**CIS 362 Project 2**

**Case Study**

**April 20, 2020**

**Team 9 -** Jehu Barbee, Eric Botelho, Ashley Famularo, Grady Landers, Michael Medeiros

**Introduction**

Documentation, or commenting code, is a touchy subject for programmers. While schools enforce the documentation of coding projects and homework assignments, various sources online say that documentation is either barely there or non-existent. Our study was conducted to find out possible reasons why this would be the case. This is considered a case study as it tries to find out what is happening to cause programmers to not comment/document their code. It attempts to identify factors that would cause them to ignore documentation all together. There was a similar case study, albeit on a much wider scale, conducted by Sam Wilmot at Balisage looking into reasons why documentation would be left at the wayside when writing programs. Their study points to it being too costly, as commenting increases the amount of work needed to be done on the program. Having to manage all the information that needs to be commented increases development time, and it also has the chance of making errors more likely.

**Conducting the Study**

Our study can be defined as a “causal” study with the reason being we are trying to find a link for various factors as to why programmers don’t comment. During the study our focus shifted slightly as to not being entirely on why they don’t comment, but what affects them commenting in general. To try and find the links, we asked each subject a series of questions through a Google form which was sent out to subjects every 1-2 days for a period of a week. Not counting the first question of “what is your first name,” the form consisted of 6 questions. The questions included asking what programming language and IDE they are currently working with, the reason as to why they are doing the project they are currently working on, whether the project stemmed from supplied code or was made from scratch, if there were reasons they wouldn’t comment such as time or space constraints, posing a hypothetical where if commenting was optional, would they do it still, and finally a question dedicated to overall thoughts on commenting your code.

Of the 6 questions, the 2nd question was the only one that had posed an issue initially, due to its phrasing. The initial phrasing was “what is the current language you are using, **the codewriter and the compiler associated with said language?**” As it can be seen, the bolded portion of the question could and was shortened to “What is the current language and IDE you are using?” which provided a much more efficient form of that idea.

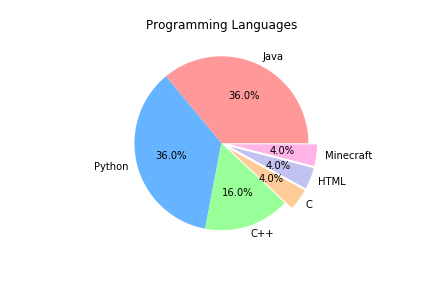
Given our topic of commenting code, our sample frame for the study would be programmers, however all programmers is an improbable number to reach out to. The sample size of the case study was 15 individuals who were given the study every 1 - 2 days over the period of a week. Many of those observed had taken the form the max amount of times they could, that being three times. However, a handful of those observed had only filled it out twice.

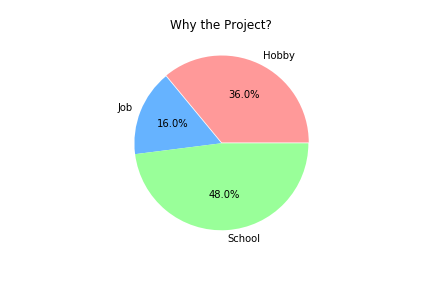
**Results**

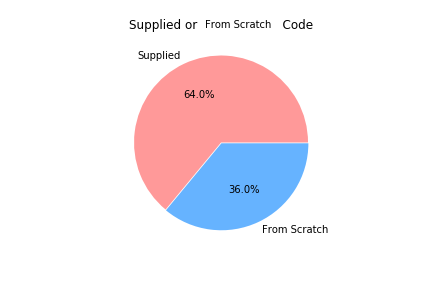
**Correlation Breakdown**

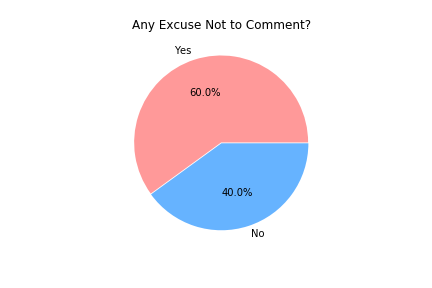
The above heatmap presents the correlations for the various factors/questions recorded from the participants. A handful of the correlations are relatively small, denoting that there is no link between those factors. This would include the -0.099 that occurs between the *Supplied or from Scratch?* and *Why the Project?* Items. However, there is a notable correlation. That correlation being the one between *Reasons you wouldn’t comment?* and *Optional?* items with that correlation being -0.51. This high correlation can be assumed because the commenting being optional could fall under a possible reasoning for not commenting your code. Another relatively high correlation is presented between *Supplied or from Scratch?* and *Optional?* at a value of 0.46. In this case, it can be assumed that the more often the code is supplied, the more often commenting will be seen as optional. This could be understood as most supplied code from places such as schools include comments to explain the already present code, so additional comments are generally not necessary.

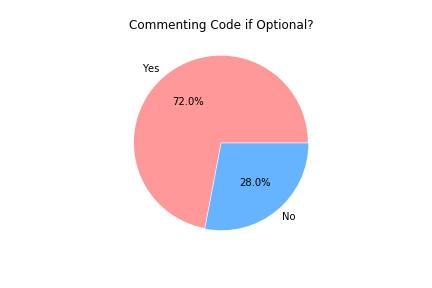
**Chart Breakdown**

This pie chart shows off the various answers we received for what programming languages our subjects are currently using. As you can see, most popular among participants were Python and Java, with C++ as a runner up. Besides those, some single answers included C, HTML and Minecraft, which we assume means they were using Java and C++ to create modifications for the game.

One question asked addressed the reason behind our participants’ coding habits. Possible answers included work, school, or hobby. As visualized by the chart above, most of the subjects (48%) primarily program as part of their studies. Next most frequent were those who program for fun (36%) and the least of them actually have a job in the field (16%).

Another question asked whether the code participants are working on was made from scratch or started out as a file supplied to them. The pie chart makes apparent that the majority of our subjects (64%) had been given code to work from, and that likely included comments. These values line up with our results as students and professional programmers are likely given code to start working from or improve upon respectively. As discussed with the heatmap, it was found that pre-written code reduced the likelihood of a programmer adding comments.

In an attempt to find correlation between participants’ programming habits and documentation, they were asked whether they believe there are any reasons to leave out comments. As you can see, 60% believe there are excuses not to document code. When analyzing the relationships between attributes it was found that those who answered positively to this question typically seemed to believe that including comments was considered optional. Subjects’ reasoning is explained further when looking at some of their open ended responses.

Lastly, one question considered whether participants comment their code even when they believe the task to be optional. Surprisingly an overwhelming amount responded that they would comment, 72% of them. This suggests that documentation is not completely lost on programmers and some do attempt to make an effort in improving the readability and ease of understanding for themselves and others that might be reading their code.

**Notable Responses**

Along with the multiple choice and short answer questions, we provided an open response area for participants to give their general feelings on commenting. Of the 27 responses, the 8 most notable were chosen. The first 4 are notable as they demonstrate instances where an individual's likelihood to not comment increases:

* “Useful/helpful but having said that, I'm still too lazy to do something I'm not being graded on”
* “If it’s going through an auto grader then it seems unnecessary to the student. If it’s a group project then it’s absolutely necessary.”

Those above 2 comments demonstrate that, for something like a school assignment, if the comments are not factored into the overall grade of the assignment then they are not necessary.

* “ A lot of the time professors give starter code that already has comments and the rest of the code is fairly self explanatory. While these school projects are usually small enough to not”
* “I like comments, but sometimes I feel commenting on code can be difficult, like riding a bike. You know the steps to ride a bike, it just takes determination and time; and quite frankly I don't have enough time or I'm just too scared to comment what I'm thinking.”

Those above 2 comments demonstrate a situation brought up with our question of *Reason to not comment?* where the programmer wouldn’t comment due to several constraints As shown above, the constraints could be the understanding of the code such as if the code is seemingly self-explanatory or difficult to describe, like riding a bike, or the constraint could be lack of time.

The following 2 comments can fall in a middle ground area where the likelihood of commenting your code can either increase or decrease depending on the code itself:

* “If methods and/or functions are repeated several times within the code then they should only need to be commented once. If the variables used are not clearly labeled then they should also be commented.”
* “if that same piece of code is repeated elsewhere in the file it shouldn’t have to be commented… but if everything is labeled clearly and values are easily identifiable, for example “TotalValue” and “InitialInput” then comments aren’t necessarily needed.”

The likelihood of not commenting can increase if the functions and variables of the code are labeled with specificity and aren’t named things such as “x,” “y,” or so on. However, if they are labeled with generic names, then the likelihood of commenting increases given they will have to explain just what exactly is going on in the code.

The last 2 comments demonstrate instances where an individual’s likelihood of commenting their code increases:

* “the more I code the more I appreciate my comments when I go back to look at past programs.”
* “Comments are essential in code. They tell the next poor soul the general idea of the code/the reason for the code to exist.”

The likelihood of commenting increases if there is a high possibility that you or someone else will be looking at your code sometime in the future. These comments would provide background to you or the other individual on what exactly is going on in the code after a period of not looking at it.

**Recommendations**

Throughout our study we noticed a few flaws in the way we went about conducting the study and have thought of a few ways to improve the study in future conductions. Firstly, maintaining an effective system for reminding the subjects of the study to continue participation. Since the subjects have to take the study every so often they need to be reminded to provide input to the study, without a proper reminder subjects tend to forget and fail to retake the study causing less data to be present in the final results. Secondly the frame size should be reduced down to a single occupation rather than just all programmers, as this is a massive field of study that covers a range of different programs and languages, so narrowing down to a smaller focus would yield better, more concise results rather than a jumble of programs and languages. Lastly, pulling from our own experience, simplifying questions down to something with a yes or no answer would be better, as giving short answer questions seems to lead to confusion from the recipients and would require revisions to the study to make the questions clearer.

**Conclusion**

With the results from the case study, it is fair to assume that many of whom we had gotten responses from find that commenting code is important, but the failure to do so lies in many other factors. For some, if they are the only ones looking at the code, they find it pointless to do so, but if they know that, in terms of school, it is weighted for their grade or that they are working in groups, then they will comment their code because either it’s required or it will help others understand their code. This makes sense, as many programmers have different styles of coding to the point where others may not understand their code at first glance, so comments are a necessity in those situations. Similar to the previously mentioned study from Sam Wilmot, it’s also important to draw a line where spending too much time on commenting is a waste of time. There is also a possibility even people who don’t program may be able to understand some snippets of code without comments or any prior knowledge of the code if it seems self-explanatory or is labeled well enough. With these factors considered, commenting code walks a fine line between being too in-depth and a waste of time, and being a necessity because without it, there will be a loss of translation and confusion.

Works cited

Wilmot, Sam. “Proceedings.” *Literate Programming: A Case Study and Observations*, Stilo International, 7 Aug. 2017, www.balisage.net/Proceedings/vol8/html/Wilmott01/BalisageVol8-Wilmott01.html.