A. T. Misirli, and N. Juristo. Are Students Representatives of Professionals in Software Engineering Experiments? In *Proceedings of the 37th International Conference on Software Engineering*, ICSE'15, pages 666–676, 2015.

[38] C. B. Seaman. Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering*, 25(4):557–572, 1999.

[39] F. Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[40] E. Shihab, Z. M. Jiang, and A. E. Hassan. On the Use of Internet Relay Chat (IRC) Meetings by Developers of the GNOME GTK+ Project. In *Proceedings of the 6th IEEE International Working Conference on Mining Software Repositories*, MSR'09, pages 107–110, 2009.

[41] E. Shihab, Z. M. Jiang, and A. E. Hassan. Studying The Use of Developer IRC Meetings in Open Source Projects. In *Proceedings of the IEEE International Conference on Software Maintenance*, ICSM'09, pages 147–156, 2009.

[42] G. Tsoumakas and I. Katakis. Multi-label Classification: An Overview. *International Journal of Data Warehousing and Mining*, 3:1–13, 2007.

[43] G. Venolia. Textual Allusions to Artifacts in Software-Related Repositories. In *Proceedings of the 2006 International Workshop on Mining Software Repositories*, MSR '06, pages 151–154, 2006.

[44] H. Wang, A. L. Johnson, and R. H. Bracewell. The Retrieval of Structured Design Rationale for the Re-use of Design Knowledge with an Integrated Representation. *Advanced Engineering Informatics*, 26(2):251–266, 2012.

[45] L. Yu, S. Ramaswamy, A. Mishra, and D. Mishra. Communications in Global Software Development: An Empirical Study Using GTK+ OSS Repository. In *Proceedings of the 2011th Confederated International Conference on On the Move to Meaningful Internet Systems*, OTM'11, pages 218–227, 2011.