2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) | 978-1-6654-6152-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/SEAA56994.2022.00054

The Impact of Forced Working-From-Home on Code Technical Debt: An Industrial Case Study

Ehsan Zabardast, Javier Gonzalez-Huerta
Software Engineering Research Lab SERL Sweden
Blekinge Institute of Technology
Karlskrona, Sweden
ehsan.zabardast@bth.se, javier.gonzalez.huerta@bth.se

Francis Palma
Department of Computer Science and Media Technology
Linnaeus University
Kalmar, Sweden
francis.palma@lnu.se

***Abstract*—*Background*: The COVID-19 outbreak interrupted regular activities for over a year in many countries and resulted in a radical change in ways of working for software development companies, i.e., most software development companies switched to a forced Working-From-Home (WFH) mode. *Aim*: Although several studies have analysed different aspects of forced WFH mode, it is unknown whether and to what extent WFH impacted the accumulation of technical debt (TD) when developers have different ways to coordinate and communicate with peers. *Method*: Using the year 2019 as a baseline, we carried out an industrial case study to analyse the evolution of TD in five components that are part of a large project while WFH. As part of the data collection, we carried out a focus group with developers to explain the different patterns observed from the quantitative data analysis. *Results*: TD accumulated at a slower pace during WFH as compared with the working-from-office period in four components out of five. These differences were found to be statistically significant. Through a focus group, we have identified different factors that might explain the changes in TD accumulation. One of these factors is responsibility diffusion which seems to explain why TD grows faster during the WFH period in one of the components. *Conclusion*: The results suggest that when the ways of working change, the change between working from office and working from home does not result in an increased accumulation of TD.**

***Index Terms*—Technical Debt, Empirical Study, Industrial Study, Case Study, COVID-19, Telework, Work From Home**

I. Introduction

Technical Debt (TD) is a metaphor to explain the long-term consequences of sub-optimal decisions taken to give priority to speed on deliveries [1] and deals primarily with non-visible aspects and issues of software development, and its maintenance [2]. Technical and design decisions influence the introduction of TD. However, we should not forget that those decisions are taken by humans (e.g., developers, architects, testers) in their daily work. The environment and the conditions in which the work is done can significantly impact whether or not we take the best solution at hand and, therefore, whether we incur TD. The communication and coordination among members within and outside the team in the organisation might be a factor that influences how developers incur and repay TD [3]. These working conditions might affect how and when we take actions to mitigate and payback TD.

In March 2020, with the COVID-19 pandemic, software development organisations had to confront a challenging situation: how to continue their activity working 100% on distance, with their employees working from home (WFH). This new way of working is different from distributed software development, in which teams are located in different sites. Therefore, in distributed software development, intra-team communication happens *face-to-face*. In contrast, inter-team communication usually should be mediated using instant messaging and video conference tools, while in WFH, even intra-team communication should be mediated with online tools. This sudden change exposed organisations to many challenges, uncertainties, and risks. There have been studies to analyse changes in the working routines of developers [4]–[7]. However, very little is known about the influence that WFH had on TD. Some companies suggested a change in their working strategy during the pandemic, enabling their employees to work from distance. Therefore, it is important to understand how software artefacts degrade considering different ways of working.

In this paper, we present the results of an industrial case study to analyse how the introduction and repayment of TD have fluctuated during the first nine months of *WFH* and compare it to the same interval working from *office*, by studying several repositories from a software development organisation. The case study comprises two different data collection methods. On the one hand, we collected quantitative data utilising archival analysis. On the other hand, we carried out a focus group with the development team to collect qualitative data. The goal of this mixed-method is to triangulate data to strengthen the evidence and identify the factors that impact TD. We address the following research question: **RQ:** *How the forced Working-From-Home mode impacted the accumulation of technical debt?* More specifically, whether and to what extent WFH impacted the accumulation of TD when developers have different ways to coordinate and communicate with peers.

The remainder of this paper is structured as follows. In Section II, we present the design of the study while Section III presents the results. Section IV discusses the main findings and

This research was supported by the KKS foundation through the SHADE KKS Hög project with ref: 20170176 and through the KKS S.E.R.T. Research Profile with ref. 2018010 project Blekinge Institute of Technology, SERL Sweden.

978-1-6654-6152-8/22/$31.00 ©2022 IEEE
DOI 10.1109/SEAA56994.2022.00054

Section V summarises the threats to validity. In Section VI, we discuss related work in the area. Finally, Section VII draws our conclusions and discusses further work.

II. Research Methodology

To address the research question, we carried out an industrial case study that combines archival and qualitative data collected from `git` repositories, SonarQube, and a focus

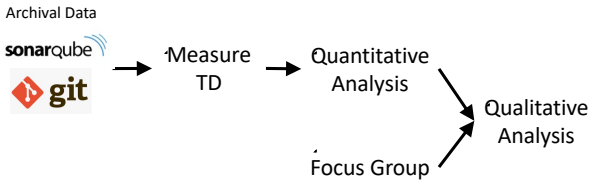

Fig. 1. The research methodology.

This study is conducted in an industrial setting and comprises the analysis of five components developed by a company that has chosen to remain anonymous. The company is a large software development company that develops web-based financial and accountancy services. It is a mature company in its development practices and has well-established, successful products. The case has been selected by convenience (availability and access). The company is interested in continuously improving its products and ways of working. Therefore, it was willing to participate in the study and learn from its results.

We are presenting an embedded case study [8] where we are analysing five components (units of analysis), which are mainly written in `Java`. The analysis period of all components spans two years (2019 and 2020).

In the following, we describe the main constructs and measurements used for the quantitative analysis in this research, as illustrated in Fig 2. We calculated the amount of accumulated TD per week for each component. We used SonarQube[1] to calculate the amount of TD in each component and calculated the amount of code using gitlog. The company uses SonarQube for controlling code quality and as a proxy for TD in the development process. Moreover, SonarQube has been used in similar studies on the topic of TD, e.g., [9], [10].

A. Data Collection

1) Quantitative Data: The data gathered from SonarQube is the issues detected by the quality profile and gate[2] selected by the company. The issues are representative of the quality profile and quality gate. The data is collected via the company's SonarQube API, i.e., the company uses the analysis of SonarQube as a proxy for the components TD. The tool provides the estimated time (i.e., effort in minutes) required to resolve the issues. The accumulated TD of a component

[1]Version 8.9.6 LTS at https://www.sonarqube.org
[2]https://docs.sonarqube.org/latest/user-guide/quality-gates/

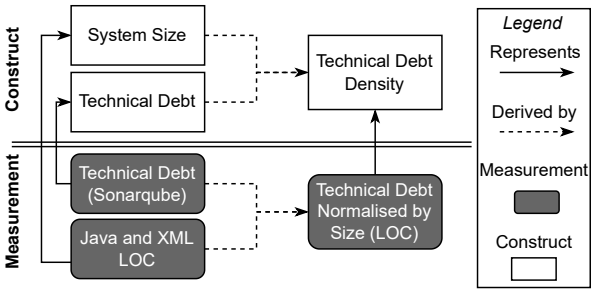

Fig. 2. Study constructs and measurements.

is the total remediation time for all the issues detected. We considered repaid TD when issues are tagged as 'fixed' and 'closed'.

To calculate TD density, we used the growth in the size of the system reported by git log, as added lines of code minus deleted lines of code. Table I presents the details of the investigated components in this study.

TABLE I
INVESTIGATED COMPONENTS' SIZE AND NUMBER OF COMMITS.

ID	Size (KLOC)	# commits in 2019	# commits in 2020
C1	6.2	314	292
C2	9	315	362
C3	4.8	163	352
C4	14	404	269
C5	6.5	149	499
Total	40.5	1,345	1,774

2) Qualitative Data: The results of the quantitative analysis were presented to the development team (see Table II) responsible for the development of the five components in a focus group session. The developers provided descriptions and explanations for what they thought were the reasons behind some of the most prominent changes in TD or size during the analysed period, and their experience with factors that impacted the accumulation of TD. The statements were collected via sticky notes and the transcript of the focus group session. The length of the focus group was 60 minutes, and six members (out of a total of 8 in the team) participated in the session. In order to complement the statements, we recorded and transcribed the conversation during the focus group. The focus group included the following steps:

TABLE II
FOCUS GROUP PARTICIPANT INFORMATION.

Role	Experience (years) In Company	Overall
Product Owner	8	21
Development Manager	2	18
Product Owner	4	9
Senior Developer	12	12
Senior Developer	2	6
Scrum Master	5	5

- Participants introduced themselves and their roles in the team.
- The authors of this paper presented a summary of the results and explained the focus group procedure.
- Participants answered questions regarding the results of the quantitative analysis.
  - Participants read each question. They had five minutes to think about each question. They wrote their answers and opinion on sticky notes and posted them at the end of the time altogether.
- Participants and researchers had a closing discussion where they discussed the details of the results.

B. Data Analysis

The data analysis is divided into two sections where we analyse the quantitative and qualitative data separately.

1) Quantitative Data Analysis: To evaluate the state of each component, we use $TD_{density}$. $TD_{density}$ is the normalised amount of TD per line of code [11], [12]. Plotting $TD_{density}$ over time will allow us to visualise the changes in TD density and their trend. The weeks from 2019-01-01 to 2020-03-11 were considered as *office* and the weeks from 2020-03-11 to 2020-12-31 were considered as WFH. The vertical black line is used to mark the week in which the company switched to forced WFH (i.e., it only applies to the year 2020).

For comparing whether there were differences in the TD density while working in the office vs WFH, we use the t-test for independent samples (at the significance of 0.05) in case the samples were normally distributed or the Mann-Whitney tests in case the samples were not normally distributed [13]. For assessing whether each sample (i.e., Office and WFH) were normally distributed, we used the Shapiro-Wilk Test [13]. The tests were conducted to compare the same interval (i.e., March to December) between *office* and *WFH* periods, although Figure 3 shows the two natural years.

2) Qualitative Data Analysis: The main input for the qualitative data analysis is the statements provided by the focus group participants. We have used *In Vivo Coding* to label the statements. In Vivo coding is suitable for labelling raw data in the first cycle coding. It prioritises the opinions of the interviewees [14]. The coding was done by two authors independently. Then we compared all the labels and discussed the conflicting cases with the help of the third author.

For the second cycle coding, we have used *Pattern Coding* to create the themes from the labels from the previous step. Pattern coding allows us to identify the themes from the data [14]. One of the authors extracted the emerging themes and later discussed them with all authors to agree on themes.

III. Results

A. Results from the Quantitative Data Analysis

Fig 3 illustrates the data for the quantitative data analysis, in which each row depicts a different component. The *blue* colour is used for year 2019 and *red* for year 2020. The vertical *black* line is used to mark the week in which the company switched to forced WFH—in 2020. The horizontal scale on each graph represents the weeks ranging from 1 to 52. For each component, we present the results using four different graphical representations (four columns). These columns are:

- **Column I: TD per Week** shows the amount of TD in a component per week. The positive and negative values represent introduced and repaid TD, respectively.
- **Column II: Accumulated TD** shows the accumulated TD during each year. The dashed blue and red lines are trends for the years 2019 and 2020, respectively, and have been calculated using the linear regression function of Scipy package[3], version 1.6.1 for Python 3.8.
- **Column III: Component Size Growth** shows the change in the component size during each year. The size is calculated only considering `Java` and `XML` files since the quality gate only considers issues related to `Java` and `XML`.
- **Column IV: TD Density** shows TD normalised per LOC during each year. The dashed blue and red lines are trends for the years 2019 and 2020, respectively, and have been calculated using the linear regression function of the Scipy package version 1.6.1 for Python 3.8. The trend lines in Column IV are all significant ($p - value < 0.5$). For **C1, C2, C3** and **C5**, the trend lines explain $< 50\%$ of the variance ($R^2 < 50\%$). However, for $C4_{WFH}$, the trend line is higher and explains 75% of the variance ($R^2 = 74.86\%$) and for $C4_{office}$ the trend line is slightly lower than 50% ($R^2 = 43.89\%$).

We observe statistically significant differences in the changes of $TD_{density}$ in all components when comparing 2019 and 2020. We observe that, when working from office, components **C1**, **C2**, **C3**, and **C5** incur more TD, i.e., on average $TD_{density}$ is higher during 2019. Therefore, the accumulated $TD_{density}$ introduced since the beginning of the year 2019 is higher than the year 2020 during the WFH period. Conversely, component **C4** incurs more TD when working from home, i.e., on average $TD_{density}$ is higher during 2020. Therefore, the accumulated $TD_{density}$ introduced since the beginning of the year 2019 is lower than the year 2020 during the WFH period.

TABLE III
DESCRIPTIVE STATISTICS FOR EACH COMPONENT FOR TD DENSITY.

Component	N	Min.	Max.	Mean	Median
$C1_{office}$	51	0.02	0.18	0.09	0.09
$C1_{WFH}$	46	0.04	0.1	0.05	0.04
$C2_{office}$	51	0.03	0.12	0.1	0.1
$C2_{WFH}$	52	0	0.09	0.07	0.08
$C3_{office}$	51	0.06	0.17	0.11	0.11
$C3_{WFH}$	50	0	0.06	0.03	0.03
$C4_{office}$	51	0.04	0.16	0.12	0.12
$C4_{WFH}$	50	0	0.37	0.27	0.3
$C5_{office}$	49	0	0.16	0.08	0.06
$C5_{WFH}$	50	0	0	0	0

Table III summarises the descriptive statistics for TD density per week. We applied the Mann-Whitney test, provided that

[3]https://www.scipy.org

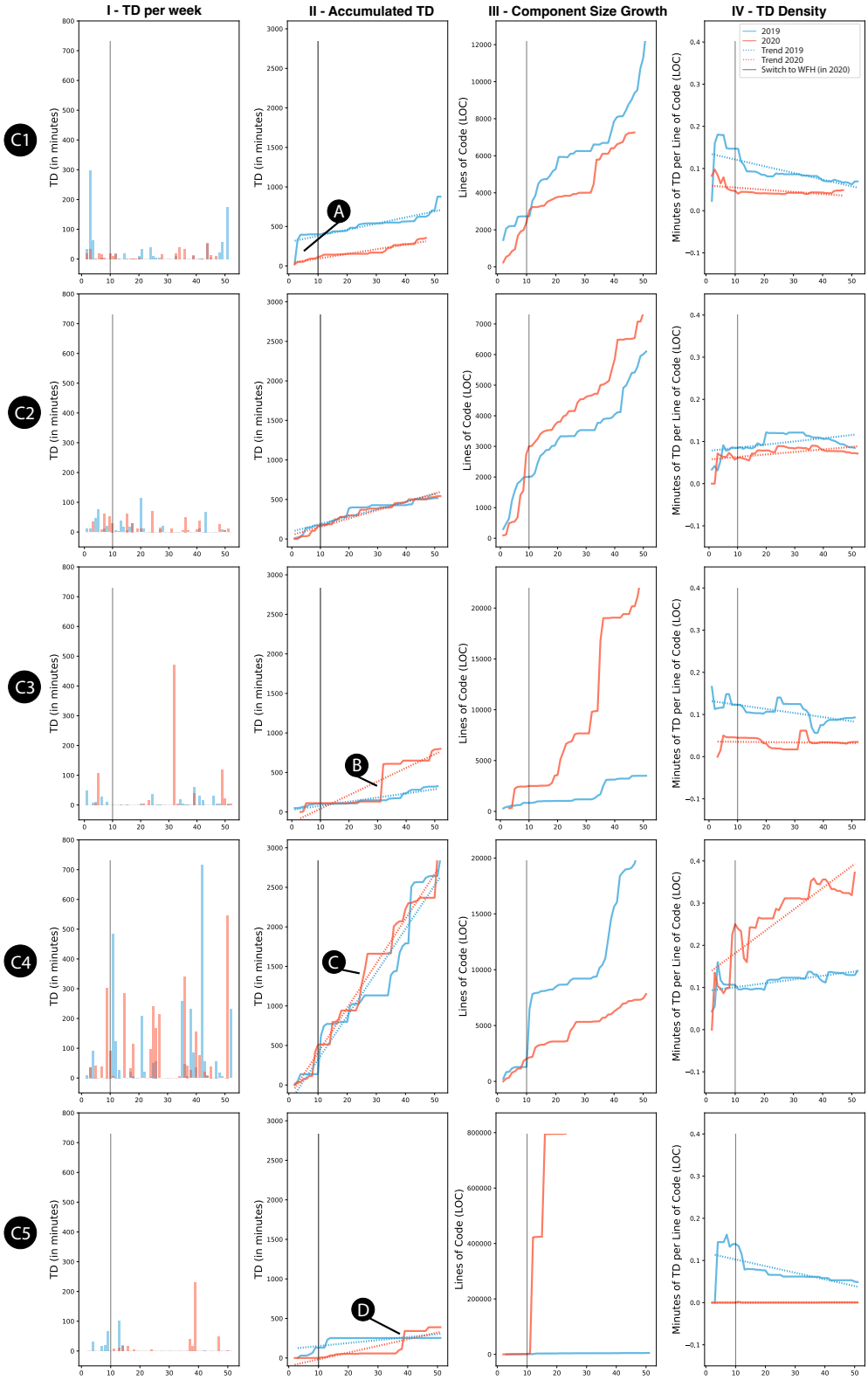

Fig. 3. Results of the quantitative analysis for five components. Column I shows TD per week; Column II shows the accumulated TD; Column III shows the component size growths; and Column IV shows the TD density. The vertical black line is used to mark the week in which the company switched to forced WFH (i.e., it only applies to the year 2020).

the samples were not normally distributed. The results of the Mann-Whitney test results are as follows:

- **C1.** There were *significant* differences in the introduced $TD_{density}$, which grew faster when working from office as compared to WFH ($W = 2203$, p=0.000).
- **C2.** There were *significant* differences in the introduced $TD_{density}$, which grew faster when working from office as compared to WFH ($W = 2307$, p=0.000).
- **C3.** There were *significant* differences in the introduced $TD_{density}$, which grew faster when working from office as compared to WFH ($W = 2544$, p=0.000).
- **C4.** There were *significant* differences in the introduced $TD_{density}$, which grew faster WFH as compared to working from office ($W = 263$, p=0.000).
- **C5.** There were *significant* differences in the introduced $TD_{density}$, which grew faster when working from office as compared to WFH ($W = 2404.5$, p=0.000).

B. Results from the Qualitative Data Analysis

In the focus group, we summarised the results presented in Section III-A to the developers, and asked them to elaborate on the results. We asked the participants to provide all the factors that impacted TD to help us understand and study the impact of those factors within the context of WFH. We recorded the number of participants who agreed on each statement (ratio of agreement in Fig 4). Example statements are provided.

- ***Question:*** *"Is there any explanation for the change of TD in each component or on the team's ways of working?"* The statements from developers suggest that there are several different reasons for the accumulation of TD in each component. The summary of possible explanations for the accumulation of TD in the investigated components are (ordered by level of agreement among participants):
  - *Changes in mob programming*[4] *(5.6/6):* The development team engaged in less mob programming. The developers state that they changed to individual programming, which is not as effective while working from home, and it impacts the accumulation of TD. *"a change from working in a mob towards working more individually"*
  - *Growth in the complexity (5/6):* Growth in complexity of the component makes TD growth faster. *"company growth, new products and projects"*
  - *Responsibility diffusion (5/6):* Having many external contributors to the repository, i.e., responsibility diffusion, impacts the accumulation of TD. *"[the component] has a lot of contributors outside our team"*
  - *Switch to another framework (4/6):* Migration to a new framework impacts the accumulation of TD. *"switch to [a new framework] lead to many changes over many components"*
  - *Growth in size (4/6):* Growth in the component via new development impacts the accumulation of TD. *"2020 seems to have had larger changes"*
  - *Multiple implementations (4/6):* Multiple implementations of (a part of) code base impact the accumulation of TD. *"multiple implementations [...] may increases TD"*
  - *Backward compatibility (4/6):* Backward compatibility might impact the accumulation of TD. *"backward compatibility may increase TD"*

There are points of interest in some of the charts presented in Fig 3, e.g., in component **C1**, Column II marked with *A* which is a significant increase in the introduction of TD at the beginning of both 2019 and 2020. To elaborate on such points of interest, we asked developers specific questions regarding such observations. These are four observations marked as *A, B, C,* and *D* in Fig 3.

- ***A: "Why are there steep increases in TD in both years at the beginning of the year?"*** The developers stated that this steep increase in TD correlates to the migration to a new framework and duplication of work because of "*project rework*". Ratio of agreement in the focus group for this answer was 5/6.
- ***"What do you think is the reason behind this steep increase in TD?"***
  - **B:** The developers state that this steep increase in TD correlates to the introduction of new code, refactoring, security maintenance work, code duplication, and TD on tests. Ratio of agreement in the focus group for this answer was 6/6.
  - **C:** The developers state that this steep increase in TD correlates to the duplication of functionality, TD on tests, development of new features and functionality, and refactoring. Ratio of agreement in the focus group for this answer was 6/6.
  - **D:** The developers state that this steep increase in TD correlates to the integration of a new external library to an already complex code. Ratio of agreement in the focus group for this answer was 6/6.

The statements provided by the participants were collected via sticky notes. We analysed the statement to extract labels and themes, as described in Section II-B. We extracted three main themes of factors that the participants in the focus group perceive might have an impact on the growth of TD including: source-code-related factors, laws-of-evolution-related factors, and ways-of-working-related factors. The results of the thematic analysis are summarised in Fig 4.

The *source-code*-related factors include code, architecture, and design decisions that impact the growth of TD such as *change of framework*, *multiple implementation of a component*, and *backward compatibility of a component*. The *laws-of-evolution*-related factors include continuous change and increasing complexity [15] that impact the growth of TD, such as *growth in size* and *growth in complexity*. The ways-of-working related factors include communication and

[4]Mob Programming is when the whole team works on the same thing, at the same time, in the same space, and on the same computer.

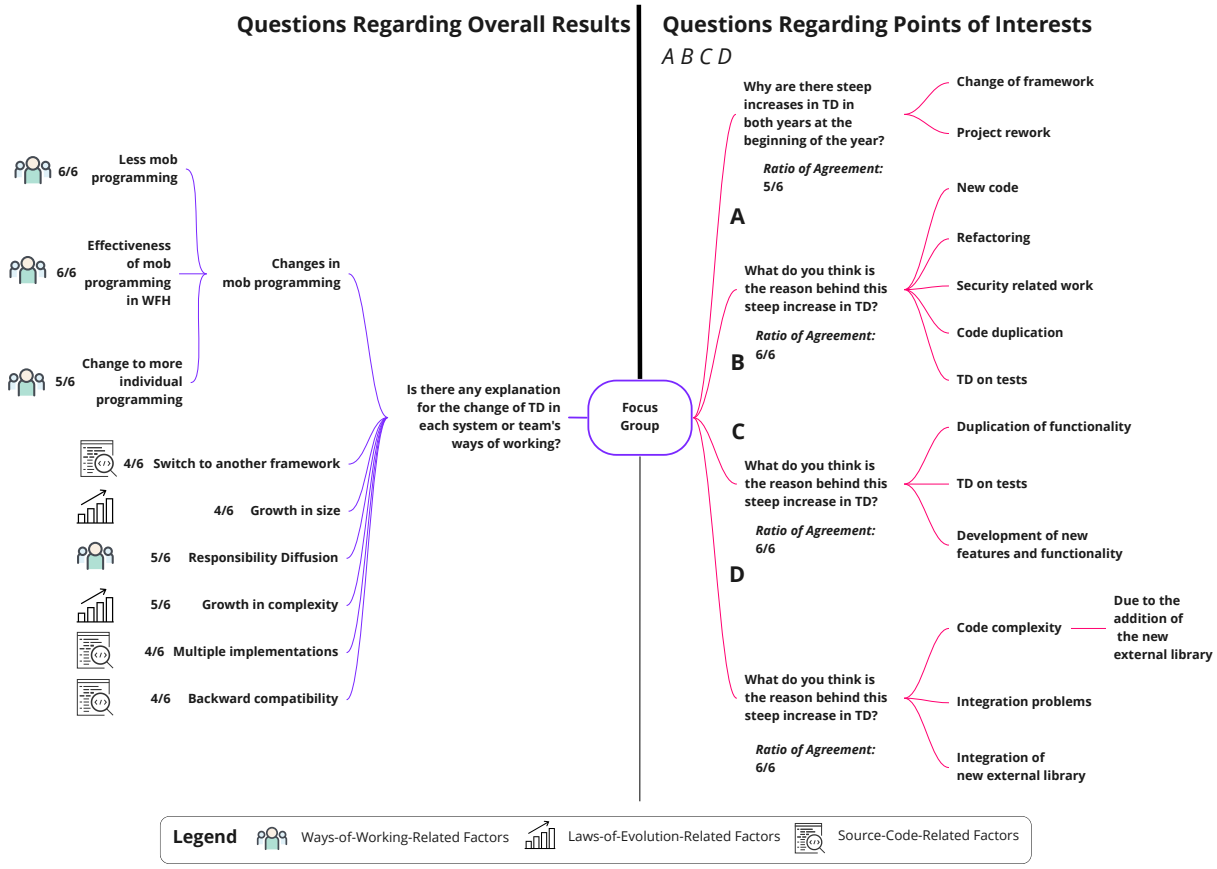

Fig. 4. Summary of the results of the qualitative analysis. The number next to the factors represent the ratio of agreement among the participants.

coordination of the team that impact the growth of TD, such as *changes in mob programming* and *contributions of developers outside of the team*.

Regarding *ways-of-working*-related factors, the participants in the focus group pointed out that the component **C4** does not have a clear ownership. The team that formally owns the component is not among the main contributors in terms of the number of revisions or the volume of code being contributed to the component (i.e., when analysing the whole history of the component, the *owning* team was the fourth in the number of total revisions). The *owning team* acts as *"gate keepers"* (i.e., in their own words) in the sense that members of this team are responsible for rating the pull requests, but the team does not have full control over the code being included in the component. This is a clear case of *responsibility diffusion* [16], where many other teams are contributing to a component, and no one takes the required code maintenance actions to keep the code in shape and/or repay TD. We observe that, apparently, in component **C4**, the effect of responsibility diffusion on TD seems to be exacerbated during the WFH period, with less pair or mob programming and more difficult inter-team communication.

IV. Discussion

As highlighted in Section III, the week-by-week statistical analysis show statistically significant differences. TD is accumulating at a faster pace in the component **C4** during the forced WFH period while in the other components TD seems to accumulate faster when working from office. The developers expected the decrease in TD accumulation: *"A gut feeling is that technical debt should not have increased during WFH - rather decreased. Sharing screens and discussing feels more natural when working remotely."* Our results show that the practices used by the developers during WFH such as screen sharing and impromptu meetings have helped decrease the accumulation of TD.

We investigated the faster increase of TD growth in component C4. Further investigation of the contributions revealed that developers from other teams are the main contributors to this component. There are 77 developers from 17 teams contributing component **C4**. The impact of responsibility diffusion on the growth of TD is highlighted as a major factor by the developers. Tornhill [16] suggests that the developers who do not own the *code* tend to disregard "*responsibility for the quality and future of a piece of code*" which is aligned with our observations and results. The gatekeeping solution

can also, in the long run, have negative consequences for the system due to review fatigue [16].

Overall, the results suggest that even when the ways of working change, e.g., less mob programming, the change between working from office and working from home does not result in increasing accumulation of TD. However, in cases where there are many different teams contributing to a component and there is *no clear ownership* of the component, the change from working from office to working from home may result in an increase pace in the accumulation of TD.

The results of the qualitative data analysis show that different factors can impact the accumulation of TD. These potential factors can be divided into *source-code*, *laws-of-evolution*, and *ways-of-working*-related factors. While we expect all the potential factors to impact TD, we observe that the impact of ways-of-working related factors can be exacerbated by forced WFH mode in the components with *less clear ownership*, as stated by the participants in the focus group.

V. Threats to Validity

In this section, we discuss the main threats to validity from four different perspectives [8]: construct validity, internal validity, external validity, and reliability.

Construct validity is the degree to which the measures studied can reflect the constructs that the researcher is aiming at and have been defined in the research questions [17]. We use SonarQube, which is a widely-adopted, *de-facto* standard used in industry to search for TD [18], and that has been used in similar other research studies (e.g., [10], [12]). We also rely on a metric $TD_{Density}$ [11], that has been used to monitor the impact of external factors (e.g., Clean Code) in [9]. The calculation of $TD_{Density}$ also takes the component's size as input, and for its calculation, we considered `Java` (source code and tests) and `XML` files (integration tests). All TD items have the same weight in our analysis and they are considered equally important. To mitigate potential threats regarding the size calculations, the selection of the file types was made in consensus with the heads of development in the studied company. The repaid TD is calculated through the issues flagged as 'fixed' and 'closed'. However, there might be cases where TD has been removed by deleting the files that might not be included in our analysis. Finally, we applied a set of commonly used statistical tests employed in the empirical SE community that relate to the constructs described in Figure 2.

Internal validity concerns whether there might be other factors that might explain some of the findings (e.g., the company is migrating to use a new framework which might impact the amount of accumulated TD). We have used a methodological triangulation [19], to try to mitigate those threats, using quantitative and qualitative sources. From the qualitative data, we understood that, for example, the lack of formal ownership and the usage of a quality gate might explain the uncontrolled growth of TD in the WFH period. However, there might be other reasons, and further studies are needed to understand better additional factors that can explain this exacerbated growth. There might be other non-studied factors that can explain the differences in TD during the studied periods.

External validity concerns the generalisability of the results into different contexts out of the investigated cases. Our results are strictly applicable to the studied case. One of the common misunderstandings about case study research is the inability to generalise from single cases [20]. While interpreting the results, we do not focus on statistical generalisation. However, we focus on comparing different components and searching for plausible explanations, trying to extract themes that can explain the observed differences.

Reliability concerns whether the data and the analysis are independent of the specific researchers. This might be a significant threat to the validity of this study. We tried to mitigate this threat by first involving the studied company in interpreting the results instead of restricting the analysis to a pure comparison of two data frames. Second, we tried to improve the reliability of the qualitative analysis by doing a separate blinded coding.

VI. Related Work

This section summarises studies from findings on WFH regarding software development and productivity, working patterns and TD, and measuring the TD trend.

Studies Analysing the Impact of WFH: One of the earliest studies during the COVID-19 pandemic by Bao et al. [6] analysed developers' activity records to understand the impact of WFH on productivity. The interest group is 139 developers in Baidu Inc, China. The dataset is obtained between Dec. 2019 and Mar. 2020 and compared with the corresponding months in 2018 and 2019. However, no generalised observation could be made, i.e., although WFH has different impacts on developers' productivity, this varies by the project age, type, language, and size. Also, the productivity of the majority of the developers when WFH is similar to the office. Smite et al. [21] studied work patterns for a geographically distributed software development company to understand how employees cope with the WFH mode. Findings suggested that during the WFH, engineers follow daily routines similar to onsite work. Moreover, there is no significant change in code production comparing the onsite and WFH code commits. The authors reported an increased flow and productivity in some instances (e.g., familiarity with the tasks) due to a lack of spontaneous interruptions from colleagues and shorter breaks.

Working Patterns and TD: In an industrial case study, Codabux and Williams [22] reported that developers primarily focus on design, testing, and defect-related TD in an agile setting. In the agile and distributed environment, developers often create their taxonomy of TD subject to the type of tasks they were assigned and their understanding. Yli-Huumo et al. [23] provided an exploratory, qualitative case study interviewing 25 engineers in eight development teams and analysing empirical data to understand TD management. TD management was done systematically for most of the teams

working in a distributed environment, i.e., managers reserved 20% of the development time to improve the code base and refactor TD issues.

Measuring the TD Trend: When it comes to analysing TD in a project, there are tools (e.g., SonarQube) that report the number of TD items. These tools can use that information to estimate the total time to remediate them, approximating the TD principal accumulated in the project. However, using this information to assess the TD evolution over time might be problematic, especially if we want to establish comparisons between two different periods or compare evolution across different systems. If we compare two periods of time in which the development activity resulted in the volume of code being added to the systems differs substantially, analysing only the accumulated TD principal might lead us to the wrong conclusions.

The density of TD or $TD_{density}$ [11], [12] defined as the TD normalised per line of code, can allow us to reason about the evolution and the impact that external factors might have on the accumulation of TD [11], [12]. In [12] analyses the impact that clean code might have on TD. In this particular case, $TD_{density}$ reflects how the TD density tends to decrease when the newly added code is cleaner. In contrast, the deletion of high-quality code causes an increase in TD density. Thus, the system's maintainability during evolution might be strongly associated with the $TD_{density}$ [12].

With forced WFH mode and the newly adopted work from anywhere, the state of TD needs to be investigated in-depth. The ways of working impact TD, e.g., whether developers are working remotely or at the designated work-space co-located with their peers [3]. Using $TD_{density}$ can allow us to draw better conclusions regarding its potential impact.

VII. Conclusion

TD reflects the long-term consequences of sub-optimal decisions taken to accomplish short-term design and development goals [1]. Due to the global COVID-19 outbreak, developers are forced to work from home worldwide. We investigated the impact of forced WFH on the accumulation of TD. We showed the accumulation of the TD in terms of TD density.

We analysed five components to study the impact of forced WFH on the accumulation of TD in an industrial case. While comparing the accumulation of TD between the pre-pandemic (in 2019) and pandemic (in 2020) times, the tests showed statistically significant increase in the accumulation of TD in one component identified with lack of ownership and responsibility diffusion.

Based on the focus group, we identified that factors related to *source-code*, *laws-of-evolution*, and *ways-of-working* are among the factors that impact the growth of TD. We plan to extend and replicate this study on other components and in other organisations, which may strengthen the evidence regarding the impact of WFH on the accumulation of TD. We want to examine other confounding factors (e.g., size of the systems, developers' experience, working conditions at home, and delivery pressure) in relation to forced WFH and TD accumulation. Finally, we plan to explore how misalignment between ownership and contribution impacts TD.

References

[1] W. Cunningham, "The wycash portfolio management system," *ACM SIGPLAN OOPS Messenger*, vol. 4, no. 2, pp. 29–30, 1993.

[2] P. Kruchten, R. Nord, and I. Ozkaya, *Managing Technical Debt: Reducing Friction in Software Development*. Pearson, 2019.

[3] R. Bavani, "Distributed agile, agile testing, and technical debt," *IEEE software*, vol. 29, no. 6, pp. 28–33, 2012.

[4] D. Smite, N. B. Moe, E. Klotins, and J. Gonzalez-Huerta, "Work Patterns of Software Engineers in the Forced Working-From-Home Mode," *arXiv preprint arXiv:2101.08315*, 2021.

[5] P. Ralph, S. Baltes, G. Adisaputri, R. Torkar, V. Kovalenko, M. Kalinowski, N. Novielli, S. Yoo, X. Devroey, X. Tan *et al.*, "Pandemic programming: how covid-19 affects software developers and how their organizations can help," *arXiv preprint arXiv:2005.01127*, 2020.

[6] L. Bao, T. Li, X. Xia, K. Zhu, H. Li, and X. Yang, "How does working from home affect developer productivity?–a case study of baidu during covid-19 pandemic," *arXiv preprint arXiv:2005.13167*, 2020.

[7] N. Forsgren, "Octoverse spotlight: An analysis of developer productivity, work cadence, and collaboration in the early days of covid-19," 2020.

[8] R. Yin, *Case Study Research: Design and Methods*, ser. Applied Social Research Methods. SAGE Publications, 2009.

[9] G. Digkas, M. Lungu, P. Avgeriou, A. Chatzigeorgiou, and A. Ampatzoglou, "How do developers fix issues and pay back technical debt in the apache ecosystem?" in *2018 IEEE 25th Int. Conf. on Software analysis, evolution and reengineering (SANER)*. IEEE, 2018, pp. 153–163.

[10] E. Zabardast, J. Gonzalez-Huerta, and D. Šmite, "Refactoring, bug fixing, and new development effect on technical debt: An industrial case study," in *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2020, pp. 376–384.

[11] M. A. Al Mamun, A. Martini, M. Staron, C. Berger, and J. Hansson, "Evolution of technical debt: An exploratory study," in *2019 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement, IWSM-Mensura 2019, Haarlem, The Netherlands, October 7-9, 2019*, vol. 2476. CEUR-WS, 2019, pp. 87–102.

[12] G. Digkas, A. N. Chatzigeorgiou, A. Ampatzoglou, and P. C. Avgeriou, "Can clean new code reduce technical debt density," *IEEE Transactions on Software Engineering*, 2020.

[13] A. Field, *Discovering statistics using IBM SPSS statistics*. Sage, 2018.

[14] J. Saldaña, *The coding manual for qualitative researchers*. Sage, 2015.

[15] M. M. Lehman, "Laws of software evolution revisited," in *European Workshop on Software Process Technology*. Springer, Berlin, Heidelberg, 1996, pp. 108–124.

[16] A. Tornhill, *Software Design X-Rays: Fix Technical Debt with Behavioral Code Analysis*. Pragmatic Bookshelf, 2018.

[17] P. Ralph and E. Tempero, "Construct validity in software engineering research and software metrics," in *22nd International Conference on Evaluation and Assessment in Software Engineering*. Christchurch, New Zealand: Association for Computing Machinery (ACM), 2018.

[18] A. Martini, T. Besker, and J. Bosch, "Technical debt tracking: Current state of practice a survey and multiple case study in 15 large organizations," *Science of Computer Programming*, vol. 163, pp. 42–61, 2018. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167642318301035

[19] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley and Sons, 2012.

[20] B. Flyvbjerg, "Five misunderstandings about case-study research," *Qualitative Inquiry*, vol. 12, no. 2, pp. 219–245, 2006.

[21] D. Smite, N. B. Moe, E. Klotins, and J. Gonzalez-Huerta, "Work patterns of software engineers in the forced working-from-home mode," *arXiv preprint arXiv:2101.08315*, 2021.

[22] Z. Codabux and B. Williams, "Managing technical debt: An industrial case study," in *2013 4th International Workshop on Managing Technical Debt (MTD)*. IEEE, 2013, pp. 8–15.

[23] J. Yli-Huumo, A. Maglyas, and K. Smolander, "How do software development teams manage technical debt?–an empirical study," *Journal of Systems and Software*, vol. 120, pp. 195–218, 2016.