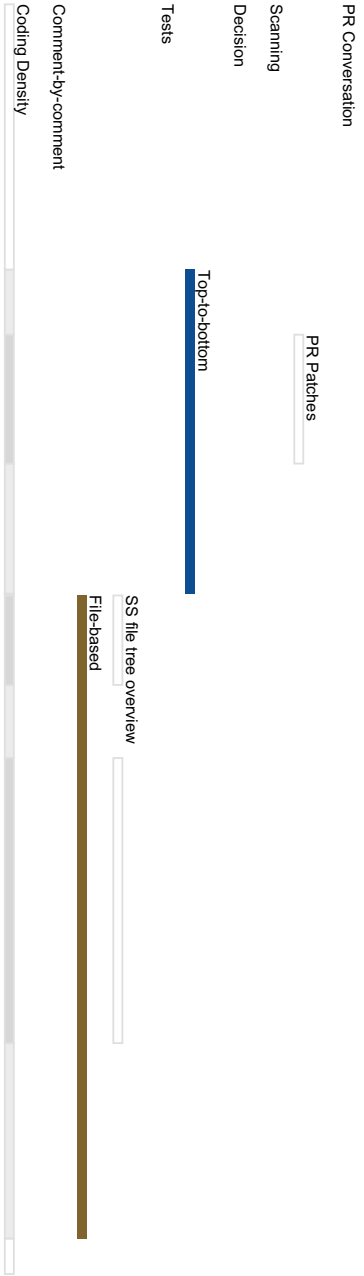
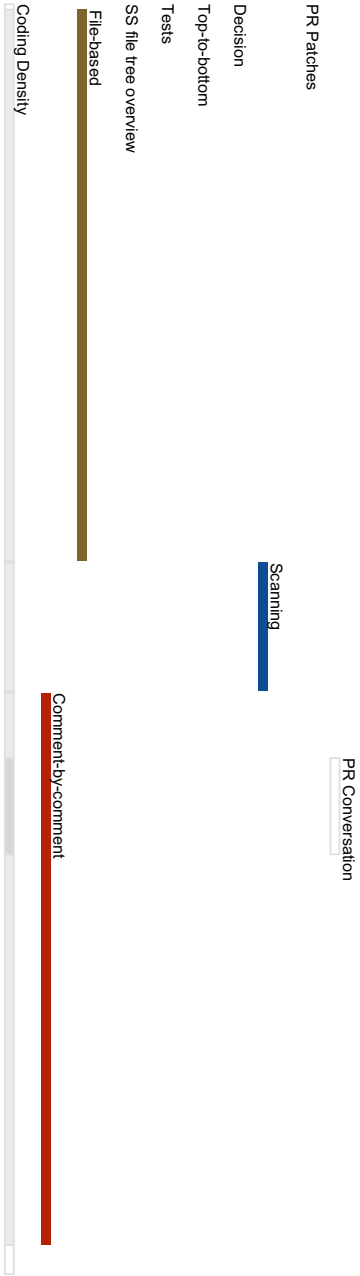
REVIEW TYPE AND GOAL
- Re-review of P2R1

CONTEXT
- The author squashed the commits so the reviewer could see better what was happening.

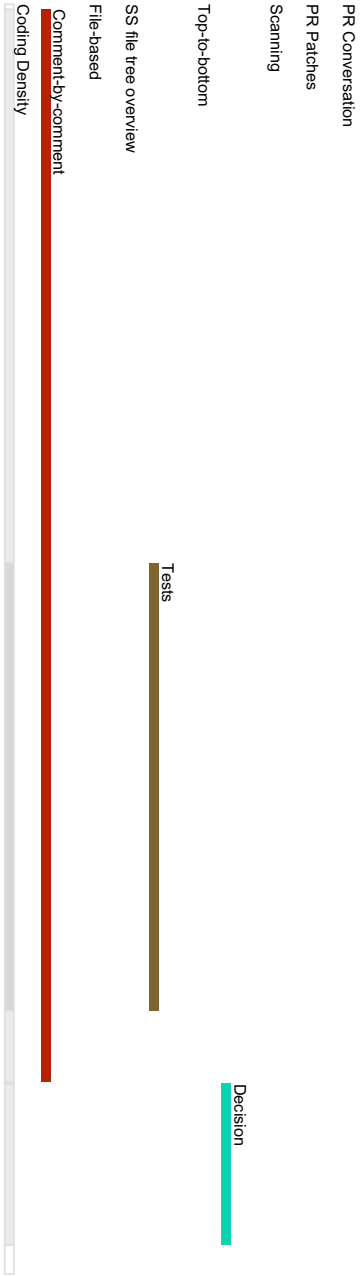
TRANSCRIPT
- Goes directly to comparing the new commit to the previous version.
- Uses the comparing commits Github function to compare the new commit directly.
- Views the changes in the Unified view and looks at the files changed in the commit.
- Sees a few first changes and then searches for a while to find the file by file view, which allows one to track which files have already been reviewed. Opens the file by file on a new tab.
- Looks at what file was changed first in the comparison view, then opens a changed file to review it in the file by file tab.
- Changed the file-by-file view to a commits view. However, gets lot looking for the file by file tab again.
- Comments that it is very helpful to see which files changed since he did his last review - it helps not to lose time with things they have already seen
- Notices a confusing notifier - on the content view of the repository, there is a cooler (brown icon) that indicates that the file has changed since the last time, but it is not consistent with the indication on the file header directly. Leave it be and review it according to the file header information.
- Appreciate one change that sorts the packages alphabetically.
- Comments that the author has there two caches, which could be better, but leaves it for later to decide or address: “Let’s see”
- Notices one of their comments was addressed -



however, they do not see the comments, so they are recalling it from memory
- Looks at the commit comparison view
- Looks at the Commit view
- Adds a comment asking to clarify characters used in the function. Suggests to add an example in the #review comment
- Goes back to the commit comparison view. Notices an updated doc string still does not make sense.
- Attempts to enter a comment on the line, but the view does not allow it.
- Switches to the file-by-file view to enter the comment. Writes in the comment the exact words that would improve the documentation. They are unsure if the suggestion is correct, so they pose a question of whether it's OK. Follows with a more abstract description of how to improve the docstring.
- Scrolls through the commits comparison view to capture other problematic issues.
- Scrolls till the end - everything seems OK. Scans one more time from top to bottom.
- Goes to the file by file view. Cannot see his previous comments so they are not sure what is happening there.
- Goes to the PR tab to look at the previous discussion, going into their comments. Confirms the newly added comments. Resolves a comment that was addressed.
- Looks at a comment and tries to find where in the last commit it was addressed. They cannot see enough context, so they go to the file by file view and try to search (using the find command) for the area (the function implementation). After finding it, they resolve another comment.
- Scrolls up and down the class which doc string they proposed to improve. Reads their comments again. Enters another comment in another line of the doctoring, asking to clarify further and describe a typical usage pattern.
- Goes back to the discussion to identify another place



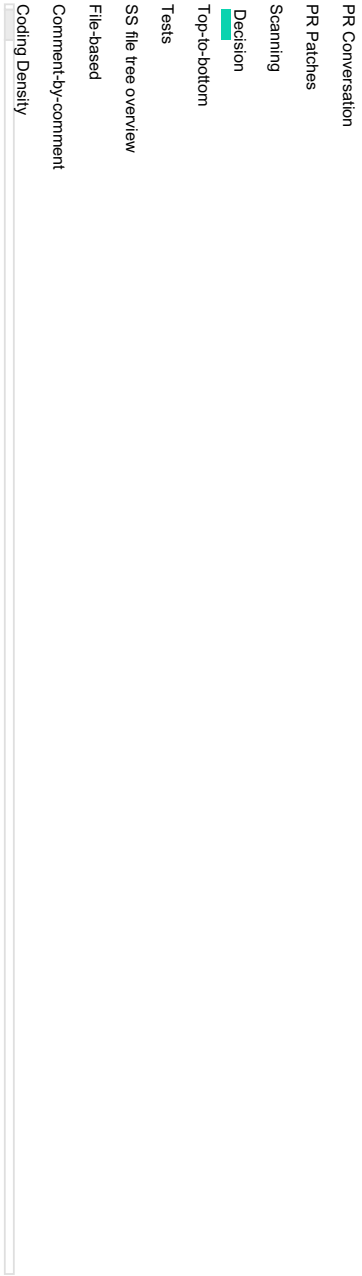
of comment that they find in the file by file view. Suggests renaming of a method.
- Asks a clarifying question in a comment on the doc string.
- Goes to his previous comment on naming and edits it to add a request to improve the code comment format, explains why and suggests how it would be more explicit.
- Scrolls down, reminds me that they insist on writing docs in an imperative form with an example.
- Looks through the whole file and notes that they would like to see how it is tested. The tests could be a better documentation of what the author is doing.
- Goes to the discussion and resolves some of their previous comments.
- Scrolls to the tests. Then goes back to the discussion and marks several of his comments as resolved.
- Goes back to see the tests. Marks some test files as viewed after viewing them. Then identifies the test file they were looking for.
- Says that the author is adding cache data to the queue. Says they are not happy that the tests assume an internal state of an object.
- Otherwise, they seem OK.
- In a comment, asks whether there is a rename to the test method that fits the naming structure of the previous method.
- The question was posed as a way to be polite, but the inconsistency exists.
- In the following tests, the issue is better.
- Marks the file as viewed.
- Goes back to the discussion. Sees there are no pending comments left to check.
- Finishes the review. Says ‘Nice work!’ States there are only minor suggestions and that after addressing the mentioned issues, the code can be merged without further review. Not requesting a further review is a step to minimise friction and time to merge.



- Marks it as ‘Approve’ and submits the review.

GENERAL STRATEGY
- Checks comment by comment and associated code to determine whether the changes were implemented.
- Skimms through the whole files in the process.
- Checks first the comment and then finds the relevant place in the code.
- Sometimes, they review code for a longer period of time without looking at the comments to keep a fresh state of mind and not to set some expectations and biases.
- This also depends on their fatigue - looking through comments is faster, but there is a risk that you might miss something.
- It is also unfair to write more comments on comments that were written previously if you do not look at the existing ones.
- For minor things, one does not need to check much; for bigger things, checks the whole file.
- All the requests they made in the comments they expect to be addressed or replied to except for the appreciative and motivational comments like “Nice work”
- When the review is a re-review with their comments, they can take shortcuts - seeing just the files with comments and throughout the iterations, more and more files are out of the scope.

NOTES
- There is not much discussion or reaction to comments from the author. The reviewer would expect that the author would indicate which comments they addressed.
- Mention the factor of author experience.The code is of lower quality because the author is a student working on a complex and big change.
- Other reviews with this author were just accepted as it was a small contribution.
- Scan the commit changes by files changed.



- Scan the changes from top to bottom.
- First, review the source files and then the test files to verify the function implementation and documentation. Test files were bigger and scrolled faster.