

The Type to Take Out a Loan? A Study of Developer Personality and Technical Debt

Lorenz Graf-Vlachy*[†] and Stefan Wagner*

**Institute of Software Engineering, University of Stuttgart*

Stuttgart, Germany

[†]*TU Dortmund University*

Dortmund, Germany

{lorenz.graf-vlachy|stefan.wagner}@iste.uni-stuttgart.de

Abstract—Background: Technical debt (TD) has been widely discussed in software engineering research, and there is an emerging literature linking it to developer characteristics. However, developer personality has not yet been studied in this context. **Aims and Method:** We explore the relationship between various personality traits (Five Factor Model, regulatory focus, and narcissism) of developers and the introduction and removal of TD. To this end, we complement an existing TD dataset with novel self-report personality data gathered by surveying developers, and analyze 2,145 commits from 19 developers. **Results:** We find that conscientiousness, emotional stability, openness to experience, and prevention focus are negatively associated with TD. There were no significant results for extraversion, agreeableness, promotion focus, or narcissism. **Conclusions:** We take our results as first evidence that developer personality has a systematic influence on the introduction and removal of TD. This has implications not only for future research, which could, for example, study the effects of personality on downstream consequences of TD like defects, but also for software engineering practitioners who may, for example, consider developer personality in staffing decisions.

Index Terms—Technical debt, personality, five factor model, Big Five, regulatory focus, narcissism

I. INTRODUCTION

A. Technical Debt and its Consequences

Ever since Cunningham famously remarked that “Shipping first time code is like going into debt” [1, p. 30], the notion of “technical debt” (TD) has gained traction in practitioner circles and academia alike [2]. Definitions of TD vary slightly, but researchers have converged on the notion that TD is the result of making technical compromises that give rise to a “collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible.” Consequently, TD “presents an actual or contingent liability” [3, p. 112].

While TD can be beneficial, especially in terms of developer velocity, the literature frequently discusses its potential downsides [4]. One of the most frequently mentioned adverse consequences of TD is its negative impact on developer morale [5]. Developers generally dislike incurring TD [6], and it has been found to be psychologically taxing [7]. This leads to TD having demotivating effects [2]. Others have widened the scope and considered other consequences like reduced developer productivity due to reduced maintainability

of code, decreased code quality (leading to more defects and subsequent costs), as well as increased uncertainty and risk [8].

B. Technical Debt as Risk-Taking to be Managed

In fact, to at least a substantial degree, the issue of TD can be interpreted as a matter of risk-taking. Prior scholars have clearly established, and it follows from the definition provided above, that TD is about making trade-off decisions [3], [9]. For one, such decisions about TD are a trade-off between saving effort in the present and having to repay this principal with interest in the future. Consequently, they constitute risk-taking, since the implied interest rate can only become fully clear in the future, which is uncertain and thus risky [10], [11]. For another, however, the risky nature of TD is exacerbated by the fact that not only is the interest amount unclear, but there is a “possibility that debt may never need to be paid back” [12, p. 52] in the first place. Perhaps one of the most lucid discussions of this issue can be found in Schmid’s work on the limits of the TD metaphor, in which he distinguishes between “potential technical debt”, which he explicitly characterizes as being “akin to a risk”, and “effective technical debt” [10, p. 64], which is the amount of TD that is actually relevant to the future evolution of a software system (i.e., the part of it that will actually require future remediation efforts, but which can of course not be perfectly identified and anticipated).

Correspondingly, TD is often viewed as something that may need to be paid down at some point, but that, ad minimum, should be very consciously managed. At least two key strands of literature exist in this context. First, researchers have spent considerable effort on measuring, that is, assessing the magnitude, of TD in a system. Researchers developed different indicators, for instance, code smells [13]. Others developed various tools to automatically measure TD, for example using code metrics [14]. Several such tools, such as SonarQube, provide composite quality indicators of maintainability, which are particularly appreciated by junior developers [15]. Ultimately, researchers unsurprisingly made considerable efforts to estimate TD costs [4], [16].

Second, researchers have concerned themselves with identifying, prioritizing, and paying down TD. Digkas *et al.*, for instance, studied how technical debt is handled [17] and paid back [18] in the Apache ecosystem. Nayebi *et al.* provide an

extensive longitudinal case study on how architectural TD is identified and paid down in the context of a healthcare communications product. Maipradit *et al.* recently developed an automated classifier to identify self-admitted TD in code [19]. Going beyond such individual studies, Alves *et al.* provide an overview of tools to identify and manage TD, including, for example, cost-benefit analysis and portfolio approaches [13]. Closely related, recent reviews of the work on the prioritization of TD (both against other TD or against new features) identified a set of strategies to prioritize and manage TD. Alfayez *et al.*, for example, identified 24 different strategies [20], whereas Lenarduzzi *et al.* aggregate the strategies they identified into four broader categories and one additional category of combination approaches [21].

C. Antecedents of Technical Debt

In the quest to best manage TD, researchers have also explicitly begun to explore its causes and antecedents. Some researchers have, for instance, identified factors such as firms' business model innovation as a driver of TD [22]. Others found, for example, the number of commits and the number of lines of code (LOC) in a project to be positively related to TD [23]. Yet another stream of research has focused on describing which incentives and punishments companies employ to motivate developers to avoid or pay down TD [24]. Overall, a myriad of factors appear to influence TD. Rios *et al.*, for example, performed a qualitative study and identified 57 factors that drive TD, with time pressure due to looming deadlines ranking highest [25]. In a large survey, Rios *et al.* later increased that number to 78 [26]. A multi-country replication of this survey subsequently derived eight overarching categories of antecedents of TD, and again identified deadlines as the most important cause of TD [2]. Interestingly, the perception among practitioners regarding the key causes of TD changes with experience. More experienced software developers focus less on technical issues as causes of TD, and more on human factors [27].

As is thus perhaps to be expected, one specific stream of research has begun to focus on individual developers to understand the origins of TD. Amanatidis *et al.*, for instance, studied various PHP projects and found that individual developers differ substantially with regard to the amounts of TD they incur [28]. Alfayez *et al.* and Codabux and Dutchyn reaffirmed these findings in other codebases and went further in explicitly profiling individual developers, identifying various characteristics that are predictive of their actions regarding the introduction and removal of TD [29], [30].

However, thus far, we are not aware of any research that links developers' personality to TD. This is surprising because it is a core tenet of psychological research that personality is strongly predictive of a wide variety of individuals' behaviors [31] and one might therefore expect it to also have a bearing on whether a given developer is more or less willing to incur or pay back TD, which in turn may have important consequences, for example, for staffing project teams. Consequently, in this study, we ask the following

research question: *How is developer personality related to introducing and removing TD?*

II. BACKGROUND AND HYPOTHESES

In this paper, we study several different aspects of developer personality and their relationship with technical debt. Specifically, we consider the personality traits covered by the Five Factor Model, regulatory focus, and narcissism. Below, we lay out the theoretical background of each, provide an overview over its prior use in software engineering research, and develop hypotheses regarding how each trait might be related to changes in technical debt.

A. Five Factor Model Personality Traits

The Five Factor Model (FFM; sometimes also referred to as the "Big Five") is arguably the most established personality trait model in psychology [32]–[34]. It was developed using a psycholexical approach and comprises five broad traits—extraversion, agreeableness, conscientiousness, emotional stability (or its inverse, neuroticism), and openness to experience—that are rather stable across different situations. The traits are universal in that empirical evidence shows that they are equally valid for different sexes, races, cultures, and age groups [35].

The FFM has been used extensively in software engineering research. The following is thus only an illustrative treatment of more recent works.¹ Rastogi and Nagappan [38], for instance, studied the FFM personality traits of GitHub contributors. They found that developers who contributed more scored higher on openness to experience, conscientiousness, and extraversion, but lower on emotional stability and agreeableness. However, Calefato *et al.* [37] later repeated this study with an improved personality measure and found no such effects.

Karimi *et al.* [39] studied the relationship between personality and programming style and performance (including code quality) in student programmers. They found that openness to experience was positively associated with a breadth-first programming style and conscientiousness was positively associated with a depth-first style. They did not explicitly link FFM traits to programming performance, but found that a breadth-first programming style is linked to superior performance.

Paruma-Pabón *et al.* [40] captured developer FFM personality, as well as developers' needs and values, from software project mailing list emails, and clustered developers with similar personality types. They demonstrated that the personality of developers with commit privileges was linked to their behavior within the projects.

Relatedly, Calefato *et al.* [37] studied the personality of Apache developers using the FFM and found that there were three common types of personality profiles. Neuroticism and agreeableness were the two traits most important for differentiating the profiles from one another. They also found that FFM traits did not vary by developer role, membership, or degree of contribution to their project, and they demonstrated

¹Several recent journal articles provide fairly extensive reviews of the use of the FFM in software engineering research [36], [37].

that developer personality was time-invariant. Further, they found a positive association between developers' openness to experience and the likelihood of making project contributions.

Finally, Iyer *et al.* [36] examined the acceptance of pull requests in open-source projects. They found that pull requests from authors who score higher in openness and conscientiousness were more likely to be approved. They further found that extraversion was negatively related to the chance of approval. In addition, they found that pull requests which are closed by programmers who score higher on conscientiousness and extraversion had a greater chance of acceptance. Emotional stability of the closer, in contrast, was linked to a reduced likelihood of acceptance.

In the following, we will link each FFM personality trait in software developers to induced TD. First, extraversion indicates the degree of engagement with the external world. Sub-facets of extraversion include friendliness, gregariousness, assertiveness, activity, and excitement-seeking. Extraverted persons can thus be described as outgoing, often feeling positive emotions, and seeking stimulating activities. In comparison, introverted persons can be described as less outgoing, shy, and preferring to spend time alone [32], [33], [41]. Since more extraverted developers are more prone to activity and excitement-seeking, we propose that they "move fast and break things", leaving behind more TD than introverted developers. Extant literature further suggests that extraversion is positively related to risk-taking [42]. Given that prior qualitative software engineering research argued that "having a higher risk appetite can influence decisions to create technical debt" [8, p. 1504], this further indicates that extraversion and TD might be positively connected. Formally put:

Hypothesis 1 (H1): Extraversion will be positively associated with induced TD.

Agreeableness refers to the degree of concern with cooperation and social harmony. Sub-facets of agreeableness include trust, morality, altruism, cooperation, modesty, and sympathy. Agreeable persons can be described as friendly, helpful, and understanding. In comparison, non-agreeable persons can be described as less friendly, not very cooperative, and initiating disagreements [32], [33], [41]. Therefore, we expect more agreeable developers to wish to support others in improving code, thus paying down TD instead of adding to it. The literature on risk-taking also suggests that agreeableness is linked to less risk-taking [42]. We therefore conjecture:

Hypothesis 2 (H2): Agreeableness will be negatively associated with induced TD.

Conscientiousness indicates the degree of control over human impulses. Sub-facets of conscientiousness include orderliness, reliability, achievement-striving, and self-discipline. Thus, conscientious people can be described as prudent, organized, and reliable. In comparison, non-conscientious people can be described as impulsive, unsystematic, and less reliable [32], [33], [41]. Consequently, we expect more conscientious developers to dislike TD, and to induce little of it, or even remove it. The psychological literature on risk-taking takes a similar stance and finds an overall negative relationship

between conscientiousness and risk-taking [42]. We posit:

Hypothesis 3 (H3): Conscientiousness will be negatively associated with induced TD.

Emotional stability is often described through its inverse, neuroticism. Neuroticism indicates the degree to which a person experiences negative feelings. Sub-facets of neuroticism include anxiety, anger, depression, self-consciousness, and vulnerability. Neurotic people can be described as tense, often in a bad mood, and emotional. In comparison, emotionally stable people can be described as calm and free from a persistent bad mood [32], [33], [41]. Neurotic individuals tend to act decisively to remedy the anxiety they are prone to experiencing in uncertain situations [43] and to minimize potential threats that may materialize, even if this means accepting some concrete negative outcomes [44]. In other words, they frequently resort to a "better safe than sorry strategy" [45, p. 1005] in uncertain situations. As discussed above, choices on TD are always made under uncertainty and involve trading off today's gains for potential future liabilities [11], suggesting that neuroticism may lead to choices that avoid uncertainty by avoiding TD. The psychological literature further discusses a potential link between neuroticism and self-oriented perfectionism [46], which may lead programmers to write whatever they perceive to be "better" code, which may well be code with less TD. Again, the literature on risk-taking concurs with this idea and shows positive links between emotional stability and risk-taking [42]. We thus hypothesize:

Hypothesis 4 (H4): Emotional stability will be positively associated with induced TD.

Openness to experience describes the degree to which individuals are imaginative and creative. Sub-facets of openness include imagination, artistic interest, adventurousness, liberalism, and intellect. Persons high in openness can be described as individualistic, non-conforming, and aware of their feelings. In contrast, persons low in openness can be described as down-to-earth, conventional, and less aware of their feelings [32], [33], [41]. Consequently, we propose that developers higher in openness to experience will feel less bound to ideas of writing supposedly "clean" code, and are thus more likely to induce TD. Furthermore, prior literature has linked openness to experience with increased risk-taking [42]. We thus propose:

Hypothesis 5 (H5): Openness will be positively associated with induced TD.

B. Regulatory Focus

Despite the comprehensiveness of the FFM, it is common in the psychology literature to study it in conjunction with other traits [47]. We thus also study regulatory focus, an established construct in personality psychology that has not yet found any attention in software engineering research. A person's regulatory focus consists of two independent self-regulatory orientations, or foci, that shape their goal-striving behavior: Prevention focus and promotion focus [48], [49].² Promotion

²Note that regulatory focus can be conceptualized as a state and a trait. As is also often done in the psychological literature, we only focus on the latter component, sometimes also referred to as "chronic" regulatory focus [48].

focus is related to being eager, risky, and oriented towards attaining gains as positive outcomes. In contrast, prevention focus is related to being careful, cautious, and oriented toward avoiding losses as negative outcomes. Given that incurring TD constitutes risk-taking, and in line with literature that links regulatory focus to risk-taking [50], we thus propose that more promotion-focused developers may be more likely to induce TD, and more prevention-focused developers less so.

Hypothesis 6 (H6): Promotion focus will be positively associated with induced TD.

Hypothesis 7 (H7): Prevention focus will be negatively associated with induced TD.

C. Narcissism

Although there is an ongoing debate about the precise definition of the construct [51], scholars broadly agree that narcissism is a personality trait that combines “a grandiose yet fragile sense of self and entitlement as well as a preoccupation with success and demands for admiration” [52, pp. 440–441]. While various blog posts about narcissistic software developers suggest that narcissism may be a topic of interest in software engineering practice, we are not aware of any extant scientific work that explicitly studies this personality trait in software engineers. Prior research in psychology, in contrast, is plentiful and has found that narcissism is related to a host of correlates, for example status-seeking [53], having little regard for others’ concerns, and viewing others as inferior [54].

A particularly prominent correlate of narcissism is risk-taking. Specifically, researchers have found associations of narcissism with a host of risky activities. These include, for instance, risky sexual behavior and sexual aggression, aggressive driving, drug and alcohol use, compulsive exercise, and gambling [55]. Given the nature of incurring technical debt as inherently risk-taking, we propose that more narcissistic developers are likely to induce more TD in their commits.

Hypothesis 8 (H8): Narcissism will be positively associated with induced TD.

III. METHOD

A. Technical Debt Measures

To measure technical debt, we relied on the “Technical Debt Dataset” in version 2 [56]. This dataset contains a comprehensive analysis of the master branches of 29 Apache open source projects using the popular tool SonarQube, and has been used in prior research [30].

Our primary measure of induced TD in a focal commit follows prior work in relying on the estimated remediation effort for all maintainability issues as identified by SonarQube [29]. Specifically, the value for the focal commit is the difference in remediation effort estimates between the focal commit and its parent commit. Positive values thus indicate increases in technical debt, and negative values indicate technical debt being paid down. In a secondary, broader measure of induced TD we additionally include the estimated remediation effort for all reliability and security issues as identified by SonarQube.

Because ratios are potentially problematic as dependent variables [57], we deviate from prior work [29] and do not divide the TD measures by the difference in lines of code (LOC) between focal and parent commit. Instead, and arguably more precisely, we control separately for the LOC added and the LOC removed in the focal commit. This accounts for the fact that more LOC may make technical debt more likely [29].

B. Survey Measures

To obtain personality data, we surveyed the developers who made at least one commit in the Technical Debt Dataset. As the dataset itself does not contain contact information of developers, the first step was to obtain information on all commits of the projects from GitHub using PyDriller [58]. This yielded 99,972 commits, which partially extended beyond the timeframe of the dataset. We identified all individuals listed as “authors” in the data, and manually cleaned the list to remove duplicates. We also merged records where individuals used different names but the same email address, or different email addresses but the same or an extremely similar name for different commits. When judgment was needed, we made decisions as conservatively as possible. We ultimately obtained a list of 1,555 unique individuals. In case of multiple email addresses per person, we selected only one, preferring personal email addresses over professional ones because the person’s commits to the projects were partially already quite old and the person might have moved organizations since.

We sent an email to all 1,555 developers to invite them to our survey [59] (which was part of a larger data collection effort for multiple studies). We sent the invitation with a personalized link to the survey. In the email, we pledged to donate US\$ 2 per completed response to the United Nations World Food Programme [60]. We also sent two reminders [59], including one that linked to an official university web page confirming the authenticity of the survey as some developers were concerned that the invitation might be a scam.

Because 165 emails bounced, we reached 1,390 developers (89.4% deliverable emails), of which 194 developers started the survey, and 124 completed it. We dropped all respondents that provided implausible values for their age (≤ 10 or ≥ 100), leaving us with a preliminary sample of 121 developers. Although this number may seem low, it is of a similar magnitude as the total number even from studies that assessed personality not based on self-reports but on mining vast corpora of emails. Calefato *et al.*, for instance, used about 1.35 million emails from 46,304 developers but only obtained full personality profiles for 211 of them ($<0.5\%$), of which only 118 had made source code commits [37]. Our response rate was 8.9%, which is similar to that of other studies surveying developers on GitHub. Graziotin *et al.*, for instance, report a 7% response rate [61]. Their reported share of deliverable emails is 96.6%, which is also similar to ours.

The first page of the survey provided participants with basic information about what kind of data would be collected. We assured the developers that their data would be treated completely confidentially and not be shared with other parties.

We highlighted that participation was voluntary and that developers could abort the survey at any point. We further explicitly clarified that the developers consented to participating in our study by proceeding to the next page. We pretested the survey with three doctoral students of software engineering and made minor modifications based on their feedback.

To capture respondents’ FFM personality traits, we employed the widely used Ten-Item Personality Measure (TIPI) [41]. Since the TIPI is designed to capture all facets of the Big Five with content and criterion validity with one item each, reliability measures like Cronbach’s α are uninformative [62]. We consequently do not report them.

To measure regulatory focus, we employed six items (three each for promotion and prevention focus) from the extensively validated Regulatory Focus Composite Scale (RF-COMP) [63]. To test for internal consistency, we performed a factor analysis on all ten items of the RF-COMP in our sample. It indicated more than the two expected distinct factors. As is commonly done in such cases, we therefore reduced the number of items until two clear factors emerged—one capturing promotion and one capturing prevention focus. We then obtained a Cronbach’s α of .66 for promotion focus and .60 for prevention focus. Given the low number of items per construct, these constitute acceptable values [64].

We measured narcissism using the short version of the Narcissistic Personality Inventory (NPI-16) [52]. Cronbach’s α was .70, constituting an acceptable value [65] and even being slightly higher than that obtained in the work that developed the measure in the first place [52].

As we measure personality after the analyzed commits were made, this might potentially raise concerns about an unclear direction of causality in our analysis. However, this is unlikely to be a problem as personality is widely considered as stable across adult life [66], [67] and the sampled developers were all between 24 and 65 years of age. In line with this, recent research specifically on the personality of developers has not found any evidence of variation over time [37].

To measure developers’ age at the time of each commit, we asked developers to simply provide their age in years. We then subtracted the difference between 2022 and the year in which the focal commit was made from the provided age.

C. Final Sample

We only retained normal commits, dropping merge and orphan commits because the former do not allow a calculation of induced TD (due to multiple parent commits) and the latter may have particular characteristics. Note that including orphan commits does not change our results.

Since our TD measures require a completed SonarQube analysis for the focal and the parent commit, and there was substantial missing data in the Technical Debt Dataset, our sample was ultimately reduced to $N = 2,145$ commits from 19 developers for which we had both full TD and personality data. While our sample is thus not large by any means, it is not too far from, for example, the sample of 28 individuals in the study of Calefato and Lanubile on personality and trust

between Apache developers [68] and much larger than Rigby and Hassan’s sample of four programmers in their study on the FFM traits of highly productive Apache developers [69].

D. Analysis Strategy

We ran panel regressions to study the association between our personality and TD measures. This method is suitable in situations in which a panel unit (the developer in our case) is observed repeatedly. We clustered standard errors at the panel unit to account for the fact that multiple commits from the same developer are not statistically independent. We included control variables for developer age at time of commit and LOC added and LOC removed in our models. In addition, we include dummy variables (fixed effects) for each project in our analyses. These variables capture all time-invariant aspects that are specific to a project, for example certain coding conventions.

E. Replication Package

Unfortunately, we cannot make the data for our analyses available to other researchers because we explicitly promised our respondents full confidentiality. Even if this were different, we would be hesitant to share the data because we could not do it anonymously given that the entire purpose of this paper is to link personality to individually identifiable contributions to software projects. The Technical Debt Dataset itself, however, is publicly available [56]. We also provide a package of all our analysis scripts.³ This package also contains a dummy data file with the precise wording of the used survey measures.

IV. RESULTS

A. Descriptive Statistics

Table I provides descriptive statistics for the final dataset. Figure 1 shows a histogram and kernel density plot of the number of commits per developer in the sample.

TABLE I
DESCRIPTIVE STATISTICS

	Mean	SD	Min.	Max.
Induced TD	65.41	1,059.15	-14,160.00	32,880.00
Induced TD (broad)	67.92	1,091.94	-14,320.00	33,829.00
Extraversion	3.57	1.36	1.00	6.00
Agreeableness	4.18	1.10	2.50	6.00
Conscientiousness	5.82	1.11	3.50	7.00
Emotional stability	4.83	1.06	1.50	6.50
Openness to experience	4.82	1.25	3.00	7.00
Promotion focus	15.30	2.39	10.00	21.00
Prevention focus	13.13	3.52	9.00	18.00
Narcissism	3.66	2.14	0.00	9.00
Age at commit	36.48	6.88	23.00	50.00
LOC added	217.63	1,565.86	0.00	40,180.00
LOC removed	81.12	732.66	0.00	25,886.00

Table II shows the correlation between all variables except for the project dummy variables. It is apparent that the two

³Available at: <https://dx.doi.org/10.6084/m9.figshare.21802968>

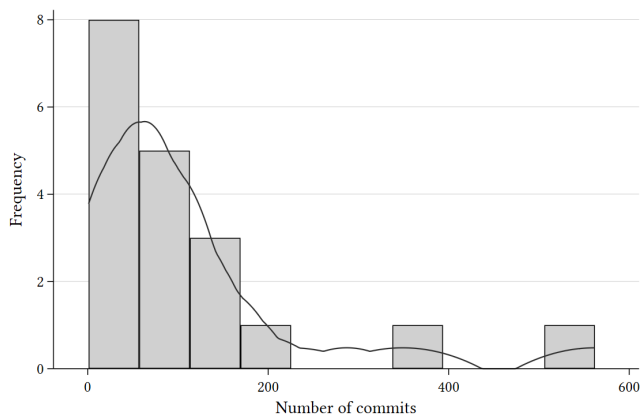


Fig. 1. Histogram and Kernel Density Plot of Commits per Developer

measures of TD are almost perfectly correlated and thus tap into essentially the same construct. Unsurprisingly, there is also a strong correlation between induced TD and LOC added.

It is well-documented and thus not surprising that FFM traits and narcissism share correlations [70]. In our sample we replicate, for instance, the positive correlation between narcissism and extraversion and the negative correlation between narcissism and agreeableness that have been widely reported in the psychological literature [70]. As we expected from regulatory focus measurements in other populations, promotion and prevention focus are correlated positively and significantly with each other [49].

B. Findings

Table III reports the regression coefficients and standard errors of our panel regression analyses, once with the primary measure of induced TD and once with the secondary, broader measure of induced TD as the dependent variable. As is evident from the table, the results are similar across both dependent variables. Specifically, the coefficients for extraversion and agreeableness are negative and not significant. These results thus do not lend support to *H1* and *H2*. The coefficient for conscientiousness is negative and significant, supporting *H3*. Counter to our hypotheses, the coefficients for both emotional stability and openness to experience are significant but negative. These findings consequently constitute evidence against *H4* and *H5*. The results regarding regulatory focus are mixed. The coefficient for promotion focus is unexpectedly negative but not significant. The coefficient for prevention focus is negative as expected, and significant. These results thus do not support *H6* but corroborate *H7*. Finally, the coefficient for narcissism is positive but not significant, lending no support to *H8*.

While there is substantial disagreement in the literature regarding whether corrections for multiple testing are necessary and whether and when they help the interpretation of results [71]–[73] (especially in the context of multiple regression), we attempted to control the false discovery rate

by computing adjusted p -values (sharpened q -values) [74] for all eight independent variables of interest. The results reaffirm our original findings.

The coefficients of our unhypothesized control variables are significant and with unsurprising signs. Developer age at the time of the commit was negatively associated with induced TD. LOC added and LOC removed were positively and negatively associated with induced TD, respectively.

V. DISCUSSION

A. Unexpected results

While some hypotheses were simply not supported by insignificant results in our regression analyses, some significant results went directly counter to our hypotheses and therefore warrant further discussion. First, emotional stability was significantly negatively related to TD. A possible explanation—in line with psychological research linking positive affect to enhanced problem solving and decision making, as well as flexible, innovative, thorough, and efficient cognitive processing [75]—would be that more emotionally stable developers’ generally positive mood allows them to find superior ways to produce highly maintainable code, inducing less TD.

Second, openness to experience was negatively related to TD. One plausible explanation would be that individuals high in openness possess a specific aesthetic sensitivity [34], which might find an expression in the creation of particularly beautiful code, which in turn may include little TD. An alternative explanation would be that such individuals are highly creative [34] and may thus simply find good ways of expressing logic in code without creating TD in the first place.

B. Threats to validity

There are several potential issues that may threaten the validity of our study. We treat them along four categories [76].

1) *Construct validity*: One may challenge the accuracy of our measurements. This holds true for both the independent and the dependent variables. When it comes to the personality measures, we resorted to using short scales. While this is likely to have contributed to a higher response rate in our survey, these measures may have introduced more measurement error than longer scales would have [77]. Additionally, our personality measures are based on self-reports. Although such measures are the most trusted and most frequently used in psychology, they are potentially not problem-free [78]. For instance, respondents may seek to present themselves in the best possible light rather than wholly accurately. Combined with limited response rates among software engineer populations, this has made researchers look to other data sources. For instance, it has recently become fashionable to infer personality from text corpora [36]–[38], [40], but such methods are at least as problematic as self reports. For one, recent work demonstrated that using psycholinguistic methods to capture personality from developer communication (e.g., mailing lists) can result in substantial inaccuracies [79] and different linguistic tools may lead researchers to very different conclusions [80]. For another, most such work has only

TABLE II
CORRELATIONS

	Induced TD	Induced TD (broad)	Extraversion	Agreeableness	Conscientiousness	Emotional stability	Openness to experience	Promotion focus	Prevention focus	Narcissism	Age at commit	LOC added
Induced TD (broad)	1.00***											
Extraversion	0.02	0.02										
Agreeableness	0.02	0.02	-0.30***									
Conscientiousness	-0.04*	-0.04*	-0.27***	-0.55***								
Emotional stability	-0.02	-0.02	-0.31***	0.12***	0.26***							
Openness to experience	0.04	0.04	0.42***	0.13***	-0.41***	-0.58***						
Promotion focus	-0.04*	-0.04*	-0.20***	0.29***	-0.03	-0.13***	-0.15***					
Prevention focus	-0.02	-0.02	0.15***	0.30***	-0.43***	-0.72***	0.46***	0.57***				
Narcissism	-0.02	-0.02	0.45***	-0.30***	0.00	-0.00	-0.06**	-0.46***	-0.18***			
Age at commit	0.02	0.02	-0.46***	0.39***	-0.33***	0.20***	-0.41***	0.02	-0.11***	-0.19***		
LOC added	0.73***	0.73***	-0.01	0.04	-0.05*	-0.02	0.07***	-0.05*	-0.01	-0.04*	0.04	
LOC removed	-0.13***	-0.13***	-0.01	0.03	-0.03	0.01	0.03	-0.01	-0.01	-0.01	-0.01	0.39***

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$. Dummy variables for projects not shown. $N = 2,145$.

studied the FFM personality traits and corresponding linguistic methods are not readily available for other personality traits.

Despite the fact that “few findings in psychology are more robust than the stability of personality” [67, p. 175], this notion has been challenged in some software engineering research, suggesting that contributors evolve as more conscientious, more extrovert and less agreeable over the years of participation [38]. While the effect sizes in this research are tiny [37], this potentially affects the validity of our personality measures.

Further, the TD estimates from SonarQube are not perfect and can likely be improved upon [81]. In fact, there are likely limits to automated detection of TD in general. Certainly, source code analysis tools cannot capture all types of technical debt that prior research has identified [8], [13], [82]. Relatedly, different technical debt detection tools frequently come to different assessments of technical debt [83]. It would thus be interesting to repeat our analysis with different measures of technical debt from other tools.

2) *Internal validity*: There are various ways in which the internal validity of our study might be limited. First, developer personality may not be the only driver of TD, and not all other relevant variables might be controlled for. This could lead to missing variable problems and hence to spurious findings in our regression analyses.

Second, our results might be affected by selection effects. Since not all authors in the studied software projects have commit privileges, the commits that were made and that we can consequently observe may differ systematically from all proposed code changes. For instance, as others have shown, the personalities of committers influence their behavior, too [36], [40]. Consequently, our data may be filtered in particular ways, and our analyses and findings might thus be distorted.

3) *External validity*: Naturally, a key question in software engineering studies on data from software repositories relates to how generalizable any findings are to other contexts. At least two potential threats to validity exist in this regard.

First, in light of recent discussions about the representativeness of samples [59], [60], we highlight that our sample is likely not representative of all software developers. In particular, we only sample open source contributors from relatively large Apache projects written largely in Java, and we achieved only a limited response rate in our survey.

Second, the Technical Debt Dataset exhibited substantial missing data. In particular, the SonarQube analyses were incomplete for many commits. Since our measure of TD requires completed analyses for not only each focal commit but also its parent commit, we experienced a lot of missing values in our dependent variables. Whether this data is missing at random remains unclear.

4) *Reliability*: The reliability of our research should be high. All scales we used to assess personality are established in the psychological literature [59]. There was very little human judgment needed in our research process. Where judgment calls were necessary, we documented the guiding principles in this article. We also provide all scripts used for data analysis. Consequently, our research should be, in principle, fully replicable. Given that confidentiality assurances to the surveyed developers prevent us from sharing the personality data, other researchers are of course not able to directly repeat our analyses. However, we deem the latter issue not a threat to the validity of our research.

C. Implications for Research

Our study has several implications for future research. First and foremost, it demonstrates that developer personality has

TABLE III
PANEL REGRESSIONS

	Induced TD	Induced TD (broad)
Extraversion	-34.80 (45.98)	-36.55 (48.32)
Agreeableness	-0.93 (25.17)	-1.91 (26.32)
Conscientiousness	-185.20** (60.77)	-191.80** (63.85)
Emotional stability	-178.47*** (21.78)	-183.47*** (22.74)
Openness to experience	-189.82*** (43.50)	-192.61*** (45.14)
Promotion focus	-31.66 (28.94)	-31.41 (30.36)
Prevention focus	-44.31*** (7.09)	-45.70*** (7.44)
Narcissism	19.77 (20.67)	22.23 (21.52)
Age at commit	-17.77* (7.51)	-17.94* (7.78)
LOC added	0.62*** (0.14)	0.64*** (0.14)
LOC removed	-0.71*** (0.14)	-0.73*** (0.14)
Constant	4500.48*** (1152.82)	4596.41*** (1203.54)
Project fixed effects	Yes	Yes
Observations	2,145	2,145
Clusters	19	19

Dependent variable indicated in top row.

Table reports coefficients, clustered standard errors in parentheses.

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

a meaningful impact on TD. We thus propose further study in this direction. In particular, we encourage replications of our study in other code bases and potentially with other TD measures, e.g., self-admitted technical debt. Further, studying other types of TD that may not be captured by the measures we used may be informative [8], [13], [82]. Moreover, potential consequences of TD like software defects or project budget overruns might also be linked to personality and warrant further study. Second, our research shows that personality traits beyond the FFM may be relevant to explain and predict the behavior of software developers. Although only prevention focus proved to be significant in our study, we suggest that regulatory focus in general and possibly even narcissism should be further studied with regard to software engineering decisions beyond the issue of TD. Finally, future research may consider the addition and the removal of TD as two distinct activities. We thus propose analyses of commits with net increases and commits with net decreases in TD [29] to see if they may potentially have different personality antecedents.

D. Implications for Practice

Our study suggests that developers with certain personality profiles might be more or less prone to incurring TD. This leads to important practical implications. First, project managers (who often know their teams well and do not need to administer potentially legally problematic personality surveys) might staff software development projects with this in mind. For instance, they might purposefully opt to assign project artifacts that already have too much TD to developers with a personality profile that likely leads to reduced TD, for example, developers high in prevention focus or conscientiousness. Similarly, they may change team composition between project phases. Following Beck's 3X model, it is for example conceivable that the "explore" phase allows for developers producing more TD, whereas "expand" calls for more careful developers, and the "extract" phase may be best handled by developers averse to TD [84]. Second, managers may allocate more time and resources to closely tracking TD in artifacts developed by programmers with a personality profile that is conducive to TD. Third, when using techniques like pair programming, project managers may wish to make personality a criterion in establishing pairs. For example, it may be desirable to have pairs of programmers that have complementary personality profiles to actively manage TD in a project. Finally, our research may allow developers themselves to become more aware of their innate tendencies. All personality measures used in this study are generally available and could be used in training programs in which developers have an opportunity to assess their personality and learn about the potential implications for their behaviors regarding TD.

VI. CONCLUSION

In conclusion, it seems that developer personality influences the creation and removal of TD. We hope our research sparks further inquiries into this intriguing issue.

ACKNOWLEDGMENT

We would like to acknowledge helpful information on the Technical Debt Dataset from Davide Taibi.

REFERENCES

- [1] W. Cunningham, "The wycash portfolio mangement system," in *Addendum to the Proceedings of OOPSLA 1992*, 1992, pp. 29–30.
- [2] R. Ramač, V. Mandić, N. Taušan, N. Rios, S. Freire, B. Pérez, C. Castellanos, D. Correal, A. Pacheco, G. Lopez, C. Izurieta, C. Seaman, and R. Spinola, "Prevalence, common causes and effects of technical debt: Results from a family of surveys with the it industry," *Journal of Systems and Software*, vol. 184, p. 111114, 2022.
- [3] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, "Managing technical debt in software engineering (dagstuhl seminar 16162)," *Dagstuhl Reports*, vol. 6, no. 4, pp. 110–138, 2016.
- [4] T. Besker, A. Martini, and J. Bosch, "The pricey bill of technical debt: When and by whom will it be paid?" in *ICSME 2017*, I. I. C. o. S. M. a. Evolution, Ed. Piscataway, NJ: IEEE, 2017, pp. 13–23.
- [5] H. Ghanbari, T. Besker, A. Martini, and J. Bosch, "Looking for peace of mind? manage your (technical) debt: An exploratory field study," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2017, pp. 384–393.
- [6] T. Besker, H. Ghanbari, A. Martini, and J. Bosch, "The influence of technical debt on software developer morale," *Journal of Systems and Software*, vol. 167, p. 110586, 2020.

- [7] J. Olsson, E. Risfelt, T. Besker, A. Martini, and R. Torkar, "Measuring affective states from technical debt," *Empirical Software Engineering*, vol. 26, no. 5, 2021.
- [8] E. Tom, A. Aurum, and R. Vidgen, "An exploration of technical debt," *Journal of Systems and Software*, vol. 86, no. 6, pp. 1498–1516, 2013.
- [9] F. Shull, "Perfectionists in a world of finite resources," *IEEE Software*, vol. 28, no. 2, pp. 4–6, 2011.
- [10] K. Schmid, "On the limits of the technical debt metaphor some guidance on going beyond," in *2013 4th International Workshop on Managing Technical Debt (MTD 2013)*, P. Kruchten, Ed. Piscataway, NJ: IEEE, 2013, pp. 63–66.
- [11] P. Kruchten, R. L. Nord, I. Ozkaya, and D. Falessi, "Technical debt: Towards a crisper definition," *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 5, pp. 51–54, 2013.
- [12] E. Allman, "Managing technical debt," *Communications of the ACM*, vol. 55, no. 5, pp. 50–55, 2012.
- [13] N. S. Alves, T. S. Mendes, M. G. de Mendonça, R. O. Spínola, F. Shull, and C. Seaman, "Identification and management of technical debt: A systematic mapping study," *Information and Software Technology*, vol. 70, pp. 100–121, 2016.
- [14] P. C. Avgeriou, D. Taibi, A. Ampatzoglou, F. Arcelli Fontana, T. Besker, A. Chatzigeorgiou, V. Lenarduzzi, A. Martini, A. Moschou, I. Pigazzini, N. Saarimaki, D. D. Sas, S. S. de Toledo, and A. A. Tsintzira, "An overview and comparison of technical debt measurement tools," *IEEE Software*, vol. 38, no. 3, pp. 61–71, 2021.
- [15] F. Gilson, M. Morales-Trujillo, and M. Mathews, "How junior developers deal with their technical debt?" in *Proceedings of the 3rd International Conference on Technical Debt*, ser. ACM Digital Library, C. Izurieta, Ed. New York, NY, United States: Association for Computing Machinery, 2020, pp. 51–61.
- [16] B. Curtis, J. Sappidi, and A. Szykarski, "Estimating the principal of an application's technical debt," *IEEE Software*, vol. 29, no. 6, pp. 34–42, 2012.
- [17] G. Digkas, M. Lungu, A. Chatzigeorgiou, and P. Avgeriou, "The evolution of technical debt in the apache ecosystem," in *Software architecture*, ser. LNCS Sublibrary: SL2 - Programming and software engineering, A. Lopes and R. de Lemos, Eds. Cham, Switzerland: Springer, 2017, vol. 10475, pp. 51–66.
- [18] 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 *25th IEEE International Conference on Software Analysis, Evolution and Reengineering*, E. IEEE International Conference on Software Analysis and Reengineering, Eds. Piscataway, NJ: IEEE, 2018, pp. 153–163.
- [19] R. Maipradit, C. Treude, H. Hata, and K. Matsumoto, "Wait for it: identifying "on-hold" self-admitted technical debt," *Empirical Software Engineering*, vol. 25, no. 5, pp. 3770–3798, 2020.
- [20] R. Alfayez, W. Alwehaibi, R. Winn, E. Venson, and B. Boehm, "A systematic literature review of technical debt prioritization," in *Proceedings of the 3rd International Conference on Technical Debt*, ser. ACM Digital Library, C. Izurieta, Ed. New York, NY, United States: Association for Computing Machinery, 2020, pp. 1–10.
- [21] V. Lenarduzzi, T. Besker, D. Taibi, A. Martini, and F. Arcelli Fontana, "A systematic literature review on technical debt prioritization: Strategies, processes, factors, and tools," *Journal of Systems and Software*, vol. 171, p. 110827, 2021.
- [22] J. Yli-Huumo, T. Rissanen, A. Maglyas, K. Smolander, and L.-M. Sainio, "The relationship between business model experimentation and technical debt," in *Software business*, ser. Lecture Notes in Business Information Processing, J. M. Fernandes, R. J. Machado, and K. Wnuk, Eds. New York NY: Springer Berlin Heidelberg, 2015, vol. 210, pp. 17–29.
- [23] J. Bedi and K. Kaur, "Understanding factors affecting technical debt," *International Journal of Information Technology*, vol. 14, no. 2, pp. 1051–1060, 2022.
- [24] T. Besker, A. Martini, and J. Bosch, "The use of incentives to promote technical debt management," *Information and Software Technology*, vol. 142, p. 106740, 2022.
- [25] N. Rios, R. Oliveira Spinola, M. G. de Mendonca Neto, and C. Seaman, "A study of factors that lead development teams to incur technical debt in software projects," in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2018)*. Piscataway, NJ: IEEE, 2018, pp. 429–436.
- [26] N. Rios, R. O. Spínola, M. Mendonça, and C. Seaman, "The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from brazil," *Empirical Software Engineering*, vol. 25, no. 5, pp. 3216–3287, 2020.
- [27] S. Freire, N. Rios, B. Perez, C. Castellanos, D. Correal, R. Ramac, V. Mandic, N. Tausan, G. Lopez, A. Pacheco, D. Falessi, M. Mendonca, C. Izurieta, C. Seaman, and R. Spinola, "How experience impacts practitioners' perception of causes and effects of technical debt," in *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering*. Piscataway, NJ: IEEE, 2021, pp. 21–30.
- [28] T. Amanatidis, A. Chatzigeorgiou, A. Ampatzoglou, and I. Stamelos, "Who is producing more technical debt?" in *Proceedings of the XP2017 Scientific Workshops*, R. Tonelli, Ed. New York, NY, USA: ACM, 2017, pp. 1–8.
- [29] R. Alfayez, P. Behnamghader, K. Srisopha, and B. Boehm, "An exploratory study on the influence of developers in technical debt," in *Proceedings of the 2018 International Conference on Technical Debt*, ser. ACM Conferences, R. L. Nord, Ed. New York, NY: ACM, 2018, pp. 1–10.
- [30] Z. Codabux and C. Dutchyn, "Profiling developers through the lens of technical debt," in *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. New York, NY, USA: ACM, 2020, pp. 1–6.
- [31] O. P. John and R. W. Robins, Eds., *Handbook of personality: Theory and research*, 4th ed. New York: The Guilford Press, 2021.
- [32] G. Matthews, I. J. Deary, and M. C. Whiteman, *Personality traits*, 2nd ed. Cambridge: Cambridge University Press, 2003.
- [33] R. R. McCrae and O. P. John, "An introduction to the five-factor model and its applications," *Journal of personality*, vol. 60, no. 2, pp. 175–215, 1992.
- [34] O. P. John, "History, measurement, and conceptual elaboration of the big-five trait taxonomy: The paradigm matures," in *Handbook of personality*, O. P. John and R. W. Robins, Eds. New York: The Guilford Press, 2021, pp. 35–82.
- [35] C. M. Ching, A. T. Church, M. S. Katigbak, J. A. S. Reyes, J. Tanaka-Matsumi, S. Takaoka, H. Zhang, J. Shen, R. M. Arias, B. C. Rincon, and F. A. Ortiz, "The manifestation of traits in everyday behavior and affect: A five-culture study," *Journal of Research in Personality*, vol. 48, pp. 1–16, 2014.
- [36] R. N. Iyer, S. A. Yun, M. Nagappan, and J. Hoey, "Effects of personality traits on pull request acceptance," *IEEE Transactions on Software Engineering*, vol. 47, no. 11, pp. 2632–2643, 2021.
- [37] F. Calefato, F. Lanubile, and B. Vasilescu, "A large-scale, in-depth analysis of developers' personalities in the apache ecosystem," *Information and Software Technology*, vol. 114, pp. 1–20, 2019.
- [38] A. Rastogi and N. Nagappan, "On the personality traits of github contributors," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2016, pp. 77–86.
- [39] Z. Karimi, A. Baraani-Dastjerdi, N. Ghasem-Aghaee, and S. Wagner, "Links between the personalities, styles and performance in computer programming," *Journal of Systems and Software*, vol. 111, pp. 228–241, 2016.
- [40] O. H. Paruma-Pabón, F. A. González, J. Aponte, J. E. Camargo, and F. Restrepo-Calle, "Finding relationships between socio-technical aspects and personality traits by mining developer e-mails," in *Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering*. New York, NY, USA: ACM, 2016, pp. 8–14.
- [41] S. D. Gosling, P. J. Rentfrow, and W. B. Swann, "A very brief measure of the big-five personality domains," *Journal of Research in Personality*, vol. 37, no. 6, pp. 504–528, 2003.
- [42] N. Nicholson, E. Soane, M. Fenton-O'Creevy, and P. Willman, "Personality and domain-specific risk taking," *Journal of Risk Research*, vol. 8, no. 2, pp. 157–176, 2005.
- [43] T. A. Judge, J. B. Rodell, R. L. Klinger, L. S. Simon, and E. R. Crawford, "Hierarchical representations of the five-factor model of personality in predicting job performance: integrating three organizing frameworks with two theoretical perspectives," *Journal of Applied Psychology*, vol. 98, no. 6, pp. 875–925, 2013.
- [44] J. B. Hirsh and M. Inzlicht, "The devil you know: neuroticism predicts neural response to uncertainty," *Psychological science*, vol. 19, no. 10, pp. 962–967, 2008.

- [45] M. J. Lommen, I. M. Engelhard, and M. A. van den Hout, "Neuroticism and avoidance of ambiguous stimuli: Better safe than sorry?" *Personality and Individual Differences*, vol. 49, no. 8, pp. 1001–1006, 2010.
- [46] G. L. Flett, P. L. Hewitt, and D. G. Dyck, "Self-oriented perfectionism, neuroticism and anxiety," *Personality and Individual Differences*, vol. 10, no. 7, pp. 731–735, 1989.
- [47] T. C. Marshall, K. Lefringhausen, and N. Ferenczi, "The big five, self-esteem, and narcissism as predictors of the topics people write about in facebook status updates," *Personality and Individual Differences*, vol. 85, pp. 35–40, 2015.
- [48] E. T. Higgins, "Beyond pleasure and pain," *The American psychologist*, vol. 52, no. 12, pp. 1280–1300, 1997.
- [49] K. Lanaj, C.-H. D. Chang, and R. E. Johnson, "Regulatory focus and work-related outcomes: a review and meta-analysis," *Psychological bulletin*, vol. 138, no. 5, pp. 998–1034, 2012.
- [50] M. R. Hamstra, J. W. Bolderdijk, and J. L. Veldstra, "Everyday risk taking as a function of regulatory focus," *Journal of Research in Personality*, vol. 45, no. 1, pp. 134–137, 2011.
- [51] M. B. Donnellan, R. A. Ackerman, and A. G. Wright, "Narcissism in contemporary personality psychology," in *Handbook of personality*, O. P. John and R. W. Robins, Eds. New York: The Guilford Press, 2021, pp. 625–641.
- [52] D. R. Ames, P. Rose, and C. P. Anderson, "The npi-16 as a short measure of narcissism," *Journal of Research in Personality*, vol. 40, no. 4, pp. 440–450, 2006.
- [53] V. Zeigler-Hill, G. A. McCabe, J. K. Vrabel, C. M. Raby, and S. Cronin, "The narcissistic pursuit of status," in *Handbook of Trait Narcissism*, A. D. Hermann, A. B. Brunell, and J. D. Foster, Eds. Cham: Springer International Publishing, 2018, pp. 299–306.
- [54] C. C. Morf and F. Rhodewalt, "Unraveling the paradoxes of narcissism: A dynamic self-regulatory processing model," *Psychological Inquiry*, vol. 12, no. 4, pp. 177–196, 2001.
- [55] M. T. Buelow and A. B. Brunell, "Narcissism and involvement in risk-taking behaviors," in *Handbook of Trait Narcissism*, A. D. Hermann, A. B. Brunell, and J. D. Foster, Eds. Cham: Springer International Publishing, 2018, pp. 233–242.
- [56] V. Lenarduzzi, N. Saarimäki, and D. Taibi, "The technical debt dataset," in *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering*, L. Minku, F. Khomh, and J. Petrić, Eds. New York, NY, USA: ACM, 2019, pp. 2–11.
- [57] S. T. Certo, J. R. Busenbark, M. Kalm, and J. A. LePine, "Divided we fall: How ratios undermine research in strategic management," *Organizational Research Methods*, vol. 23, no. 2, pp. 211–237, 2020.
- [58] D. Spadini, M. Aniche, and A. Bacchelli, "Pydriller: Python framework for mining software repositories," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, G. T. Leavens, A. Garcia, and C. S. Păsăreanu, Eds. New York, NY, USA: ACM, 2018, pp. 908–911.
- [59] S. Wagner, D. Mendez, M. Felderer, D. Graziotin, and M. Kalinowski, "Challenges in survey research," in *Contemporary Empirical Methods in Software Engineering*, M. Felderer and G. H. Travassos, Eds. Cham: Springer International Publishing, 2020, pp. 93–125.
- [60] S. Baltes and P. Ralph, "Sampling in software engineering research: a critical review and guidelines," *Empirical Software Engineering*, vol. 27, no. 4, 2022.
- [61] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson, "On the unhappiness of software developers," in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, E. Mendes, Ed. New York, NY: ACM, 2017, pp. 324–333.
- [62] S. D. Gosling, "A note on alpha reliability and factor structure in the tipi." [Online]. Available: <https://gosling.psy.utexas.edu/scales-weve-developed/ten-item-personality-measure-tipi/a-note-on-alpha-reliability-and-factor-structure-in-the-tipi/>
- [63] K. L. Haws, U. M. Dholakia, and W. O. Bearden, "An assessment of chronic regulatory focus measures," *Journal of Marketing Research*, vol. 47, no. 5, pp. 967–982, 2010.
- [64] J. M. Cortina, "What is coefficient alpha? an examination of theory and applications," *Journal of Applied Psychology*, vol. 78, no. 1, pp. 98–104, 1993.
- [65] J. C. Nunnally, *Psychometric theory*, 2nd ed., ser. McGraw-Hill series in psychology. New York and London: McGraw-Hill, 1978.
- [66] P. T. Costa and R. R. McCrae, "Personality in adulthood: A six-year longitudinal study of self-reports and spouse ratings on the neo personality inventory," *Journal of personality and social psychology*, vol. 54, no. 5, pp. 853–863, 1988.
- [67] R. R. McCrae and P. T. Costa, "The stability of personality: Observations and evaluations," *Current Directions in Psychological Science*, vol. 3, no. 6, pp. 173–175, 1994.
- [68] F. Calefato and F. Lanubile, "Establishing personal trust-based connections in distributed teams," *Internet Technology Letters*, vol. 1, no. 4, p. e6, 2018.
- [69] P. C. Rigby and A. E. Hassan, "What can oss mailing lists tell us? a preliminary psychometric text analysis of the apache developer mailing list," in *Fourth International Workshop on Mining Software Repositories (MSR'07:ICSE Workshops 2007)*. IEEE, 2007, p. 23.
- [70] B. A. Visser, "Narcissism and the big five/hexaco models of personality," in *Handbook of Trait Narcissism*, A. D. Hermann, A. B. Brunell, and J. D. Foster, Eds. Cham: Springer International Publishing, 2018, pp. 205–212.
- [71] K. J. Rothman, "No adjustments are needed for multiple comparisons," *Epidemiology*, vol. 1, no. 1, pp. 43–46, 1990.
- [72] R. Bender and S. Lange, "Adjusting for multiple testing—when and how?" *Journal of clinical epidemiology*, vol. 54, no. 4, pp. 343–349, 2001.
- [73] D. J. O'Keefe, "Colloquy: Should familywise alpha be adjusted? against familywise alpha adjustment," *Human Communication Research*, vol. 29, no. 3, pp. 431–447, 2003.
- [74] M. L. Anderson, "Multiple inference and gender differences in the effects of early intervention: A reevaluation of the abecedarian, perry preschool, and early training projects," *Journal of the American Statistical Association*, vol. 103, no. 484, pp. 1481–1495, 2008.
- [75] A. M. Isen, "An influence of positive affect on decision making in complex situations: Theoretical issues with practical implications," *Journal of Consumer Psychology*, vol. 11, no. 2, pp. 75–85, 2001.
- [76] L. Gren, "Standards of validity and the validity of standards in behavioral software engineering research," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, M. Oivo, D. Méndez, and A. Mockus, Eds. New York, NY, USA: ACM, 2018, pp. 1–4.
- [77] F. L. Schmidt and J. E. Hunter, "Measurement error in psychological research: Lessons from 26 research scenarios," *Psychological Methods*, vol. 1, no. 2, pp. 199–223, 1996.
- [78] D. L. Paulhus and S. Vazire, "The self-report method," in *Handbook of Research methods in Personality Psychology*, R. W. Robins, R. C. Fraley, and R. F. Krueger, Eds. New York, NY: Guilford Press, 2007, pp. 224–239.
- [79] F. C. van Mil, A. Rastogi, and A. Zaidman, "Promises and perils of inferring personality on github," in *Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, F. Lanubile, Ed. New York, NY, USA: ACM, 2021, pp. 1–11.
- [80] F. Calefato and F. Lanubile, "Using personality detection tools for software engineering research: How far can we go?" *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 3, pp. 1–48, 2022.
- [81] V. Lenarduzzi, A. Martini, D. Taibi, and D. A. Tamburri, "Towards surgically-precise technical debt estimation: early results and research roadmap," in *Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation - MaLTeSQuE 2019*, F. A. Fontana, B. Walter, A. Ampatzoglou, F. Palomba, G. Perrouin, M. Acher, M. Cordy, and X. Devroey, Eds. New York, New York, USA: ACM Press, 2019, pp. 37–42.
- [82] N. Rios, M. G. de Mendonça Neto, and R. O. Spínola, "A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners," *Information and Software Technology*, vol. 102, pp. 117–145, 2018.
- [83] J. Lefever, Y. Cai, H. Cervantes, R. Kazman, and H. Fang, "On the lack of consensus among technical debt detection tools," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2021, pp. 121–130.
- [84] K. Beck, "The product development triathlon," 2018. [Online]. Available: https://medium.com/@kentbeck_7670/the-product-development-triathlon-6464e2763c46