Unfolding the Rationale for Code Commits

Khadijah Al Safwan

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

Francisco Servant, Chair
Eli Tilevich
Na Meng

May 9, 2018
Blacksburg, Virginia

Keywords: software engineering, revision control systems, rationale, empirical study

Unfolding the Rationale for Code Commits

Khadijah Al Safwan

ABSTRACT

One of the main reasons why developers investigate code history is to search for the rationale for code commits. Existing work found that developers report that rationale is one of the most important aspects to understand code changes and that it can be quite difficult to find. While this finding strongly points out the fact that understanding the rationale for code commits is a serious problem for software engineers, no current research efforts have pursued understanding in detail what specifically developers are searching for when they search for rationale. In other words, while the rationale for code commits is informally defined as, "Why was this code implemented this way?" this question could refer to aspects of the code as disparate as, "Why was it necessary to implement this code?"; "Why is this the way in which it was implemented?"; or "Why was the code implemented in that moment?" Our goal with this study is to improve our understanding of what software developers mean when they talk about the rationale for code commits, *i.e.*, how they "unfold" rationale. We additionally study which components of rationale developers find important, which ones they normally need to find, which ones they consider specifically difficult to find, and which ones they normally record in their own code commits. This new, detailed understanding of the components of the rationale for code commits may serve as inspiration for novel techniques to support developers in seeking and accurately recording rationale.

Unfolding the Rationale for Code Commits

Khadijah Al Safwan

General Audience Abstract

Modern software systems evolution is based on the contribution of a large number of developers. In version control systems, developers introduce packaged changes called code commits for various reasons. In this process of modifying the code, the software developers make some decisions. These decisions need to be understood by other software developers. The question "why the code is this way?" is used by software developers to ask for the rationale behind code changes. The question could refer to aspects of the code as disparate as, "Why was it necessary to implement this code?"; "Why is this the way in which it was implemented?"; or "Why was the code implemented at that moment?" Our goal with this study is to improve our understanding of what software developers mean when they talk about the rationale for code commits, *i.e.*, how they "unfold" rationale. We additionally study which components of rationale developers find important, which ones they normally need to find, which ones they consider specifically difficult to find, and which ones they normally record in their own code commits. This new, detailed understanding of the components of the rationale for code commits will allow researchers and tools builders to understand what the developers mean when they mention rationale. Therefore, assisting the development of tools and techniques to support the developers when seeking and recording rationale.

Acknowledgments

I would like to thank my advisor, Francisco Servant, who guided me through my Master's journey. Thank you for providing continuous support and feedback. I really appreciate your willingness to meet with me whenever I needed help.

Heartfelt thanks go to my husband, Mohammed Alaboalirat, for being a supportive person who helped and encouraged me from the start to the end. Thank you for being a better companion than I could ever wish for. I am also grateful for all your financial support. Thank you for making Blacksburg a home far from home.

I would like to thank my parents, Ahmad and Yusra Al Safwan, who planted the love of learning in my heart when I was a small child. Thank you for teaching me how to be a good student and for raising me with good manners and moral values. I am also deeply thankful for my siblings and my extended family for being THE family.

My sincere appreciation goes to the scholarship program of the Custodian of the Two Holy Mosques King Salman bin Abdulaziz and to the Saudi Arabian Cultural Mission (SACM) for awarding me a full scholarship for my degree. Thank you for supporting my studies to be a "qualified individual capable of achieving the country's goals of progress and development."

I am indebted to the professors who taught me and contributed to my education. Thanks to those of you who helped me enter Virginia Tech by writing recommendation letters: Dilek Dustegor, Naya Nagy, Karima Makhluf, and Imran Mahmood. I also extend my gratitude to those who taught me: Barbara Ryder, Eli Tilevich, Aditya Prakash, Francisco Servant, Rex Hartson, and Chris North.

Warm thanks go to my friends, lab members, and classmates for filling the journey with joy and memorable events. I am glad we crossed paths and met each other. Thank you for extending the list of countries of my friends from only Saudi Arabia to Saudi Arabia, Bahrain, Kuwait, Oman, Egypt, Palestine, India, Korea, China, Colombia, and the United States.

Last but not least, I would like to thank the participants in my research study for giving up some of their valuable time to participate in the interview or fill out the survey. Thank you for sharing your opinions, knowledge, and experience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Rationale in software engineering is regarded with a great deal of importance. Software development relies on the decision making of the stakeholders in every step of the development process. Rationale is a resource that supports the development process [1, 2]. One fundamental component of software development and evolution is software changes [3]. Modern software is modified in packaged changes called code commits in revision control systems. Searching for the rationale for code commits is one of the developers' main motivations for examining code history [4], which can be quite difficult to find [5, 6].

Codoban *et al.* [4] found that developers often examine old code history to "recover the rationale behind a snippet of code." Their survey results show that around 58% of the participants selected "Why is this 'this way'?" as a motivation for examining old history. Tao *et al.* [5] regarded the rationale for change as the most important part of change understanding. LaToza and Myers [6] found that rationale is the most frequently reported category of hard-to-answer questions about the code. Although these findings strongly point out the fact that understanding the rationale for code commits is a serious problem for software engineers, no current research efforts have pursued understanding in detail the specific information that developers are searching for when they search for rationale.

While the rationale for code commits is informally defined as, "Why was this code implemented this way?" this question could refer to aspects of the code as disparate as, "Why was it *necessary* to implement this code?"; "Why *is this the way* in which it was implemented?"; or "Why was the code implemented *at that moment*?"

We hypothesize that the rationale for code commits can be divided into components. Discovering the components of the rationale for code commits is the primary goal of our study. Understanding specifically what developers mean by rationale will have many benefits. First, it will allow researchers and practitioners to disambiguate conversations or information-seeking processes about code-change rationale. Additionally, it will enable the development of tools to support developers in finding, recording, and assessing code-change rationale. All these applications can potentially provide a strong improvement of development productivity because the rationale is important for developers, they search for it often, and it can be really hard to find. In addition to identifying the components of code-change rationale, we also investigate which components of rationale developers regard as important, which ones they normally need to find, which ones they consider specifically difficult to find, and which ones they normally record in their own code commits.

We used two research methods to perform our study. First, we used one-on-one interviews to identify the components of the rationale for code commits and understand them in depth, and then we performed a survey to validate the identified components from a larger number of practitioners. We combined the responses of our interview and survey participants to understand the components of the rationale for code commits, practitioners' habits about recording and finding the rationale for code commits, and the importance of and need for such components.

The interview participants built the model of the rationale for code commits and identified its components (Tables 3.1, 4.1, and 4.2). In combining the interview and survey participants' answers, we identified the components of the rationale for code commits most important to the developers. These components are goal, need, location, and modifications. We also found that developers need the rationale for code commits and spend a considerable amount of time searching for it. Finally, the interview and online survey results specified that software developers usually only record the components of the rationale for code commits that they consider the most important, and they rarely record the remaining ones.

Our newly discovered detailed understanding of the components of the rationale for code commits motivates creating tools and techniques to support developers in accurately recording and finding the rationale. Further research may include studies to discover developers' strategies for recording and finding the rationale. In addition, studies to validate the frequency of recording dierent components of rationale are encouraged.

Our goal with this study is to improve our understanding of what software developers mean when they talk about the rationale for code commits, i.e., how they "unfold" rationale. This master's thesis examines the following thesis statement:

"The rationale for code changes can be divided into components, each of which has different characteristics."

The main contributions of this study are as follows:

1. Unfolding the rationale for code commits into separate components.
2. Discovering the importance of every component of the rationale for code commits.
3. Identifying the components of the rationale for code commits that developers need.

4. Identifying the components of the rationale for code commits that are difficult to find.

5. Identifying the recorded or unrecorded components of the rationale for code commits.

The rest of this document is organized as follows. First, we present related work in Chapter 2. Chapter 3 discusses our study's methodology. In Chapters 4 and 5, we present our study's results and our discussion of the results. Finally, Chapter 6 states threats to validity, Chapter 7 proposes future work based on our results, and Chapter 8 concludes this thesis.

Chapter 2

Literature Review

We group the work related to our study into three parts: empirical studies of software history, rationale studies, and software change-understanding research.

2.1 Software History Studies

Researchers have performed several empirical studies involving software history and developers' information needs in the last decade [4, 7, 5, 8, 6, 9, 10]. In these studies, researchers explored the motivation for examining software history (evolved software), the methods and strategies used in the software-history investigation, the challenges developers face when studying evolved software, and the needs and questions developers have about software history.

Our work builds on these empirical studies, whose findings establish a strong demand for rationale. For our goal of defining and conceptualizing the rationale for code commits, we created an initial model of the rationale for code commits with reference to these empirical studies, where associations were made between questions/needs and rationale/reasoning.

Tao *et al.* [5] highlighted the question, "What is the rationale behind this code change?" [11] and found that rationale is the most important information need. Ko *et al.* [8] identified questions developers ask when reasoning about design: "What is the purpose of this code?"; "What is the program supposed to do?"; "Why was this code implemented this way?"; and "What are the implications of this change?" LaToza and Myers [6] identified the hard-to-answer rationale questions, "Why was it done this way?"; "Why wasn't it done this other way?"; and "Was this intentional, accidental, or a hack?" Finally, Fritz and Murphy [7] discovered the developers' question, "Why were [these changes] introduced?"

Other researchers studied the impact and risk of changes [12, 13, 14], which is one other need of developers. Understanding the impact and risk of a code change is the main reasoning component of changes, which is why we included it in our model of rationale.

2.2 Rationale Studies

Dutoit *et al.* [1] published a book that discusses "Rationale Management (RM) in Software Engineering." "The aim of [the] book is to encourage software engineers to explore different ways for RM in research and practice and help to make RM a well-recognized ingredient for successful software engineering." Similarly, Burge *et al.* 's [2] book "Rationale-Based Software Engineering" discusses the rationale for software development and the software engineering lifecycle, including requirements engineering, software design, software architecture, and bodies of knowledge.

These two books motivated us to study rationale at the code-commit level. Although software history studies, as presented in Section 2.1, regarded rationale with importance, a clear definition of software-change rationale is missing.

One recent empirical study [15] tried to capture how developers discuss rationale in Internet Relay Chat (IRC) channels of open-source software (OSS). They identified rationale in IRC by following an Issue-Based Information System (IBIS), analyzing rationale elements as issues, alternatives, pro-arguments, con-arguments, and decisions.

2.3 Change-Understanding Research

Software engineering research has proposed a number of techniques and tools to facilitate developers' change-understanding practice. Our work complements the existing research by empirically studying the rationale for code changes, identifying the components of the rationale and the developers' need for these components.

Researchers developed techniques to summarize and document code changes. Some focused on generating natural language descriptions of code changes to replace incomplete or inaccurate commit messages [11], clarify the rationale behind code changes [16, 17], or answer "what" and "why" questions about code changes [18, 19, 20].

Other researchers mined software repositories to characterize commits [21, 22, 23, 24]. The categories they used to classify commits are the same as the rationale components we found. An example is the use of the commit goal (e.g., fix, add, test, bug, patch) to classify commits.

Chapter 3

Methodology

To establish the methodology of our study, we employ successful methods from prior research in the field of software engineering [4, 5, 25, 26]. Our study consists of two main research methods: interviews followed by an online survey. We first interviewed software developers in one-to-one sessions to build the model of the rationale for code commits and identify its components. We also asked the interview participants about their experience with recording and finding rationale components and the importance of and need for such components. Then, we distributed an online survey to a larger number of developers, populating the answers and findings of the interview questions.

3.1 Research Questions

Our exploratory study of the rationale for code commits was designed to answer the following research questions:

RQ1: What are the **components** of the rationale for code commits?

RQ2: How **important** is each rationale component to the rationale for code commits?

RQ3: What is the developers' **need** for rationale and its components?

RQ4: What is the developers' experience with **finding** rationale and its components?

RQ5: What are the rationale components that are **recorded** or unrecorded by software developers?

3.2 Interviews

Interviews are the qualitative method that we used to elicit the developers' knowledge about the rationale for code commits. One-to-one interviews are the first and primary method of our study. We interviewed 20 practitioners, seeking information about the rationale for code commits and its components. Information about the interview preparations, structure, recruitment, and participants is presented in the following sections.

3.2.1 Preparations

We built an initial model of the rationale for code commits as preparation for the interviews. The model was created based on the research involving software history. When reviewing the software history studies, we recognized a lack of a consistent and clear definition of the rationale for code commits. We searched empirical studies [8, 6, 5, 7] for associations between questions/needs and rationale/reasoning. Such associations were created by researchers to categorize or label the questions and needs of software developers. Table 3.1 is our initial model of the rationale for code commits.

Table 3.1: Initial Model of the Rationale for Code Commits

Component	**Component Expressed as Question**	**Example Answer**
Goal	What did you want to achieve?	I want to modify our usage of try/catch blocks in a way that they all account for unexpected exceptions.
Need	Why did you need to achieve that?	A new company directive requires that all our try/catch block statements include the Exception case by June 1st.
Alternatives	What other alternatives did you have?	Spend the time to consider the best way to handle Exception for each instance of a try/catch block.
Selected Alternative	Why did you make those specific changes and not others?	A general exception message works for now. It was more efficient to apply the same implementation to all locations.
Location	What artifacts were changed?	I made changes in every catch/block that I could find in the core branch.
Modifications	What specific changes were performed in the artifacts?	I added an "Exception (e)" case to all of them and handled it by logging a general exception message.
Validation	How did those specific changes achieve the goal?	This partially achieved our goal, because we may be able to handle some of these exceptions in a more specific way.
Benefits	What is the benefit of what you want to achieve?	This will increase the quality of our system by now having decided how to handle exceptions that we did not think about before.
Costs	What risks could come from these changes?	Some test cases may now fail because of the new exception messages in the logs.

We then developed our interview script by performing five pilot interviews. After each pilot, we made changes to improve the script structure, clarity of questions, and supporting material. The pilots helped us identify the scope of the study, define the interview structure, phrase the interview questions, prepare clarification questions, and rehearse for the actual sessions.

3.2.2 Questions and Structure

The final interview questions are based on the research questions presented in Section 3.1. The parts of the interview script and their purpose are presented below in the order we asked them during the interview. The interview script of the interviewer and interviewee are available in Appendix A.

In the first part of our interview script, we ask the participant for examples of investigating code commits for rationale. Providing examples helps the participants engage in the discussion and interview. It also clarifies the scope of the study.

We asked the participants to think about rationale in a more general sense after providing the examples. This second part of the interview captured the participants' own conceptual (abstract-level) definition of the rationale and its components. To avoid bias, it was important to ask for the participants' general thoughts about rationale before introducing the initial model of rationale.

In the third part of the interview, we presented the initial model of the rationale for code commits to the participants. We requested that they review, understand, think about, and ask for clarifications of the model. It was important that the participants understand the model before asking for their modifications in the following part of the interview.

Part four of the interview is when we asked the participants for their modifications to the model. We requested that they share their ideas about the model of the rationale for code commits, delete components that should not be part of the rationale for code commits, and add missing components of the rationale for code commits. The participants were allowed to update the model to reflect their definition of the rationale for code commits.

The last part of the interview addressed the importance of and need for rationale and its components, as well as how rationale and its components are recorded and found, based on the participants' experience. We asked the participants to select on a scale of four or five Likert points the importance of rationale for work completion, rationale usual- and hard-case search time, the components' importance for rationale, and the rationale and the components' frequency of recording, frequency of need, maximum frequency of need, frequency of finding, and difficulty of finding.

3.2.3 Recruitment

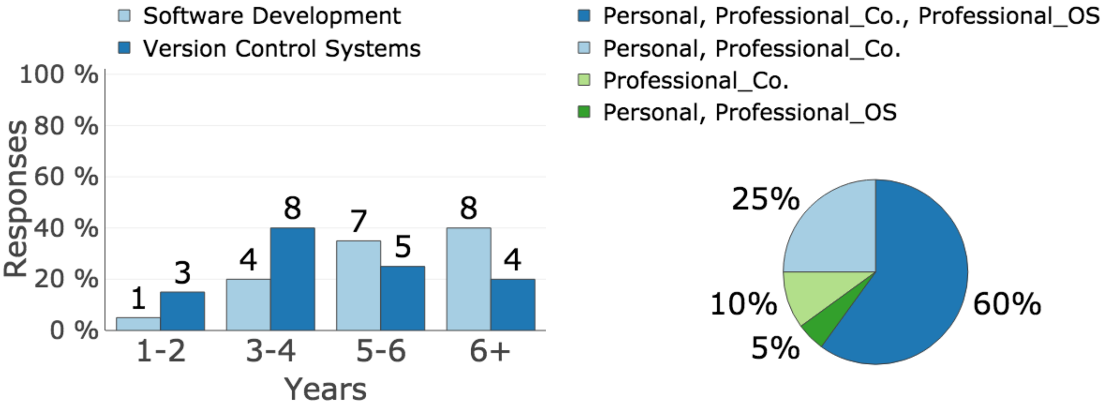
The recruitment of interview participants was done in two ways. We sent out an advertisement email about the study, and at the end of the study session, we asked every interview participant for referrals to people who might participate (snowball sampling). We asked for participants with revision control systems experience and experience investigating code changes to understand something about them.

All subjects who were interested in participating were asked to fill out a screening questionnaire. In the screening questionnaire, we asked demographic questions about age, gender, software development experience (number of years and type), revision control systems experience (number of years, type, and systems used), and the frequency of investigating code commits for rationale (own commits and other developers' commits). We interviewed everyone who filled out the screening questionnaire and replied to our scheduling emails.

3.2.4 Participants

We interviewed 28 developers in total. Each interview lasted between 60 and 120 minutes, and the participants were given $20 Amazon.com gift cards at the end of the interview as compensation for their time. Figure 3.1 shows our analysis of the participants' demographic information.

We analyzed the data of 20 interview sessions after excluding the five pilot interviews and three other interview sessions. In the first one, the participant did not complete the interview. The participant was unable to provide an example of a situation where he/she needed to understand the rationale for code commits. In the second excluded session, the participant voluntarily stated lack of experience and knowledge multiple times during the interview session. The last one was excluded after we noticed that the participant had knowledge about our initial model before the interview was performed. We excluded this person because he/she could have been biased toward supporting our initial model.



(a) Years of Experience (b) Type of Experience

Figure 3.1: Interview Participants' Demographics

3.3 Survey

We used an online survey as our second method of data collection. While we used our interviews to understand (in depth) the components into which developers break down rationale, we used our surveys to reach a larger number of practitioners to get more information about those components. We collected the responses of 24 survey participants about recording and finding rationale components, and the importance of and need for such components.

3.3.1 Preparations

We designed the online survey with 16 questions to quantify our findings from the interviews. All the survey questions were multiple choice. The questions were the same as the demographic questions from the screening questionnaire (see Section 3.2.3) and the last part of the interview (see Section 3.2.2). The exact survey questions are available in Appendix ??. We asked the survey participants about the updated model of rationale that we obtained from the interview participants. The final model of the rationale for code commits presented to the survey participants is shown in Table 4.1.

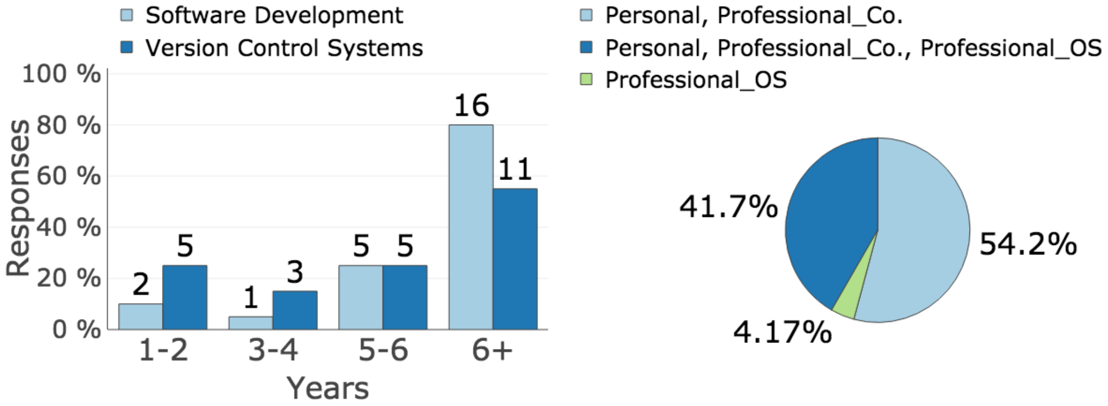
We ran a pilot of the survey by asking four participants with various levels of familiarity and experience with version control systems to fill out the survey. We asked one of the pilot participants to complete the survey carelessly and quickly while still reading the questions. The goal was to test the minimum time needed to fill out the survey. Data from any surveys completed in less than that amount of time was discarded in the analysis of the survey data. The rest of the pilot participants were asked to give feedback on the timing, the clarity of the rationale for code commits model, and the clarity of the questions.

3.3.2 Recruitment

To recruit participants for the online survey, invitations were sent through public channels. The interview participants were sent the advertisement email to forward it to people who might participate but not to fill it out themselves. To maximize participation, we offered a raffle of $50 Amazon.com gift cards to survey participants.

3.3.3 Participants

We received a total of 35 complete survey responses and 55 incomplete responses. We excluded 11 of the complete responses from our analyses. Two of the pilot surveys were discarded, as their goal was to set a minimum bar for survey duration to accept the responses. This minimum bar of 10 minutes was used to discard another two of the completed surveys. An additional five of the completed surveys were excluded because the participants did not consent. Finally, we excluded two responses where the participants specified that they have a greater number of years of experience in version control systems than software development. Figure 3.2 shows the demographic information of the 24 analyzed survey participants' data.



(a) Years of Experience

(b) Type of Experience

Figure 3.2: Online Survey Participants' Demographics

Chapter 4

Results

In this chapter, we present the results of the interviews and online survey, which answer our research questions and address our thesis statement. We give an overview of the results, including the final model of the components of the rationale for code commits and the characteristics of these components. Afterward, we provide a detailed report on the modifications and improvements to the initial model of the rationale for code commits, the components of rationale developers consider important, the components they normally need to find, the components they consider specifically difficult to find, and the components they normally record in their own code commits.

4.1 Overview

The model of the rationale for code commits that we discovered in this study is presented in Table 4.1. We started with the model in Table 3.1, and Table 4.2 shows the components proposed by the interview participants. The final updated model's components are described by a question and an example answer to the question. The example answers are possible answers to the expressive question of each component. All the example answers are related to a simple hypothetical commit. The questions and answers for the components added to the model by participants are included as provided by the participants. Most of the final model's components are a merge of proposed components that refer to the same matter. Some of the updated model's components are an abstraction of the specific cases that the participants mentioned.

The components of the rationale for code commits can be categorized into four themes. First, the change-objective theme includes the goal, need, and benefits components. The second theme is change-design (pre-implementation assessment), including the constraints, alternatives, selected alternative, and dependency components. The third is the change-execution theme, including the committer, time, location, modifications, and explanation

Table 4.1: Updated Model of the Rationale for Code Commits

Component	**Component Expressed as Question**	**Example Answer**
Goal	What did you want to achieve?	I wanted to implement functionality to sort the product list by price.
Need	Why did you need to achieve that?	Our user requested to be able to sort the list of products by price.
Benefits	What is the benefit of what you want to achieve?	The new option of sorting products by price will be useful for many customers in addition to the one who requested it.
Constraints	What were the constraints limiting your implementation choice?	The sorting algorithm had to be space efficient because it should work in embedded devices.
Alternatives	What other alternatives did you have?	I could have used the bucket sort algorithm, but this option was not feasible because I would not have known the maximum price before sorting.
Selected Alternative	Why did you make those specific changes and not others?	I implemented heap sort because it is space efficient and it has a predictable speed.
Dependency	What other changes does this change depend on?	This change depends on the API that provides the product list to be updated to use JSON format.
Committer	Who changed the code?	Developer X, who is responsible for the “products” page.
Time	Why were the changes made at that time?	This change happened before our 3.0 release to meet the customer contract for that release.
Location	What artifacts were changed?	The “product” class was updated.
Modifications	What specific changes were performed in the artifacts?	I added a “sort” method in the “product” class implementing heap sort and now the “listProduct” method calls “sort” first.
Explanation of Modifications	What are the details of the implementation?	The code sorts the products by price by performing the following steps: 1- Build a heap from a list of “products” in O(n) operations. 2- Swap the first list-element with the final list-element of the list. 3- Decrease the considered range of the list by one. 4- Shift the new first element to its appropriate index in the heap based on the “price. ” 5- Repeat step (2) unless the considered range of the list is one element.
Validation	How did those specific changes achieve the goal?	By using the heap sort algorithm, our customers can now see a sorted product list in their memory-limited hardware.
Maturity Stage	How mature is this code?	The change is an initial implementation, which still has to be fully tested after the API for the products list is updated.
Side Effects	What are the side effects of the change?	The integration test will fail if the API that provides the product list is not updated. At the same time, merging this change with the main branch after updating the API might break the existing code. Also, our implementation of heap sort may be too complex for beginners and may slow down maintenance.

of modifications components. The final theme is change-evaluation (post-implementation assessment), including validation, maturity stage, and side effects.

Figure 4.1 shows an overview of the results for the characteristics of the rationale for code commits. For every component of the rationale for code commits, we show five boxes indicating the component's importance, frequency of recording, frequency of need, frequency of finding, and difficulty of finding. The boxes show statistics about the participants' responses covering the minimum, maximum, median, and outliers. The raw data is available in Appendix C.

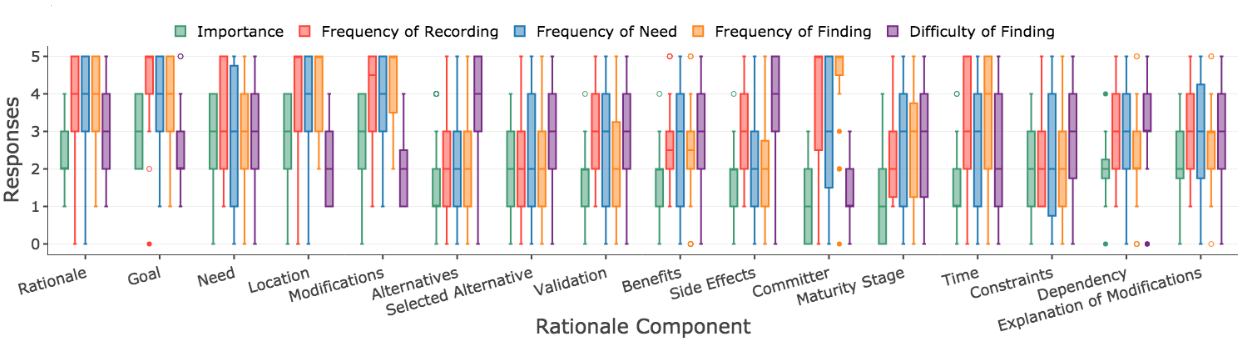


Figure 4.1: Rationale Components' Importance, Frequency of Recording, Frequency of Need, Frequency of Finding, and Difficulty of Finding

4.2 Components of the Rationale for Code Commits

The interview participants were asked to list and describe the components of the rationale for code commits (**RQ1**). After that, we showed them our initial model (Table 3.1) of components of the rationale and asked them for their opinion. Most participants shared similar thoughts about the initial model being detailed and comprehensive. Then, we asked participants if they wanted to change their answer after having seen our model. We specifically told them that they should feel free to remove, add, or modify components from their model. Most participants used our model as a reference (since most felt that it was more comprehensive than theirs) and adapted it by removing or adding components, as they felt necessary.

The participants provided positive comments about the initial model of the rationale for code commits. They indicated that the model is "a good model," "detailed," "thorough," "comprehensive," "holistic," and "exhaustive." They also thought it "formally define[s] rationale" and is "a logical frame that would work." One participant said that the model "clarifies the wide open concept which is hard to think about."

When we asked the participants about components that should not be part of the rationale for code commits, they revisited the model and spent some time to think. Below, we explain some examples of changes that participants proposed. One participant thought that the goal and the need can be the same most of the time and preferred to merge them together, deleting the "goal" component. Another participant thought the need is included in the benefits and cost, deleting the need component. One participant deleted the benefits component because it is included in the goal component. Four of the participants disagreed with the location component being part of the rationale for code commits. One of them considered the location to be part of the modification component. Another one of the four stated the reason for deletion as, "The location tells what changed, not why you changed it." The last two of the four also disagreed with the modification component. One said, "Rationale is a high-level concept, and the content of the commit is low-level implementation," deleting both the location and modification components. The other said, “Modifications are implementation details. The details are not the reason the code changed.” Four of the participants did not consider the alternatives component part of the rationale. One of them clarified, “Alternatives is not something that you actually implement!” For the selected alternative component, one participant thought this component is extra information and not part of the rationale. One participant thought that “validation answers why the code is correct, not the rationale,” deleting the validation component. Finally, one participant doubted the cost component but decided to keep it in the end.

As for any missing components of the rationale for code commits, most of the participants proposed new components to be added to the model. In total, the participants added 18 components to the initial model. Table 4.2 shows the proposed components along with the participant-provided component questions and example answers. Some of these components mean the same thing as other participants’ added components or our initial model’s components.

To update the initial model with the participants’ proposed components, we merged some components together in two steps. First, we grouped components that mean the same thing. Second, we abstracted away some of the specific cases that the participants mentioned into a more abstract component. Figure 4.2 shows the transition from the initial to the updated model, considering the proposed components. The timeliness proposed component is divided into two parts to be grouped. The final updated model, which is composed of 15 components, is shown in Table 4.1.

Table 4.2: Proposed Components of the Rationale for Code Commits

Component	**Component Expressed as Question**	**Example Answer**
Technical Requirement	Do I need to do the commit as part of my version control software?	Git requires that I make a merge commit.
Documentation	Why did you modify these parts of the code?	I modified these parts of code similarly to X because the implementation needs to follow this Y specific order.
Guidelines	Does the code follow all the company-defined coding guidelines?	Remove all hard-coded strings from the code.
Non-feasible alternative	Were there alternatives initially chosen by the developer that could not complete the requirement?	I tried to use HTML table to tabulate the changes needed but couldn't do that in the time frame. I used Bootstrap to wrap the changes to all the tables.
Opinion Selected Alternative	Why did you choose this implementation rather than another quality effective implementation?	I prefer generic error messages over unspecified errors and/or attempts to predict error types.
Constraints	What else does this change impact or not impact?	This change improves the exception handling but does so in a way that doesn't change any subsequent processing steps. Another approach might have required further changes elsewhere in the system.
Timeliness	Why is this change necessary now ? What other chunks of work need to take place before/after?	Suppose a commit includes a call to an API of a external system. The other system also needs changes in order for this commit to function properly. Has the other commit reached production yet?
Dependency	What is this change dependent on?	This change is dependent on a version 1.x of history y.
Person Author Committer	Who changed the code? Who was it who made this change (and when?) Who made the change? Why them?	I made this commit at the end of my internship; or, I made this commit on May 31st to hurriedly meet the June 1st deadline. Person A because they are familiar with error handling.
Time/Date	When was the change introduced? Who was it who made this change (and when?) When were the changes made?	I made this commit at the end of my internship; or, I made this commit on May 31st to hurriedly meet the June 1st deadline. Right after Bug A manifested.
Explanation of Modifications	Explain how the modifications work and why the code is like this.	Explanation of the specific syntax and purpose of it. (for .. i+=2 && var) Why i+=2 instead of i+1 Why i++ instead of ++i
Result	What observed results were noticed when running the code?	Due to the exception message handling, we see 10% more messages due to location not being observed before.
Environment	Which environment does the commit need to be pushed to?	(Dev → Test → Val → Prod)
Scope for future development	Did you identify future implementations that can be done in next cycle sprint?	We can improve our code further to cover all the test cases that couldn't be done due to the upcoming deadline.
Quality	How well is the code written from software engineering? Is it just a hack or small fix as opposed to a more thorough solution to fix a problem from previous commits?	Instead of using print statements in "try/catch" block, ideally, a logging library would have been a better choice.
Merge Conflict Success	Will there be a merge conflict with the master branch or previous commits of other collaborators?	Yes, if somebody else tried to remedy a fix or added other alternatives or extra code in parallel to you for the same code location.
Limitation	What are the known limitations of this commit?	This commit can address that for only specific inputs the intended goal would be accomplished or in certain cases, the expected output or runtime could be different from the general case.
Impact	How did this change impact the rest of the system?	This change requires John to update his code where he calls API X.

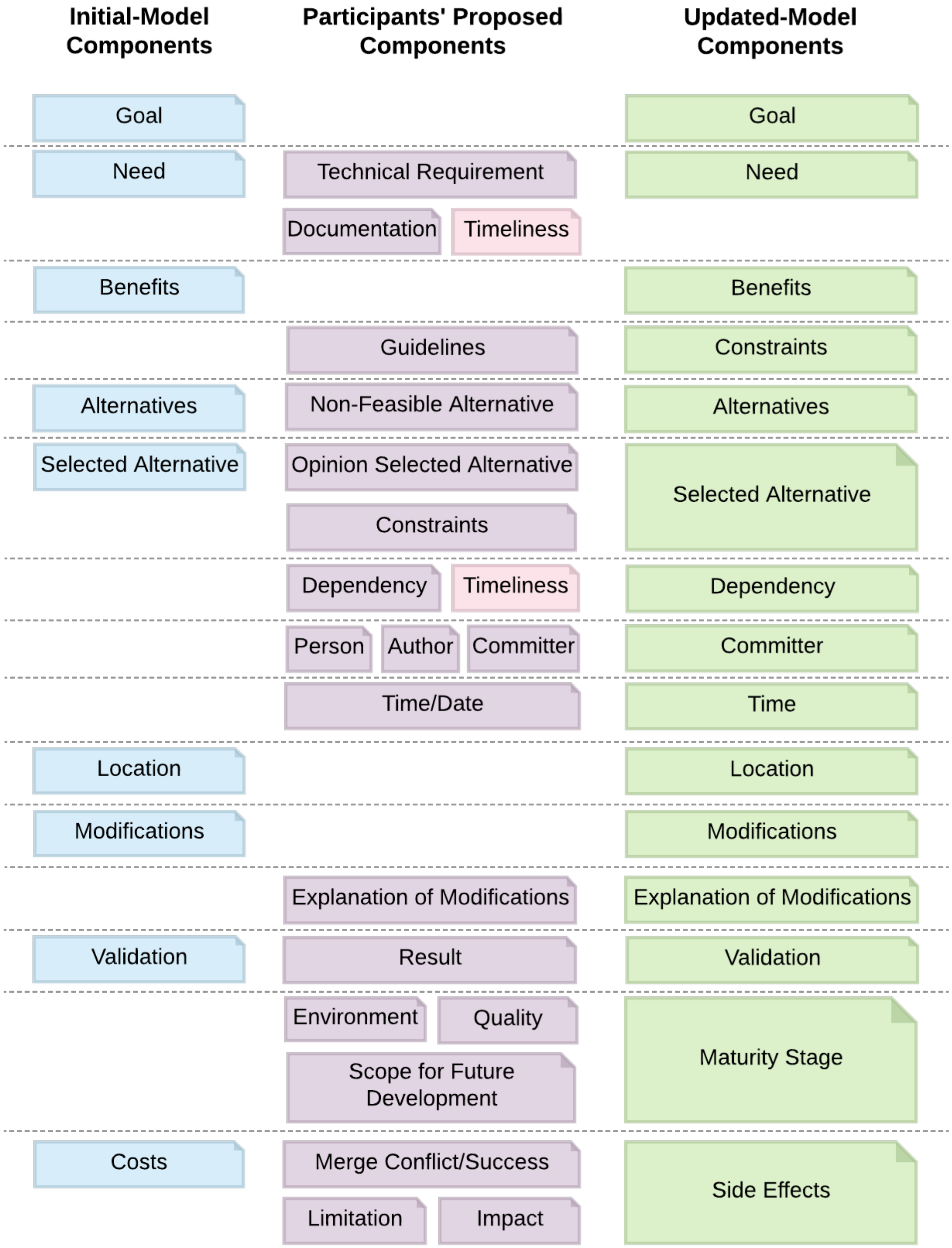


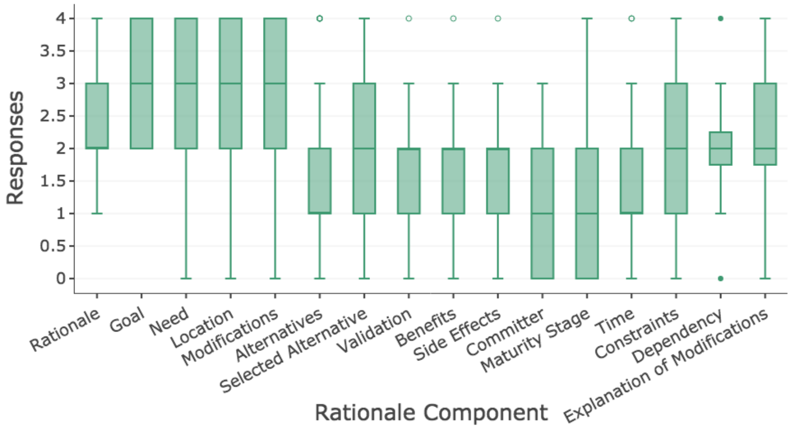
Figure 4.2: The Process of Updating the Rationale for Code Commits Model

4.3 Importance of Each Rationale Component

Not all the rationale components presented in Tables 3.1, 4.2, and 4.1 are considered to be equally important to the rationale. We asked the interview and online survey participants to identify how important each component is to the rationale for code commits (**RQ2**). We asked the participants to choose from the following scale:

(0) I would **easily know** the rationale for code commits **without finding** this component.
(1) I would **know** the rationale for code commits **without finding** this component.
(2) I would **better know** the rationale for code commits **when finding** this component.
(3) It's **hard to know** the rationale for code commits **without finding** this component.
(4) I **can't know** the rationale for code commits **without finding** this component.

The green boxes in Figures 4.1 and 4.3 show a summary of the participants' answers. Considering the median value of the participants' answers, the components of the rationale for code commits are of three levels of importance. The first level is the most important rationale components and includes the goal, need, location, and modifications. The second level is less important rationale components and includes benefits, constraints, selected alternative, dependency, explanation of modifications, validation, and side effects. The third level is the least important rationale components and includes alternatives, committer, time, and maturity stage.



(0) I would **easily know** the rationale for code commits **without finding** this component
(1) I would **know** the rationale for code commits **without finding** this component
(2) I would **better know** the rationale for code commits **when finding** this component
(3) It's **hard to know** the rationale for code commits **without finding** this component
(4) I **can't know** the rationale for code commits **without finding** this component

Figure 4.3: Rationale Components' Importance

4.4 The Need for Rationale and its Components

To learn about the developers' need for rationale and its components (**RQ3**), we asked the interview and online survey participants two questions. First, we asked them about the importance of rationale for the completion of their work. Second, we asked them about the frequency of their need for rationale and its components.

Figure 4.4 and the green boxes of rationale in Figures 4.1 and 4.3 show the participants' selected level of importance or need for rationale in general. Almost 50% of the participants need the rationale for code commits, but they can complete their work without it. A little more than 30% of the participants really need the rationale for code commits, and they struggle to complete their work without it.

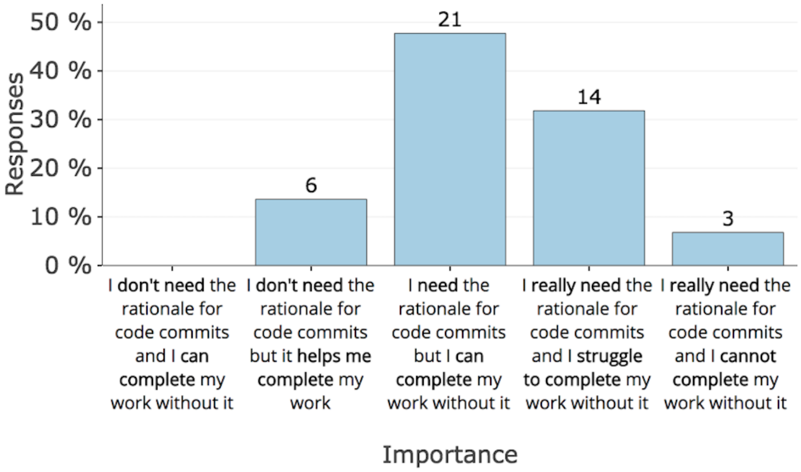
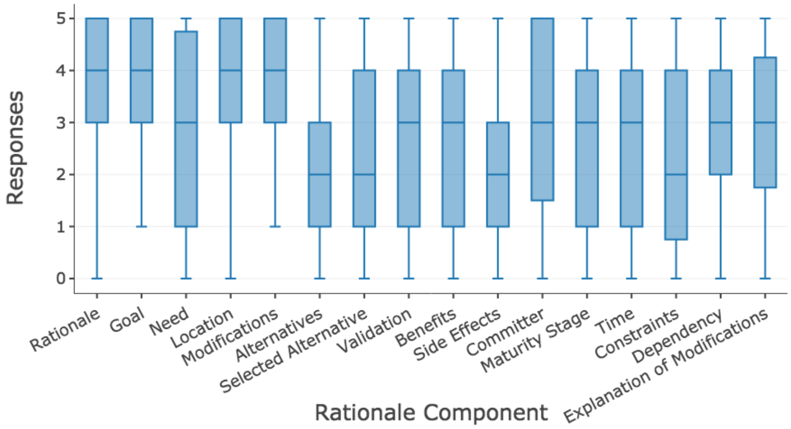


Figure 4.4: Rationale Importance for Work Completion

The blue boxes in Figures 4.1 and 4.5 shows a summary of the participants' answers about the frequency of their need for rationale and its components. We asked the participants to select the common frequency of need from a scale of five Likert points: (0) N/A, (1) a few times per year, (2) multiple times per year, (3) multiple times per month, (4) multiple times per week, or (5) multiple times a day. The average need for rationale, in general, is multiple times per week. The participants on average need the goal, location, and modifications components multiple times per week. They need the need, benefits, dependency, committer, time, explanation of modifications, validation, and maturity stage components multiple times per month. Finally, participants need the constraints, alternatives, selected alternative, and side effects components multiple times per year.



(0) N/A (1) A few times per year (2) Multiple times per year (3) Multiple times per month
(4) Multiple times per week (5) Multiple times per day

Figure 4.5: Frequency of Need for Rationale Components

4.5 Finding Rationale and its Components

To explore the developers' experience with finding rationale and its components (**RQ4**), we asked the interview and online survey participants about the amount of time they spend searching for the rationale for code commits, the frequency of finding rationale and its components, and the difficulty of finding rationale and its components.

Figure 4.6 shows the rationale search times spent by the participants. The figure includes both usual and hard-case search times. The participants' usual search time is five to ten minutes on average. In the hard cases, almost 50% of the participants spend more than 30 minutes searching for the rationale.

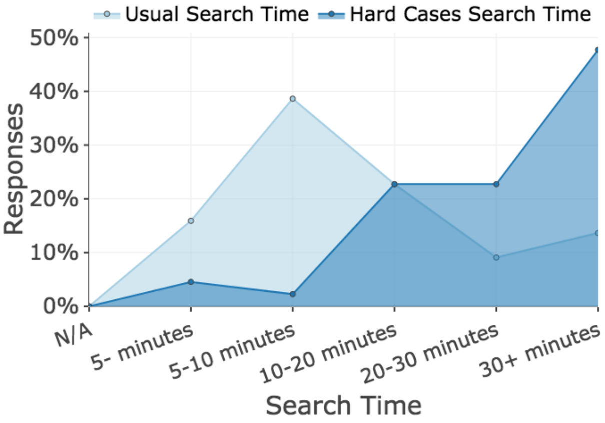
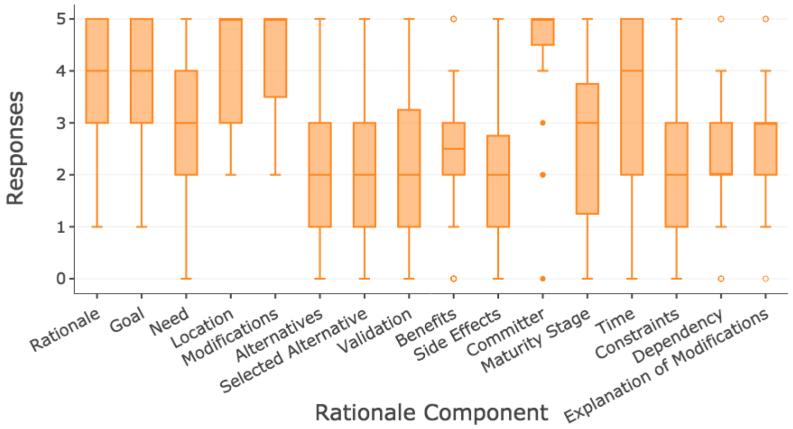


Figure 4.6: Time Spent Searching for Rationale

When we asked the interview participants, "When do you give up searching for rationale?" half of the participants indicated that they do not really give up searching for the rationale for code commits. In the context of this question, five participants mentioned making an effort to reach the author or reaching out to other developers on the team for help. Four participants come up with their own judgment of the rationale by looking at and revising the code and making sense of it. Another four of the participants completely give up when the author is no longer available, the author is no longer writing code, they have exhausted all the possibilities, they cannot run the code, or the code base is huge and no documentation is available.

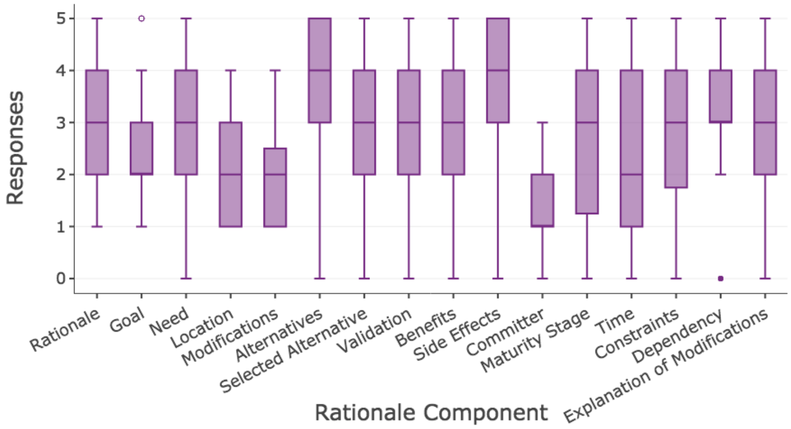
For the frequency of finding rationale and its components, we asked the participants to select from a scale of five Likert points: (0) N/A, (1) almost never, (2) rarely, (3) sometimes, (4) often, or (5) almost always. The orange boxes in Figures 4.1 and 4.7 summarize the participants' answers. The participants almost always find the committer, location, and modifications. They often find the goal and time. Sometimes they find the need, benefits, explanation of modifications, and maturity stage. Finally, the six components that they rarely find are constraints, alternatives, selected alternative, dependency, validation, and side effects.



(0) N/A (1) Almost never (2) Rarely (3) Sometimes (4) Often (5) Almost always

Figure 4.7: Frequency of Finding Rationale Components

For the difficulty of finding rationale and its components, we asked the participants to select from a scale of five Likert points: (0) N/A, (1) very easy, (2) easy, (3) neutral, (4) difficult, or (5) very difficult. The purple boxes in Figures 4.1 and 4.8 summarize the participants' answers. The participants think it is relatively easy to find the goal, committer, time, location, and modifications components. On the other hand, the participants find it relatively difficult to find the alternatives and side effects components. The rest of the rationale components have a neutral finding difficulty.

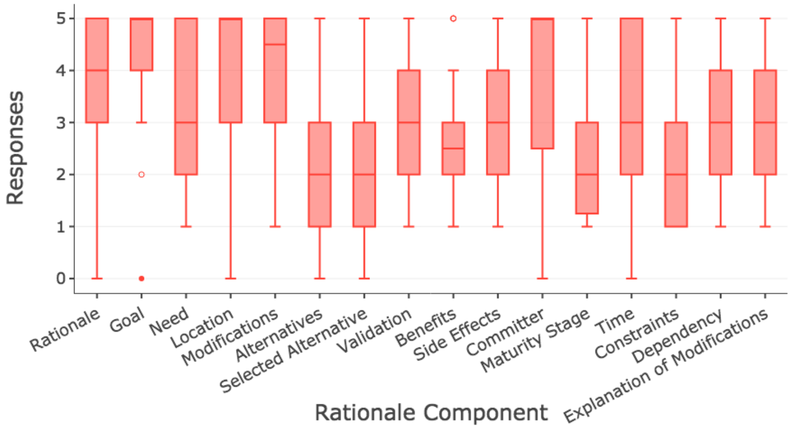


(0) N/A (1) Very easy (2) Easy (3) Neutral (4) Difficult (5) Very difficult

Figure 4.8: Difficulty of Finding Rationale Components

4.6 Recording Rationale and its Components

To find the participants' recording or lack of recording of the rationale for code commits (**RQ5**), we asked them to select the frequency with which they record rationale and its components. For this question, we presented choices as a scale of five Likert points: (0) N/A, (1) almost never, (2) rarely, (3) sometimes, (4) often, or (5) almost always. The red boxes in Figures 4.1 and 4.9 show a summary of the participants' answers. The participants almost always record the goal, committer, location, and modifications components. Sometimes they record the need, benefits, dependency, time, explanation of modifications, validation, and side effects components. Finally, they rarely record the constraints, alternatives, selected alternative, and maturity stage components.



(0) N/A (1) Almost never (2) Rarely (3) Sometimes (4) Often (5) Almost always

Figure 4.9: Recording of Rationale Components

Chapter 5

Discussion

In this section, we highlight some of the results presented in Chapter 4 and discuss the implications of these results.

5.1 Components of the Rationale for Code Commits

The interview participants proposed that a large number of components be included in the model of the rationale for code commits, indicating a diversity of the meaning of rationale. When different developers request the rationale for code commits, they probably are referring to different components of rationale.

For the model of rationale, some interview participants shared their concern about the model's usability or applicability and the ability to record the components for every single commit. One participant expressed that concern by saying that the model "seems like a good model, especially when I read the component and the example answer. It clarifies everything [developers] has trouble [answering]. I never thought of it like that even if I am doing it. If I look at commits and find answers to all the components, it would make my life a lot easier. I know it might not be doable or possible because no one will ever answer all these in a commit. However, it is a good model." Another participant mentioned, "I have included in my commit messages a description of the benefits and cost, including benchmarking numbers and a description of the benchmarks I ran and what the numbers were. I was asked to remove that from my commit message!"

A number of participants were concerned about the overlap between the components of the rationale for code commits (see Section 4.2). It is true that the answers for different components can be the same in some cases. Because components of the same theme (see Section 4.1) are similar, the answers to these components may be the same for some commits. However, we wanted this model to be general and comprehensive.

5.2 Importance of Rationale Components

The participants identified the goal, need, location, and modification components as the most important components of the rationale for a code commit. These most important components are the commit as presented in most version control systems. A commit in GitHub, for example, is presented as a commit message that specifies the goal and need of the changes, a list of changed files which specifies the location of the change, and the difference between the old and new versions which is the modifications. It may be that developers identify the primary parts of a commit as the rationale because it is what they see.

5.3 The Need for Rationale and its Components

Although a lack of rationale is not stopping more than 50% of the participants from completing their work (Figure 4.4), the interview participants strongly clarified that they do not give up seeking rationale (see Section 4.5). The rationale for code commits might not be extremely important for the completion of work. However, the rationale for code commits is something that developers spend a considerable amount of time searching for, especially when it is hard to find. The amount of time spent by the developers searching and not giving up the search for rationale implies a strong need for rationale. The rationale for code commits is valuable for software developers, even if they can still manage to complete their work without it.

5.4 Finding Rationale and its Components

Spending five to ten minutes searching for the rationale for code commits is a reasonable amount of time. Although that is the usual amount of time, there are hard cases where software developers spend a significant amount of time (20 to 30 minutes on average) searching for the rationale. Making rationale finding a less time-consuming process is an area that should be explored (see future work in Chapter 7).

Figure 5.1 shows the average frequency of need for and the frequency of finding the rationale and its components. The participants indicated that they are finding needed components except for the dependency and validation components, which are frequently needed but not found. The average frequency of recording these components indicates that the participants rarely record dependency and validation. Studying the need for, recording of, and finding of both components is discussed further in Chapter 7 on future work.

Figure 5.2 shows the average frequency of need and the difficulty of finding the rationale and its components. The participants indicated easy to neutral difficulty in finding needed rationale components, except for the alternatives and side effects components. Making rationale finding a simpler process is an area that should be explored (see Chapter 7 on future work).

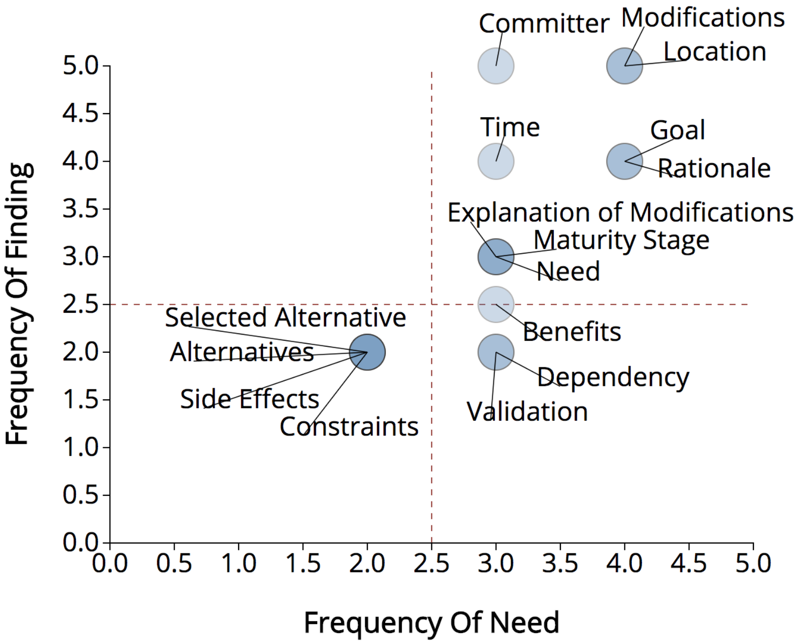


Figure 5.1: Frequency of Need vs. Frequency of Finding Rationale Components

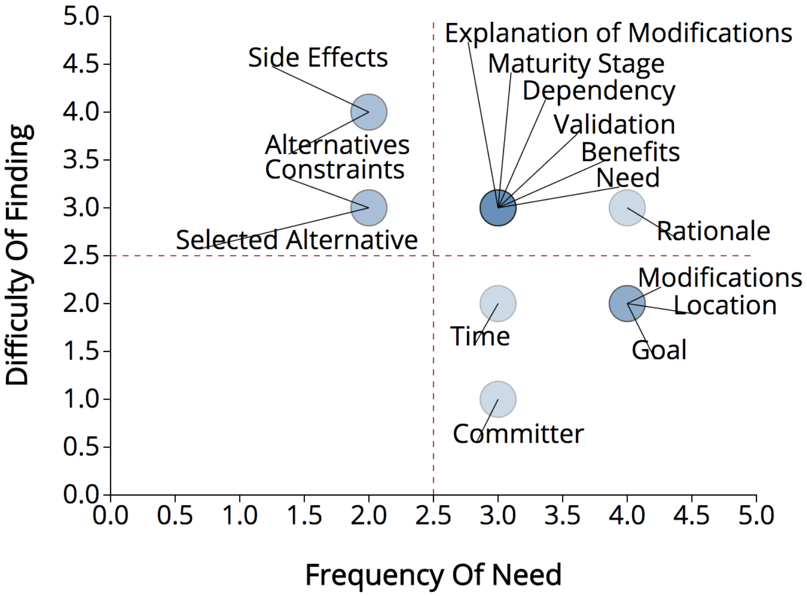


Figure 5.2: Frequency of Need vs. Difficulty of Finding Rationale Components

5.5 Recording Rationale and its Components

Figure 5.3 shows that the developers are recording the parts of rationale they consider to be important. Here we raise a question about the relationship between the recorded components and their importance. Are the developers not identifying unrecorded components as important because these components are not usually seen?

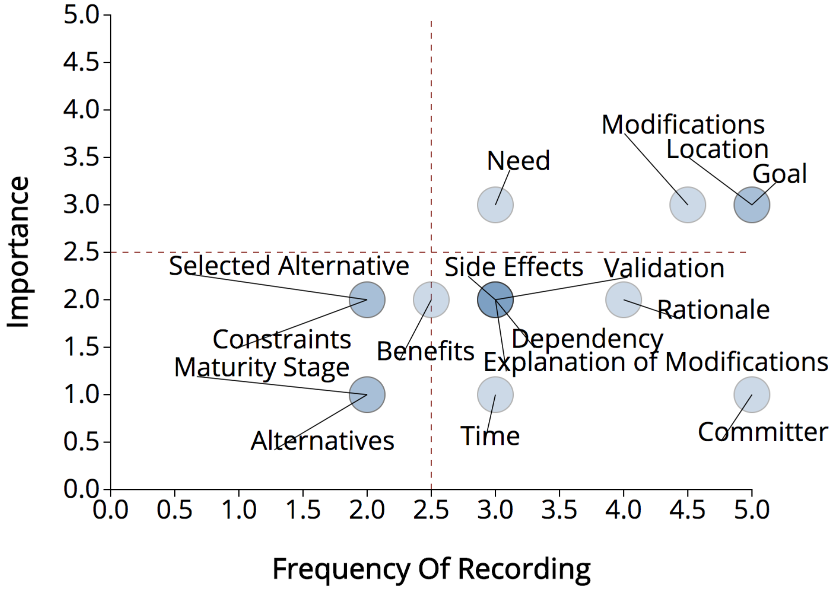


Figure 5.3: Frequency of Recording vs. Importance of Rationale Components

Chapter 6

Threats to Validity

This chapter presents the threats to the validity of our study, including construct, internal, external, and replicability threats.

6.1 Construct

Are we asking the right questions? To answer our research questions, we asked both open and quantitative questions. We scheduled the interview sessions to be relatively long (two hours), making sure that we gave the participants enough time to express their ideas and share their thoughts. At the beginning of each interview section, we asked the participants to "answer the questions in [their] own words and provide as much detail as [they] feel is relevant to address each question." We also placed an open question at the end of the interview to allow the participants to share any additional information about the topic.

6.2 Internal

Is the accuracy of the results skewed by the method with which we collected and analyzed them? The methods we used in our study, interviews and surveys, can be affected by bias and inaccurate responses. This effect could be intentional or unintentional. We gave gift cards to the interview participants, and some of the survey participants won gift cards, which could have biased our results. To mitigate these concerns, we clearly indicated that the compensation is for the time spent and not the given answers. We repeatedly and constantly used phrases to encourage the participants to provide their own honest opinions. We used the phrase "based on your experience" in most of the questions. We also clearly indicated that the participants should "feel free to change/add/delete or not." Sometimes, we also indicated that "there is no right or wrong answer; we are interested in what you think and your perspective."

6.3 External

Are our study results generalizable? By interviewing a group of developers, we cannot understand the whole developer population. To mitigate this threat to validity, we tried to recruit as diverse a population as possible. Our interview participants have diverse types and amounts of experience.

6.4 Replicability

Can the results of this study be replicated by others? It is difficult to replicate qualitative studies.The exact interview scripts and survey questions are available in Appendix A and B. We cannot share the raw data and transcripts of the interviews because we told the interview participants we would not share individual participants' data. However, counts of the participants' responses to the quantitative questions are available in Appendix C.

Chapter 7

Future Work

The results of our study motivate the development of novel techniques and tools to support developers in accurately recording and seeking the rationale for code commits. Further exploratory studies are encouraged to discover developers' strategies for recording and finding the rationale. The processes that developers follow to find the rationale for code commits, the tools they use, and the locations where they find rationale is outside the scope of our study. However, the participants in our study provided some points about their strategies in the context of the questions we asked. Future studies may use interviews, surveys, and shadowing or observing developers while working to discover their strategies.

Future research may also further validate the frequency of recording different components of rationale expressed by our participants by searching for evidence for it through mining of open-source project repositories and their related management and bug-tracking systems. This approach may be also interesting to also validate the reported importance of every rationale component by mining the recorded discussions, conversations, and messages between projects' software developers.

Finally, a detailed understanding of the components of the rationale for code commits may serve as inspiration for novel techniques to support developers in accurately recording and seeking rationale. Making rationale finding a simpler and less time-consuming process is an area that should be explored.

Chapter 8

Conclusion

In this study, we discovered that the rationale for code changes can be divided into components. We interviewed software developers to identify the rationale for code commits and its components. We created an initial model based on the findings of software history studies to start the discussion with the developers, asked the study participants to update the model to reflect what they mean by the rationale for code commits, and analyzed the participants' proposed components to create a final model of the rationale for code commits.

Additionally, we identified different characteristics of these components. We determined which components of rationale developers consider important, which components are needed more than others, which components are difficult to find, and which components are recorded by software developers.

An exploration of developers' strategies for recording and finding the rationale for code commits would be a study that naturally follows ours. Our results will assist future work in creating tools and techniques to support developers in accurately recording and efficiently finding the rationale. The identification of the rationale for code commits components and their characteristics are essential for building such tools and techniques, which will potentially improve the software developers' productivity.

Bibliography

[1] A. H. Dutoit, R. McCall, I. Mistrík, and B. Paech, *Rationale management in software engineering.* Springer Science & Business Media, 2007.

[2] J. E. Burge, J. M. Carroll, R. McCall, and I. Mistrik, *Rationale-based software engineering.* Springer, 2008.

[3] V. Rajlich, "Software evolution and maintenance," in *Proceedings of the on Future of Software Engineering*, ser. FOSE 2014. New York, NY, USA: ACM, 2014, pp. 133–144. [Online]. Available: http://doi.acm.org/10.1145/2593882.2593893

[4] M. Codoban, S. S. Ragavan, D. Dig, and B. Bailey, "Software history under the lens: A study on why and how developers examine it," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sept 2015, pp. 1–10.

[5] Y. Tao, Y. Dang, T. Xie, D. Zhang, and S. Kim, "How do software engineers understand code changes?: An exploratory study in industry," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, ser. FSE '12. New York, NY, USA: ACM, 2012, pp. 51:1–51:11. [Online]. Available: http://doi.acm.org/10.1145/2393596.2393656

[6] T. D. LaToza and B. A. Myers, "Hard-to-answer questions about code," in *Evaluation and Usability of Programming Languages and Tools*, ser. PLATEAU '10. New York, NY, USA: ACM, 2010, pp. 8:1–8:6. [Online]. Available: http://doi.acm.org/10.1145/1937117.1937125

[7] T. Fritz and G. C. Murphy, "Using information fragments to answer the questions developers ask," in *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ser. ICSE '10. New York, NY, USA: ACM, 2010, pp. 175–184. [Online]. Available: http://doi.acm.org/10.1145/1806799.1806828

[8] A. J. Ko, R. DeLine, and G. Venolia, "Information needs in collocated software development teams," in *29th International Conference on Software Engineering (ICSE'07)*, May 2007, pp. 344–353.

[9] T. Roehm, R. Tiarks, R. Koschke, and W. Maalej, "How do professional developers comprehend software?" in *Proceedings of the 34th International Conference on Software Engineering*, ser. ICSE '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 255–265. [Online]. Available: http://dl.acm.org/citation.cfm?id=2337223.2337254

[10] K. Y. Sharif and J. Buckley, "Observation of open source programmers' information seeking," in *2009 IEEE 17th International Conference on Program Comprehension*, May 2009, pp. 307–308.

[11] R. P. Buse and W. R. Weimer, "Automatically documenting program changes," in *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '10. New York, NY, USA: ACM, 2010, pp. 33–42. [Online]. Available: http://doi.acm.org/10.1145/1858996.1859005

[12] S. Zhang and M. D. Ernst, "Which configuration option should i change?" in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 152–163. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568251

[13] S. Jiang, C. McMillan, and R. Santelices, "Do programmers do change impact analysis in debugging?" *Empirical Software Engineering*, vol. 22, no. 2, pp. 631–669, Apr 2017. [Online]. Available: https://doi.org/10.1007/s10664-016-9441-9

[14] C. Rosen, B. Grawi, and E. Shihab, "Commit guru: Analytics and risk prediction of software commits," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2015. New York, NY, USA: ACM, 2015, pp. 966–969. [Online]. Available: http://doi.acm.org/10.1145/2786805.2803183

[15] R. Alkadhi, M. Nonnenmacher, E. Guzman, and B. Bruegge, "How do developers discuss rationale?" in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2018, pp. 357–369.

[16] L. F. Corts-Coy, M. Linares-Vsquez, J. Aponte, and D. Poshyvanyk, "On automatically generating commit messages via summarization of source code changes," in *2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation*, Sept 2014, pp. 275–284.

[17] M. Linares-Vsquez, L. F. Corts-Coy, J. Aponte, and D. Poshyvanyk, "Changescribe: A tool for automatically generating commit messages," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2, May 2015, pp. 709–712.

[18] S. Jiang, A. Armaly, and C. McMillan, "Automatically generating commit messages from diffs using neural machine translation," in *Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE 2017. Piscataway, NJ, USA: IEEE Press, 2017, pp. 135–146. [Online]. Available: http://dl.acm.org/citation.cfm?id=3155562.3155583

[19] S. Jiang and C. McMillan, "Towards automatic generation of short summaries of commits," in *Proceedings of the 25th International Conference on Program Comprehension*, ser. ICPC '17. Piscataway, NJ, USA: IEEE Press, 2017, pp. 320–323. [Online]. Available: https://doi.org/10.1109/ICPC.2017.12

[20] S. Rastkar and G. C. Murphy, "Why did this code change?" in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13. Piscataway, NJ, USA:

IEEE Press, 2013, pp. 1193–1196. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486788.2486959

[21] A. Alali, H. Kagdi, and J. I. Maletic, "What's a typical commit? a characterization of open source software repositories," in *2008 16th IEEE International Conference on Program Comprehension*, June 2008, pp. 182–191.

[22] L. P. Hattori and M. Lanza, "On the nature of commits," in *Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE'08. Piscataway, NJ, USA: IEEE Press, 2008, pp. III–63–III–71. [Online]. Available: https://doi.org/10.1109/ASEW.2008.4686322

[23] A. Hindle, D. M. German, M. W. Godfrey, and R. C. Holt, "Automatic classication of large changes into maintenance categories," in *2009 IEEE 17th International Conference on Program Comprehension*, May 2009, pp. 30–39.

[24] R. Goyal, G. Ferreira, C. Kästner, and J. Herbsleb, "Identifying unusual commits on github," *Journal of Software: Evolution and Process*, vol. 30, no. 1, 2018.

[25] M. Hilton, N. Nelson, T. Tunnell, D. Marinov, and D. Dig, "Trade-offs in continuous integration: Assurance, security, and flexibility," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2017. New York, NY, USA: ACM, 2017, pp. 197–207. [Online]. Available: http://doi.acm.org/10.1145/3106237.3106270

[26] F. Shull, J. Singer, and D. I. Sjøberg, *Guide to advanced empirical software engineering.* Springer, 2007.

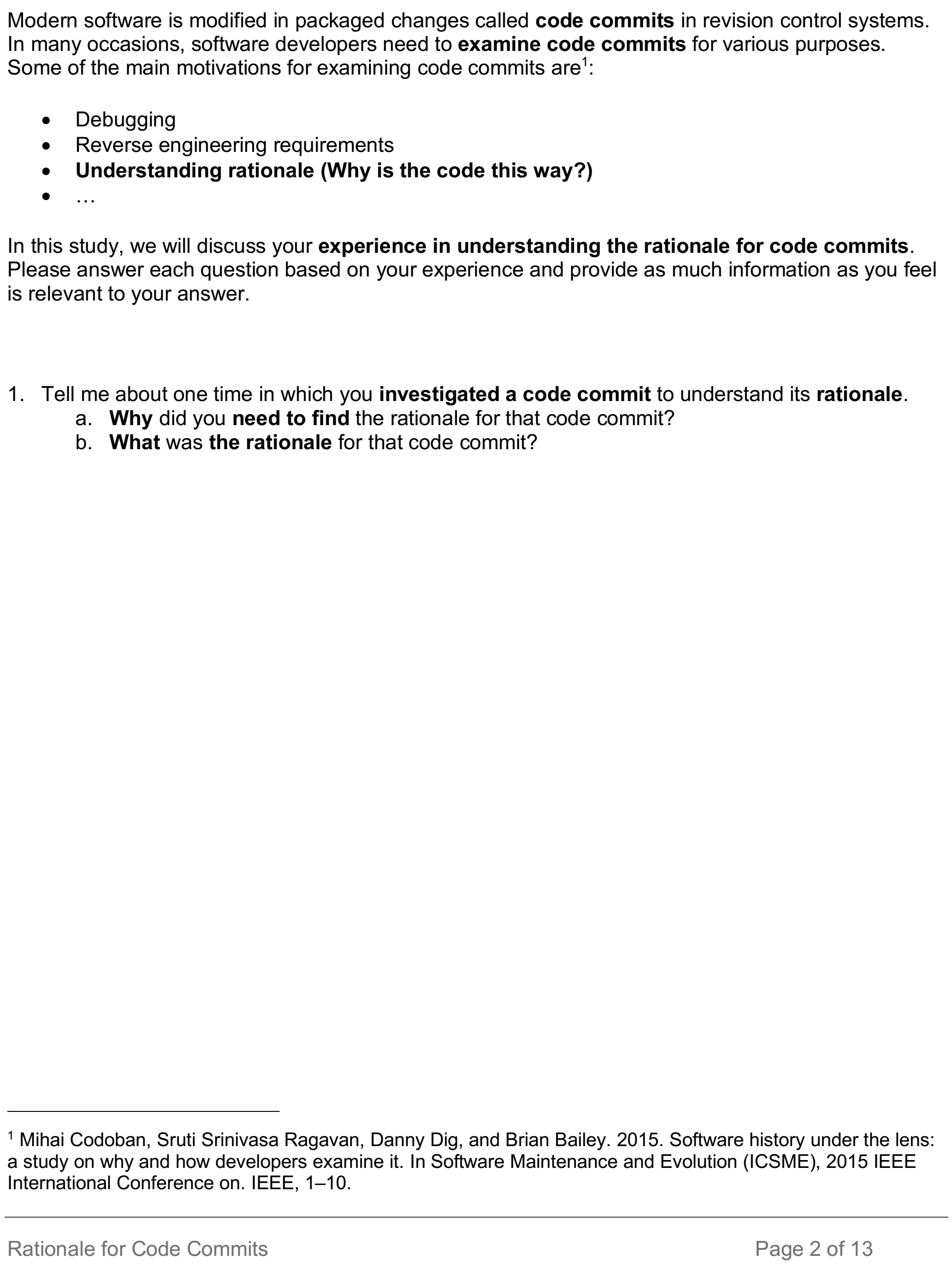
Appendix A

Interview Script

A.1 Interviewee Version

Rationale for Code Commits Interview

Please answer the questions in your own words and provide as much detail as you feel is relevant to address each question. Some questions have choices. I will instruct you to choose the answer for the ones with choices.



Modern software is modified in packaged changes called **code commits** in revision control systems. In many occasions, software developers need to **examine code commits** for various purposes. Some of the main motivations for examining code commits are[1]:

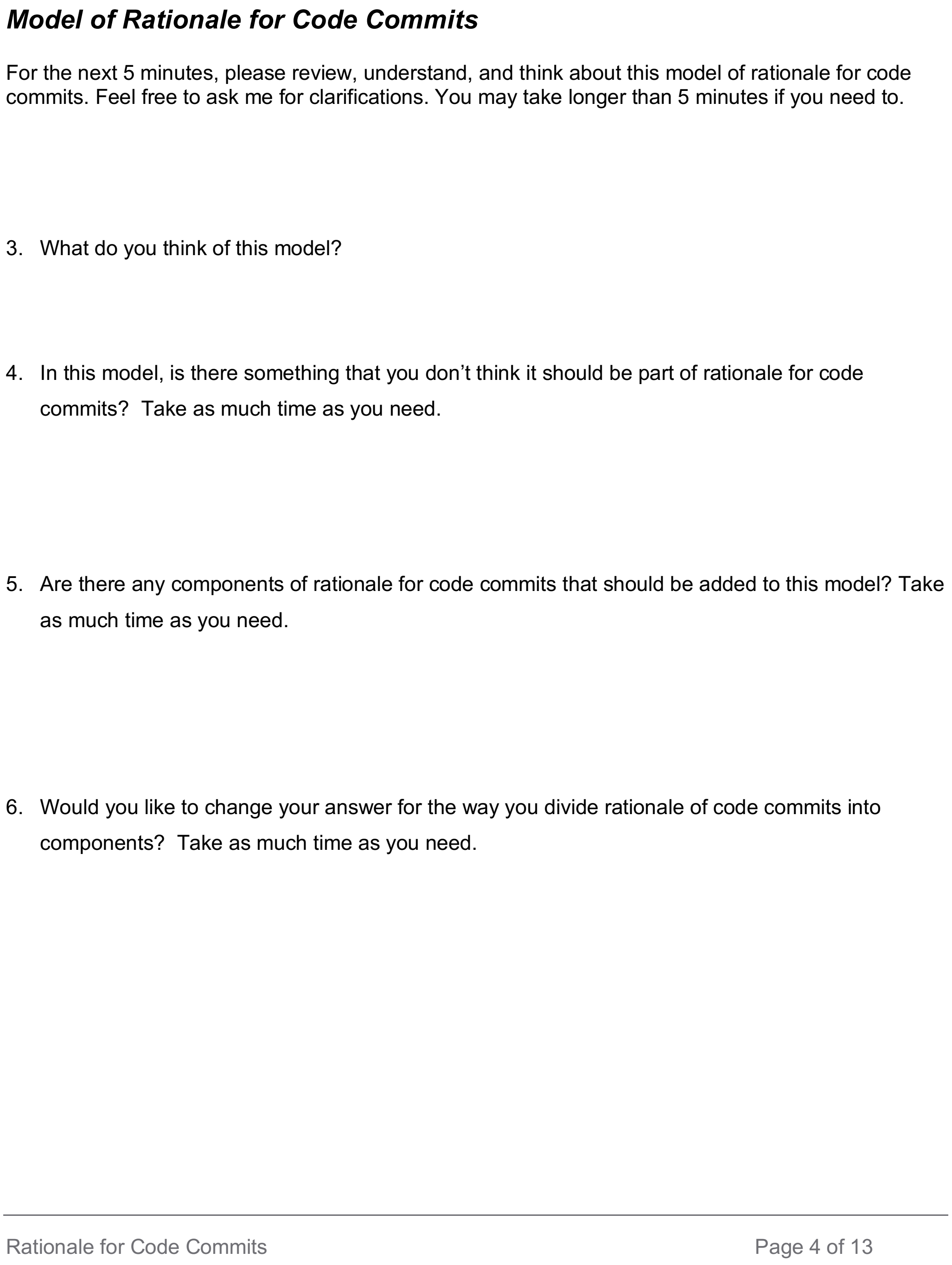
- Debugging
- Reverse engineering requirements
- **Understanding rationale (Why is the code this way?)**
- …

In this study, we will discuss your **experience in understanding the rationale for code commits**. Please answer each question based on your experience and provide as much information as you feel is relevant to your answer.

1. Tell me about one time in which you **investigated a code commit** to understand its **rationale**.
   a. **Why** did you **need to find** the rationale for that code commit?
   b. **What** was **the rationale** for that code commit?

[1] Mihai Codoban, Sruti Srinivasa Ragavan, Danny Dig, and Brian Bailey. 2015. Software history under the lens: a study on why and how developers examine it. In Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on. IEEE, 1–10.

2. If you had to **divide rationale into components**, what are the components of rationale for code commits? Take as much time as you need.



Model of Rationale for Code Commits

For the next 5 minutes, please review, understand, and think about this model of rationale for code commits. Feel free to ask me for clarifications. You may take longer than 5 minutes if you need to.

3. What do you think of this model?

4. In this model, is there something that you don’t think it should be part of rationale for code commits? Take as much time as you need.

5. Are there any components of rationale for code commits that should be added to this model? Take as much time as you need.

6. Would you like to change your answer for the way you divide rationale of code commits into components? Take as much time as you need.

Finding and Recording Rationale for Code Commits

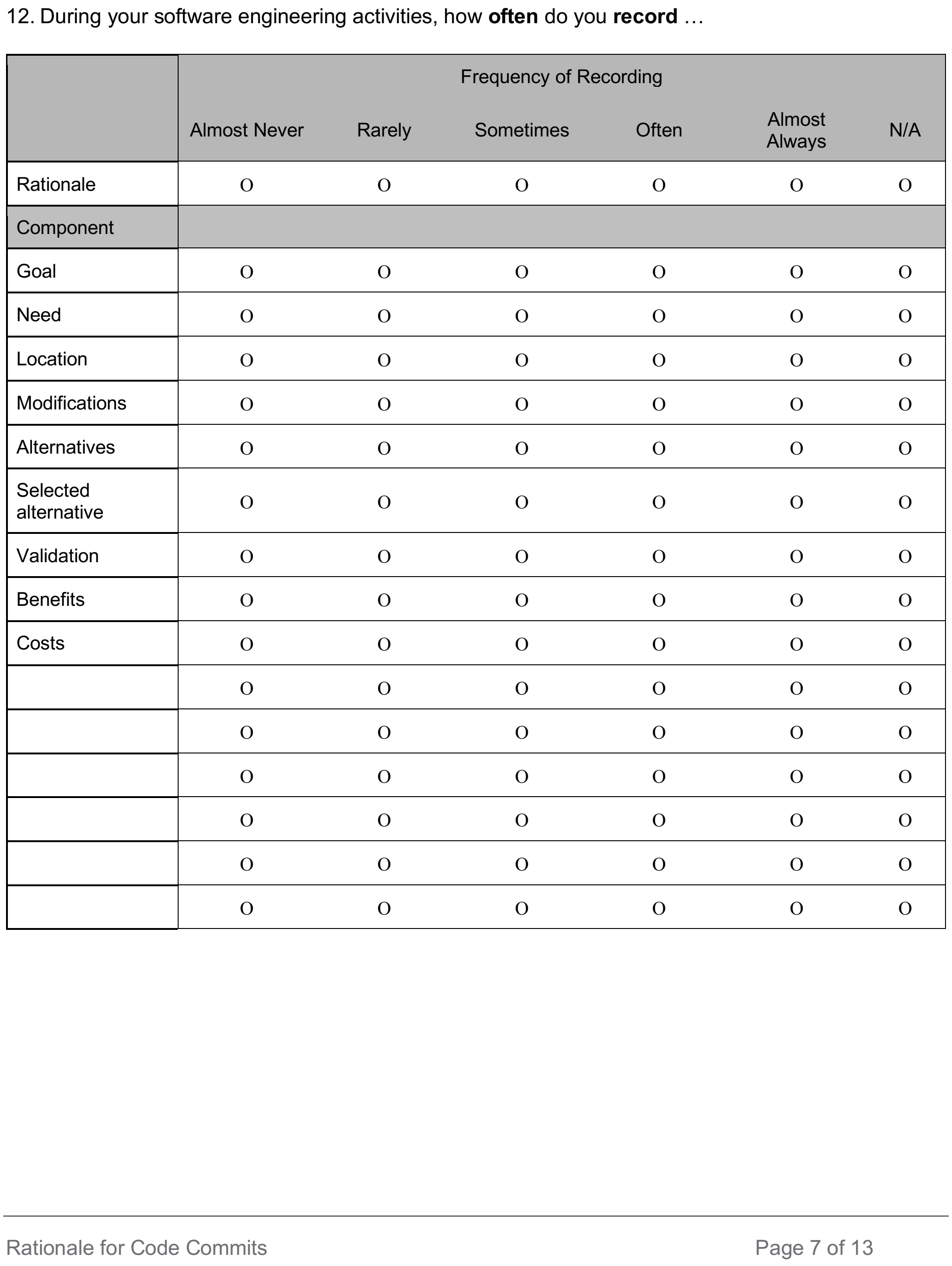
7. What are your **strategies for finding** the rationale of code commits? Please include:
    a. Your **steps** for finding rationale of code commits.
    b. The **location** of which you find the rationale for code commits.
    c. The **tools** you use to find the rationale for code commits.
    d. The **rationale components** that fit every strategy.

8. When your strategy does not lead you to the rationale of code commits, what do you do?

9. In what **order** do you follow your **strategies**?

10. What would make your life easier in this process?

11. As a developer, what are your strategies for **keeping a record** of the rationale for your code commits? Please include:
    a. Your **steps** for recording rationale of code commits.
    b. The **location** of which you record the rationale for code commits.
    c. The **tools** you use to record the rationale for code commits.
    d. The **rationale components** that fit every strategy.

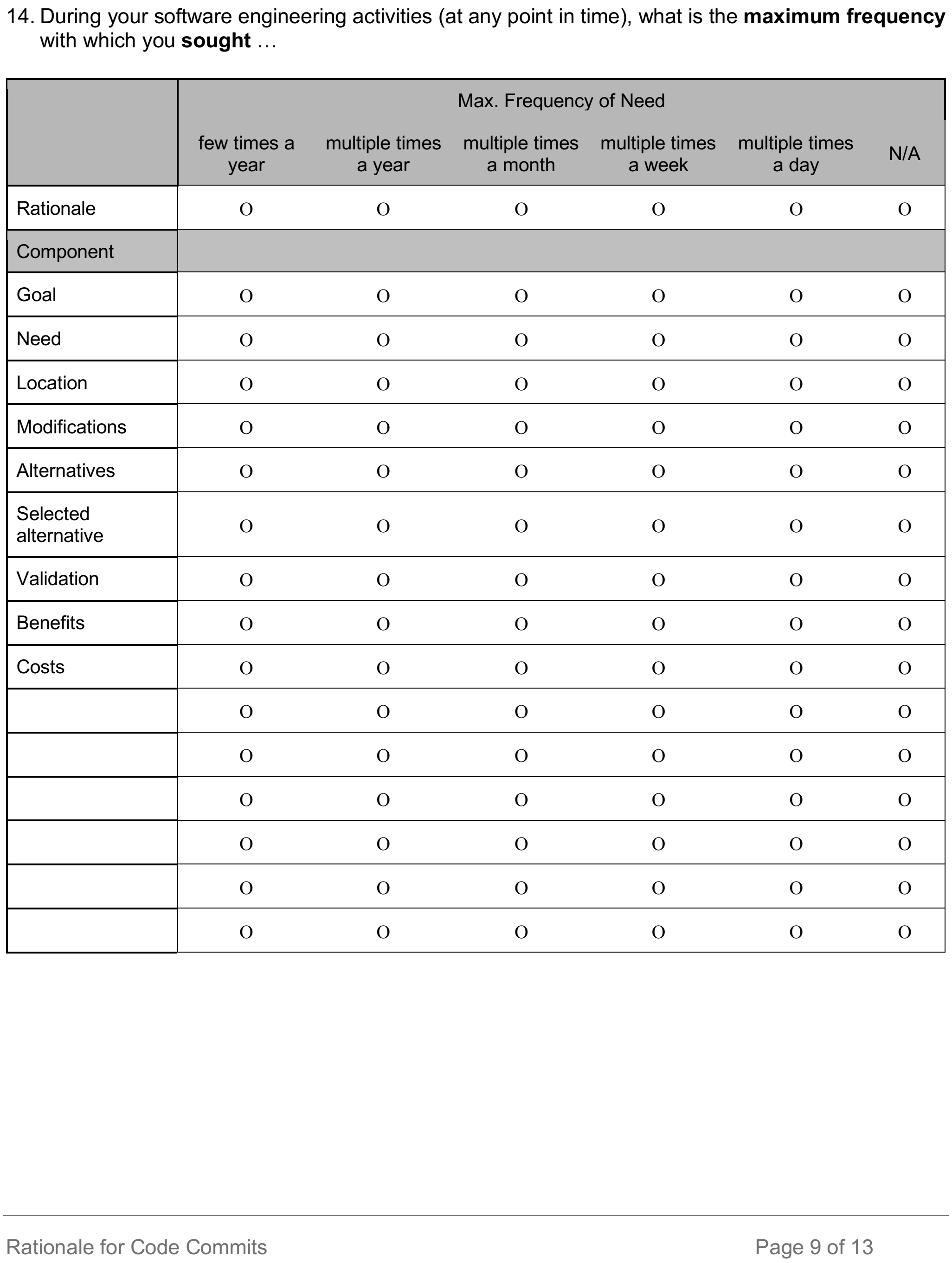


12. During your software engineering activities, how **often** do you **record** …

	Frequency of Recording					
	Almost Never	Rarely	Sometimes	Often	Almost Always	N/A
Rationale	O	O	O	O	O	O
Component						
Goal	O	O	O	O	O	O
Need	O	O	O	O	O	O
Location	O	O	O	O	O	O
Modifications	O	O	O	O	O	O
Alternatives	O	O	O	O	O	O
Selected alternative	O	O	O	O	O	O
Validation	O	O	O	O	O	O
Benefits	O	O	O	O	O	O
Costs	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O

13. During your software engineering activities, which **frequency** best reflects how often you **sought** …

	Frequency of Need					
	few times a year	multiple times a year	multiple times a month	multiple times a week	multiple times a day	N/A
Rationale	O	O	O	O	O	O
Component						
Goal	O	O	O	O	O	O
Need	O	O	O	O	O	O
Location	O	O	O	O	O	O
Modifications	O	O	O	O	O	O
Alternatives	O	O	O	O	O	O
Selected alternative	O	O	O	O	O	O
Validation	O	O	O	O	O	O
Benefits	O	O	O	O	O	O
Costs	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O

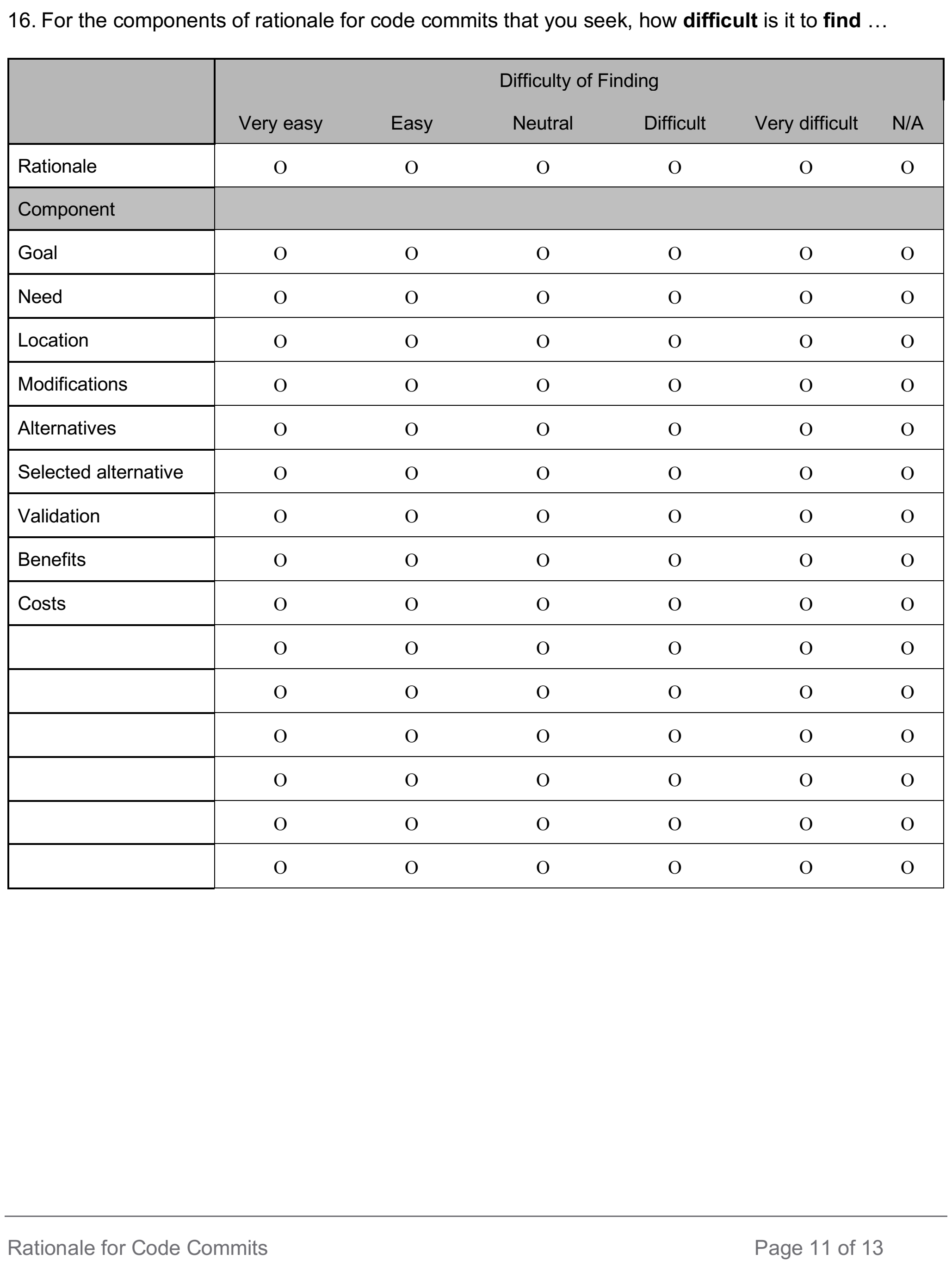


14. During your software engineering activities (at any point in time), what is the **maximum frequency** with which you **sought** …

	Max. Frequency of Need					
	few times a year	multiple times a year	multiple times a month	multiple times a week	multiple times a day	N/A
Rationale	○	○	○	○	○	○
Component						
Goal	○	○	○	○	○	○
Need	○	○	○	○	○	○
Location	○	○	○	○	○	○
Modifications	○	○	○	○	○	○
Alternatives	○	○	○	○	○	○
Selected alternative	○	○	○	○	○	○
Validation	○	○	○	○	○	○
Benefits	○	○	○	○	○	○
Costs	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○

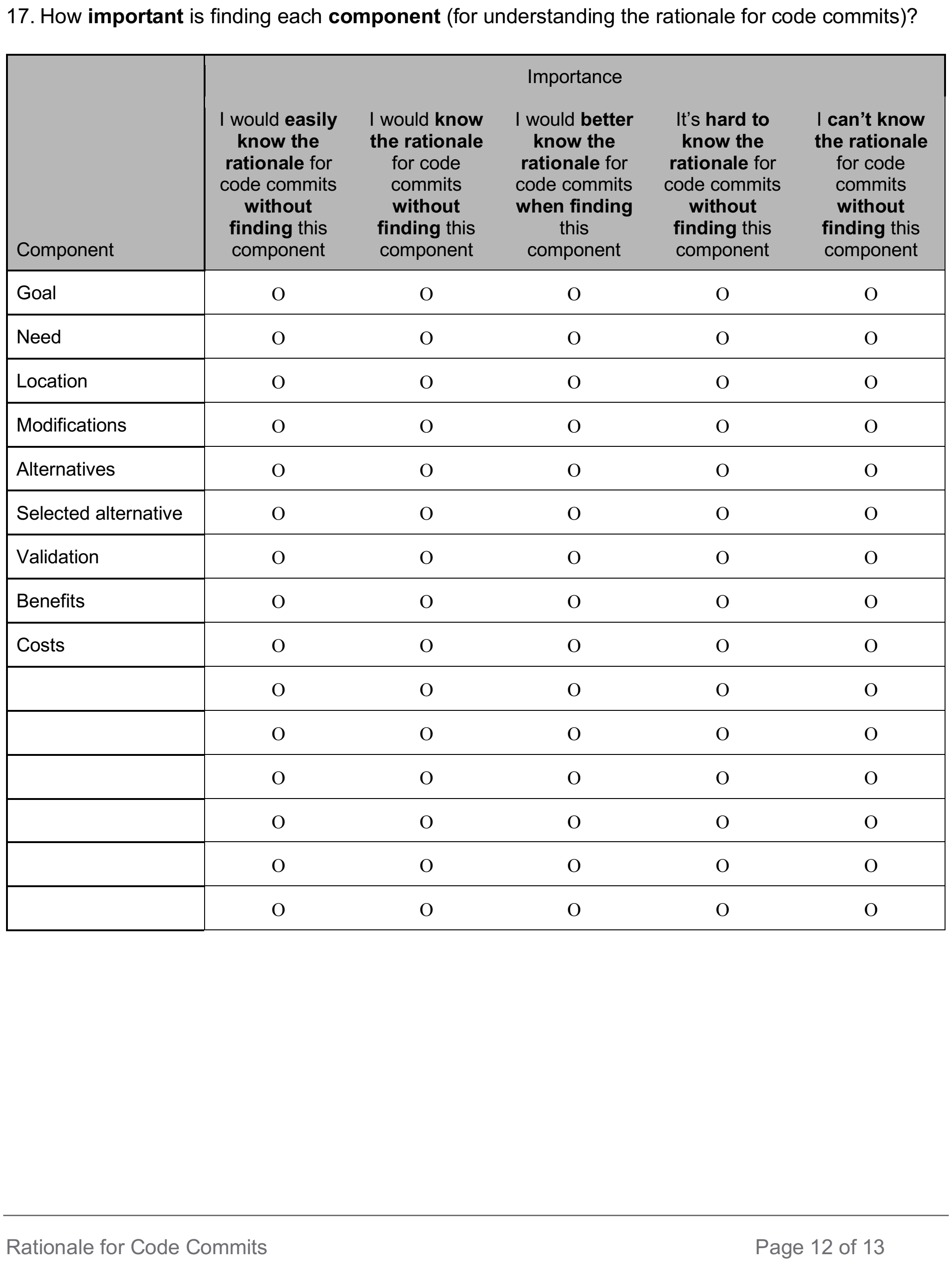
15. For the components of rationale for code commits that you seek, how **often** do you usually **find** …

	Frequency of Finding					
	Almost Never	Rarely	Sometimes	Often	Almost Always	N/A
Rationale	O	O	O	O	O	O
Component						
Goal	O	O	O	O	O	O
Need	O	O	O	O	O	O
Location	O	O	O	O	O	O
Modifications	O	O	O	O	O	O
Alternatives	O	O	O	O	O	O
Selected alternative	O	O	O	O	O	O
Validation	O	O	O	O	O	O
Benefits	O	O	O	O	O	O
Costs	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O



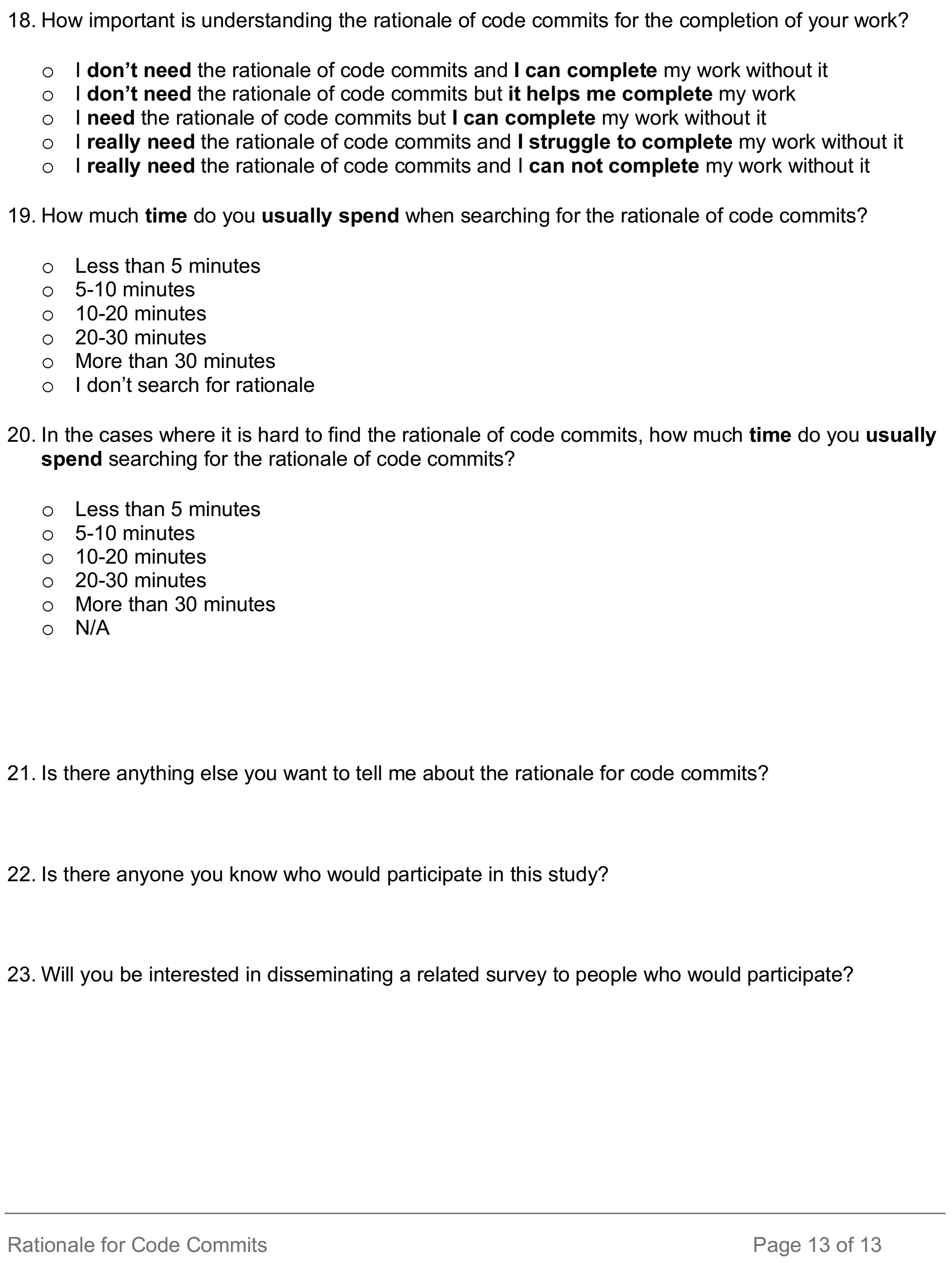
16. For the components of rationale for code commits that you seek, how **difficult** is it to **find** …

	Difficulty of Finding					
	Very easy	Easy	Neutral	Difficult	Very difficult	N/A
Rationale	O	O	O	O	O	O
Component						
Goal	O	O	O	O	O	O
Need	O	O	O	O	O	O
Location	O	O	O	O	O	O
Modifications	O	O	O	O	O	O
Alternatives	O	O	O	O	O	O
Selected alternative	O	O	O	O	O	O
Validation	O	O	O	O	O	O
Benefits	O	O	O	O	O	O
Costs	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O
	O	O	O	O	O	O



17. How **important** is finding each **component** (for understanding the rationale for code commits)?

	Importance				
Component	I would **easily know the rationale** for code commits **without finding** this component	I would **know the rationale** for code commits **without finding** this component	I would **better know the rationale** for code commits **when finding** this component	It's **hard to know the rationale** for code commits **without finding** this component	I **can't know the rationale** for code commits **without finding** this component
Goal	O	O	O	O	O
Need	O	O	O	O	O
Location	O	O	O	O	O
Modifications	O	O	O	O	O
Alternatives	O	O	O	O	O
Selected alternative	O	O	O	O	O
Validation	O	O	O	O	O
Benefits	O	O	O	O	O
Costs	O	O	O	O	O
	O	O	O	O	O
	O	O	O	O	O
	O	O	O	O	O
	O	O	O	O	O
	O	O	O	O	O
	O	O	O	O	O



18. How important is understanding the rationale of code commits for the completion of your work?

○ I **don't need** the rationale of code commits and **I can complete** my work without it
○ I **don't need** the rationale of code commits but **it helps me complete** my work
○ I **need** the rationale of code commits but **I can complete** my work without it
○ I **really need** the rationale of code commits and **I struggle to complete** my work without it
○ I **really need** the rationale of code commits and I **can not complete** my work without it

19. How much **time** do you **usually spend** when searching for the rationale of code commits?

○ Less than 5 minutes
○ 5-10 minutes
○ 10-20 minutes
○ 20-30 minutes
○ More than 30 minutes
○ I don't search for rationale

20. In the cases where it is hard to find the rationale of code commits, how much **time** do you **usually spend** searching for the rationale of code commits?

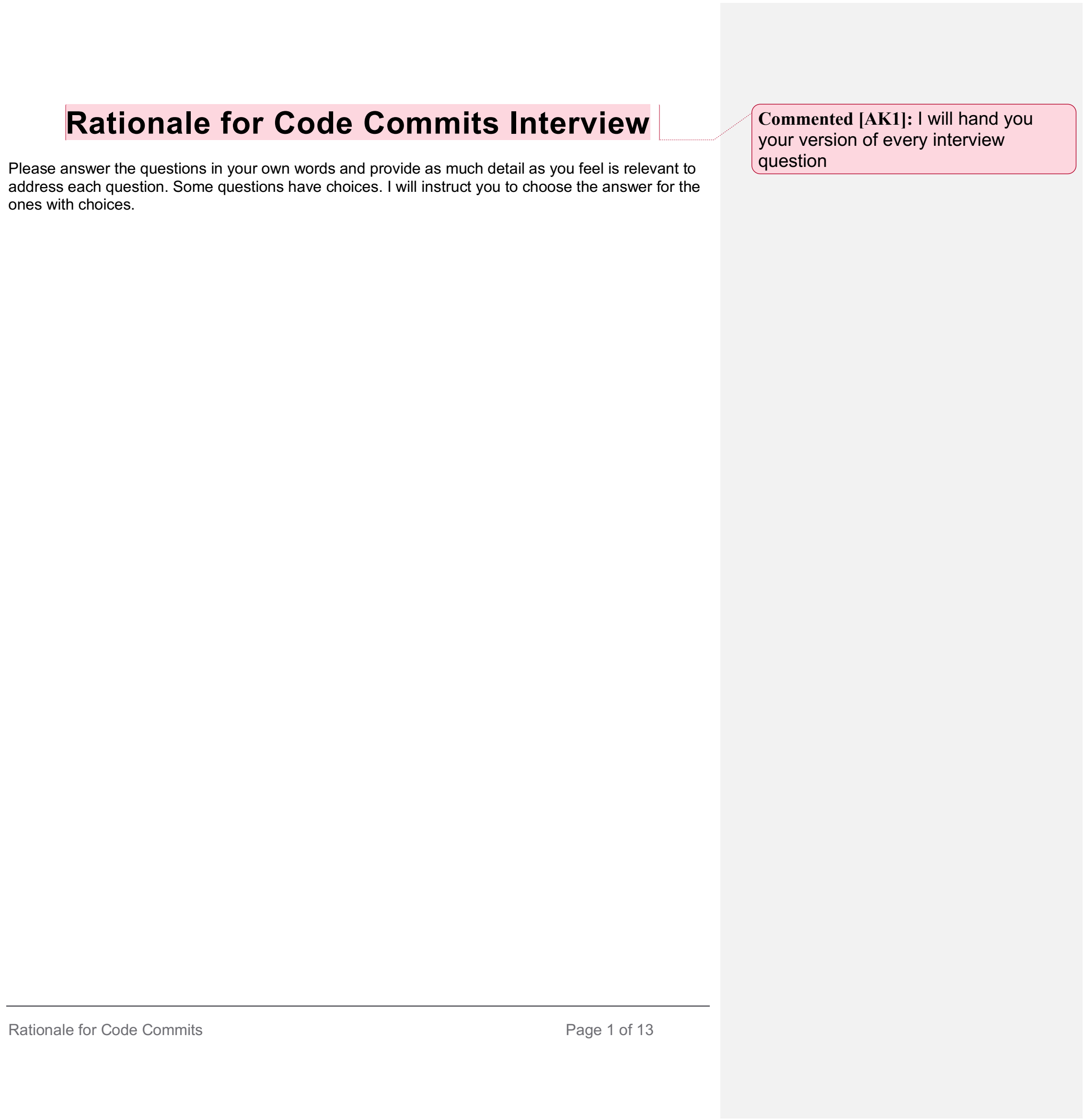
○ Less than 5 minutes
○ 5-10 minutes
○ 10-20 minutes
○ 20-30 minutes
○ More than 30 minutes
○ N/A

21. Is there anything else you want to tell me about the rationale for code commits?

22. Is there anyone you know who would participate in this study?

23. Will you be interested in disseminating a related survey to people who would participate?

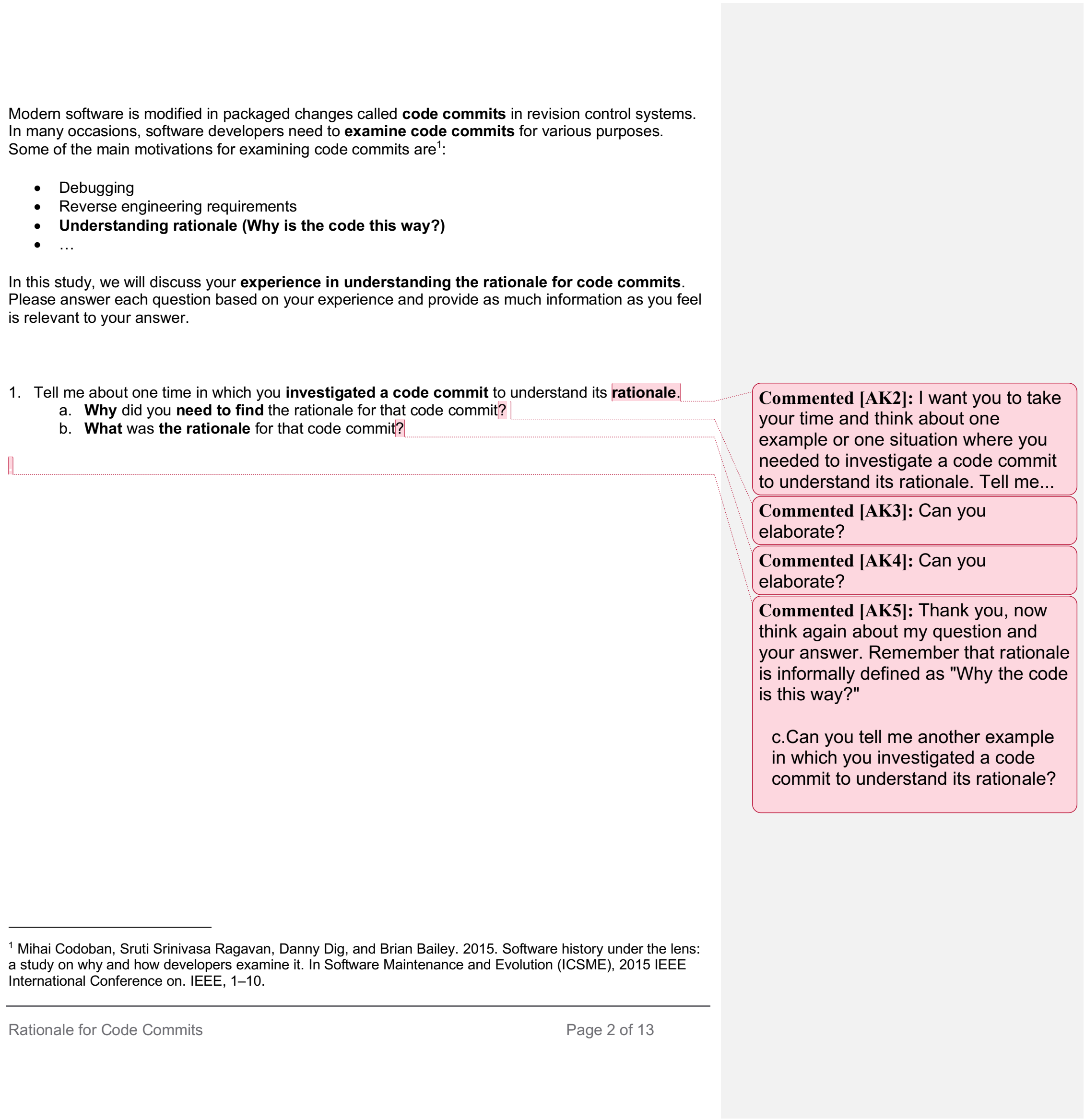
A.2 Interviewer Version



Rationale for Code Commits Interview

Commented [AK1]: I will hand you your version of every interview question

Please answer the questions in your own words and provide as much detail as you feel is relevant to address each question. Some questions have choices. I will instruct you to choose the answer for the ones with choices.



Modern software is modified in packaged changes called **code commits** in revision control systems. In many occasions, software developers need to **examine code commits** for various purposes. Some of the main motivations for examining code commits are[1]:

- Debugging
- Reverse engineering requirements
- **Understanding rationale (Why is the code this way?)**
- …

In this study, we will discuss your **experience in understanding the rationale for code commits**. Please answer each question based on your experience and provide as much information as you feel is relevant to your answer.

1. Tell me about one time in which you **investigated a code commit** to understand its **rationale**.
   a. **Why** did you **need to find** the rationale for that code commit?
   b. **What** was **the rationale** for that code commit?

Commented [AK2]: I want you to take your time and think about one example or one situation where you needed to investigate a code commit to understand its rationale. Tell me...

Commented [AK3]: Can you elaborate?

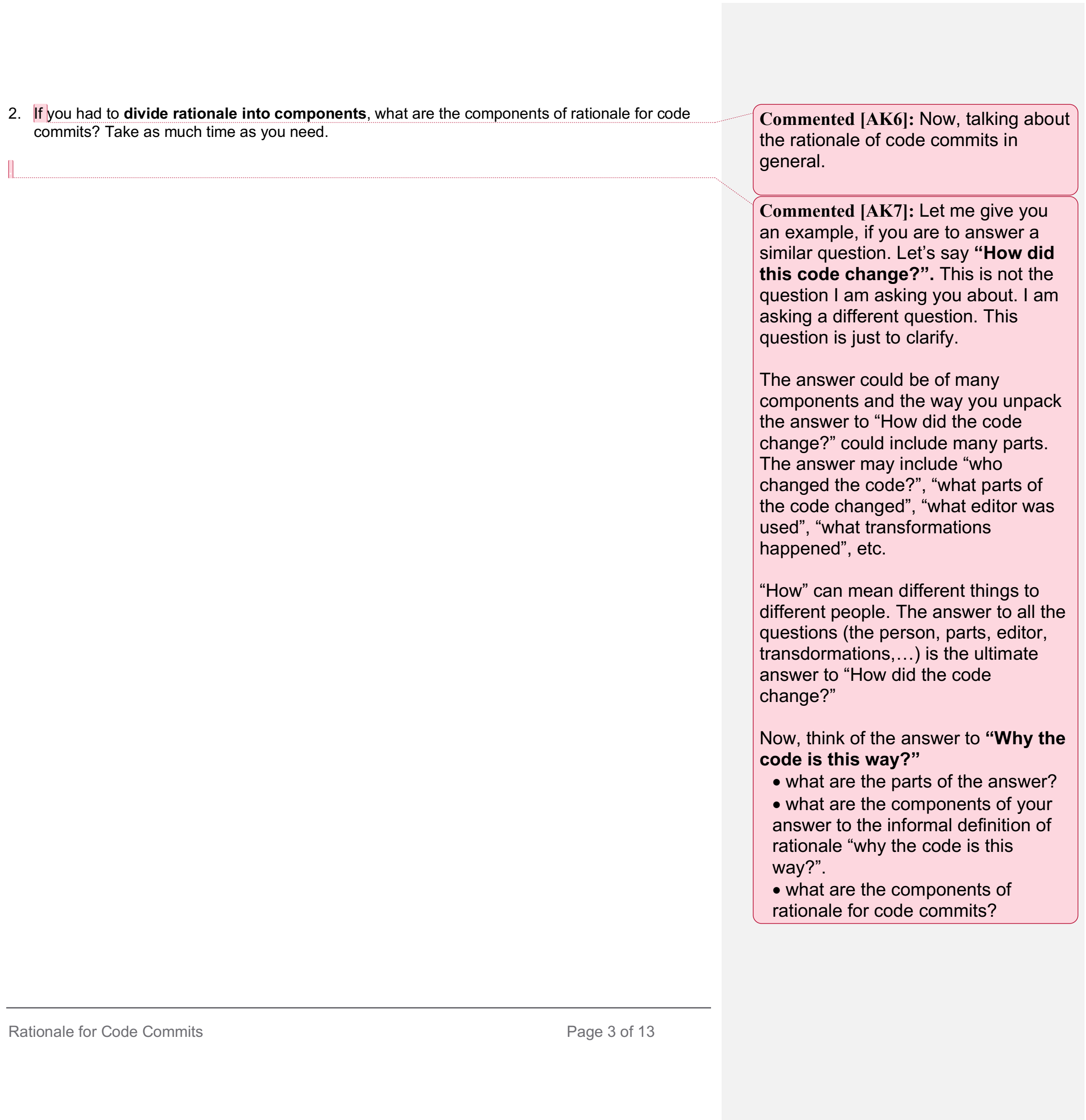
Commented [AK4]: Can you elaborate?

Commented [AK5]: Thank you, now think again about my question and your answer. Remember that rationale is informally defined as "Why the code is this way?"

c.Can you tell me another example in which you investigated a code commit to understand its rationale?

[1] Mihai Codoban, Sruti Srinivasa Ragavan, Danny Dig, and Brian Bailey. 2015. Software history under the lens: a study on why and how developers examine it. In Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on. IEEE, 1–10.

Rationale for Code Commits Page 2 of 13



2. If you had to **divide rationale into components**, what are the components of rationale for code commits? Take as much time as you need.

Commented [AK6]: Now, talking about the rationale of code commits in general.

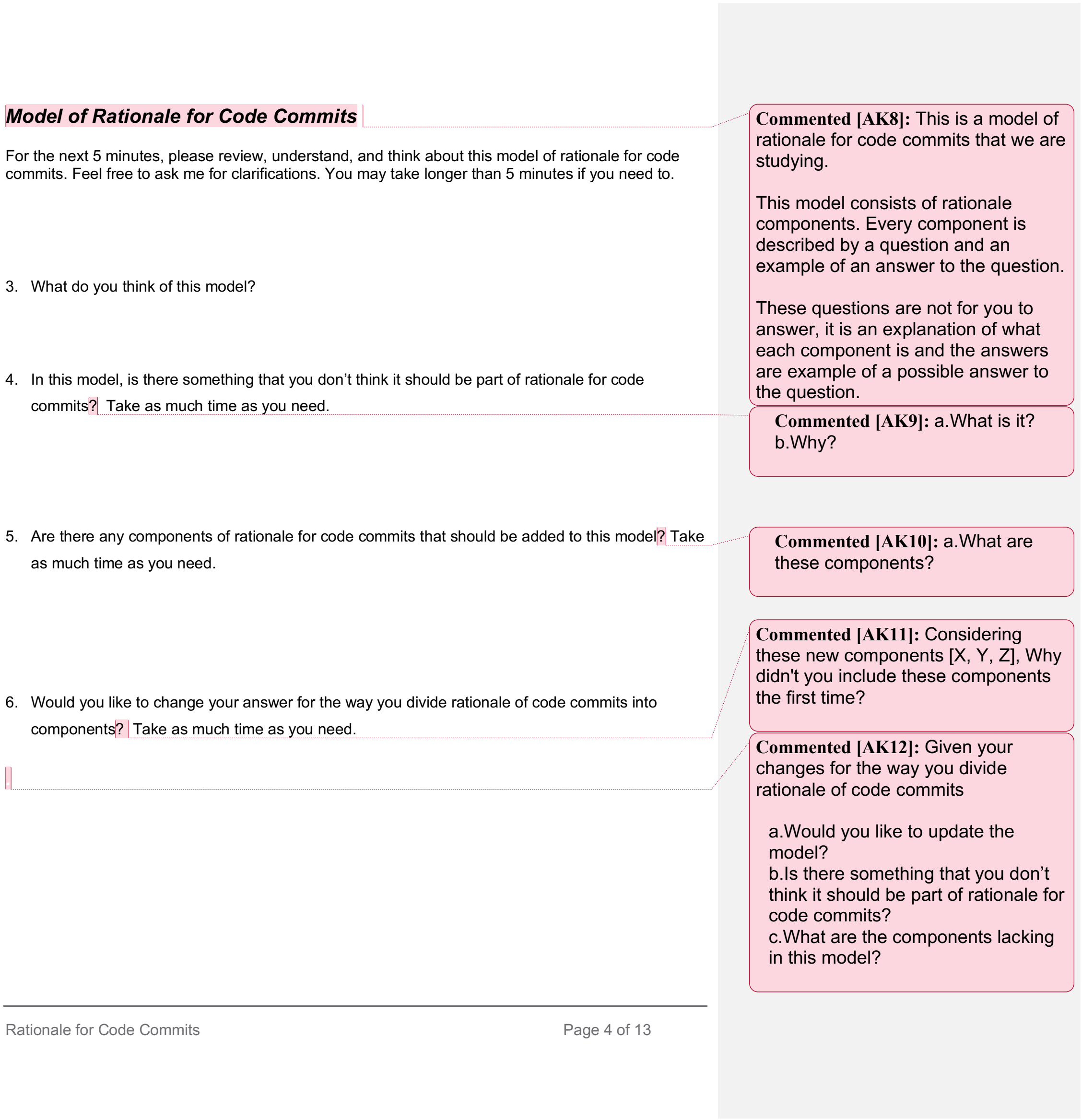
Commented [AK7]: Let me give you an example, if you are to answer a similar question. Let's say **"How did this code change?".** This is not the question I am asking you about. I am asking a different question. This question is just to clarify.

The answer could be of many components and the way you unpack the answer to "How did the code change?" could include many parts. The answer may include "who changed the code?", "what parts of the code changed", "what editor was used", "what transformations happened", etc.

"How" can mean different things to different people. The answer to all the questions (the person, parts, editor, transdormations,…) is the ultimate answer to "How did the code change?"

Now, think of the answer to **"Why the code is this way?"**

- what are the parts of the answer?
- what are the components of your answer to the informal definition of rationale "why the code is this way?".
- what are the components of rationale for code commits?



Model of Rationale for Code Commits

Commented [AK8]: This is a model of rationale for code commits that we are studying.

This model consists of rationale components. Every component is described by a question and an example of an answer to the question.

These questions are not for you to answer, it is an explanation of what each component is and the answers are example of a possible answer to the question.

For the next 5 minutes, please review, understand, and think about this model of rationale for code commits. Feel free to ask me for clarifications. You may take longer than 5 minutes if you need to.

3. What do you think of this model?

4. In this model, is there something that you don’t think it should be part of rationale for code commits? Take as much time as you need.

Commented [AK9]: a.What is it?
b.Why?

5. Are there any components of rationale for code commits that should be added to this model? Take as much time as you need.

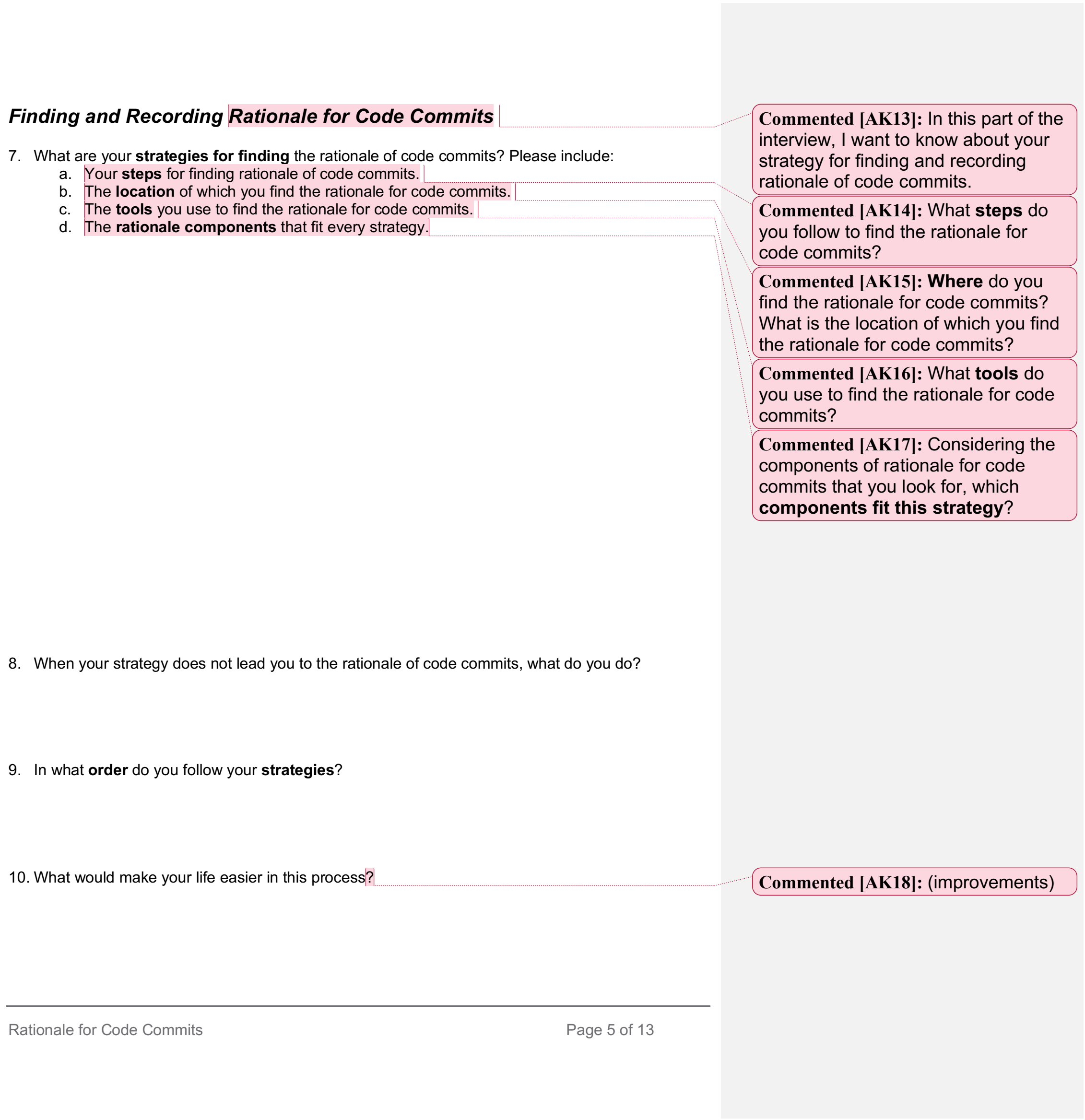
Commented [AK10]: a.What are these components?

Commented [AK11]: Considering these new components [X, Y, Z], Why didn't you include these components the first time?

6. Would you like to change your answer for the way you divide rationale of code commits into components? Take as much time as you need.

Commented [AK12]: Given your changes for the way you divide rationale of code commits

a.Would you like to update the model?
b.Is there something that you don’t think it should be part of rationale for code commits?
c.What are the components lacking in this model?



Finding and Recording Rationale for Code Commits

7. What are your **strategies for finding** the rationale of code commits? Please include:
   a. Your **steps** for finding rationale of code commits.
   b. The **location** of which you find the rationale for code commits.
   c. The **tools** you use to find the rationale for code commits.
   d. The **rationale components** that fit every strategy.

8. When your strategy does not lead you to the rationale of code commits, what do you do?

9. In what **order** do you follow your **strategies**?

10. What would make your life easier in this process?

Commented [AK13]: In this part of the interview, I want to know about your strategy for finding and recording rationale of code commits.

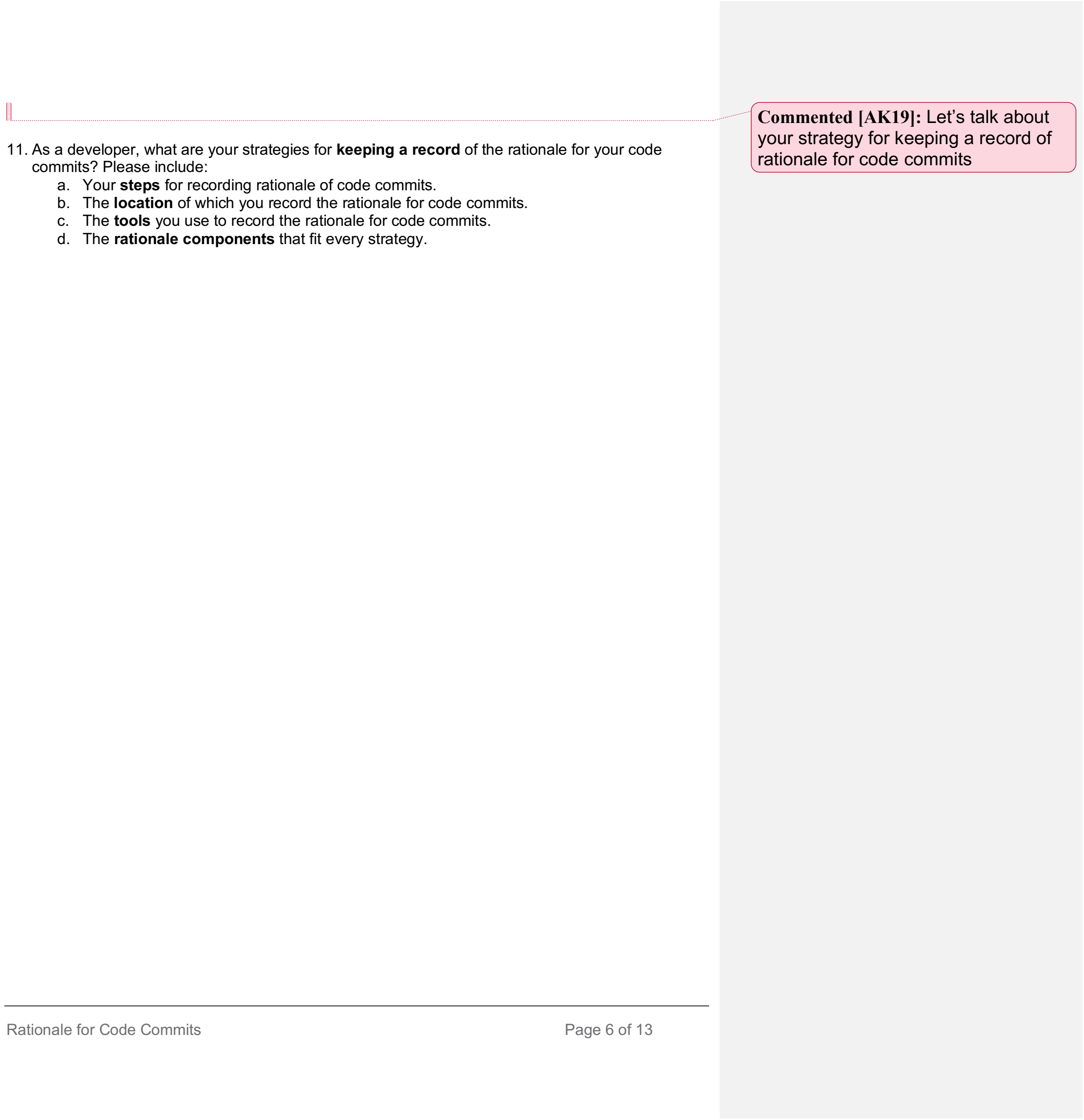
Commented [AK14]: What **steps** do you follow to find the rationale for code commits?

Commented [AK15]: **Where** do you find the rationale for code commits? What is the location of which you find the rationale for code commits?

Commented [AK16]: What **tools** do you use to find the rationale for code commits?

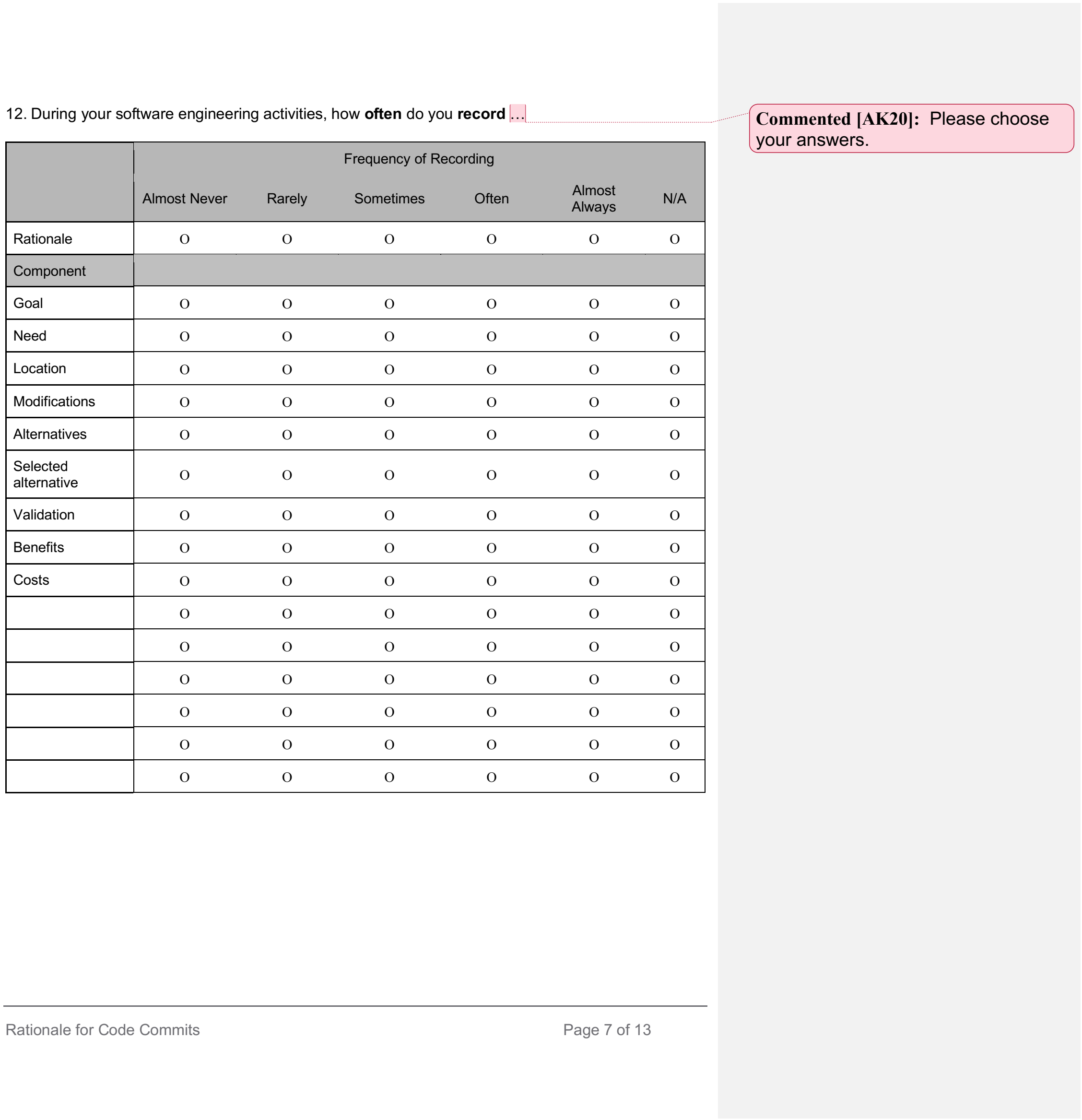
Commented [AK17]: Considering the components of rationale for code commits that you look for, which **components fit this strategy**?

Commented [AK18]: (improvements)



11. As a developer, what are your strategies for **keeping a record** of the rationale for your code commits? Please include:
    a. Your **steps** for recording rationale of code commits.
    b. The **location** of which you record the rationale for code commits.
    c. The **tools** you use to record the rationale for code commits.
    d. The **rationale components** that fit every strategy.

Commented [AK19]: Let's talk about your strategy for keeping a record of rationale for code commits

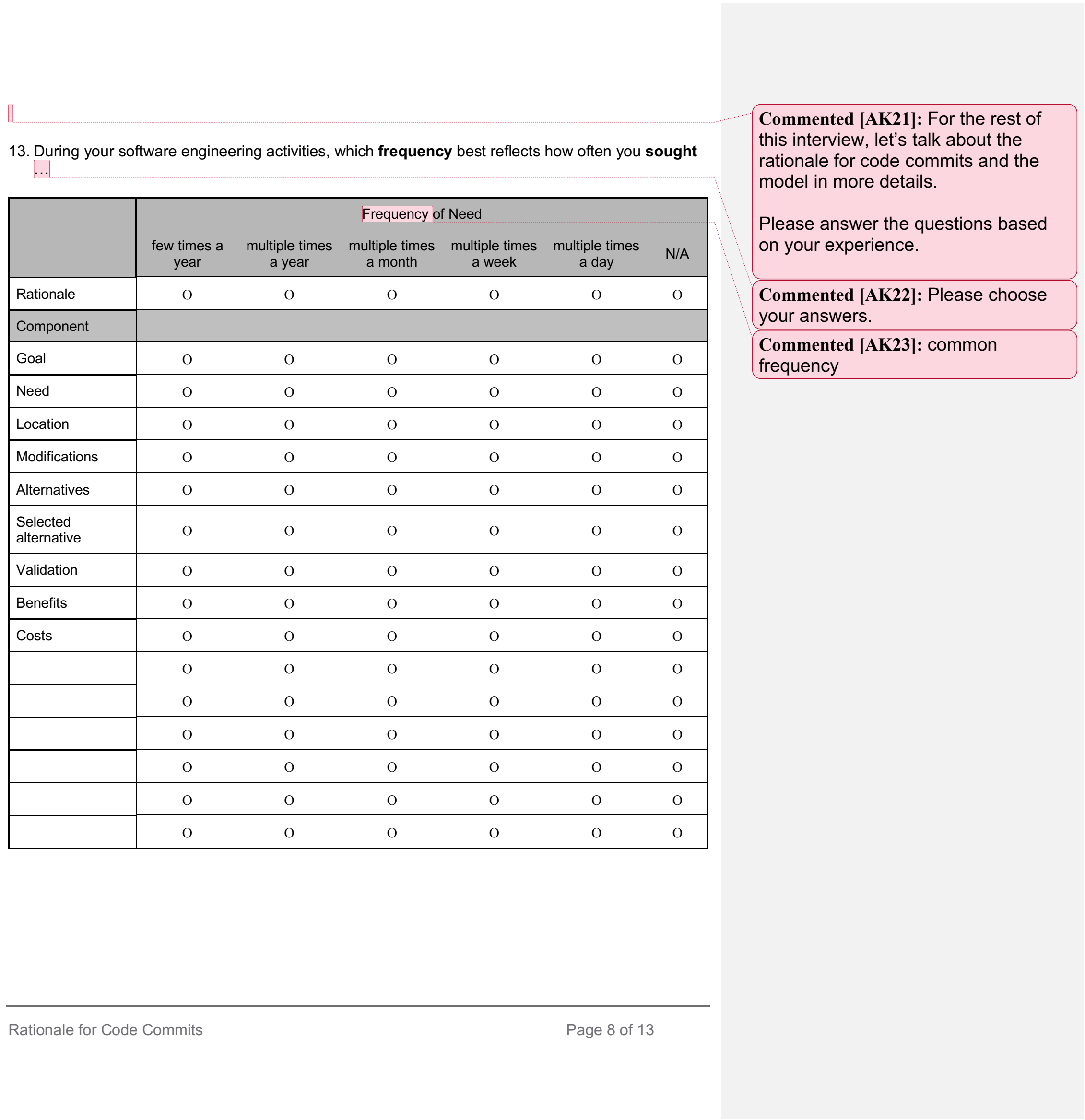


12. During your software engineering activities, how **often** do you **record** …

Commented [AK20]: Please choose your answers.

	Frequency of Recording					
	Almost Never	Rarely	Sometimes	Often	Almost Always	N/A
Rationale	○	○	○	○	○	○
Component						
Goal	○	○	○	○	○	○
Need	○	○	○	○	○	○
Location	○	○	○	○	○	○
Modifications	○	○	○	○	○	○
Alternatives	○	○	○	○	○	○
Selected alternative	○	○	○	○	○	○
Validation	○	○	○	○	○	○
Benefits	○	○	○	○	○	○
Costs	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○

Rationale for Code Commits Page 7 of 13



13. During your software engineering activities, which **frequency** best reflects how often you **sought** …

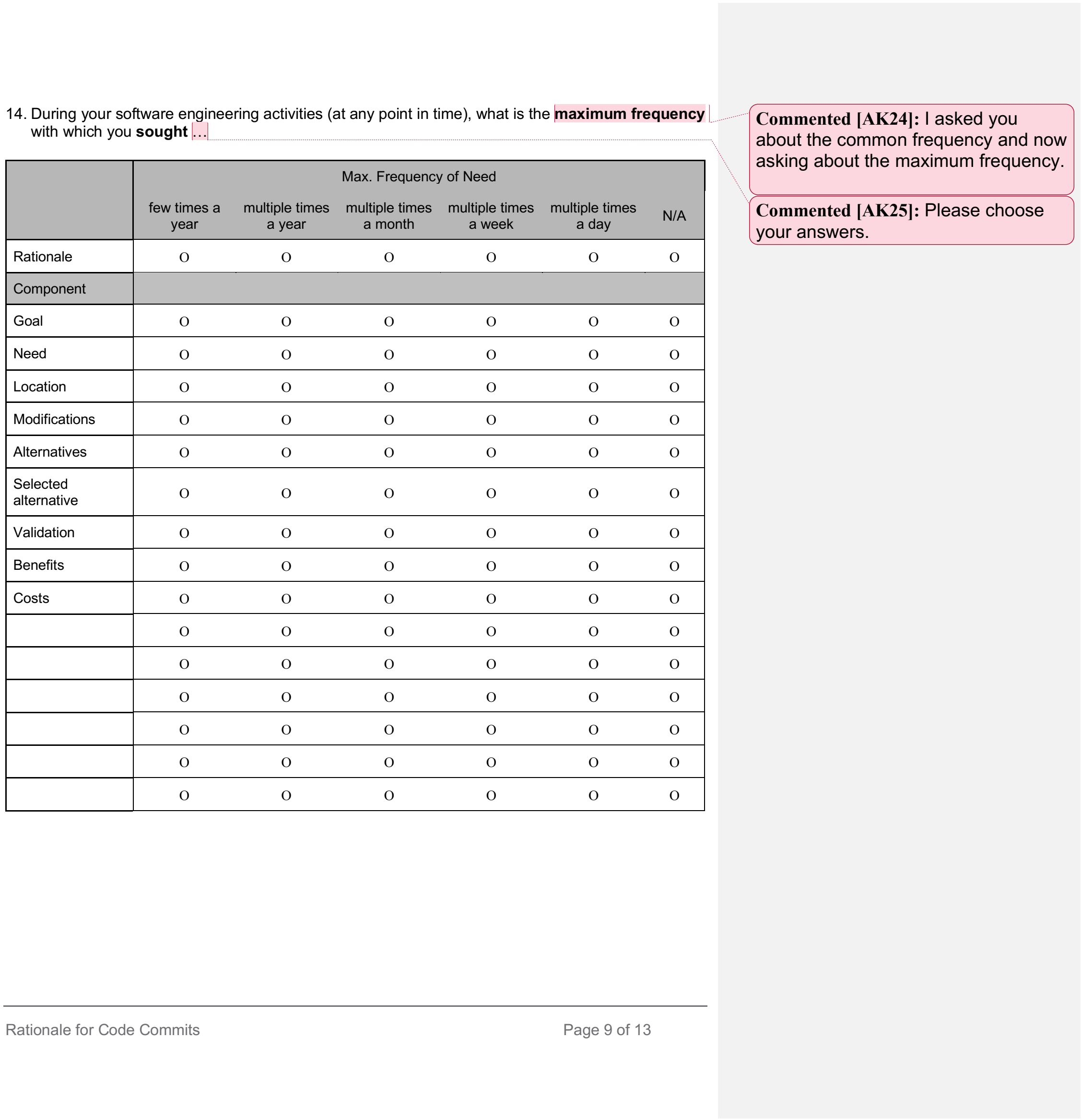
	Frequency of Need					
	few times a year	multiple times a year	multiple times a month	multiple times a week	multiple times a day	N/A
Rationale	o	o	o	o	o	o
Component						
Goal	o	o	o	o	o	o
Need	o	o	o	o	o	o
Location	o	o	o	o	o	o
Modifications	o	o	o	o	o	o
Alternatives	o	o	o	o	o	o
Selected alternative	o	o	o	o	o	o
Validation	o	o	o	o	o	o
Benefits	o	o	o	o	o	o
Costs	o	o	o	o	o	o
	o	o	o	o	o	o
	o	o	o	o	o	o
	o	o	o	o	o	o
	o	o	o	o	o	o
	o	o	o	o	o	o
	o	o	o	o	o	o

Commented [AK21]: For the rest of this interview, let's talk about the rationale for code commits and the model in more details.

Please answer the questions based on your experience.

Commented [AK22]: Please choose your answers.

Commented [AK23]: common frequency

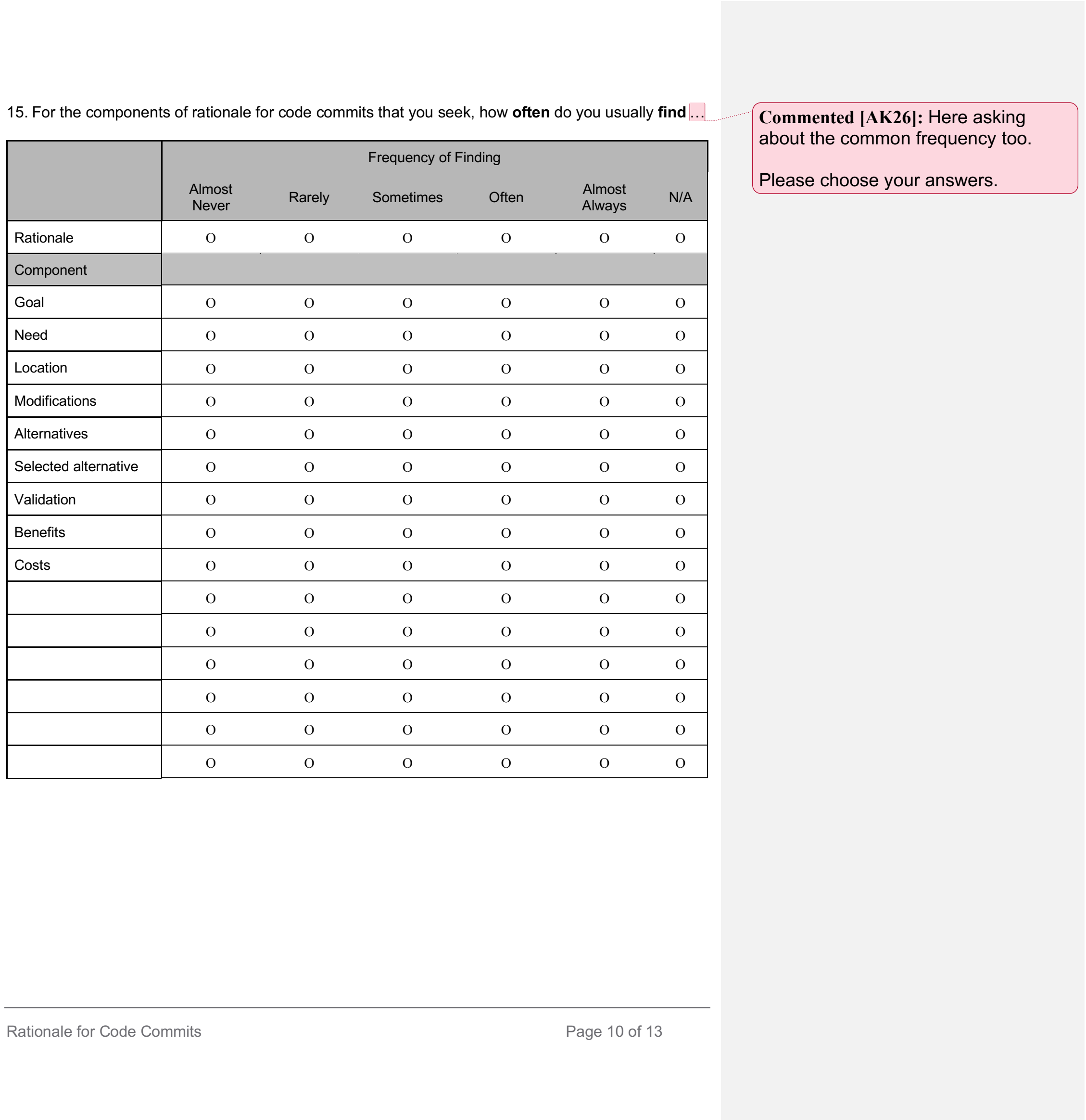


14. During your software engineering activities (at any point in time), what is the **maximum frequency** with which you **sought** …

	Max. Frequency of Need					
	few times a year	multiple times a year	multiple times a month	multiple times a week	multiple times a day	N/A
Rationale	○	○	○	○	○	○
Component						
Goal	○	○	○	○	○	○
Need	○	○	○	○	○	○
Location	○	○	○	○	○	○
Modifications	○	○	○	○	○	○
Alternatives	○	○	○	○	○	○
Selected alternative	○	○	○	○	○	○
Validation	○	○	○	○	○	○
Benefits	○	○	○	○	○	○
Costs	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○

Commented [AK24]: I asked you about the common frequency and now asking about the maximum frequency.

Commented [AK25]: Please choose your answers.

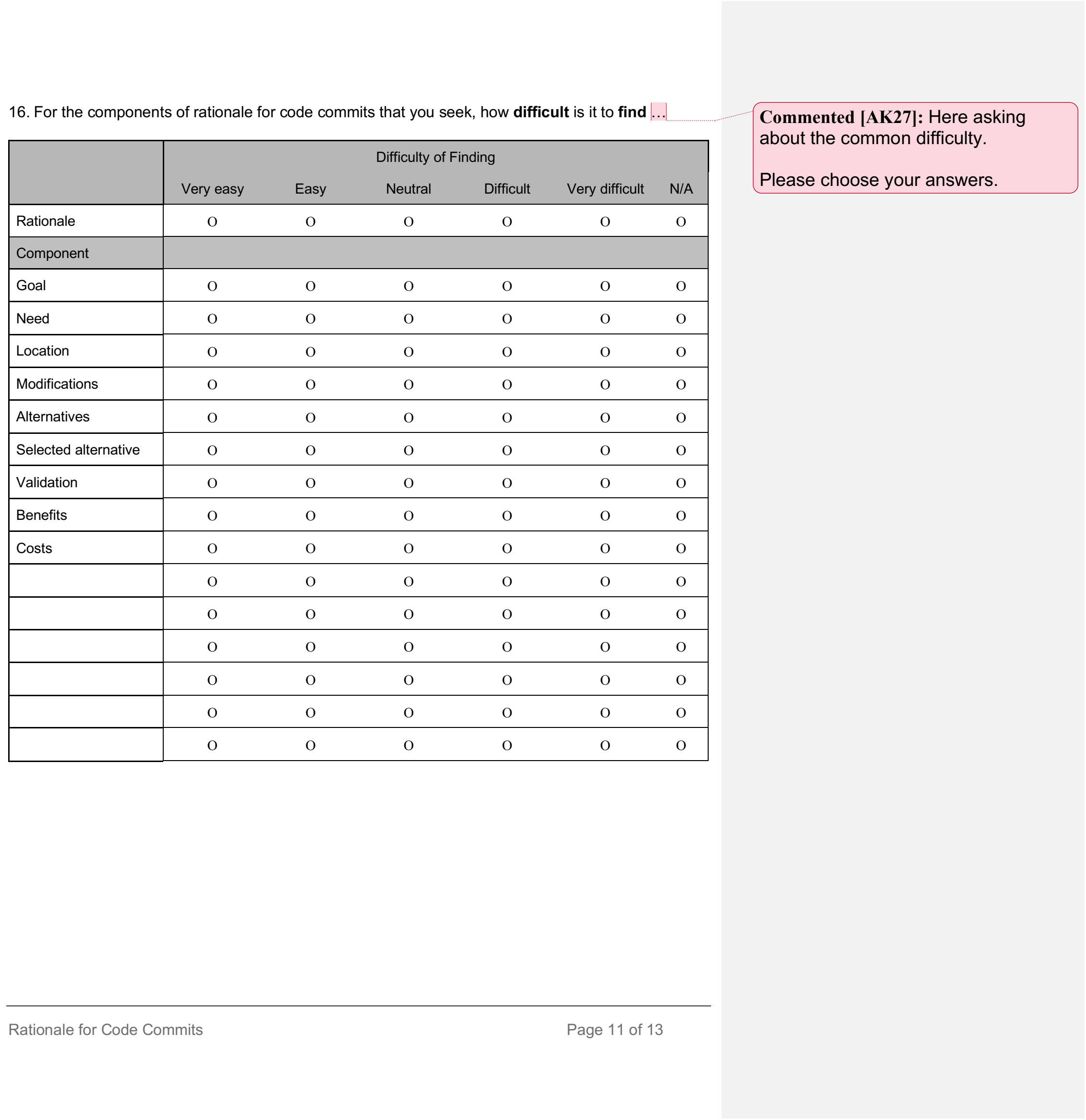


15. For the components of rationale for code commits that you seek, how **often** do you usually **find** …

Commented [AK26]: Here asking about the common frequency too.

Please choose your answers.

	Frequency of Finding					
	Almost Never	Rarely	Sometimes	Often	Almost Always	N/A
Rationale	○	○	○	○	○	○
Component						
Goal	○	○	○	○	○	○
Need	○	○	○	○	○	○
Location	○	○	○	○	○	○
Modifications	○	○	○	○	○	○
Alternatives	○	○	○	○	○	○
Selected alternative	○	○	○	○	○	○
Validation	○	○	○	○	○	○
Benefits	○	○	○	○	○	○
Costs	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○

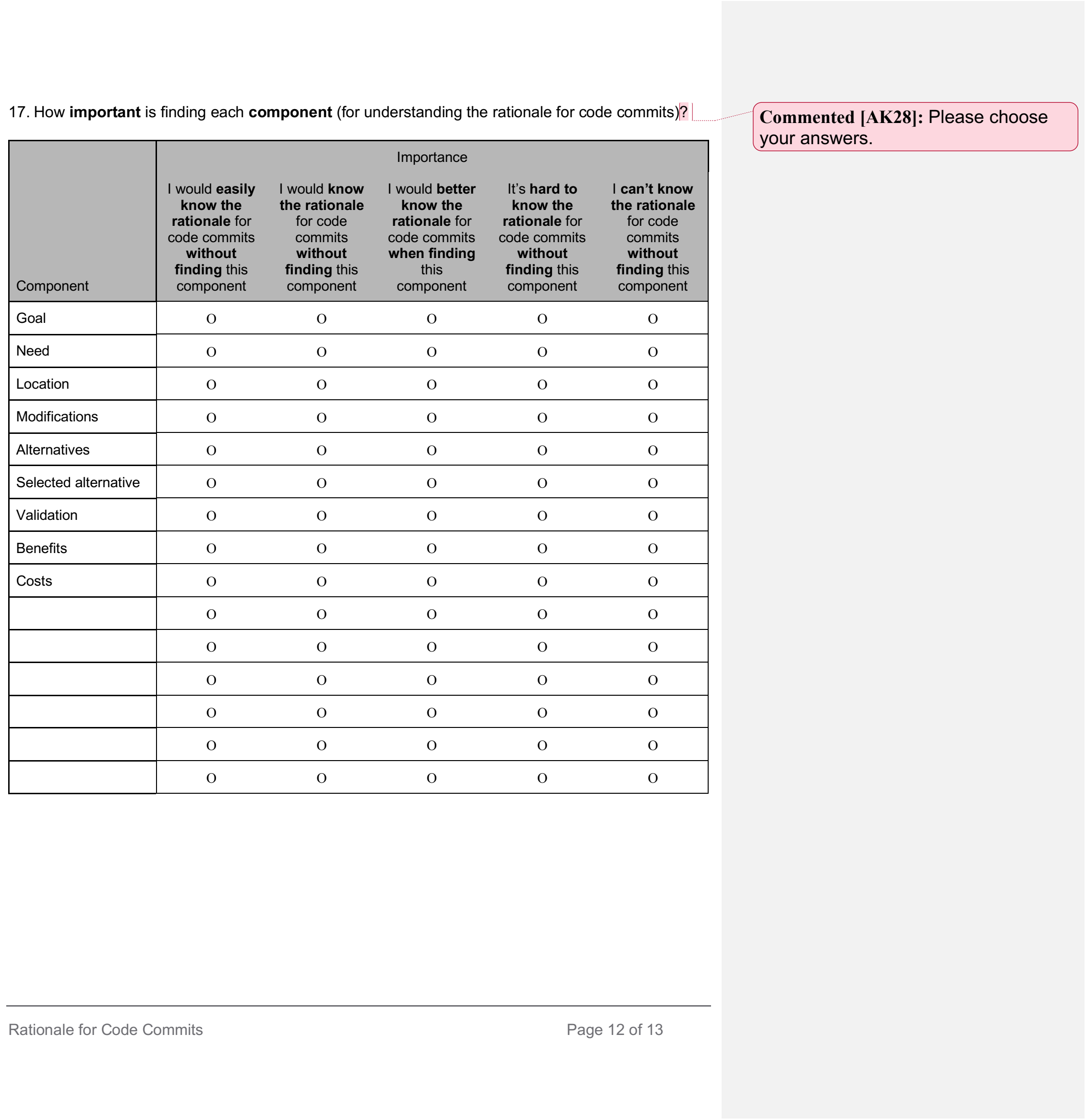


16. For the components of rationale for code commits that you seek, how **difficult** is it to **find** …

	Difficulty of Finding					
	Very easy	Easy	Neutral	Difficult	Very difficult	N/A
Rationale	○	○	○	○	○	○
Component						
Goal	○	○	○	○	○	○
Need	○	○	○	○	○	○
Location	○	○	○	○	○	○
Modifications	○	○	○	○	○	○
Alternatives	○	○	○	○	○	○
Selected alternative	○	○	○	○	○	○
Validation	○	○	○	○	○	○
Benefits	○	○	○	○	○	○
Costs	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○
	○	○	○	○	○	○

Commented [AK27]: Here asking about the common difficulty.

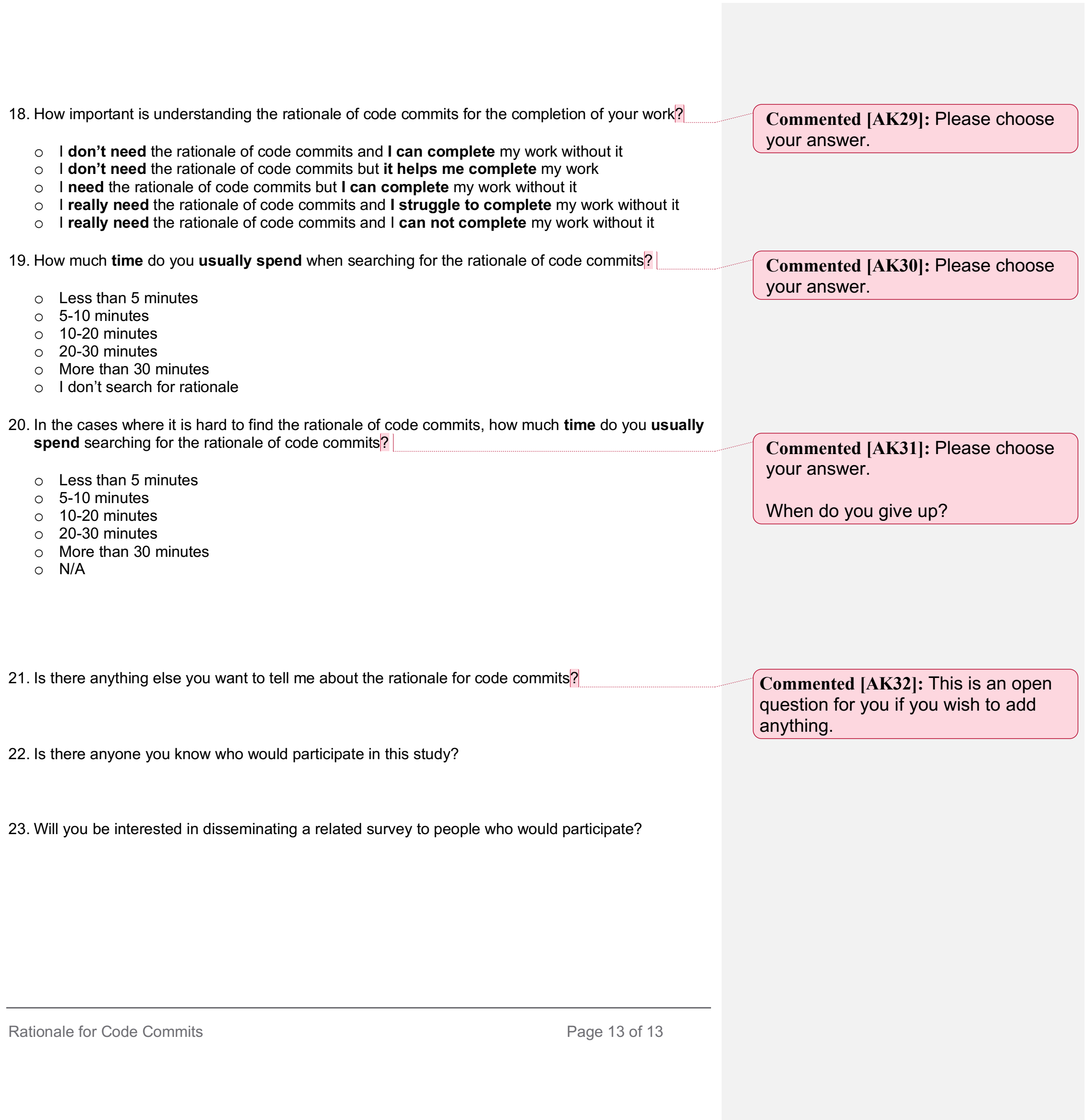
Please choose your answers.



17. How **important** is finding each **component** (for understanding the rationale for code commits)?

Commented [AK28]: Please choose your answers.

	Importance				
Component	I would **easily know the rationale** for code commits **without finding** this component	I would **know the rationale** for code commits **without finding** this component	I would **better know the rationale** for code commits **when finding** this component	It's **hard to know the rationale** for code commits **without finding** this component	I **can't know the rationale** for code commits **without finding** this component
Goal	○	○	○	○	○
Need	○	○	○	○	○
Location	○	○	○	○	○
Modifications	○	○	○	○	○
Alternatives	○	○	○	○	○
Selected alternative	○	○	○	○	○
Validation	○	○	○	○	○
Benefits	○	○	○	○	○
Costs	○	○	○	○	○
	○	○	○	○	○
	○	○	○	○	○
	○	○	○	○	○
	○	○	○	○	○
	○	○	○	○	○
	○	○	○	○	○



18. How important is understanding the rationale of code commits for the completion of your work?

○ I **don't need** the rationale of code commits and **I can complete** my work without it
○ I **don't need** the rationale of code commits but **it helps me complete** my work
○ I **need** the rationale of code commits but **I can complete** my work without it
○ I **really need** the rationale of code commits and **I struggle to complete** my work without it
○ I **really need** the rationale of code commits and I **can not complete** my work without it

Commented [AK29]: Please choose your answer.

19. How much **time** do you **usually spend** when searching for the rationale of code commits?

○ Less than 5 minutes
○ 5-10 minutes
○ 10-20 minutes
○ 20-30 minutes
○ More than 30 minutes
○ I don't search for rationale

Commented [AK30]: Please choose your answer.

20. In the cases where it is hard to find the rationale of code commits, how much **time** do you **usually spend** searching for the rationale of code commits?

○ Less than 5 minutes
○ 5-10 minutes
○ 10-20 minutes
○ 20-30 minutes
○ More than 30 minutes
○ N/A

Commented [AK31]: Please choose your answer.

When do you give up?

21. Is there anything else you want to tell me about the rationale for code commits?

Commented [AK32]: This is an open question for you if you wish to add anything.

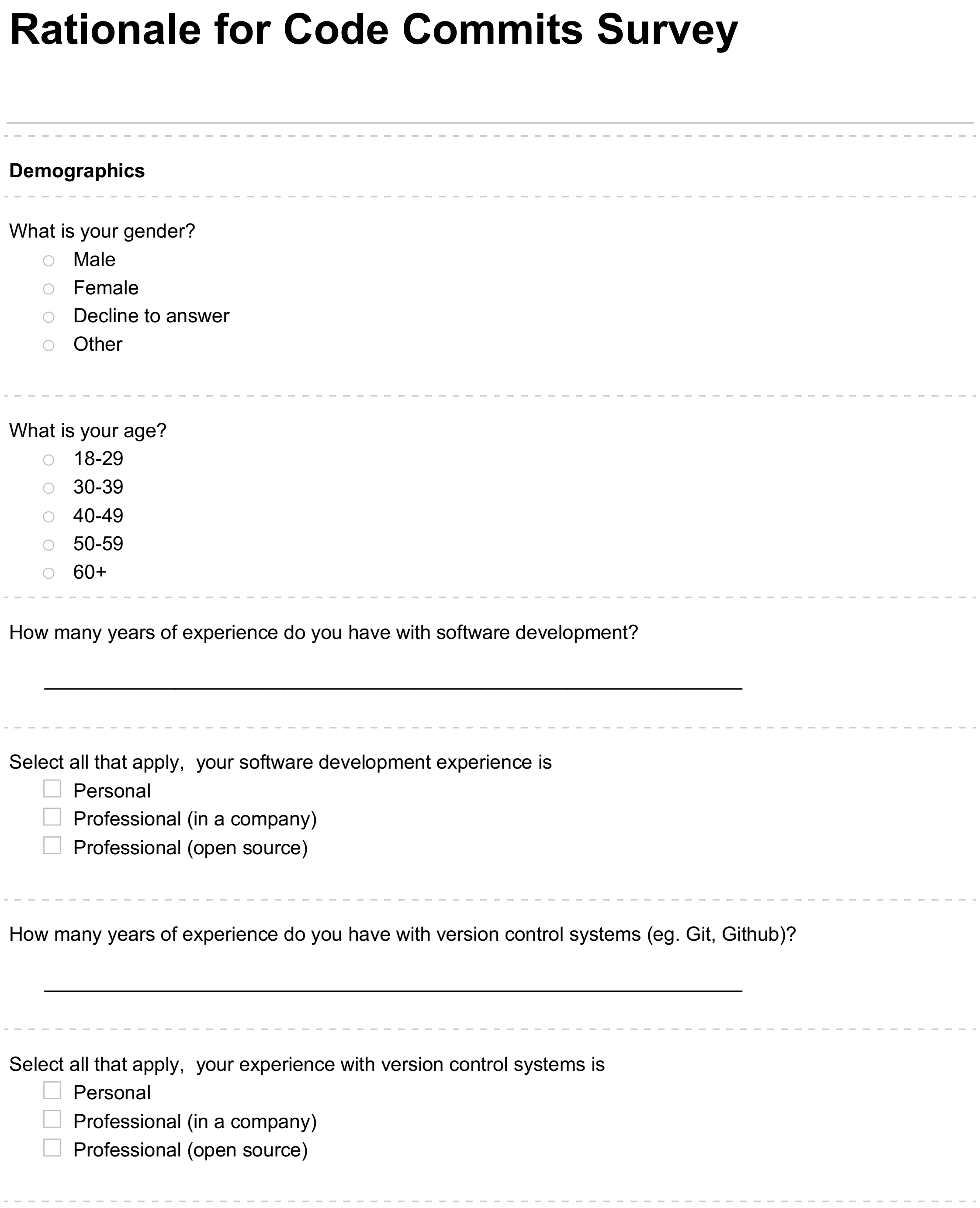
22. Is there anyone you know who would participate in this study?

23. Will you be interested in disseminating a related survey to people who would participate?

Rationale for Code Commits Page 13 of 13

Appendix B

Online Survey Questions



Rationale for Code Commits Survey

Demographics

What is your gender?

○ Male
○ Female
○ Decline to answer
○ Other

What is your age?

○ 18-29
○ 30-39
○ 40-49
○ 50-59
○ 60+

How many years of experience do you have with software development?

____

Select all that apply, your software development experience is

□ Personal
□ Professional (in a company)
□ Professional (open source)

How many years of experience do you have with version control systems (eg. Git, Github)?

____

Select all that apply, your experience with version control systems is

□ Personal
□ Professional (in a company)
□ Professional (open source)

What version control systems have you used for software development?

□ CVS
□ Git
□ Mercurial
□ Perforce
□ Subversion
□ Other (please specify) ____

Rationale for Code Commits

Modern software is modified in packaged changes called **code commits** in revision control systems. In many occasions, software developers need to **examine code commits** for various purposes. Some of the main motivations for examining code commits are:

- Debugging
- Reverse engineering requirements
- **Understanding rationale (Why is the code this way?)**
- ...

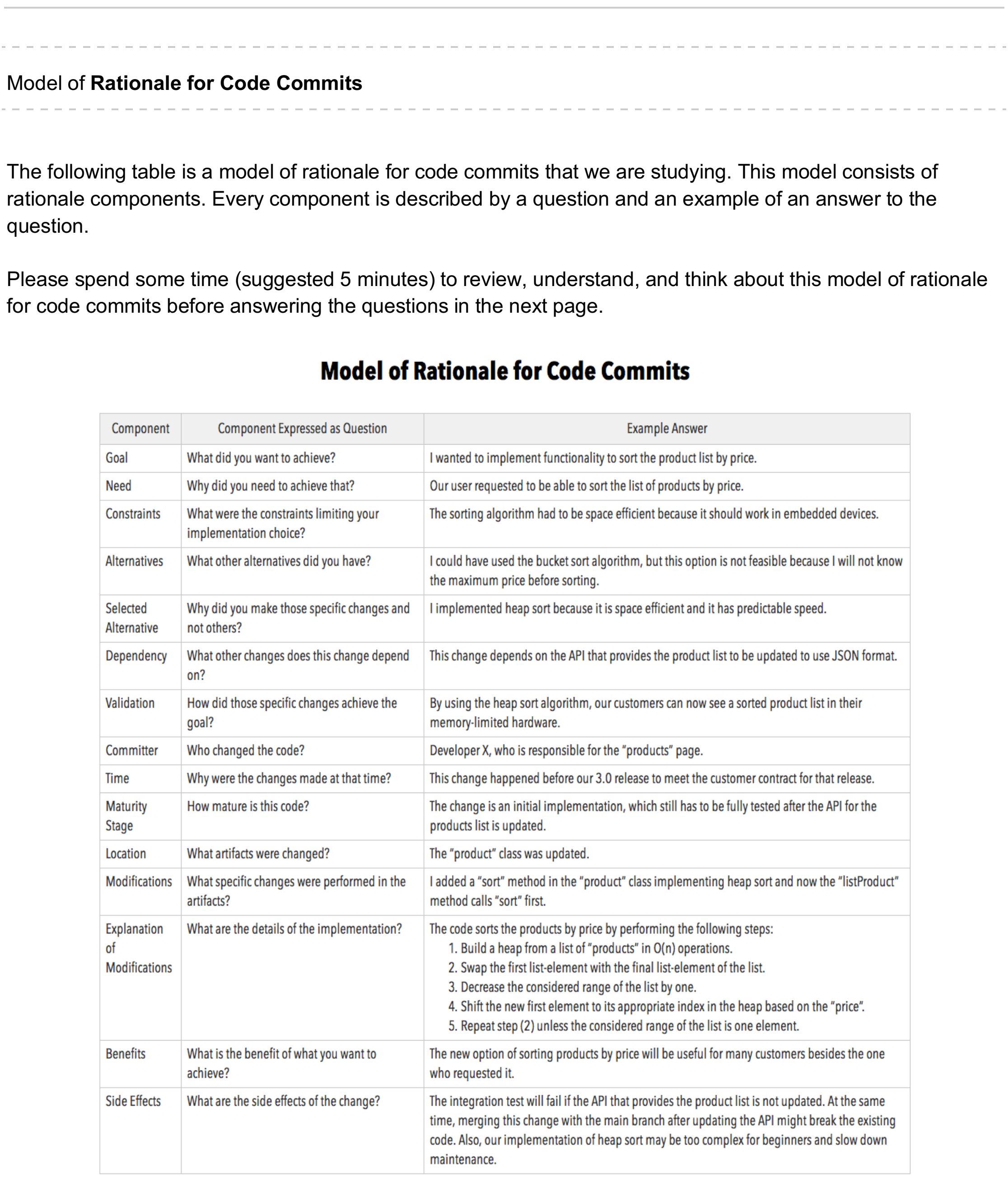
In this study, we want to understand developers' experiences with rationale for code commits and its components.

During your software engineering activities, how often do you **inspect your own code commits to understand their rationale (Why the code is this way)**?

○ A few times per year
○ Multiple times per year
○ Multiple times per month
○ Multiple times per week
○ Multiple times per day

During your software engineering activities, how often do you **inspect other developers' code commits to understand their rationale (Why the code is this way)**?

○ A few times per year
○ Multiple times per year
○ Multiple times per month
○ Multiple times per week
○ Multiple times per day



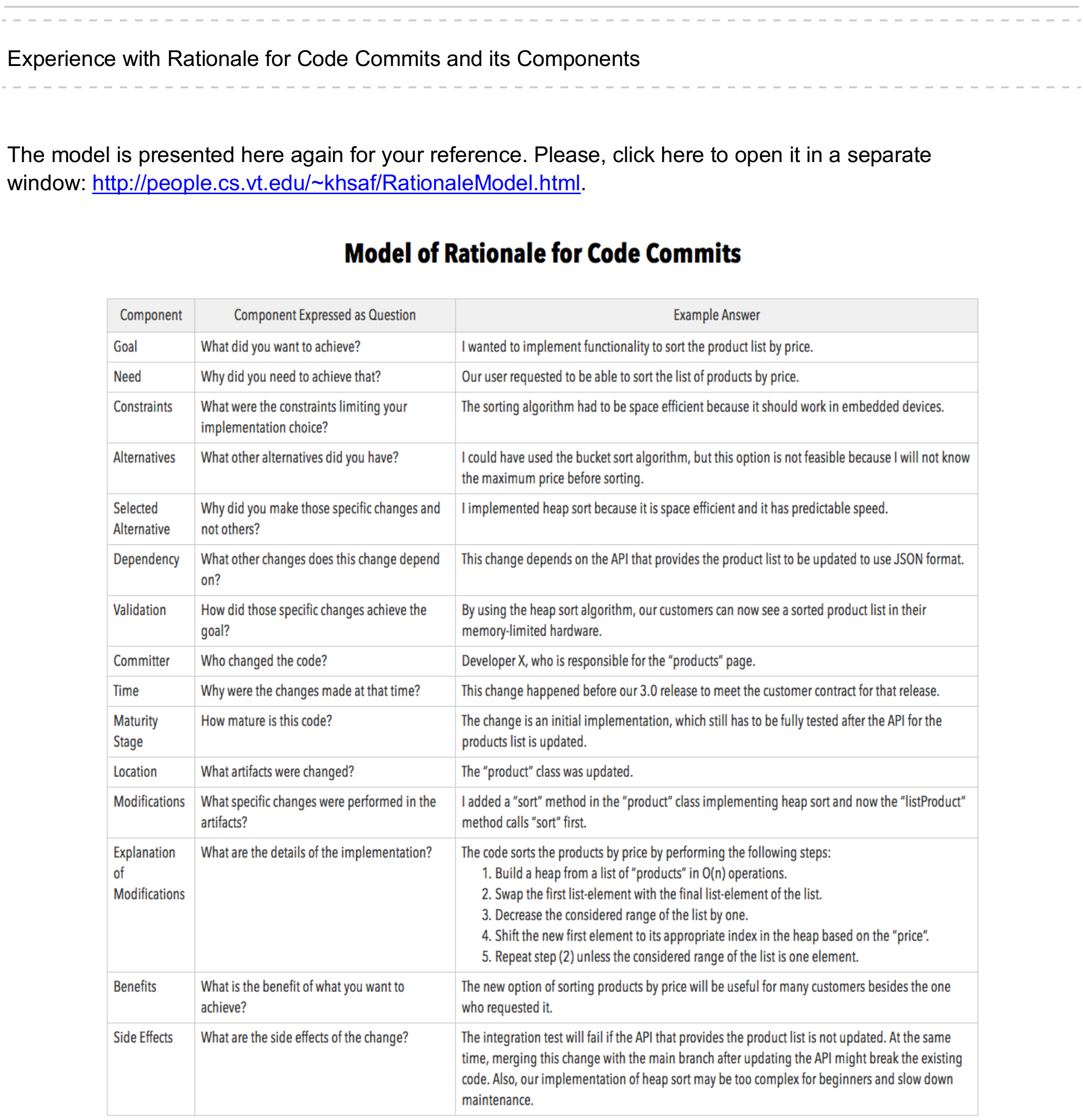
Model of **Rationale for Code Commits**

The following table is a model of rationale for code commits that we are studying. This model consists of rationale components. Every component is described by a question and an example of an answer to the question.

Please spend some time (suggested 5 minutes) to review, understand, and think about this model of rationale for code commits before answering the questions in the next page.

Model of Rationale for Code Commits

Component	Component Expressed as Question	Example Answer
Goal	What did you want to achieve?	I wanted to implement functionality to sort the product list by price.
Need	Why did you need to achieve that?	Our user requested to be able to sort the list of products by price.
Constraints	What were the constraints limiting your implementation choice?	The sorting algorithm had to be space efficient because it should work in embedded devices.
Alternatives	What other alternatives did you have?	I could have used the bucket sort algorithm, but this option is not feasible because I will not know the maximum price before sorting.
Selected Alternative	Why did you make those specific changes and not others?	I implemented heap sort because it is space efficient and it has predictable speed.
Dependency	What other changes does this change depend on?	This change depends on the API that provides the product list to be updated to use JSON format.
Validation	How did those specific changes achieve the goal?	By using the heap sort algorithm, our customers can now see a sorted product list in their memory-limited hardware.
Committer	Who changed the code?	Developer X, who is responsible for the "products" page.
Time	Why were the changes made at that time?	This change happened before our 3.0 release to meet the customer contract for that release.
Maturity Stage	How mature is this code?	The change is an initial implementation, which still has to be fully tested after the API for the products list is updated.
Location	What artifacts were changed?	The "product" class was updated.
Modifications	What specific changes were performed in the artifacts?	I added a "sort" method in the "product" class implementing heap sort and now the "listProduct" method calls "sort" first.
Explanation of Modifications	What are the details of the implementation?	The code sorts the products by price by performing the following steps: 1. Build a heap from a list of "products" in O(n) operations. 2. Swap the first list-element with the final list-element of the list. 3. Decrease the considered range of the list by one. 4. Shift the new first element to its appropriate index in the heap based on the "price". 5. Repeat step (2) unless the considered range of the list is one element.
Benefits	What is the benefit of what you want to achieve?	The new option of sorting products by price will be useful for many customers besides the one who requested it.
Side Effects	What are the side effects of the change?	The integration test will fail if the API that provides the product list is not updated. At the same time, merging this change with the main branch after updating the API might break the existing code. Also, our implementation of heap sort may be too complex for beginners and slow down maintenance.

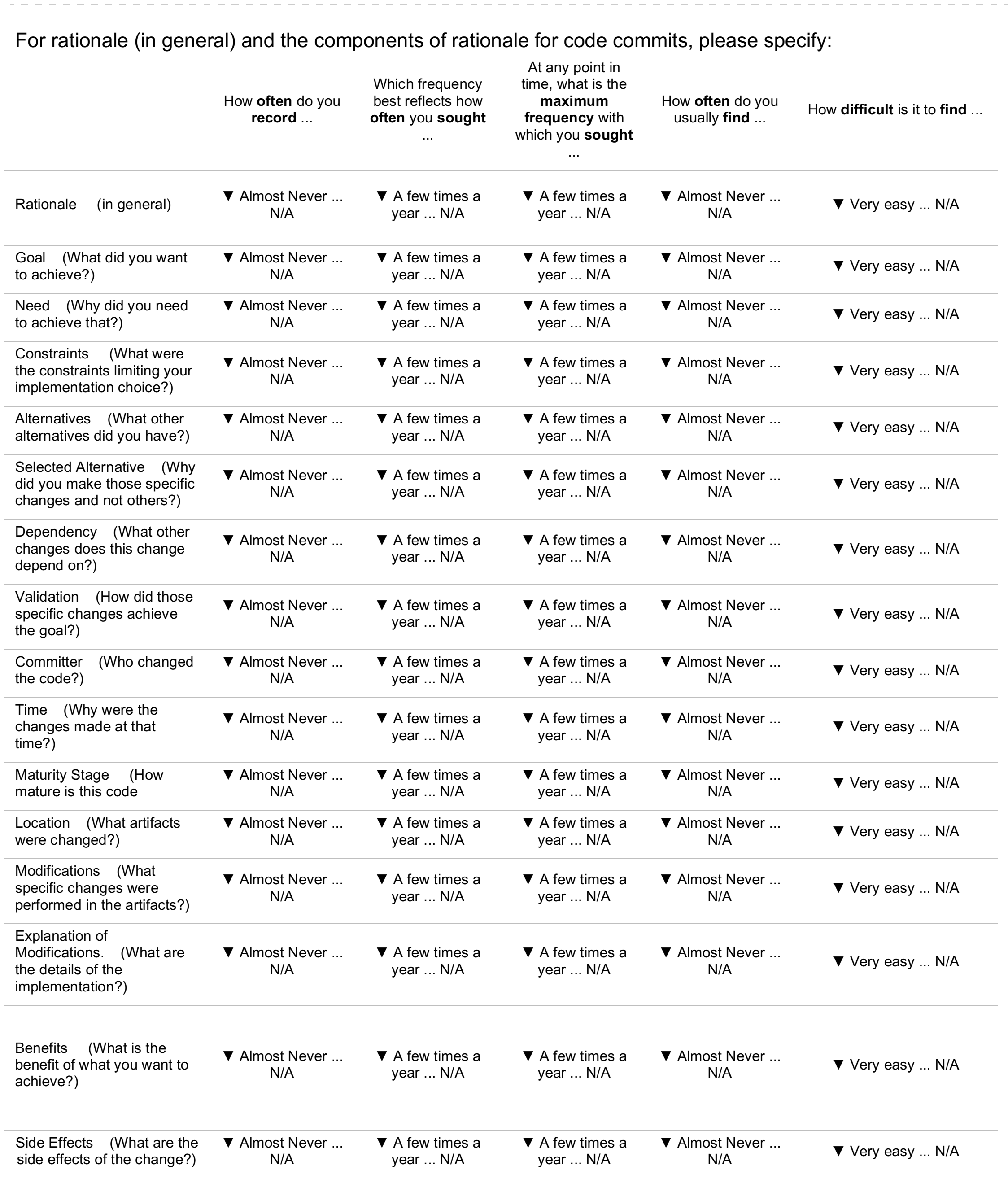


Experience with Rationale for Code Commits and its Components

The model is presented here again for your reference. Please, click here to open it in a separate window: http://people.cs.vt.edu/~khsaf/RationaleModel.html.

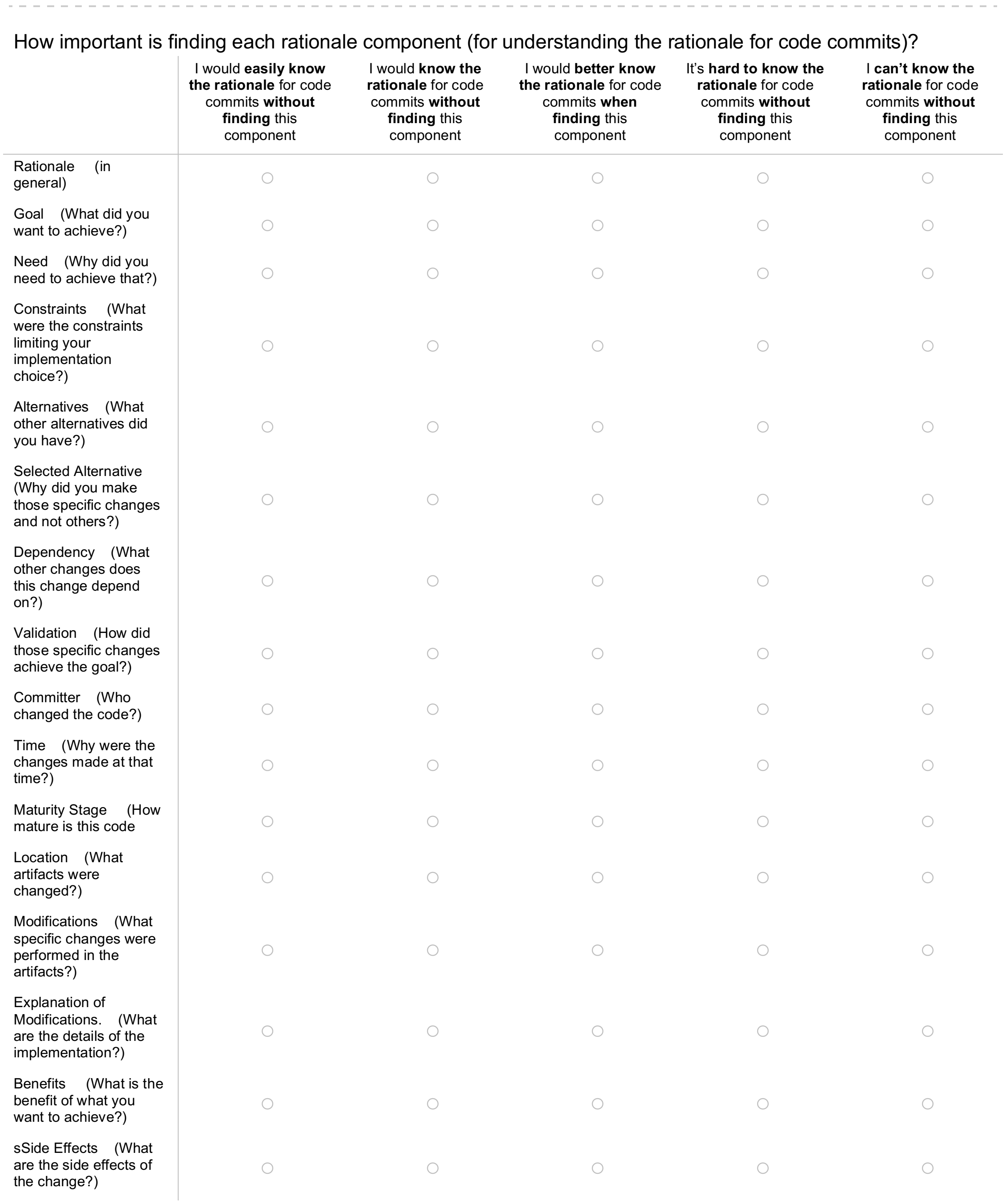
Model of Rationale for Code Commits

Component	Component Expressed as Question	Example Answer
Goal	What did you want to achieve?	I wanted to implement functionality to sort the product list by price.
Need	Why did you need to achieve that?	Our user requested to be able to sort the list of products by price.
Constraints	What were the constraints limiting your implementation choice?	The sorting algorithm had to be space efficient because it should work in embedded devices.
Alternatives	What other alternatives did you have?	I could have used the bucket sort algorithm, but this option is not feasible because I will not know the maximum price before sorting.
Selected Alternative	Why did you make those specific changes and not others?	I implemented heap sort because it is space efficient and it has predictable speed.
Dependency	What other changes does this change depend on?	This change depends on the API that provides the product list to be updated to use JSON format.
Validation	How did those specific changes achieve the goal?	By using the heap sort algorithm, our customers can now see a sorted product list in their memory-limited hardware.
Committer	Who changed the code?	Developer X, who is responsible for the "products" page.
Time	Why were the changes made at that time?	This change happened before our 3.0 release to meet the customer contract for that release.
Maturity Stage	How mature is this code?	The change is an initial implementation, which still has to be fully tested after the API for the products list is updated.
Location	What artifacts were changed?	The "product" class was updated.
Modifications	What specific changes were performed in the artifacts?	I added a "sort" method in the "product" class implementing heap sort and now the "listProduct" method calls "sort" first.
Explanation of Modifications	What are the details of the implementation?	The code sorts the products by price by performing the following steps: 1. Build a heap from a list of "products" in O(n) operations. 2. Swap the first list-element with the final list-element of the list. 3. Decrease the considered range of the list by one. 4. Shift the new first element to its appropriate index in the heap based on the "price". 5. Repeat step (2) unless the considered range of the list is one element.
Benefits	What is the benefit of what you want to achieve?	The new option of sorting products by price will be useful for many customers besides the one who requested it.
Side Effects	What are the side effects of the change?	The integration test will fail if the API that provides the product list is not updated. At the same time, merging this change with the main branch after updating the API might break the existing code. Also, our implementation of heap sort may be too complex for beginners and slow down maintenance.



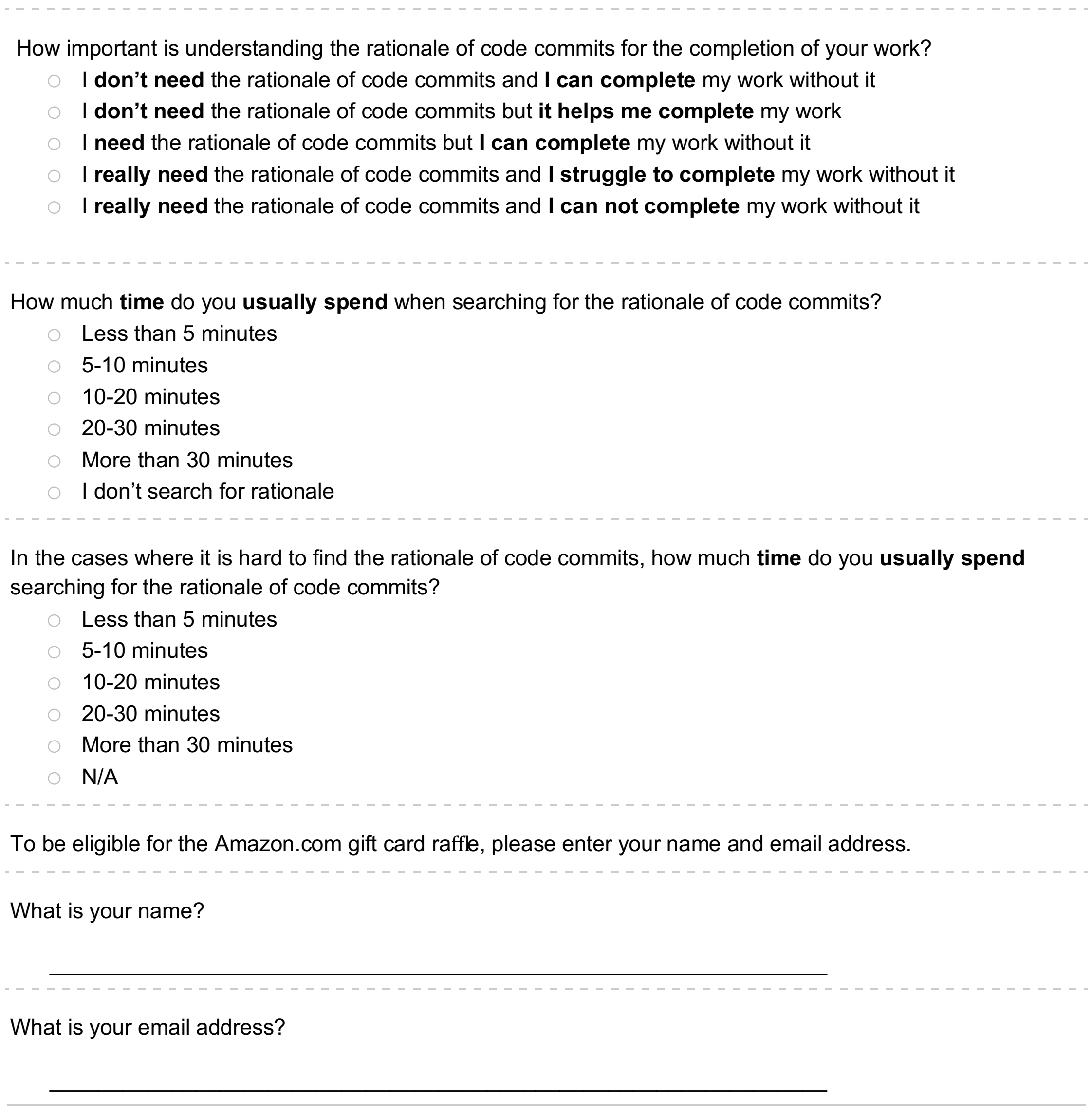
For rationale (in general) and the components of rationale for code commits, please specify:

	How **often** do you **record** ...	Which frequency best reflects how **often** you **sought** ...	At any point in time, what is the **maximum frequency** with which you **sought** ...	How **often** do you usually **find** ...	How **difficult** is it to **find** ...
Rationale (in general)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Goal (What did you want to achieve?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Need (Why did you need to achieve that?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Constraints (What were the constraints limiting your implementation choice?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Alternatives (What other alternatives did you have?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Selected Alternative (Why did you make those specific changes and not others?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Dependency (What other changes does this change depend on?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Validation (How did those specific changes achieve the goal?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Committer (Who changed the code?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Time (Why were the changes made at that time?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Maturity Stage (How mature is this code	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Location (What artifacts were changed?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Modifications (What specific changes were performed in the artifacts?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Explanation of Modifications. (What are the details of the implementation?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Benefits (What is the benefit of what you want to achieve?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A
Side Effects (What are the side effects of the change?)	▼ Almost Never ... N/A	▼ A few times a year ... N/A	▼ A few times a year ... N/A	▼ Almost Never ... N/A	▼ Very easy ... N/A



How important is finding each rationale component (for understanding the rationale for code commits)?

	I would **easily know the rationale** for code commits **without finding** this component	I would **know the rationale** for code commits **without finding** this component	I would **better know the rationale** for code commits **when finding** this component	It's **hard to know the rationale** for code commits **without finding** this component	I **can't know the rationale** for code commits **without finding** this component
Rationale (in general)	○	○	○	○	○
Goal (What did you want to achieve?)	○	○	○	○	○
Need (Why did you need to achieve that?)	○	○	○	○	○
Constraints (What were the constraints limiting your implementation choice?)	○	○	○	○	○
Alternatives (What other alternatives did you have?)	○	○	○	○	○
Selected Alternative (Why did you make those specific changes and not others?)	○	○	○	○	○
Dependency (What other changes does this change depend on?)	○	○	○	○	○
Validation (How did those specific changes achieve the goal?)	○	○	○	○	○
Committer (Who changed the code?)	○	○	○	○	○
Time (Why were the changes made at that time?)	○	○	○	○	○
Maturity Stage (How mature is this code	○	○	○	○	○
Location (What artifacts were changed?)	○	○	○	○	○
Modifications (What specific changes were performed in the artifacts?)	○	○	○	○	○
Explanation of Modifications. (What are the details of the implementation?)	○	○	○	○	○
Benefits (What is the benefit of what you want to achieve?)	○	○	○	○	○
sSide Effects (What are the side effects of the change?)	○	○	○	○	○



How important is understanding the rationale of code commits for the completion of your work?

○ I **don't need** the rationale of code commits and **I can complete** my work without it
○ I **don't need** the rationale of code commits but **it helps me complete** my work
○ I **need** the rationale of code commits but **I can complete** my work without it
○ I **really need** the rationale of code commits and **I struggle to complete** my work without it
○ I **really need** the rationale of code commits and **I can not complete** my work without it

How much **time** do you **usually spend** when searching for the rationale of code commits?

○ Less than 5 minutes
○ 5-10 minutes
○ 10-20 minutes
○ 20-30 minutes
○ More than 30 minutes
○ I don't search for rationale

In the cases where it is hard to find the rationale of code commits, how much **time** do you **usually spend** searching for the rationale of code commits?

○ Less than 5 minutes
○ 5-10 minutes
○ 10-20 minutes
○ 20-30 minutes
○ More than 30 minutes
○ N/A

To be eligible for the Amazon.com gift card raffle, please enter your name and email address.

What is your name?

____

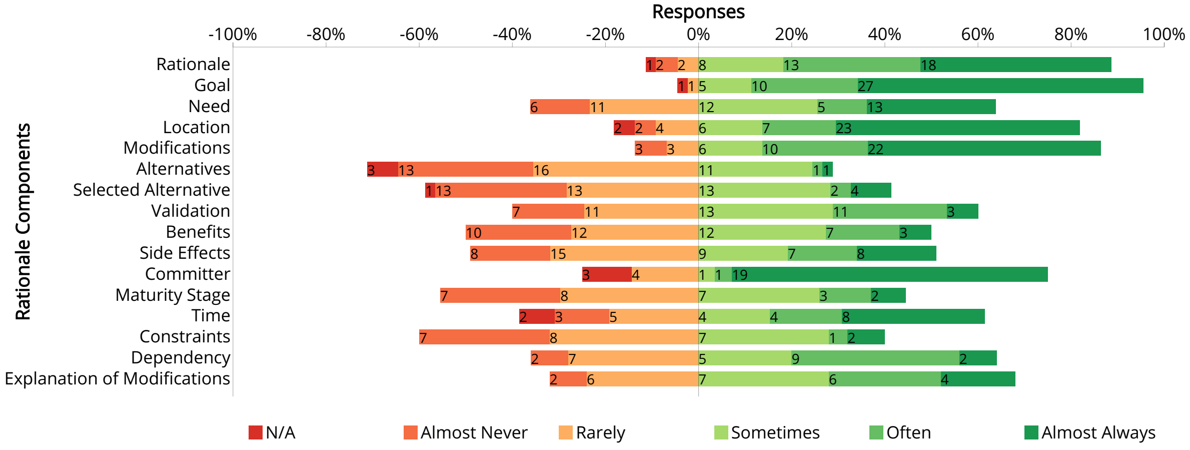
What is your email address?

____

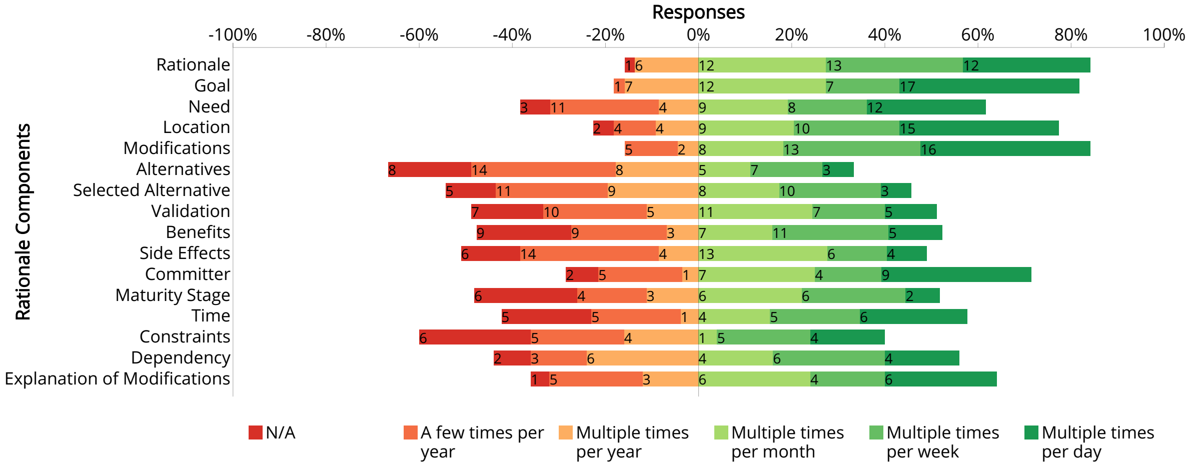
Appendix C

Quantitative Questions Report

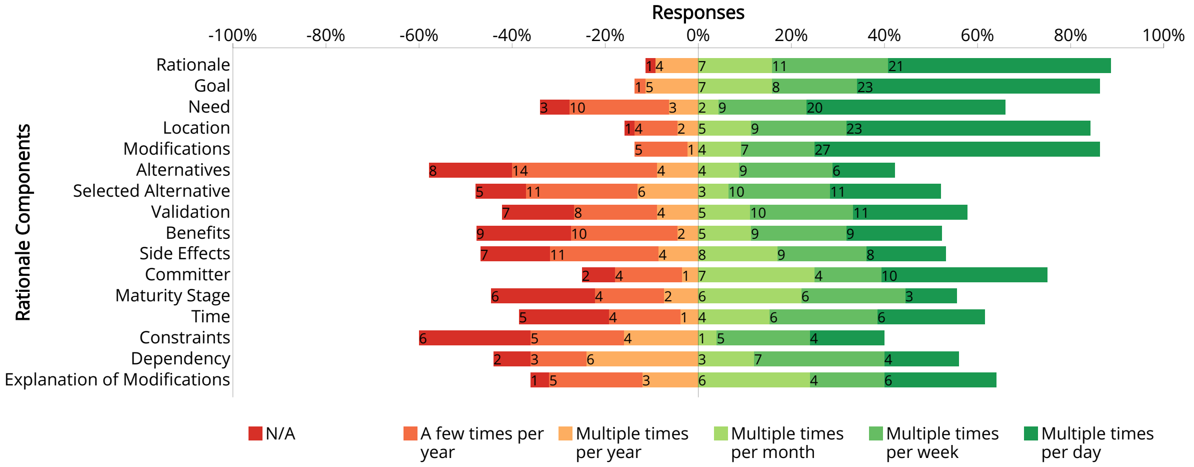
Rationale Components Frequency of Recording



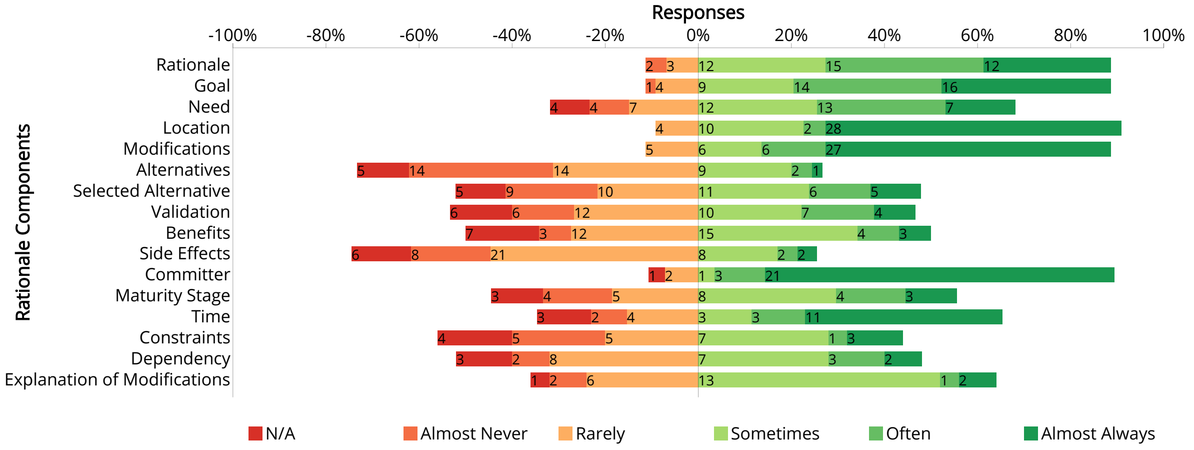
Rationale Components Frequency of Need

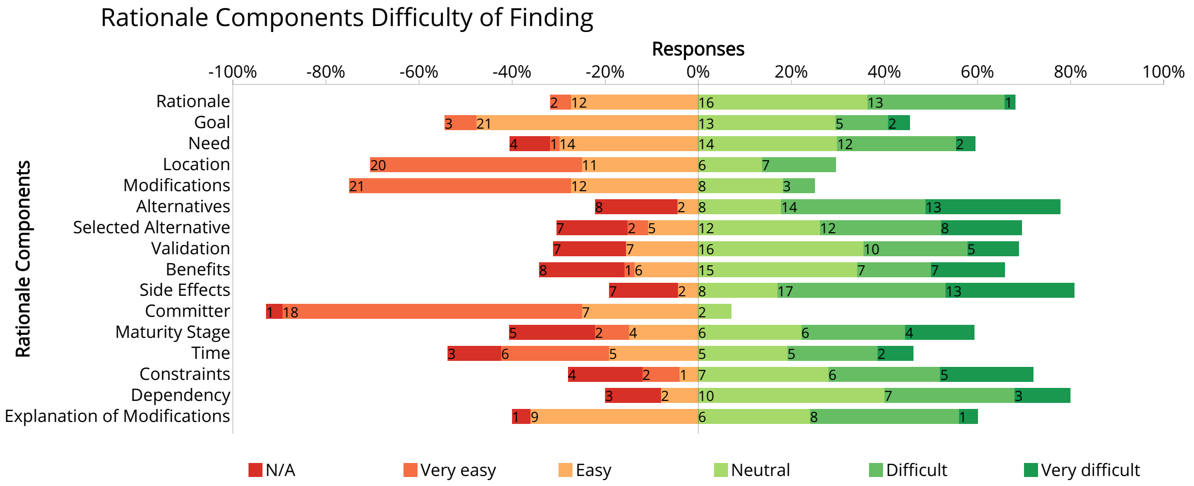


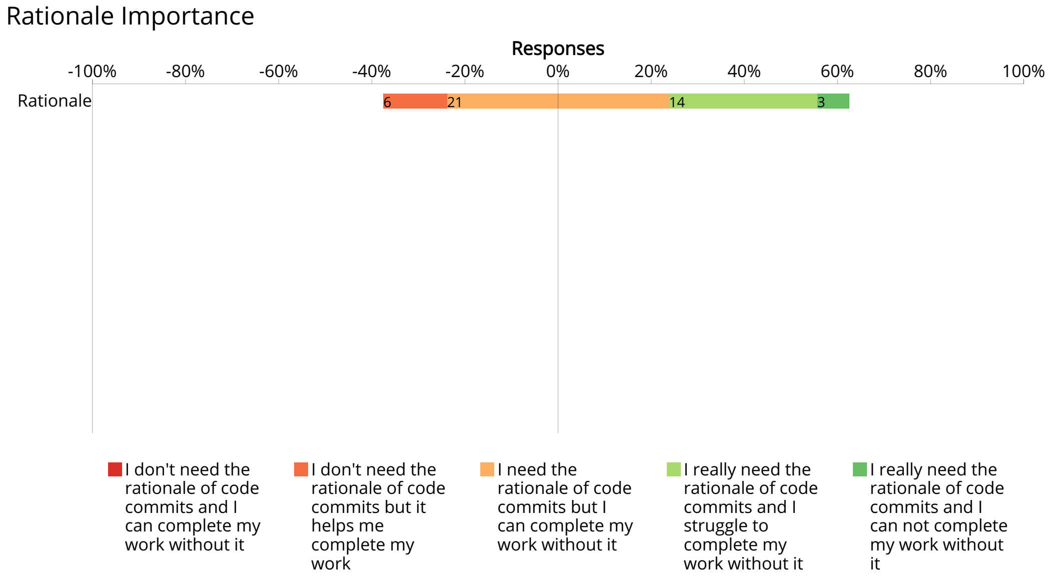
Rationale Components Maximum Frequency of Need

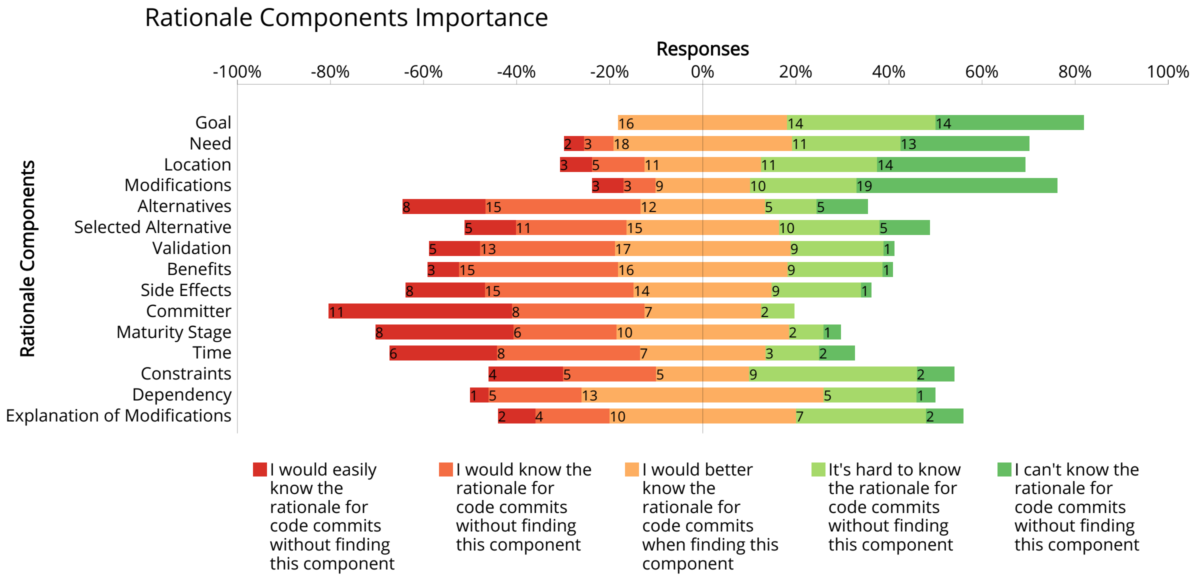


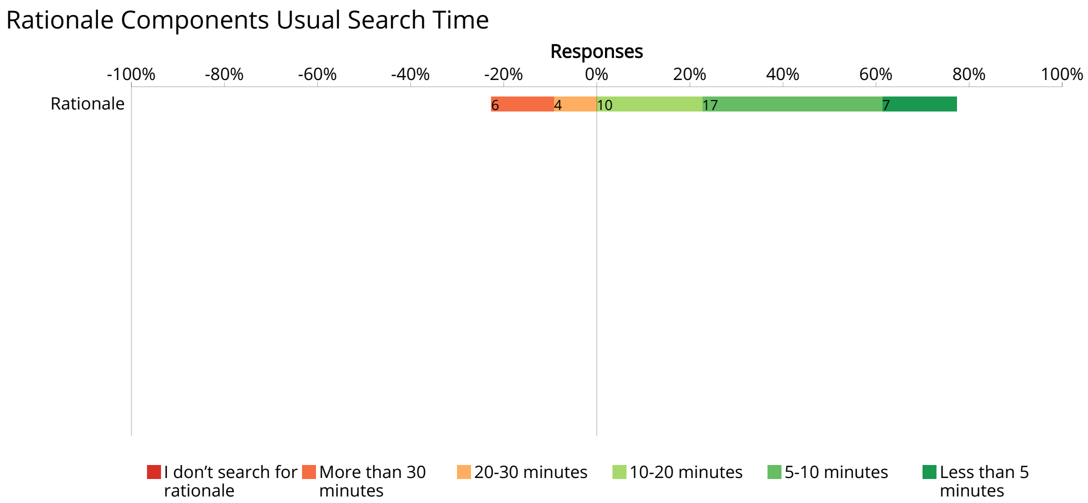
Rationale Components Frequency of Finding

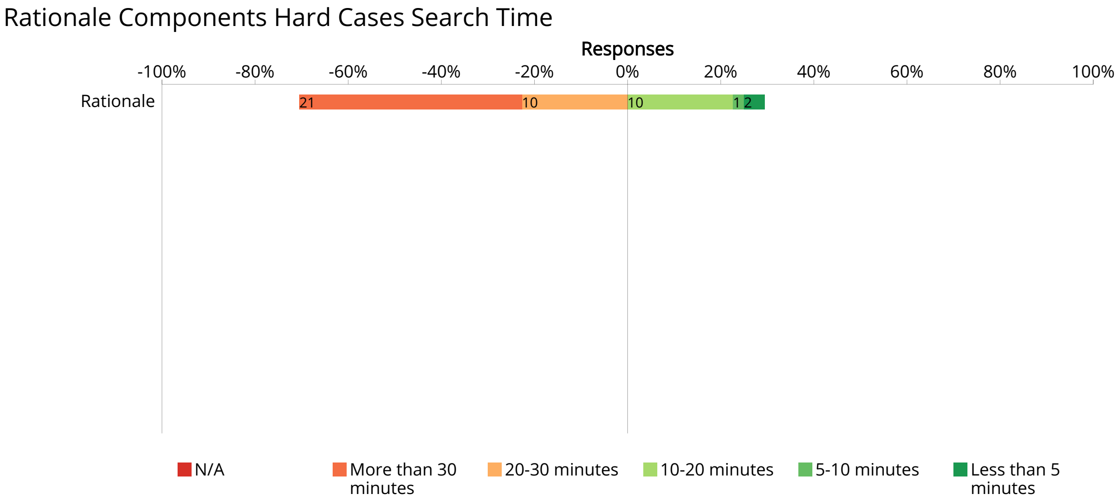



Rationale Components Difficulty of Finding
Responses
-100%
-80%
-60%
-40%
-20%
0%
20%
40%
60%
80%
100%
Rationale Components
Rationale 2 12 16 13 1
Goal 3 21 13 5 2
Need 4 1 14 14 12 2
Location 20 11 6 7
Modifications 21 12 8 3
Alternatives 8 2 8 14 13
Selected Alternative 7 2 5 12 12 8
Validation 7 7 16 10 5
Benefits 8 1 6 15 7 7
Side Effects 7 2 8 17 13
Committer 1 18 7 2
Maturity Stage 5 2 4 6 6 4
Time 3 6 5 5 5 2
Constraints 4 2 1 7 6 5
Dependency 3 2 10 7 3
Explanation of Modifications 1 9 6 8 1
N/A
Very easy
Easy
Neutral
Difficult
Very difficult


Rationale Importance
Responses
-100%
-80%
-60%
-40%
-20%
0%
20%
40%
60%
80%
100%
Rationale 6 21 14 3
I don't need the rationale of code commits and I can complete my work without it
I don't need the rationale of code commits but it helps me complete my work
I need the rationale of code commits but I can complete my work without it
I really need the rationale of code commits and I struggle to complete my work without it
I really need the rationale of code commits and I can not complete my work without it

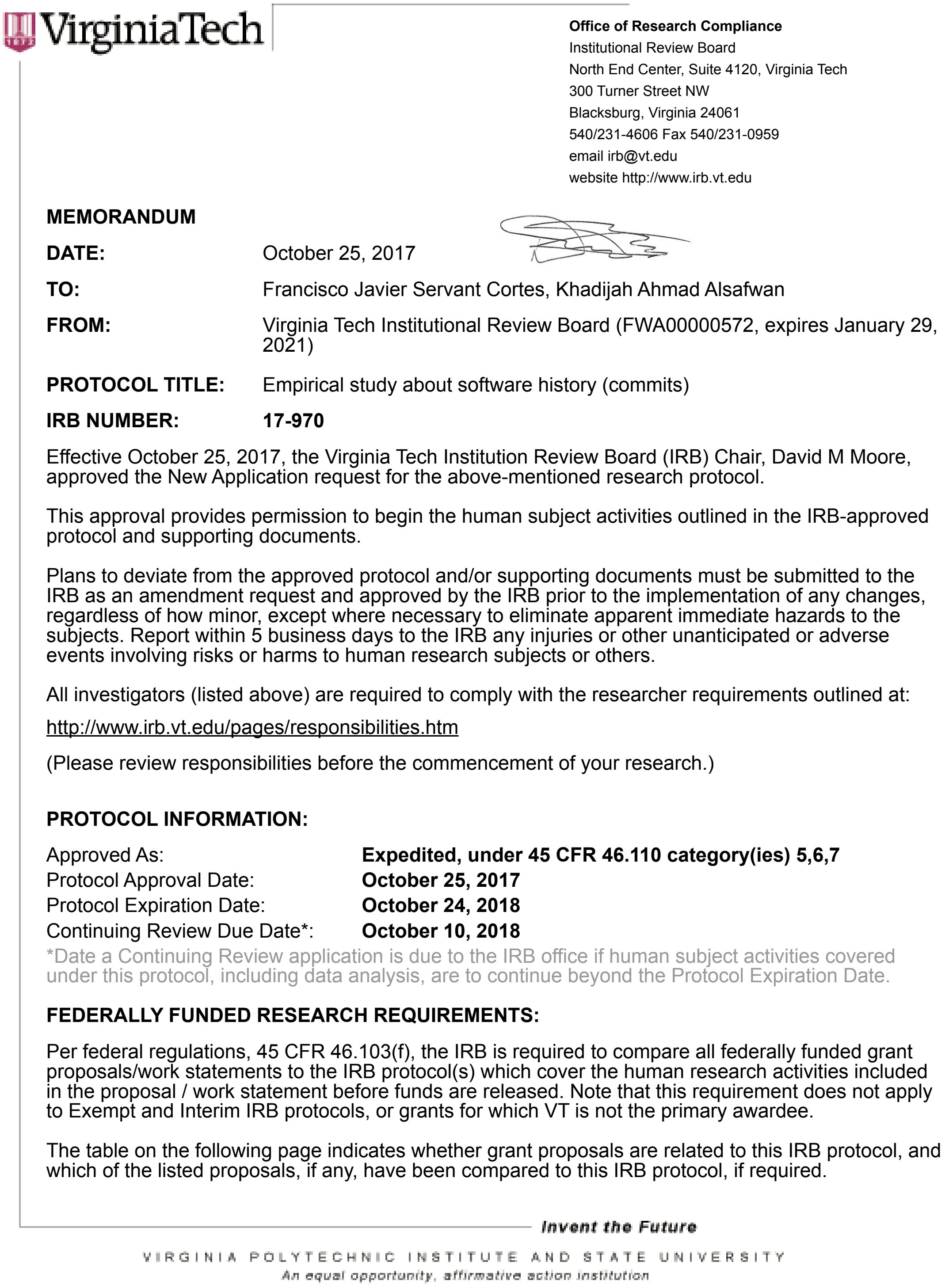

Rationale Components Importance
Responses
-100%
-80%
-60%
-40%
-20%
0%
20%
40%
60%
80%
100%
Rationale Components
Goal 16 14 14
Need 2 3 18 11 13
Location 3 5 11 11 14
Modifications 3 3 9 10 19
Alternatives 8 15 12 5 5
Selected Alternative 5 11 15 10 5
Validation 5 13 17 9 1
Benefits 3 15 16 9 1
Side Effects 8 15 14 9 1
Committer 11 8 7 2
Maturity Stage 8 6 10 2 1
Time 6 8 7 3 2
Constraints 4 5 5 9 2
Dependency 1 5 13 5 1
Explanation of Modifications 2 4 10 7 2
I would easily know the rationale for code commits without finding this component
I would know the rationale for code commits without finding this component
I would better know the rationale for code commits when finding this component
It's hard to know the rationale for code commits without finding this component
I can't know the rationale for code commits without finding this component


Rationale Components Usual Search Time
Responses
-100%
-80%
-60%
-40%
-20%
0%
20%
40%
60%
80%
100%
Rationale 6 4 10 17 7
I don't search for rationale
More than 30 minutes
20-30 minutes
10-20 minutes
5-10 minutes
Less than 5 minutes


Rationale Components Hard Cases Search Time
Responses
-100%
-80%
-60%
-40%
-20%
0%
20%
40%
60%
80%
100%
Rationale
21
10
10
1
2
N/A
More than 30 minutes
20-30 minutes
10-20 minutes
5-10 minutes
Less than 5 minutes

Appendix D

Virginia Tech IRB Approval



VirginiaTech

Office of Research Compliance
Institutional Review Board
North End Center, Suite 4120, Virginia Tech
300 Turner Street NW
Blacksburg, Virginia 24061
540/231-4606 Fax 540/231-0959
email irb@vt.edu
website http://www.irb.vt.edu

MEMORANDUM

DATE:	October 25, 2017
TO:	Francisco Javier Servant Cortes, Khadijah Ahmad Alsafwan
FROM:	Virginia Tech Institutional Review Board (FWA00000572, expires January 29, 2021)
PROTOCOL TITLE:	Empirical study about software history (commits)
IRB NUMBER:	**17-970**

Effective October 25, 2017, the Virginia Tech Institution Review Board (IRB) Chair, David M Moore, approved the New Application request for the above-mentioned research protocol.

This approval provides permission to begin the human subject activities outlined in the IRB-approved protocol and supporting documents.

Plans to deviate from the approved protocol and/or supporting documents must be submitted to the IRB as an amendment request and approved by the IRB prior to the implementation of any changes, regardless of how minor, except where necessary to eliminate apparent immediate hazards to the subjects. Report within 5 business days to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.

All investigators (listed above) are required to comply with the researcher requirements outlined at:

http://www.irb.vt.edu/pages/responsibilities.htm

(Please review responsibilities before the commencement of your research.)

PROTOCOL INFORMATION:

Approved As:	**Expedited, under 45 CFR 46.110 category(ies) 5,6,7**
Protocol Approval Date:	**October 25, 2017**
Protocol Expiration Date:	**October 24, 2018**
Continuing Review Due Date*:	**October 10, 2018**

*Date a Continuing Review application is due to the IRB office if human subject activities covered under this protocol, including data analysis, are to continue beyond the Protocol Expiration Date.

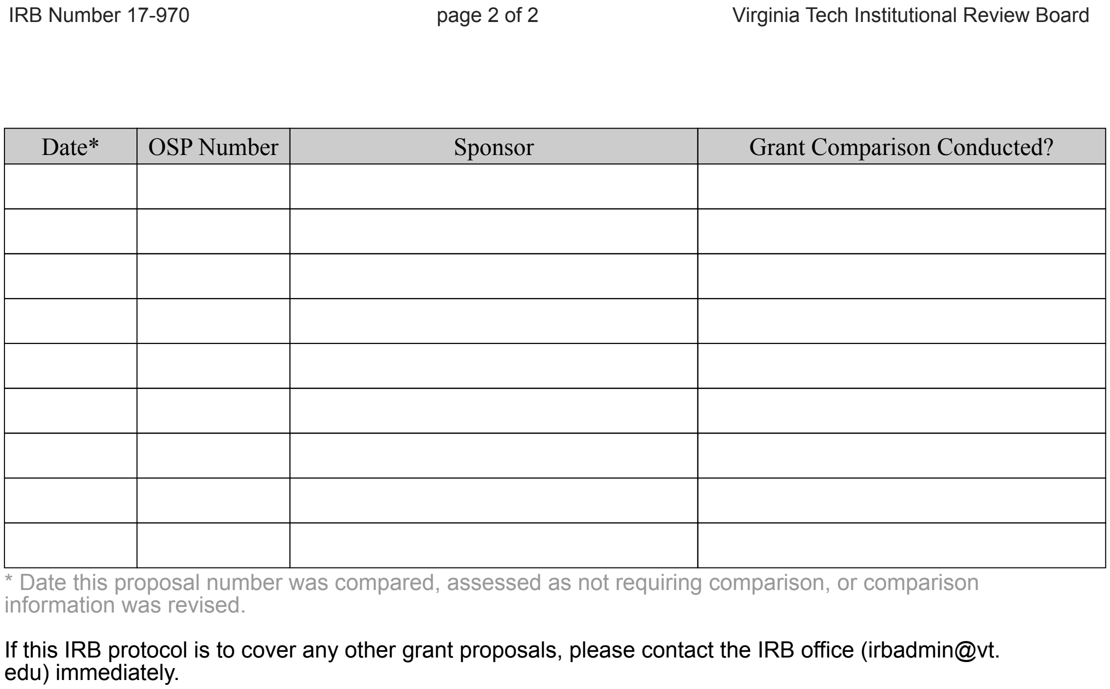
FEDERALLY FUNDED RESEARCH REQUIREMENTS:

Per federal regulations, 45 CFR 46.103(f), the IRB is required to compare all federally funded grant proposals/work statements to the IRB protocol(s) which cover the human research activities included in the proposal / work statement before funds are released. Note that this requirement does not apply to Exempt and Interim IRB protocols, or grants for which VT is not the primary awardee.

The table on the following page indicates whether grant proposals are related to this IRB protocol, and which of the listed proposals, if any, have been compared to this IRB protocol, if required.

Invent the Future

VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
An equal opportunity, affirmative action institution



IRB Number 17-970 page 2 of 2 Virginia Tech Institutional Review Board

Date*	OSP Number	Sponsor	Grant Comparison Conducted?

* Date this proposal number was compared, assessed as not requiring comparison, or comparison information was revised.

If this IRB protocol is to cover any other grant proposals, please contact the IRB office (irbadmin@vt.edu) immediately.