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 <u>your own</u> 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 <u>other developers'</u> 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: [URL].

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?

_____