CIS 5930 Project

Jordan Gethers, Sineha Aneel, Aditi Allady

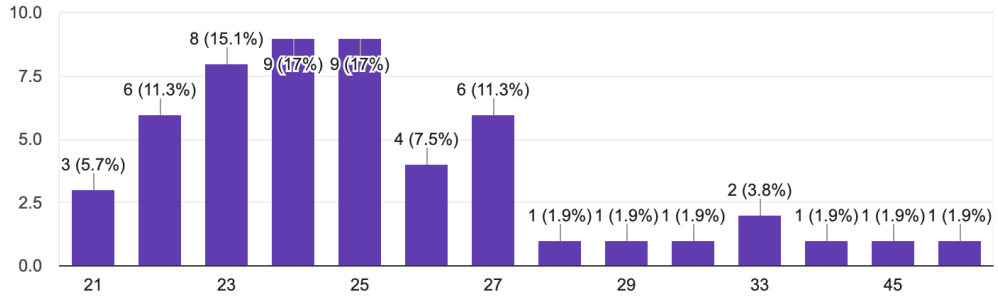
Abstract

Our goal was to replicate a study that measured how technical debt may affect developers' productivity. We wanted to know things such as, how their managers viewed technical debt as well as what tools developers use to see if there were any common links between the technologies developers use and their individual thoughts on how technical debt may affect their work. In this replication study, we will go over some of our main findings and analyze how the data matches up together. We will also compare our results to the original study to see if, and how the results of our studies differ from the original.

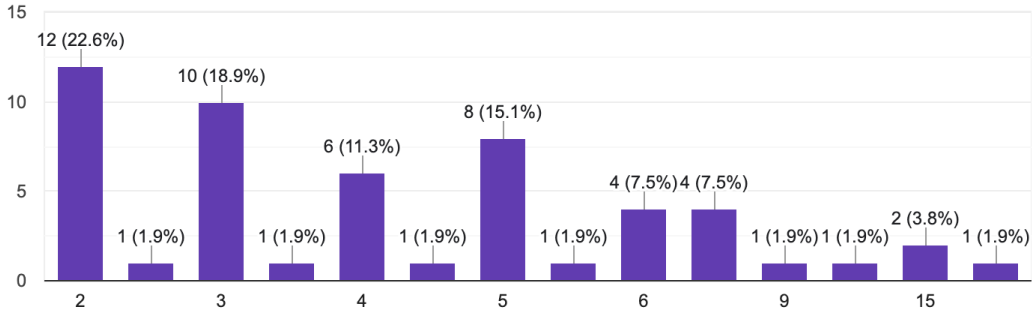
Introduction

Where we differ from the original study is that we do not ask the participants the same questions repeatedly over time, which will serve as one of the limitations of this replicated study. In our result findings, we also differed in that the most time wasted by developers came from meetings instead of additional testing, this could be due to the factor of the average age of our respondents to the survey. It is possible that although we sent the survey out to a wide range of individuals, the people most likely to respond to us could be fellow students or recent grads who just recently entered their respective career fields. In the original study, the most commonly cited activity that contributed to technical debt was additional testing. A majority of this sampling and opinion difference could be due to the fact that over 50% of sample respondents in the original study had 10+ years of developer experience. It is likely that they find activities such as additional testing to be mundane because they are very experienced workers and would likely be content with moving on to another task, knowing that the work they provided was quality.

Ages of Developers:

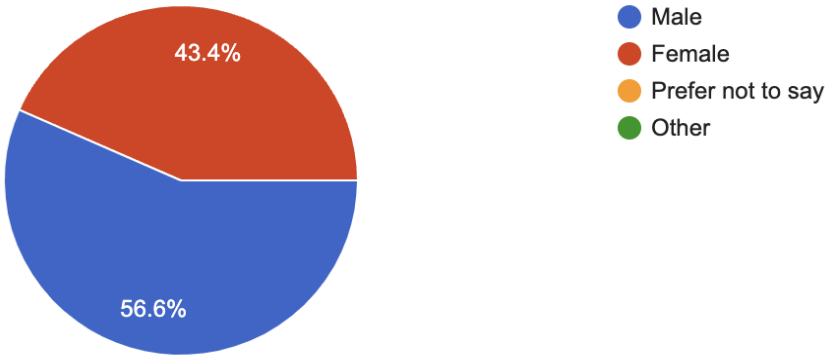


Developers years of experience programming:



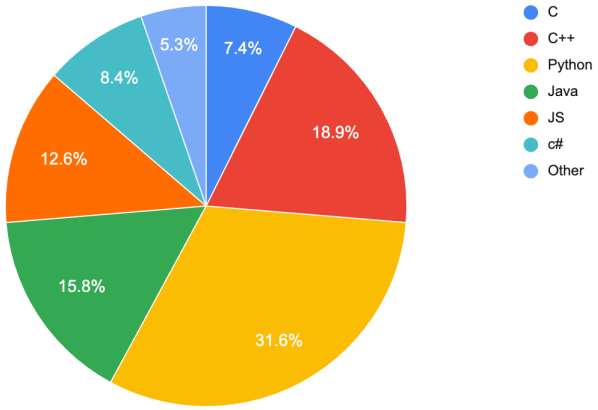
When it came to gender, we had slightly more male respondents than females. Even with 15% or so more male than female respondents, it is still a much more balanced pool than in the original study where over 80% of respondents were male. This could have many ramifications on the other responses, but our study is more likely to represent the current opinions of computer scientists as the group is becoming increasingly more balanced and female.

Gender Demographics of Developers Surveyed:



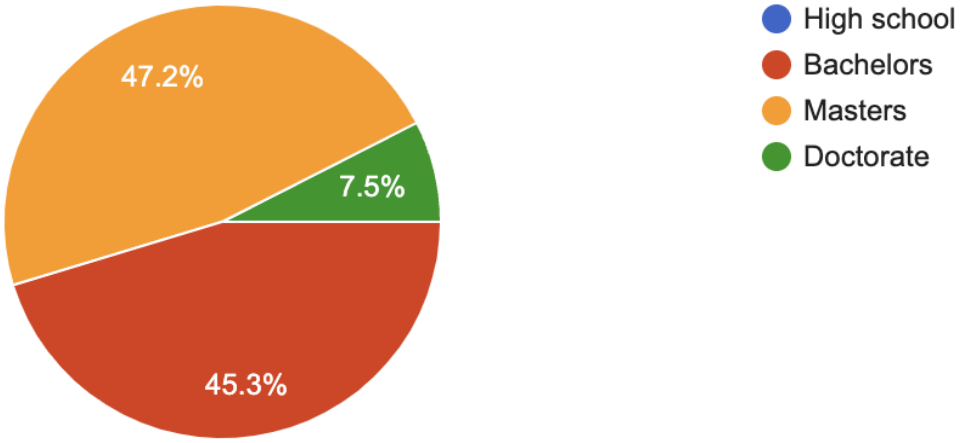
It is also likely that in the original study, additional testing took much more time because over half of the developers surveyed wrote their code in C and C++, both languages of which mistakes can be more difficult to find and much more catastrophic in nature. Whereas today, and in our responses, we found that many more people work with higher-level languages such as Python that may require less testing to be serviceable in production.

Programming Languages Used:



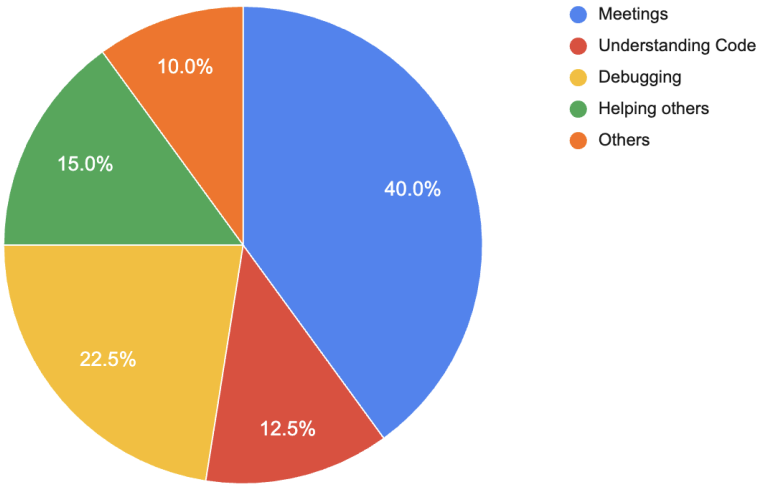
An interesting demographic is in the original study over 80 percent of their participants had achieved a master's degree. Their population overall seems very homogenous and may not represent most developers today. In our findings, there was about a 50-50 split between those that had obtained their master's and bachelor's degrees.

Education Levels:

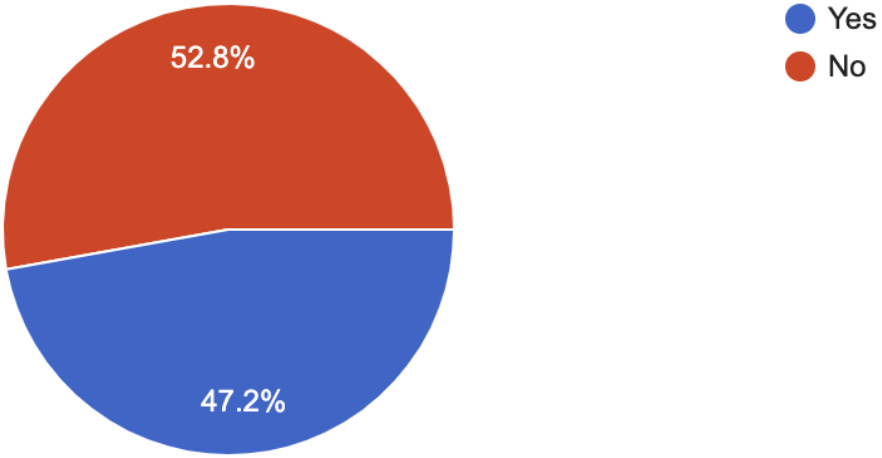


Another interesting statistic about our pool of respondents is that about half work from home (remotely) at least some of the time. This could be one reason why meetings were seen as the biggest waste of time. Video call meetings tend to be less engaging and may take longer than if everyone is in person. It is also more likely that co-workers were already messaging each other or sending emails throughout the day to one another, so meetings could just be reiterations of what was already discussed and understood throughout the day.

Main Sources of Technical Debt:



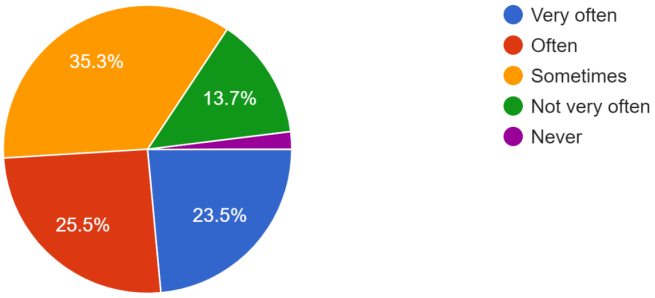
Do you work remotely or not? Responses:



Only 17% of our respondents claimed that they are typically not given enough time to finish projects. No one said that they strongly feel as though they aren't given enough time to complete projects. Looking at the data, it could suggest that the main technical debt factor outside of meetings could be working with old or deprecated code and systems as almost 75% of our respondents claimed that they are "forced" to work with legacy systems. It would require additional studies for replication, but the data seems to suggest that the number one thing slowing down the coding process as it's happening is working with outdated systems and code.

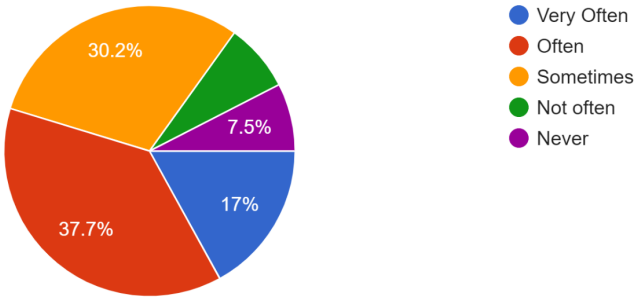
How often are you say you are “forced” to use legacy code or outdated versions of applications because of hardware /company limitations?

51 responses

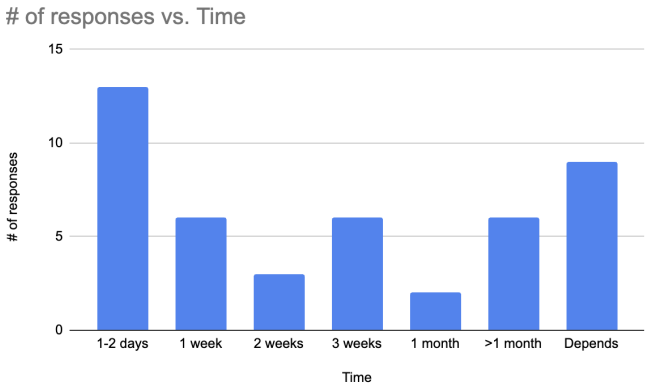


How often would you say you feel rushed when submitting code for production and or review?

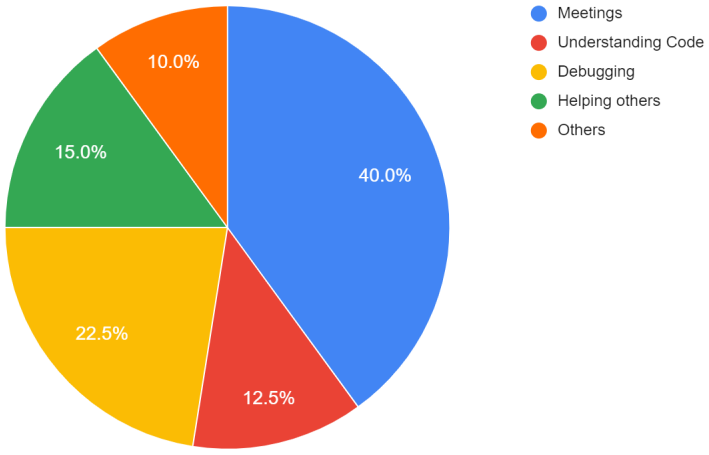
53 responses



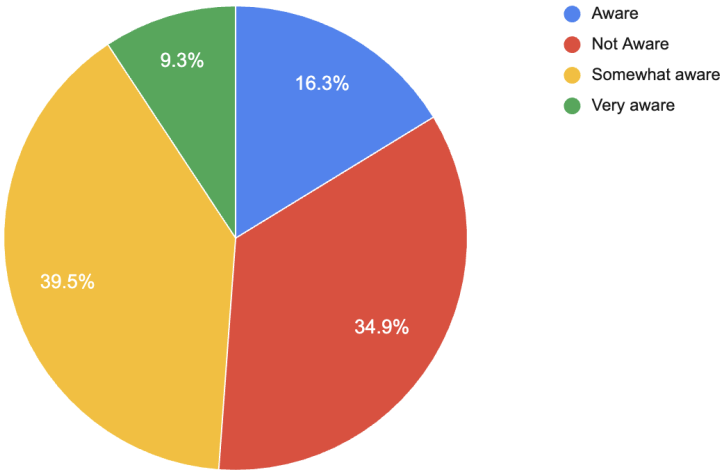
How long would you say that the projects take if technical debt occurs?



On which extra activities is the wasted time spent?



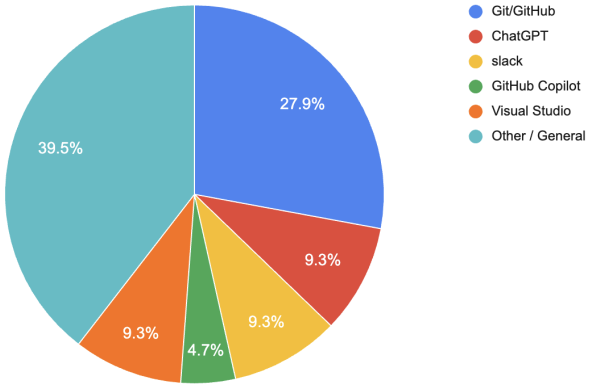
How aware are the developers and their managers about the wasted time due to Technical Debt?



Do you feel team meetings or meetings in general lead to wasting time and not accomplishing anything?

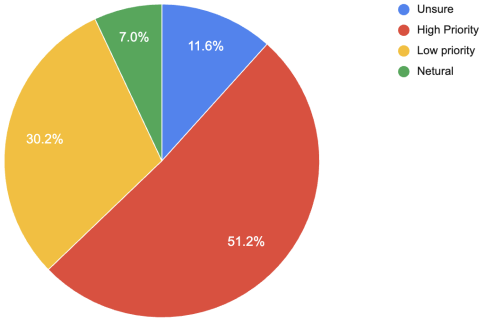
What productivity tools do you use to attempt to reduce technical debt?
In this question, our survey had a short answer requirement hence we had to aggregate it, the most highly used productivity tool was Git and our responses had a variety of different answers.

Productivity Tools Used:

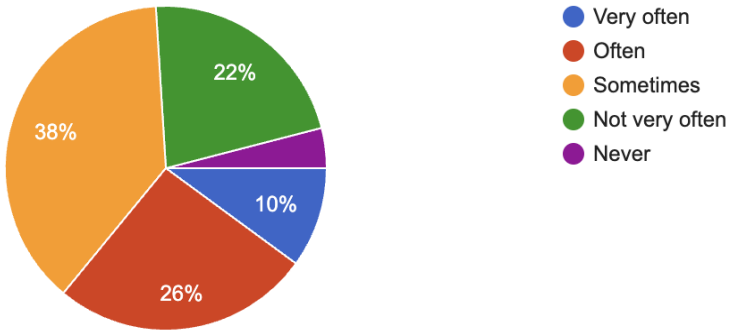


How would/does your manager react if/when they saw a bug or potential security vulnerability in your code (even if minor)?

In this question, our survey had a short answer requirement hence we had to aggregate it and found insightful results that managers tend to give high priority to security vulnerabilities.



How often does the amount of technical debt affect your mental health?



Compare/Contrast to the paper "Technical Debt Cripples Software Developer Productivity"

Wasted time

In order to determine how much of the total development time of software engineers is lost to TD and whether the distribution of the lost time exhibits any trends, we developed two distinct sets of questions. 47 developers filled out a general questionnaire during the longitudinal data collection phase to report the amount of time they lost working because of TD.

Developers are compelled to carry out supplemental measures in terms of carrying out extra tasks when they experience TD when working on developing software. Consequently, if the TD had not been there, none of these various activities would have been required. The respondents were questioned about the extra activities they engaged in throughout the phase of collecting longitudinal data. The responders decided which activities accounted for each reported occurrence's wasted time.
Meetings, followed by understanding code and additional refactoring, is the activity that is most strongly linked to wasted time.When absolutely no time was available, the "None" option was wasted.

There are numerous varieties of TD, including Code TD, Test TD, Architectural TD, Documentation TD, Requirement TD, and Infrastructure TD, and each of these TD types may have a varied level of ill effects on the amount of time spent. In this part, we examine how various TD types affect lost time as well as which TD type, as seen from the perspective of developers, has the most detrimental effects on spent time.

This inquiry will center on the awareness of the detrimental effects TD has on the routine software development work and whether (and how) the developers and managers value the knowledge of the lost time. Apart from taking part in this study, none of the developers specifically measured, recorded, or reported the time they wasted, yet they still believed they had a high level of awareness of the time they lost because of TD. Overall, managers and developers shared a similar perspective on the advantages of being aware of how much time is spent. The developers explained the advantages by saying that the quantifiable wasted time could aid in identifying and anticipating the need for additional quality enhancements. Some developers also emphasized the advantages of being able to motivate and defend it to their managers, as well as the advantages of having better forecasting and capacity planning.

This finding suggests that the current TD significantly increases the amount of time wasted during the total development work, even if this study did not examine whether and how the first introduction of TD influenced the productivity of the development work. Additionally, this suggests that if the software companies are unaware of this time and have not made

preparations for it, they could easily find themselves under time pressure and be compelled to add more TD. In fact, according to the developers, a quarter of all TD they encounter compel them to build more TD in order to fix the present TD. This finding suggests that the amount of TD may potentially expand continuously, and, in the worst case, this may result in a vicious cycle of TD growth, depending on the degree to which software companies are able to remediate TD.

Additional Activities

This study aimed to investigate which activities consumed the wasted time and whether the quantity of wasted time was associated with any particular activity. The outcome reveals that completing additional testing, followed by additional source code analysis and additional refactoring, is the task on which the extra time was typically spent. This result suggests that the time spent on these operations may be decreased if the systems lacked TD. Additionally, having to carry out many of these additional tasks throughout the software development could, as a result, be a sign that a system has TD as well as a sign of the level of interest.

Technical Debt types

Aside from requirement issues, the questionnaire reveals that all other TD types are significant and strongly correlated with the amount of wasted time, with source code TD having the strongest correlation with wasted time. This finding highlights the need of paying attention to all forms of TD. These findings may be explained by the fact that developers are more aware of Source Code TD and hence perceive the negative effects of it to be more pronounced. Additionally, the findings indicate that developers encounter fewer Infrastructure TD and Requirement TD, suggesting that developers are less likely to attribute their wasted time to those TD types. This finding further suggests that software businesses should concentrate on a variety of TD and, less frequently today, should not just concentrate on TD that is related to code.

Awareness and Challenges

The study also examines the developers' and their manager's levels of awareness of the time lost to TD, the advantages of this knowledge, and the internal messaging they use inside their businesses. The findings show that, despite their lack of attempts to monitor, track, or quantify it, software engineers are reasonably aware of the amount of time they waste during the development phase. However, the managers of developers are much less aware of how much time the developers waste and both professions appear to have different ideas about how much time is appropriate and inappropriate to spend on TD. Managers are unable to respond and take necessary action in response to the lost time if they are unaware of how much time engineers lose on software development due to TD. This implies that, in the worst-case situation, developer productivity and the amount of wasted time could actually grow. It is also observed that TD

affects' the mental health of software engineers to some extent with over 26% claiming it happens oftenly.

Future Work

To reproduce the findings in different geographic locations and software development cultures, more research is required.

Conclusions

This is the first continuous study on technical debt, and 53 developers answered questions regarding the amount of time they lose to TD, the additional tasks they engaged in during that time, and the specific types of TD that contributed to the lost time. This study offers proof that TD impedes software engineers by creating a lot of unnecessary time waste. This lost time has a detrimental impact on the software's viability and development productivity. This study demonstrates that TD also contributes to the requirement for executing time-consuming additional tasks, and developers report that TD wastes, on average, some of their working time to software development. Furthermore, developers frequently have to conduct additional testing, source code analysis, and refactoring due to the presence of TD during the development work. This study also demonstrates that, in some of the cases where developers come across TD, they are compelled to implement new TD as a result of the preexisting TD. The hardship of having to spread more TD serves as evidence of how contagious TD is. These results suggest that in order for software businesses to measure TD interest, they must be prepared with techniques and proactive management. Making smarter, more knowledgeable decisions to balance TD accumulation and repayment could be the outcome of such a plan.

CIS 5930 Productivity in Software Engineering Project Team Member Contributions

Aditi Allady

- Created the Github
- Created the Technical Debt Questionnaire Form
- Wrote some of the Technical Debt Questions
- Wrote some of the project proposal
- Asked people to answer a questionnaire in the class, and other people in other classes and also who work in the cs department to answer the questionnaire(got 17 people to answer)
- Sent a reminder to the participants who have not filled out the form yet
- Merged short answer question for better results visualization
- Wrote the compare and contrast, future work and conclusion section
- Proofread and formatted the report
- Participated in team meetings for general discussion and helping other teammates

Sineha Aneel

- Asked people to answer a questionnaire in the class, and other people in other classes and also who work in the cs department to answer the questionnaire(got 17 people to answer)
- Wrote some of the project proposal
- Wrote some of the Technical Debt Questions
- Sent a reminder to the participants who have not filled out the form yet
- Merged 4 short answer questions for better results visualization
- Created visualizations for merged questions
- Wrote the compare and contrast, future work and conclusion section
- Proofread and formatted the report
- Participated in team meetings for general discussion and helping other teammates

Jordan Gethers

- Asked people to answer a questionnaire in the class, and other people in other classes and also who work in the cs department to answer the questionnaire(got 17 people to answer)
- Wrote some of the project proposal
- Wrote some of the Technical Debt Questions
- Sent a reminder to the participants who have not filled out the form yet
- Merged short answer question for better results visualization
- Wrote the abstract, introduction and general analysis of the report
- Proofread and formatted the report
- Participated in team meetings for general discussion and helping other teammates

Questionare:
https://docs.google.com/forms/d/e/1FAIpQLScQ7yb1JHjtUS4z17GagzN9vAc30r5hBqCJL5JWb07GDyHBgg/viewform?usp=pp_url

Link to Paper:
https://dl.acm.org/doi/pdf/10.1145/3194164.3194178