

Results of Thematic Analysis

Initial Data	Code	Theme
We wrote up exactly what the problem was and pinpointed the exact code that needed to be changed. (D1)	way to reproduce	Clear Issue Description
The description may clearly explain where the issue is. (D15)		
Newcomers will know how to reproduce the bug. (D18)		
Context can be explained in the issue descriptions. (D21)		
very simple changes that we pretty much spelled out in the description. (D2)	detail description	
Good frist issues usually have a detailed descriptions. It will make newcomers easier to understand(D23)		
Issue description had also already suggestions about how this could get fixed. (D6)	way to fix	
the approach to take was already outlined in the issue. (D13)		
The description may clearly explain how to fix it. (D15)		
Newcomers will know how to fix just after reading the descriptions. (D18)		
Clearly written issue which identifies specifically what needs to change in the behavior of the software or application. (D19)		
The issue to be addressed was described clearly in the issue. (D7)	clear description	
The issues with clear descriptions can be identified as good first issues. (D3)		
The description of issue is clear and specific. (D12)		
Newcomers will feel easy when the issues have clear descriptions. (D16)		
Newcomers don't feel puzzled when reading the description (D22)		
The description is very clear that clarify what need to be changed. (D10)	what to change	
The issue itself has already clarify what need to be changed, which we usually treat as good first issues. (D17)		
the scope of understanding necessary. E.g. I'm more likely to label extension code (which no other libraries depend on) as GFI and not runtime code (which a lot of extensions depend on). (D1)	scope of understanding	Self-Contained Change
The issues do not require changes in multiple files, which usually is difficult for newcomers to figure out. (D2)	not involve	

The change is self-contained: (1) it touches a small part of the codebase. (D4)	multiple files/functions	
it was rather self-contained to a single file. (D6)		
They all require changes to be made in a single file and the changes need to be made in a single function in that file. (D7)		
Issues which can be tackled with small amounts of code that is to say they are self-contained. (D8)		
Its a short, self-contained task that should only touch very few files and the approach to take was already outlined in the issue. (D13)		
Ideally, it doesn't need to modify mutiple files or functions. (D15)		
Newcomers are less possible to sovlve a issue with complex changes that involve many files. (D16)		
I usually treated the issues that with a few changes, ideally to a single file as GFI. (D17)		
The changes don't need complex implemmentation, such as documentation, minor fix in a single function or files. (D18)		
Newcomers usually feel easy when the changes do not need deep investigation and do not spread in multiple files. (D19)		
Its a short, self contained task that should only touch very few files and the approach to take was already outlined in the issue. (D20)		
The change that is ioslated to a single file is more likely to be sovled by newcomers. (D22)		
This issues is just a minor fix to a single file and do not need deep understanding of the whole project. (D24)		
(2) it doesn't change wide-spread APIs that require chasing callers; (D4)	not involve wide-spread APIs	
(3) ideally, it's unit-testable. (D4)	unit-testable change	
Difficult also can be, if adding feature breaks something else. (D5)	minimal impact	
minimal interconnections with the rest of the app. (D10)		
I would consider it a good issue to start with if I new that the impact on other code would be minimal. (D12)		
Small, more or less self-contained tasks where the team already has a vague idea of how the fix might look like. (D14)	self-contained	

The change only on the UI part which is isolated and easy to verify. (D21)	change	
This issue requires some basic understanding of Java (understanding a stacktrace and reading through some existing code to fix an exception arising in a particular situation). (D2)	basic knowledge needed	Limited Skills Needed
It doesn't require inside out familiarity with the whole codebase, doesn't imply deep knowledge of RFCs or technical standards, and doesn't require guru-level coding skills. (D4)		
Issues which won't require any substantial knowledge of the specific programming language or the overall application.(D5)		
Resolving any of these issues will require someone to amend some existing Java code - so some knowledge of programming is essential but do not need deep investigation. (D9)		
Issues which won't require any substantial knowledge of the specific programming language or the overall application. (D15)		
It only requires intermediate Ruby programming. (D17)		
This issue only need to have experience with python dependency management in general, or with that particular library.(D23)		
This issues is just a minor fix to a single file and do not need deep undertanding of the whole project. (D24)		
it didn't require internal implementation knowledge, it was just about documenting a thing better. (D6)	no internal implementation knowledge needed	
It required some expertise from the library consumer's side but it didn't require internal knowledge. (D8)		
"Good first issues" should not tackle any large architectural or behavioural changes. Documentation changes, test fixes, small nonurgent bugxes and internal refactoring are good candidates. (D13)		
don't require any understanding of the wider context - except in the challenge of locating where code changes need to be made in the first place which might prove to be the most difficult part of the challenge. (D10)		
Is this something that someone without a lot of project-specific context could pick up and work with? (D20)		
Issues don't require additional interpretation or understanding to tackle the issue. (D12)	no additional interpretation needed	
Repetitive task - once you've done one, doing the rest should be quite straightforward. (D19)	repetitive task	

The amount of code needed to be written is minimal (a few lines at most).(D2)	limited lines of code	Less Workload
Most times I consider issue to be beginner friendly when it only requires few lines of changes which should be more or less trivial to do. (D6)		
Issues which can be tackled with small amounts.(D8)		
Small, more or less self-contained tasks where the team already has a vague idea of how the fix might look like. (D14)		
Usually, it is only need to modify a few lines of code.(D16)		
Newcomers are more likely to sovlе the issues with less workload, for example, just a few lines of moifications. (D18)		
should only touch very few files and the approach to take was already outlined in the issue. (D20)	limited time	
As for what we use to judge if it's suitable for newcomers, usually it's the amount of time it would take to implement it (usually < 3h). (D5)		
Good first issues usually can be fixed with limlited time.(D12)		
Repetitive task - once you've done one, doing the rest should be quite straightforward. (D19)	limited investigation	
It is a small change and can be solved with less investigation. (D10)		
"Good first issues" should not tackle any large architectural or behavioural changes. Documentation changes, test fixes, small nonurgent bugxes and internal refactoring are good candidates. (D13)		
Ideally, it doesn't need to modify mutple files or functions. (D15)		
This issues is just a minor fix to a single file and do not need deep undertanding of the whole project. (D24)	rough idea	Available Support
I treat the issues as good first issues that whether I have a rough idea of the implementation so I can provide support for newcmers. (D2)		
I can support newcomers if they get stuck and lead the decisions. (D4)	available support	
Whether can provided guidance is important to whether the issues can be sovled by newcomers. (D10)		
Newcomers will feel easy when there is avaiable support when they meet the difficulties. (D16)		
I usually expect a issues can be fixed by newcomers under the support by our experts. (D17)		
Is someone on the maintainer team willing and available to support someone new to the project to get the context they'd need to solve this by themselves? (D18)		

We point them to the tutorial and suggest they post questions right in the thread – which if they do choose to engage, someone on our team does a great job responding. (D20)		
we expect newcomers to be able to get to the point where they can contribute by a few rounds of asking additional questions and we will provide timely support. (D24)		
It has a visible final impact on users. This is a meta-property of the issue, and I try to take that into consideration to avoid assigning boring tasks. Also, newcomers want to “see” the end result. (D8)	visible final impact	Motivating Newcomers
Issues with medium-high value: it can provide a chance to learn about and practice with a new feature.(D12)	medium-high value	
Could this provide an interesting challenge? Does would solving this help someone new learn a little bit about our codebase? (D20)	interesting change	
Low urgency but medium-high value.(D8)	low urgency	Low Urgency
We usually treat issues that are low urgency as good first issues because newcomers need more time to get familiar with projects.(D12)		
I usually don't labelled the issues with high priority as GFIs.(D17)		