The Sky is Not the Limit: Multitasking on GitHub Projects

Bogdan Vasilescu[†], Kelly Blincoe[§], Qi Xuan[‡], Casey Casalnuovo[†], Daniela Damian[♯], Premkumar Devanbu[†], Vladimir Filkov[†]

[†]Dept. Computer Science, University of California, Davis. Davis, CA 95616, USA
[§]Dept. Computer & Mathematical Sciences, Auckland University of Technology, New Zealand
[‡]Dept. Automation, Zhejiang University of Technology, Hangzhou 310023, China
[♯]Dept. Computer Science, University of Victoria, Victoria, BC, Canada
{vasilescu, ccasal, ptdevanbu, vfilkov}@ucdavis.edu
kblincoe@acm.org, xuanqi@zjut.edu.cn, danielad@uvic.ca

ABSTRACT

Software development has always inherently required multitasking: developers switch between coding, reviewing, testing, designing, and meeting with colleagues. The advent of software ecosystems like GitHub has enabled something new: the ability to easily switch *between* projects. Developers also have social incentives to contribute to many projects; prolific contributors gain social recognition and (eventually) economic rewards. Multi-tasking, however, comes at a cognitive cost: frequent context-switches can lead to distraction, sub-standard work, and even greater stress. In this paper, we gather ecosystem-level data on a group of programmers working on a large collection of projects. We develop models and methods for measuring the rate and breadth of a developers' context-switching behavior, and we study how context-switching affects their productivity. We also survey developers to understand the reasons for and perceptions of multitasking. We find that the most common reason for multitasking is interrelationships and dependencies between projects. Notably, we find that the *rate of switching* and *breadth* (number of projects) of a developer's work matter. If a developer is able to focus on few projects per day, they can increase their productivity by working on more projects. However, if they attempt to switch projects too much during the course of a day, their productivity *declines* as they work on more projects. Despite these findings, developers perceptions of the benefits of multitasking are varied.

1. INTRODUCTION

Multitasking is a staple of high performing professionals, programmers included. It is the ability to stop working on a task, switch to another, and return eventually to the first one, as needed or as scheduled. The goal is to optimize the human resource, while reprioritizing tasks dynamically [17].

When done well, or at least in a disciplined way, multitasking can yield dividends [1, 4]. Clearly, if a task in the queue has a higher priority than the current one, switching them can lead to better performance. E.g., programmers encounter this when they begin a new coding task, while their previous, urgently needed fix undergoes testing. If bugs are found during testing, the urgent fix will require attention, and the new coding task will be put on the back-burner.

Multitasking comes at a cost though [9]. Humans, programmers included, have a certain, limited amount of *cognitive flexibility*, the mental ability to switch between thinking about one concept to thinking about another. Limitations apply to both the number of concepts we can juggle, as well as to the difficulty in switching between them. For example, switching between two small matrix multiplication problems is more difficult than switching between two small integer additions. Likewise, coding simultaneously 7 simple functions is easier than coding 8 of them. As we can imagine, reaching our innate limitations can result in decreased performance on all tasks and perhaps even diminished quality. It is unknown how far can multitasking be pushed safely, although some anecdotal evidence is available.[1]

Software developers have long been pushing the limits on multitasking [26] because of the innate modularity of the software development process and the independence of module processing (*e.g.*, coding can be going on while testing is being executed; similarly with code reviews, discussions, *etc.*). With the advent of GitHub, a new imposition on the cognitive flexibility of programmers has arisen: contributing to multiple software projects in the same time period. It is not uncommon to find prolific developers contributing code to 5-10 GitHub projects in the same week. In fact, contributing to as many GitHub projects as possible is an accomplishment, valued by peers and employers alike [30].

There are various reasons why developers are more prolific on GitHub compared to other platforms. The features and usability provided by GitHub play a big role [32]. It is one example of how novel technology benefits programmers, and it empowers them; GitHub's platform is responsible for the inception of many projects, that otherwise wouldn't have existed [33]. And the payoffs are also substantial [33]. With so many drivers for multitasking, it is easy to see how one could cross the limits from safe, productive multitasking into an overloaded mode, where code output falls and bugs start to multiply. The question really becomes where are those limits, and what are their determinants?

Multitasking is a complex phenomenon, with costs along different dimensions. This paper reports on a mixed meth-

[1]E.g., there is a myth that our brain can hold up to a high single digit number of concepts at the same time.

ods study of multitasking and focus shifting. Through analysis of longitudinal data, we investigate how productivity of prolific GitHub programmers is determined by the number of projects they work on, how much they focus on each (relative to the others), and how diverse the projects they contribute to are in terms of programming languages.Further, a survey of 128 of these prolific developers was used to better understand the reasons for and perceptions of multitasking. The highlights of our findings are that:

- 98% of prolific programmers contribute to multiple projects per day at least once, but the patterns are nuanced, and the reasons vary.
- Multitasking leads to a higher productivity so long as the focus is on fewer projects, but developers' perception is mixed.
- Productivity does not decrease so long as developers work on 5 or fewer projects per week. Yet, developers have an irrational ambition and often want to contribute to more projects regardless of their current workload.

We introduce our research questions and related works in Sec. 2, followed by our research methods (Sec. 3) and results (Sec. 4). We discuss the significance of our contributions in Sec. 5 and offer some concluding remarks in Sec. 6.

2. BACKGROUND

Multitasking and Performance Multitasking, defined as the act of working on multiple projects or tasks simultaneously, has become increasingly more common among knowledge workers [17, 29, 36]. Recent research across many disciplines has begun to investigate the effects of multitasking on individual and team productivity [1, 29]. While there is some disagreement between studies, it is generally believed that multitasking has non-monotonic (concave) effects on individual productivity [1, 4], with plenty of theoretical evidence from Psychology, Management, and Organizational Behavior (*e.g.*, [1, 36]).

Positive effects are attributed to several factors. First, multitasking may increase productivity since inevitable lulls in one project (*e.g.*, periods of inactivity when workers are waiting for information from customers or colleagues, or raw materials from a supplier; intervals when software developers are waiting for the build process to finish) can be filled with tasks related to other projects [1, 4, 36]. By keeping multiple projects active at the same time, one can switch focus between those projects during periods of relative down-time, enabling them to utilize their time more efficiently, therefore increasing their productivity.

Second, multitasking may increase productivity through cross-fertilisation and learning. If knowledge and skills required to carry out work on a project are transferrable to other projects as well, workers may experience decreasing costs of contributing to multiple projects simultaneously, as they are able to realize such knowledge transfers [28]. Similarly, exposure to more projects brings about exposure to different environments, which offers workers more opportunities to develop transferrable skills. We already found evidence during prior work on GitHub [10] supporting this effect of knowledge transfers on productivity: past experience in a newly-joined project's languages has positive effects.

On the other hand, several theories have been proposed to explain the cognitive processes that account for decreased performance in multitasking conditions, *e.g.*, memory-for-goals [2]. Performing a task often requires maintaining some information mentally.[2] This information is stored in the central part of the working memory, an area of the brain that can be accessed without delay, but its capacity is limited to a single task at a time [3]. Memory-for-goals explains how initiating a task requires strengthening its goal ("a mental representation of an intention to accomplish a task" [2]) in memory to the extent to which its activation rises above other competing goals [47]. Therefore, in a multitasking context, switching to another task involves first retrieving its goal from memory (especially if multiple tasks require storing state [9], which is not uncommon for programming), and this takes time. With more multitasking, one starts to incur cognitive switching costs for interrupting one task and resuming another [9]. Ultimately, too much multitasking leads to mental congestion, which negatively affects productivity.[3] As a result, workers start to experience "project overload" [56], and become increasingly slower and more error prone [8, 16, 44, 46]. The duration of the interruption, its complexity, and the moment when it occurs, all play an important role in how disruptive task switching is [2, 18].

In addition, switching between GitHub projects also involves a social component. Not only can different projects involve different technological contexts, but they may also involve different social contexts and different teams, all of which require adjusting to. Research has found that the more diverse the "working spheres" associated with each team are, the more disruptive switching between those teams becomes, and the more it hinders productivity [29]. GitHub is known to be a particularly diverse [51], social [14] ecosystem, wherein many social attributes become salient [50] and can impact impression formation and collaboration [31].

In summary, we expect there would be significant benefits to productivity for those who engage in multitasking, through load balancing, more efficient work practices, learning, and cross-fertilisation. However, there will be a point of diminishing returns, after which these benefits will be offset by cognitive overload.

Focus Shifting in Software Engineering Software developers shift focus frequently due to interruptions [21, 26, 57]. Interruptions can be external or self-inflicted [13]. For example, technical dependencies often exist between tasks, and this can cause developers to interrupt one another for coordination purposes [5, 11, 20]. The patterns of how developers shift focus within a project are very complex [54]. Focus shifts occur when changing from one task to another, but also when stopping work to coordinate or talk to someone else on the team [53].

When software developers resume a task after being interrupted, they must reconstruct the context of the task [37, 39–41]. Cues or tools can help developers pick up where they left off, by helping them quickly remember the task details [25, 38], but even with this support there is considerable overhead involved when switching. For example, editing many files concurrently is found to negatively impact software quality [55]. Similarly, fragmented within-project work drives down developer productivity [48]. In this pa-

[2]Not all tasks do; *e.g.*, while computing the tenth Fibonacci number involves keeping the eighth and ninth in the problem state, riding a bike does not require maintaining state.

[3]The phenomenon is often illustrated with a highway metaphor: initially, throughput increases as more cars enter an empty highway; as the arrival rate of new cars increases beyond the highway's bandwidth, congestion eventually forms and throughput suffers.

per, we look beyond individual projects and investigate focus shifting patterns and their impact on productivity when developers contribute *across* multiple projects.

Furthermore, a recent study found that developers associate minimal interruptions and context switches with higher productivity. Despite these beliefs, they reportedly feel productive even when significant task shifting occurs [34]. Our study continues this investigation of *perceived* effects of multitasking on productivity, and also triangulates the findings with repository analysis on multitasking and productivity.

2.1 Research Questions

Our study was guided by two research questions. First, we seek a deeper understanding of multitasking and its effects for software developers active on GitHub.

RQ1. What are the trends, reasons for, and effects of multitasking and focus shifting on developer productivity in GitHub?

Second, having identified the important multitasking and focus shifting effectors on programmer productivity, we proceed to investigate how these dimensions interact, and which tradeoffs between them exist.

RQ2. Are there limits on multitasking (and what are they) before productivity is impacted?

3. METHODS

To answer our research questions, we followed a mixed-methods approach characterized by a sequential explanatory strategy [15]. We analyzed development activity and perceptions of prolific GitHub developers. We combined: (1) an analysis and regression modeling of repository data, to quantitatively examine the effects of multi-tasking and focus shifting on productivity; and (2) a user survey, to garner additional qualitative insight into the developers' perceptions of multitasking, focus shifting, and their effects.

3.1 Data Set

Prolific Developers We utilized a GitHub dataset collected during prior work that contains information on prolific developers with a long and active contribution history [10]. Such developers were identified using GHTorrent [19], as those with: at least 5 years between their earliest and latest recorded commits; at least 500 commits in total; made to at least 10 different non-fork repositories. The dataset contains details about the date, size (LOC added and removed), and contents (files touched) of all commits authored by 1,255 developers across 58,092 public repositories accessible on GitHub at the time of mining. Commit data was obtained by cloning each repository locally and parsing its git logs, and was used for our repository analysis. Multiple aliases used by a single developer were resolved using a series of heuristics [52]. The languages of source code files were determined using file extensions and some contextual information [10]. In addition, we invited the developers in the most active three-quartiles (by total count of days active) for the user survey.

Conceptualization of a Project Since our analysis focuses on multitasking across projects, we need to conceptualize how we define the boundaries of a software project. The naive approach would be to consider each GitHub repository as its own separate project. However, we observed that in some cases, software projects are organized into multiple separate repositories on GitHub.[4] Such repositories are conceptually and technically related, or even interdependent [7]. It is arguably less costly to switch between such related repositories (in terms of context switches) than other, less related repositories on GitHub.

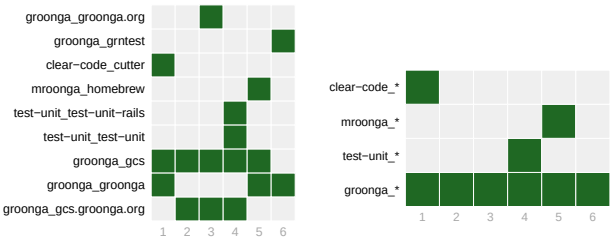

Figure 1: *Left*: Sample from one developer's daily contributions during a week in 2012. *Right*: Aggregation of the data into projects. The x-axis is the day index.

To account for this technicality, we conservatively group all repositories owned by the same GitHub user or having exactly the same name[5] into sets of repositories, which we hereafter refer to as *projects*. All data about individual contributions to these repositories is then correspondingly aggregated at project level. This transformation ensures that we do not overestimate the number of focus shifts occurring during short time intervals. For example, the left part of Figure 1 depicts a sample from a developer's daily activity across different repositories in week in 2012 (a day [column] – repository [row] cell is highlighted if the developer has contributed commits to that repository that day; overall inactive days have been omitted). In total, the developer has contributed commits to 9 different repositories over the course of that week. On the busiest day, she contributed to 4 of these. The resulting aggregation of this data, using our conceptualization of a project, depicted in the right part of Figure 1, shows contributions to only 4 projects.

Note that our definition of *project* as a collection of repositories owned by the same GitHub user is orthogonal to the definition of project as a main repository together with all its forks, proposed in the literature [24].

3.2 Multi-project Multitasking Productivity

For the quantitative analysis, we developed a model to investigate the relationship between productivity and a multitude of factors relating to multitasking and focus shifting. Here we describe the factors and the model itself.

3.2.1 Temporal Resolution

An important question when studying multitasking and focus shifting is what time interval to consider. During a longer period of time, be it objective (*e.g.*, week, month, year) or context-dependent (*e.g.*, project duration, length of employment, major release), a developer may experience many focus shifts without her necessarily also "juggling" many tasks over a shorter interval (*e.g.*, hour, day, working session). For example, suppose Alice and Bob both work on four different projects in a given week, and are free to schedule their tasks on each of the projects; when viewed at this resolution, both can be said to switch contexts (shift focus) between their four projects over the course of a week; how-

[4]The Ruby Standard Library project is a typical example. `https://github.com/rubysl`

[5]Recall that repositories on GitHub are identified by the `user_login/repository_name` stub.

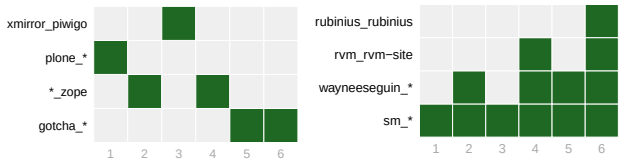

Figure 2: Two developers contributed to 4 projects in a given week. *Left*: Sequentially; *Right*: Multitasking.

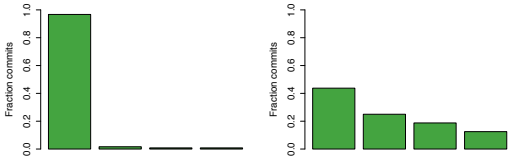

Figure 3: Distributions of commits over projects for two developers contributed to 4 projects in a given week. Both worked sequentially, on not more than one project each day. *Left*: $S_{\text{Focus}} = 0.25$. *Right*: $S_{\text{Focus}} = 1.85$

ever, Alice may choose to schedule her tasks sequentially, such that she never works on more than one project each day, while Bob may prefer to interleave his daily tasks, by contributing to multiple projects each day.

We modeled the interplay between focus shifting at two temporal resolutions—daily and weekly,[6] and using different measures, as described next.

3.2.2 Multitasking Multidimensions

Projects per day. At the finest temporal resolution (day), we measured multitasking activity using the number of different projects contributed to that day. Clearly, this represents a lower bound on the number of switches per day: a developer contributing to k projects in one day has to shift focus at least k times that day.[7] Similar count-based metrics are used in other studies to represent task switching [1].

At the coarsest temporal resolution (week), we used the average number of projects per day (denoted AvgProjectsPerDay) as an aggregate measure of multitasking, as per [4] (inactive days are excluded). For example, if Alice was active on GitHub for 6 days during a particular week, with the distribution of projects day_1: 2 different projects; day_2: 3; day_3: 1; day_4: 1; day_5: 3; and day_6: 2, then we say that Alice contributed to $(2+3+1+1+3+2)/6 = 2$ projects per day on average that week.

AvgProjectsPerDay captures the distinction between developers who, over the course of a week, tend to work on projects sequentially day-to-day (AvgProjectsPerDay=1 in Figure 2 left), and those who interleave contributions to multiple projects each day, *i.e.*, multitask (AvgProjectsPerDay=2.2 in Figure 2 right).

Weekly Focus. While effective at identifying periods of sequential and interleaved contributions, the AvgProjectsPerDay measure is not useful to distinguish how evenly developers divide their attention among their projects. This differentiation is important because it has been found that more narrowly focused developers have less cognitive burden, resulting in higher productivity and quality [43, 54]. For example, Alice and Bob both contributed to 4 projects one week, and both worked sequentially (*i.e.*, AvgProjectsPerDay=1); however, over the course of that week, Alice focused her efforts on only one of the 4 projects (Figure 3 left), *i.e.*, most of her commits went to one project; in contrast, Bob contributed evenly to each of his 4 projects (Figure 3 right).

From an information theoretic point of view, Alice's weekly focus shifting behavior is very predictable: knowing the historical distribution of her commits to different projects, one can accurately predict the object of Alice's next focus, *i.e.*, her next contribution is more likely to be in a project to which she contributes frequently. Stated differently, Alice is less likely to shift focus between projects, since she spends most of her time contributing to a single project. In contrast, there is much more uncertainty in Bob's focus shifting behavior: since he contributes more uniformly to all of his projects, he is approximately equally likely to contribute to any one of them in the next time period. Bob is more likely to shift focus than Alice.

We measured the uncertainty in a developer's focus shifting behavior (or the diversity of focus shifts) in a given week using the Teachman/Shannon entropy index, a commonly used measure in many scientific disciplines [4, 6, 54]. We denote this measure S_{Focus}, defined as:

$$S_{\text{Focus}} = -\sum_{i=1}^{N} p_i \log_2 p_i, \tag{1}$$

where p_i is the fraction of the developer's commits this week,[8] made to project i, and N is the total number of projects this week. S_{Focus} ranges between 0, when a developer contributes to a single project that week, and $\log_2 N$, when a developer contributes equally (*i.e.*, $p_i = 1/N$) to all N projects.

Similarly, we measured a developer's language entropy S_{Language}, defined analogously over the L different programming languages of the files touched that week:

$$S_{\text{Language}} = -\sum_{i=1}^{L} p_i \log_2 p_i \tag{2}$$

S_{Language} is a proxy for the overall complexity of one's contributions to different projects: all other things being equal, it is reasonable to assume that one can be more productive when writing code in only a few programming languages; otherwise, in addition to switching focus between different projects (which would involve restoring project-specific contexts), one would also have to switch focus between different languages (which may involve different skills).

Day-to-Day Focus. To capture the diversity of focus shifts also at a finer temporal resolution (day), we adapted the focus shifting networks (FSNs) developed by Xuan *et al.* [54] for file-level focus. In our case, for a developer contributing to N projects in a given week, her FSN is a weighted directed graph over all projects. Two projects i and j are connected by an edge if they have been committed to by the developer on successive, but not necessarily consecutive, days. Edges point from the earlier-commit project to the later. The weight of the edge is the number of times the shift from i and j occurred.[9]

[6] One week is considered a relevant period in software development [36]. For convenience, our two temporal resolutions both use objective time intervals; subjective time intervals cannot be directly inferred from repository data.

[7] We assume that going from a neutral state, before contributing to any projects that day, to a state of focus imposed by contributing to the first project of the day, also counts as one switch.

[8] p_i can also be defined in terms of files touched, or LOC added/removed; all tend to be highly correlated.

[9] Xuan *et al.* [54] used slightly different weights.

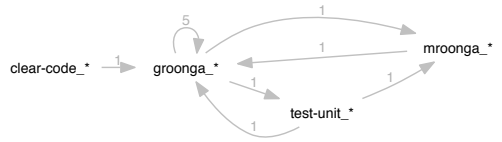

Figure 4: Corresponding FSN for Figure 1 (right). Number of commits to a project in parentheses.

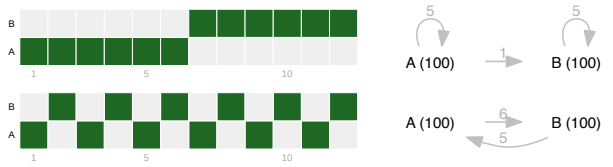

Figure 5: FSNs capture different aspects of one's focus shifting behavior than AvgProjectsPerDay and S_{Focus}.

For example, recall the activity of the developer in Figure 1-right. Her corresponding FSN is depicted in Figure 4. There are 4 nodes corresponding to the 4 different projects she contributed to, over the 6 active days that week: `clear-code_*`, `groonga_*`, `test-unit_*`, `mroonga_*`. The day-to-day shift from `clear-code_*` to `groonga_*` occurred once (day_1–day_2); the shift from `groonga_*` to `groonga_*` is also recorded, since it occurred 5 times (all subsequent days except the last); other edges are constructed similarly.

FSNs capture additional information about a developer's day-to-day multitasking and focus shifting behavior (relative to AvgProjectsPerDay and S_{Focus}), since they are based on time-series of commits rather than just aggregate counts. To illustrate this, consider the two scenarios in Figure 5. Alice and Bob both contribute equally to two projects A and B (100 commits to each; $S_{\text{Focus}} = 1$), and they never multitask (AvgProjectsPerDay $= 1$). However, in the first scenario Alice finishes all her tasks on project A before starting work on project B, while in the second scenario Bob alternates between days focused on A, and days focused on B. While neither AvgProjectsPerDay nor S_{Focus} distinguish between these scenarios, the corresponding FSNs (shown on the right in Figure 5) capture the two behaviors.

To leverage this, we considered focus shifting as a Markov process—the next state is completely determined by the current one, and we measured the diversity of a developer's *day-to-day* focus shifts (as opposed to aggregated at week-level with S_{Focus}) using Markov entropy [54]. We refer to the measure as S_{Switch}, defined as:

$$S_{\text{Switch}} = -\sum_{i=1}^{N} \left[p_i \sum_{j \in \pi_i} p(j|i) \log_2 p(j|i) \right], \qquad (3)$$

where π_i is the set of outgoing neighbors of node i (*e.g.*, $\pi_A = \{A, B\}$; $\pi_B = \{B\}$ for the top FSN in Figure 5); $p(j|i)$ is the conditional probability that the developer shifts focus from i to j, defined as $p(j|i) = \frac{w_{ij}}{\sum_{k \in \pi_i} w_{ik}}$; w_{ij} is the weight of the edge $i \to j$; and p_i is defined as above.

S_{Switch} can be seen as a measure of the cyclicity of one's focus shifts: the lower the value, the more cyclic one's day-to-day behavior is. To illustrate this, consider the scenario in Figure 6, in which Charlie contributes to both projects A and B every day ($S_{\text{Switch}} = 2$). Charlie's working style is the opposite of Bob's (Figure 5; $S_{\text{Switch}} = 0$): while Bob cycles back and forth between A and B on subsequent days, Charlie uniformly contributes to both projects throughout

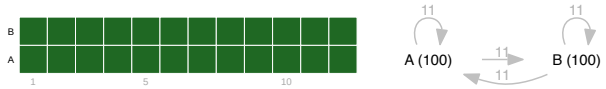

Figure 6: Example of a non cyclical behavior.

the entire period. Relative to Charlie, Alice's behavior is closer to Bob's ($S_{\text{Switch}} = 0.325$).

3.2.3 Regression Analysis

We modeled the variability in multitasking productivity (i.e. number of file touches), as dependent on control measures and the three dimensions of multitasking, projects per day, weekly focus, and day-to-day focus. For each developer, our data consists of measurements of the different variables across multiple multitasking weeks (only weeks with Projects > 1, were modeled).

As customary in regression modeling with skewed predictors, as some of ours are, we first removed outliers. Whenever one of our variables x was well fitted by an exponential distribution, we rejected as outliers values that exceeded $k(1+2/n)\text{median}(x)+\theta$ [42], where θ is the exponential parameter [45], and k is computed such that not more than 0.5% of values are labeled as outliers, for each variable. Table 1 presents summary statistics for our filtered data set.

Then we fit a linear mixed-effects model with a random-effects term for developer. This allows us to capture developer-to-developer variability in the response (productivity), (*e.g.*, some developers being naturally more productive than others), rather than assessing the contributions of specific developers, which we are less interested in. Additionally, we allow for deviations in slope of a developer's time trend from the population values (*i.e.*, we allow for the possibility that, for example, developers with higher initial productivity may, on average, be less strongly affected by time passing). We also tested, but found insignificant, the inclusion of a random-effect term for the time window in which the measurement was taken, to capture longitudinal, week-to-week variability (*e.g.*, some weeks developers may be more productive due to other, unobserved variables). All other variables were modeled as fixed effects. We used multiple linear mixed-effects models, as implemented in the functions `lmer` and `lmer.test` in R. Coefficients are considered important if they were statistically significant ($p < 0.05$). Their effect sizes are obtained from ANOVA analyses. We evaluate our model's fit using a marginal (R^2_m) and a conditional (R^2_c) coefficient of determination for generalized mixed-effects models [22, 35], as implemented in the `MuMIn` package in R: R^2_m describes the proportion of variance explained by the fixed effects alone; R^2_c describes the proportion of variance explained by both the fixed and random effects.

To check collinearity among the predictors we use the VIF, or variance inflation factor; all were below 4, except for those between the interaction terms and their comprising factors, which is expected. We conclude collinearity between our variables is not an issue. To ensure compliance with `lmer`'s modeling assumptions, we also checked the QQ-plot for our model, which showed good match with a normal distribution. The residuals between the observed and model fitted values for productivity showed no difference in variance variability across the range.

Regression Variables.

Table 1: Summary statistics for week-level data (78,562 rows; 1,193 developers; outliers removed).

Statistic	Mean	St. Dev.	Min	Median	Max
GlobalTime	1,142.76	74.23	991	1,150	1,263
UserTime	308.04	194.12	1	271	2,307
Repositories	3.70	2.22	2	3	18
Projects	2.57	0.98	2	2	9
DaysActive	3.98	1.63	1	4	7
Languages	3.19	1.43	1	3	14
Commits	23.89	30.06	2	15	943
FileTouches	88.55	174.30	2	39	2,737
LOCAdded	4,150.83	15,690.22	0	635	265,702
LOCDeleted	2,282.01	9,216.86	0	247	148,977
S_{Language}	0.82	0.52	0.00	0.84	2.92
AvgProjectsPerDay	1.44	0.48	1.00	1.33	6.00
S_{Focus}	0.92	0.44	0.02	0.92	3.01
S_{Switch}	0.64	0.53	0.00	0.69	2.83

Our response is **FileTouches**, as a measure of productivity. It sums the count of files touched per commit, over all commits in a given week (the same file, if touched in different commits, is counted each time). We also experimented with Commits and LOCAdded. All are highly correlated (rho $\simeq 0.82$ in both cases). Using FileTouches results in the best model fit, consistent with our previous experience [10].

The main predictors are **AvgProjectsPerDay** (Multitasking), $\boldsymbol{S_{\text{Focus}}}$ (Weekly Focus), and $\boldsymbol{S_{\text{Switch}}}$ (Day-to-Day Focus), discussed above, and their interactions.

Our controls are:

GlobalTime: Week index of the current week, with respect to 1990-01-01 (chosen arbitrarily for simplicity). Controls for potential generic environmental changes with time.

Projects: Total number of projects contributed to in a given week, as a control.

Languages: Number of different programming languages across all files touched that week, as a control. Touching files in more programming languages confounds FileTouches.

$\boldsymbol{S_{\text{Language}}}$: Controls for the overall complexity of one's contributions.

UserTime: Per developer index of the current week, relative to one's first ever recorded GitHub contribution. Controls for changes in developer experience with time, that may have affected individual productivity.

3.2.4 Hypothesis Testing

To test for a difference in the means between two populations we use the non parametric Wilcoxon-Mann-Whitney test, for unpaired samples, and the Mann-Whitney paired test for paired samples. We interpret the results using p-values, indicating the likelihood of a hypothesis being true by chance, and supplement those with the Hodges-Lehmann point estimate for effect sizes, $\hat{\Delta}$. We also use the multiple contrast test procedure $\widetilde{\mathbf{T}}$ [27] for Tukey-type contrasts at 95% confidence level, to distinguish significant from non-significant differences between all pairs of distributions.

3.3 User Survey

We sent an online survey to 851 GitHub users selected from the set of prolific developers described earlier. The survey included multiple choice, Likert scale, and open-ended questions. We asked users about their software development experience in general, and with GitHub; which factors influence their contributing to new repositories; what makes them switch between projects; and their perceptions of the impacts of contributing to multiple projects.[10] We received 128 responses (15% response rate). We iterated through the survey responses using grounded theory methods [12], to categorize them and identify themes that emerged from the open-ended questions. The survey participants reported' development experience was 17.2 years on average (median 15; range 7 to 40), while their GitHub experience was 5.9 years on average (median 6; range less than 1 to since GitHub).

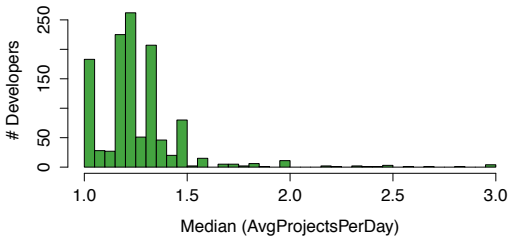

Figure 7: Distribution of median(AvgProjectsPerDay).

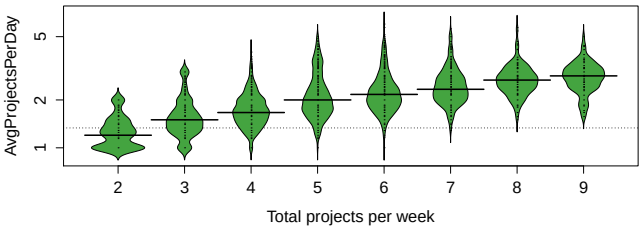

Figure 8: Beanplots showing the distributions of AvgProjectsPerDay, for different values of Projects. The solid horizontal lines represent the medians in each group. The dotted horizontal line is the overall median. Log y-axis.

4. RESULTS

4.1 RQ1: Multitasking, summarized.

4.1.1 Amount of Multitasking

Do developers multitask? To identify how much developers multitask across projects, we examined their commit activity and also asked survey participants to report the number of projects they contribute to in an average day and week. From the repository analysis, we found that multitasking across projects over the course of a week is not uncommon: developers contributed to multiple projects in 37% of the developer–weeks in our dataset (or 78,562 out of 132,277). The answer is not quite as clear for daily multitasking. Consider the distribution of each developer's median weekly AvgProjectsPerDay in weeks where they contributed to multiple projects (Figure 7). The distribution is right-skewed ($\gamma_1 = 3.08$). Even when contributing to multiple projects during a week, some developers still frequently focus on only few projects each day: 15% (183 out of 1,193) did not contribute (on average) to more than one project each day, at least half of the time (median = 1); 95% (1,129 out of 1,193) did not contribute (on average) to more than 1.5 projects each day. Still, almost all developers (98%; 1,165 out of 1,193) contributed to multiple projects per day at least once.

[10] A complete list of questions is available at `http://kblincoe.github.io/survey/Focus_Shifting_Survey.pdf`.

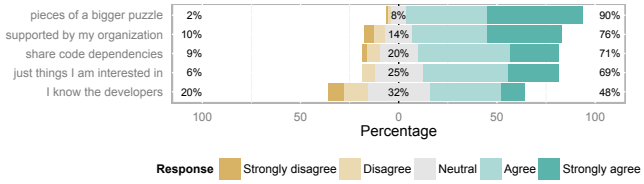

Figure 9: Reasons for multitasking. Survey responses on how projects switched between are related.

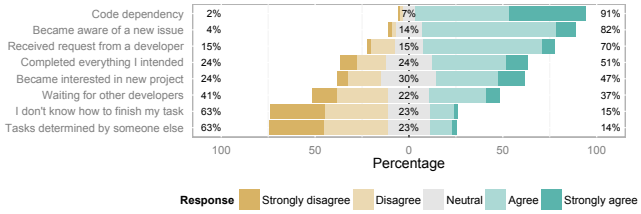

Figure 10: Reasons for switching between projects. Survey responses on why developers switch from one project to another when working on multiple projects.

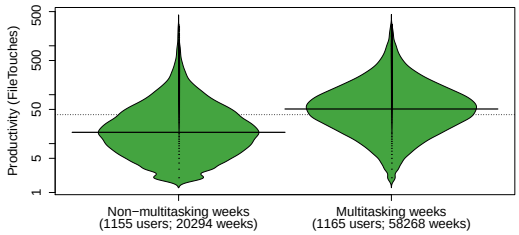

Figure 11: Beanplots comparing the distributions of FileTouches during periods of sequential working (left) and multitasking (right) Solid horizontal lines: medians in each group. Dotted horizontal line: overall median. Log y-axis.

Our survey participants reported contributing to an average of 2.7 projects per day (median 2; range 0–10). Over the course of a week, respondents stated that they contribute to an average of 6 projects (median 5; range 0–30).

Does within-day multitasking scale with the number of projects per week? We investigated whether contributing to more projects each week is also associated with contributing to more projects daily (therefore with more within-day focus shifts), as one would expect, or if there is evidence that developers avoid daily multitasking. Figure 8 summarizes the distribution of AvgProjectsPerDay for different values of Projects (each beanplot shows the distribution of AvgProjectsPerDay for all users over all weeks). Our finding is that *developers do significantly more daily context switches as they participate in more projects per week.* All groups are significantly different pairwise using the $\widetilde{\mathbf{T}}$ [27] procedure at 95% confidence level, and higher Projects corresponds to higher AvgProjectsPerDay: *e.g.*, developers contributing to a median of 5 projects per week switch between a median of 2 each day, although in fact they could have focused on only one each day.

4.1.2 Reasons for Multitasking

Our survey investigated reasons for working on multiple projects. We asked the respondents to indicate, using a five-point Likert-type scale, their agreement to a number of statements about how the projects they contribute to are related. As shown in Figure 9, interdependencies, personal interest, and social relationships were all stated as strong reasons for multitasking. Participants were also able to provide other ways the projects they contribute to are related: common responses were that they are an end user of the software tool and want to fix bugs impacting it, and that they contribute to a mix of projects due to their job as well as their personal interests.

We also asked about reasons for switching between projects when working on multiple. The strongest were all related to a need being identified in another project, either because of dependencies, newly created issues, or a request from another developer. The summarized responses are shown in Figure 10. Again, there was space for participants to describe other ways they schedule their work on multiple projects. The most common answer was that they simply liked to change focus for the sake of working on something different. For example, one participant said "I find it easier to produce quality code if I break up long stretches working on a single codebase with time spent on a completely different problem. It freshens up the mind." [P57]

4.1.3 Productivity Effects

Is multitasking associated with higher productivity? Through a preliminary quantitative analysis, we found that more focus shifts are associated with higher productivity. Splitting the data into two groups (weeks of sequential work—AvgProjectsPerDay=1—and weeks with multitasking; Figure 11), we find using the WMW rank-sum test that periods of multitasking are significantly more productive in terms of FileTouches per week[11] ($p < 2.2\text{e-}16$; $\hat{\Delta} = 27$).

Still, this analysis does not consider many confounds. As a refinement, we turn to the multiple regression analysis. Table 2 shows the effects of our independent and control variables for the multitasking productivity model; the response is log(FileTouches). In addition to the model coefficients and corresponding standard errors, the table shows the sum of squares, a measure of variance explained, for each variable. All predictors have been z-transformed to reduce potential collinearity in the interaction terms. The statistical significance is indicated by stars. The model fits the data well; it explains 43% of the variability in the data using only the fixed effects ($R^2_m = 0.43$), and 63% when considering both the fixed and random effects ($R^2_c = 0.63$).

We start by discussing the effects associated with our controls. As expected, Projects and Languages have significant, positive effects on the response. *Developers who contribute to more projects and who use more programming languages are associated with higher productivity.* Together, these two effects explain about 60% of the variance explained.

More interestingly, S_{Focus} has a strong, negative effect on productivity, accounting for 18% of the variance explained. Recall that S_{Focus} captures a developer's overall project focus at week level (the lower the value, the more of one's time goes into a single project; the higher the value, the more evenly one divides their attention across projects). This suggests that *developers keeping fewer projects as their focus, rather than spreading themselves thin, are associated with higher productivity.* S_{Language} paints a similar picture, albeit with a smaller effect: focusing on fewer languages requires less context switching between them, and is associated with higher productivity.

[11]The observation remains valid for other operationalizations of productivity: Commits, LOCAdded.

	Coeffs (Errors)	Sum Sq.
(Intercept)	0.037 (0.012)**	
GlobalTime	−0.028 (0.005)***	6.29***
Projects	0.371 (0.006)***	3791.21***
Languages	0.440 (0.004)***	12146.37***
S_{Focus}	−0.424 (0.004)***	4874.55***
S_{Language}	−0.067 (0.003)***	186.14***
AvgProjectsPerDay	0.103 (0.004)***	675.65***
S_{Switch}	0.272 (0.003)***	3294.50***
Projects:S_{Focus}	0.037 (0.002)***	2.59***
Languages:S_{Language}	−0.032 (0.002)***	251.87***
Projects:S_{Language}	−0.030 (0.003)***	117.82***
Projects:AvgProjectsPerDay	−0.035 (0.003)***	192.21***
S_{Language}:AvgProjectsPerDay	0.029 (0.003)***	92.35***
Projects:S_{Switch}	−0.127 (0.003)***	562.66***
AvgProjectsPerDay:S_{Switch}	0.080 (0.003)***	385.20***
S_{Language}:S_{Switch}	−0.030 (0.003)***	38.04***
AIC = 157380; BIC = 157566; LogLik = -78670		
Num. obs.	78562	
Num. groups: fUserID	1193	
Variance: fUserID.(Intercept)	0.128	
Variance: fUserID.UserTime)	0.097	
Variance: Residual	0.410	

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

Table 2: Multitasking productivity model. The response is log(FileTouches) per week. $R^2_m = 0.43$. $R^2_c = 0.63$.

Within-day multitasking (AvgProjectsPerDay) has a significant positive effect on productivity (2.5% of the variance explained). *More multitasking is associated with higher productivity.* Switching projects allows developers to make more efficient use of their time when one project reaches a lull, and provides them with opportunities for learning and knowledge transfer, as discussed in the Background section.

Day-to-day focus switching (S_{Switch}) also has a strong, significant effect (12% of the variance explained). Recall that S_{Switch} captures the diversity of focus shifts from one day to the next (higher values mean less repetitive switches). The positive coefficient suggests that *repetitive, predictable day-to-day switches are detrimental to productivity.* This is consistent with research in psychology explaining that performance is suboptimal at low levels of arousal [49], which can be associated with very repetitive work; instead, task switching constitutes additional stimulation [23], which leads to higher productivity.

4.1.4 Perceived Impacts of Focus Shifting

Positive Impacts. We asked developers about the impacts of contributing to multiple projects in parallel. The responses are summarized in Figure 12. The impact with the highest level of agreement (47%) was that contributing to multiple projects increases the chances of each project succeeding. There was a weak correlation between the number of projects a developer reported contributing to each week and their perceived impact of this behavior on project success (rho = 0.25, $p = 0.005$). This shows that the more cross-project focus shifting a developer does, the more they believe this will increase the project's success. This could indicate that developers, especially those who do multitask frequently, are aware of the knowledge transfer benefits [28]. In fact, many respondents described this benefit. For example, one participant said "Working on different projects, especially when the ecosystem (programming language, runtime environment) varies greatly, increases the chance to take advantage of things learned from one project in another." [P7]

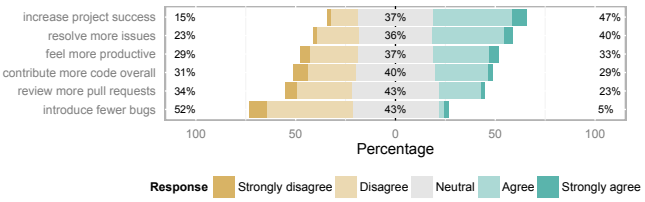

Figure 12: Perceived impacts of multitasking. Survey responses on the impacts of contributing to multiple projects in parallel (compared to focusing on fewer projects).

40% of respondents also agreed that when working on multiple projects they are able to resolve more issues. Respondents noted several reasons for this, such as the reduced wait time when working on multiple projects. For example, " I can switch to another project when I get stuck and need some time to work out a good solution" [P96], and "I don't get blocked [waiting] on a dependency (by contributing to the dependency [myself])."[P6] Other positive impacts noted in the open-ended responses include an increased social network and reduced overall effort, for example, due to fixing a bug in the correct location rather than implementing multiple workarounds in many dependent projects.

Negative Impacts. However, participants were not very positive about the effects of multitasking on the quality of their code: only 5% agreed that they introduce *fewer* bugs when focus shifting. In the open-ended responses, the most commonly cited negative impact was related to the cost of context switches; *e.g.*, one participant stated "Working on multiple projects also requires more 'context switching' mentally, which can be a drag on productivity." [P107]

Confusion. While our repository analysis showed significant effects on productivity, developers perceptions around this were mixed. For three of the statements relating to the impact on productivity ("contribute more code"—the response variable we studied also quantitatively; "review more pull requests"; and "feel more productive"), there was a high amount of variance in the responses: some believe they are more productive while working on multiple projects, while nearly as many believe that they are not. We did observe a difference in responses based on how many projects developers reportedly contribute to. Comparing the responses of those who shift focus across many projects, with those that do not (work on only 1 project per day, or 1-2 projects per week), we find that the more frequent focus shifters are more likely to report positive benefits from this multitasking; results in Table 3.

4.2 RQ2: Limits on multitasking.

4.2.1 Quantitative Analysis

While the discussion above paints a general picture of the directionality of the various multitasking effects on productivity, the model in Table 2 also yields interesting two-way interactions. We illustrate the interpretation of such interaction terms with the example Projects—AvgProjectsPerDay. Keeping all other variables constant, the effect of the interaction on the response can be described by the equation:

$$\begin{aligned}\text{Productivity} =& \alpha_1 \text{Projects} + \alpha_2 \text{AvgProjectsPerDay} \\ &+ \alpha_3 \text{Projects} \cdot \text{AvgProjectsPerDay}, \end{aligned} \quad (4)$$

where α_1, α_2, and α_3 are the coefficients in the model.

To understand this interaction, we follow the standard practice of varying one term while holding the other con-

	Day			Week		
Projects	> 1	1	U	> 2	1-2	U
n	102	19		104	17	
Feel more productive	0.11	-0.37	1242*	0.13	-0.59	1269.5**
Contribute more code	0.02	-0.63	1377**	0	-0.59	1219**
Review more pull reqs	-0.04	-0.68	1348.5**	-0.05	-0.71	1236**
Resolve more issues	0.23	0.05	1091.5	0.27	-0.24	1177*
Introduce fewer bugs	-0.49	-0.74	1144	-0.49	-0.76	1084
Increase project success	0.43	-0.05	1259*	0.46	-0.29	1309.5***

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

Table 3: Impacts of multitasking segregated based on respondents who reported multitasking (>1 project per day or >2 projects per week) compared to all other responses. Table shows the mean response (and Mann-Whitney U) where responses were coded as follows: strongly disagree (-2), disagree (-1), neutral (0), agree (1), strongly agree (2). Multitaskers reported more positive benefits from multitasking.

stant, and vice versa. For example, fixing Projects (which in our data set varies between 2 and 9; recall we only modeled weeks with some amount of multitasking) allows us to examine the effects of increasing AvgProjectsPerDay on the response (*what happens to productivity the more daily multitasking there is?*), for different levels of Projects. We illustrate this process in Figure 13, where the two regression lines shown in black correspond to the extremes of Projects. When Projects is at its lowest (2), increasing AvgProjectsPerDay is always associated with an increase in the response, since the slope is positive ($\beta = 0.33$). In other words, more daily multitasking is associated with higher productivity for developers that only take on 2 projects per week. At the other end of the scale, when Projects = 9, AvgProjectsPerDay is always associated with a decrease in the response, since the slope is negative. In other words, the more daily multitasking, the lower one's productivity, if developers take on too many different projects that week. Resolving $\alpha_2 + \alpha_3 \text{Projects} = 0$ allows us to estimate the point of diminishing return, *i.e.*, Projects = 5 (horizontal regression line shown in gray). Combining the two findings, we conclude: *higher average daily multitasking is associated with higher productivity (albeit with diminishing returns), as long as developers don't take on more than 5 projects per week.*[12]

Similarly, we express tradeoffs between the other dimensions of multitasking and focus shifting by investigating the interaction between two other pairs: (1) AvgProjectsPerDay and S_{Switch} and (2) Projects and S_{Switch}.

The interaction between AvgProjectsPerDay and S_{Switch} reveals that *higher average daily multitasking is associated with higher productivity when one's day-to-day focus switching is more repetitive.* This suggests that one can handle more complexity each day, as long as one doesn't have to also handle more complexity day-to-day.

The interaction between Projects and S_{Switch} shows *higher average weekly multitasking is associated with higher productivity when developers don't follow too repetitive day-to-day switching patterns.* This suggests that working on many projects can prevent boredom and be beneficial; however, if it gets too repetitive, productivity goes down.

4.2.2 Developer Perceptions on Limits

[12]We find a similar result when modeling at month level: higher average daily multitasking is associated with higher productivity, as long as developers don't take on more than 7 projects per month.

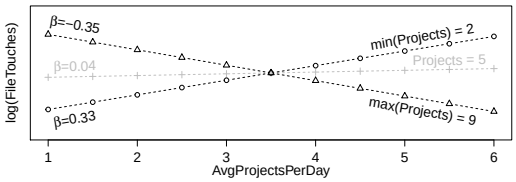

Figure 13: Interaction Projects (week) : AvgProjectsPerDay

Complementary to this repository analysis, we asked the survey respondents if they would prefer to contribute to more projects, fewer projects, or their current number of projects. Participants are most likely to want to increase the number of projects they contribute to. Only 14.6% of respondents indicated that they would prefer to contribute to fewer projects (46.9% to more; 38.5% current amount).

However, a Spearman's rank correlation test (due to shows the number of projects a developer already contributes to in an average day and week do not correlate with their desire to contribute to more, fewer, or the same number of projects (day: rho = 0.10, p = 0.29; week: rho = 0.01, p = 0.86). Developers who contribute to the most projects are no more likely to want to contribute to fewer and vice versa. This could imply that developers do not have a good sense of when to stop multitasking, despite the negative effects to productivity outlined by theory and confirmed by our model.

However, we find that *participants who want to contribute to more projects are more optimistic about the impacts of multitasking.* The exception to this is the impact to code quality (bugs introduced), to which all participants responded equally negatively. Figure 14 shows the survey responses relating to the impacts of multitasking, grouped by the participant's reported desire to contribute to more, fewer, or their current number of projects. Thus, for each statement, three bars show the aggregated responses for each of these three groups of participants (want to contribute to more/fewer/same number of projects). The responses for the "want more" group are skewed to the right compared to the other two groups (for all statements other than "introduce fewer bugs"), showing the responses from this group are more positive.

5. DISCUSSION

We found that developers do a significant amount of multitasking across projects over the course of a week. Developers indicated that the most common reason for this multitasking is the interrelationships between projects. Focus switches often occur because of a need identified on another project, indicating that such focus shifting is not planned, but occurs because of necessity. Yet, we found that despite these unplanned interruptions, developers who shift focus across projects are more productive than those who do not. However, developers' perceptions towards multitasking and its effect on productivity are varied. Those who do multitask are more likely to believe multitasking can be beneficial.

However, we found that limits exist on the amount of focus switching a developer can participate in before their productivity begins to decline. Yet, developers seem to have an *irrational ambition*: they lack a good sense of when to stop multitasking. Nearly half of all survey participants would prefer to contribute to more than their current number of projects. We saw no differences in response based on the

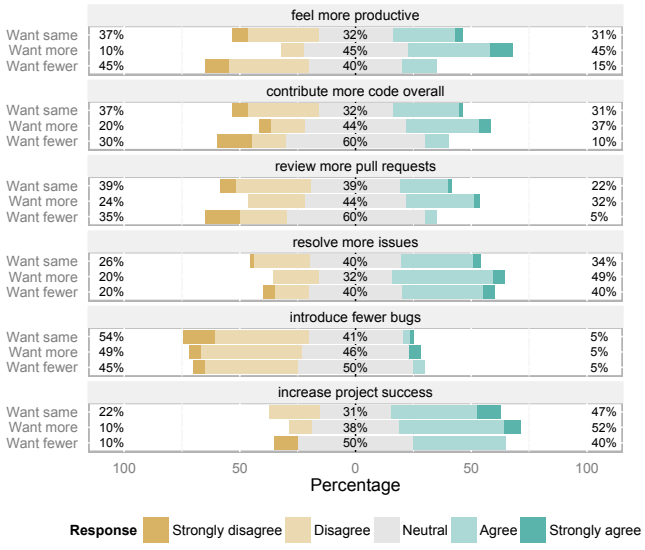

Figure 14: Survey responses on the impacts of multitasking, grouped based on the respondents' desire to increase/decrease multitasking. Those who wish to increase multitasking are more positive about the benefits.

number of projects a developer already contributes to, indicating they do not seem to be aware of the limits of multitasking.

5.1 Research Agenda

While our investigation resulted in significant new insights on multitasking and focus shifting across projects, there are still many unanswered research questions, opening some interesting avenues for future research.

Predicting productivity impacts. Our model was successful in identifying the effects of various features of multitasking on developer productivity. A logical next step would be to use these models to predict the productivity of individual developers based on their past contribution patterns. A predictive model could be the foundation of a tool that can monitor developer focus shifting behavior, and provide warnings to developers (or their managers) when risky patterns emerge that could indicate productivity will be negatively impacted.

Further investigation of non-multitaskers. Our survey participants who reported no or very little multitasking, were more pessimistic about the impacts of multitasking on their productivity and project success. Future work can investigate why developers choose not to multitask. Do non-multitaskers contribute to projects with fewer dependencies to other projects, resulting in a reduced need for multitasking? If the projects do not have fewer dependencies, how do non-multitaskers avoid switching focus when these dependencies necessitate changes in another project? Do they simply create an issue on that project and wait for someone else to make the change, resulting in reduced productivity? The answers to these questions can result in the conceptualization of methods and tools designed to change the perceptions and habits of these developers, to promote an increase in multitasking to reap the associated productivity benefits.

Further investigation of reasons for focus shifts. We identified limits to multitasking. Contributing to more projects per week results in higher productivity, as long as developers do not work on too many projects in a single day. Future work can investigate reasons for this finding. Perhaps the reasons for focus shifting is different for those who contribute to many projects per day, compared to those who contribute to only a small number of projects per day. The differences between these two groups may provide additional insight, revealing which focus shifting reasons are more likely to result in increased productivity. With a greater understanding of the beneficial focus shifts, tools can be devised to help developers distinguish between the various interruptions they face.

5.2 Threats to Validity

All developers in our study are highly prolific contributors, and our survey respondents were self-selected. Thus, our results may not generalize beyond this research setting. The thematic analysis of the survey was performed by only one person introducing a possible risk of unreliable results.

We define projects by aggregating repositories related by name. Given the ease with which different projects can depend on each other on GitHub, determining the exact boundaries of a project is difficult. Our definition may overly combine repositories, but we believe it makes context switches more likely to be between distinct entities. The results may be different if a coarser or finer grained definition of project was used.

The aliased user identities refer to the original author of the commit, while the dates chosen are from the committer fields, which mark when was finally added to the main repository. These fields can be different if the commit was originally created in a fork. We found the author and committer to be different in only 1.78% commits, so we do not believe this will significantly affect the results.

Finally, the projects we extracted were based on data from GitHub and GHTorrent, which limits the generalizability.

Measuring productivity as the number of file touches, or any other count of user interaction with the artifact, is certainly somewhat naive, and incomplete. But it is certainly not unprecedented nor unjustified by prior research [54].

6. CONCLUSIONS

Programming is a multi-faceted task, and context-switching is inherently involved. With the advent of ecosystems like Github, another tier of context-switching becomes possible: switching between projects. Using a mixed-methods approach (survey+quantitative analysis of mined data) we study this phenomenon. We find that up to a certain point, switching between projects actually raises productivity; but beyond that, it appears that overall output of developers declines. Paradoxically, the survey responses indicate that developers seem to believe it's always better to work on evermore projects, regardless of how many they already work on! Our work is *actionable*: both managers and developers can take advantage of our findings to manage both the number of projects they work on and the intensity of their context-switching.

7. REFERENCES

[1] R. F. Adler and R. Benbunan-Fich. Juggling on a high wire: Multitasking effects on performance. *International Journal of Human-Computer Studies*, 70(2):156–168, 2012.

[2] E. M. Altmann and J. G. Trafton. Memory for goals: An activation-based model. *Cognitive Science*, 26(1):39–83, 2002.

[3] J. R. Anderson. Human symbol manipulation within an integrated cognitive architecture. *Cognitive science*, 29(3):313–341, 2005.

[4] S. Aral, E. Brynjolfsson, and M. V. Alstyne. Information, technology, and information worker productivity. *Information Systems Research*, 23(3-part-2):849–867, 2012.

[5] E. T. Barr, C. Bird, P. C. Rigby, A. Hindle, D. M. German, and P. Devanbu. Cohesive and isolated development with branches. In *FASE*, pages 316–331. Springer, 2012.

[6] R. Benbunan-Fich. An entropy index for multitasking behavior. In *ICIS*. AIS, 2011.

[7] K. Blincoe, F. Harrison, and D. Damian. Ecosystems in GitHub and a method for ecosystem identification using reference coupling. In *MSR*, pages 202–211. IEEE, 2015.

[8] J. P. Borst, N. A. Taatgen, and H. van Rijn. The problem state: a cognitive bottleneck in multitasking. *Journal of Experimental Psychology: Learning, memory, and cognition*, 36(2):363, 2010.

[9] J. P. Borst, N. A. Taatgen, and H. van Rijn. What makes interruptions disruptive? a process-model account of the effects of the problem state bottleneck on task interruption and resumption. In *CHI*, pages 2971–2980. ACM, 2015.

[10] C. Casalnuovo, B. Vasilescu, P. Devanbu, and V. Filkov. Developer onboarding in GitHub: The role of prior social links and language experience. In *FSE*, pages 817–828. IEEE, 2015.

[11] M. Cataldo and J. D. Herbsleb. Coordination breakdowns and their impact on development productivity and software failures. *IEEE TSE*, 39(3):343–360, 2013.

[12] J. Corbin and A. Strauss. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage Publications, 2014.

[13] L. Dabbish, G. Mark, and V. M. González. Why do i keep interrupting myself? environment, habit and self-interruption. In *CHI*, pages 3127–3130. ACM, 2011.

[14] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Social coding in GitHub: transparency and collaboration in an open software repository. In *CSCW*, pages 1277–1286. ACM, 2012.

[15] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian. Selecting empirical methods for software engineering research. In *Guide to Advanced Empirical Software Engineering*, pages 285–311. Springer, 2008.

[16] S. J. Gilbert and T. Shallice. Task switching: A PDP model. *Cognitive Psychology*, 44(3):297–337, 2002.

[17] V. M. González and G. Mark. Managing currents of work: Multi-tasking among multiple collaborations. In *ECSCW*, pages 143–162. Springer, 2005.

[18] A. J. Gould. *What makes an interruption disruptive? Understanding the effects of interruption relevance and timing on performance*. PhD thesis, UCL (University College London), 2014.

[19] G. Gousios and D. Spinellis. GHTorrent: Github's data from a firehose. In *MSR*, pages 12–21. IEEE, 2012.

[20] J. D. Herbsleb and A. Mockus. An empirical study of speed and communication in globally distributed software development. *IEEE TSE*, 29(6):481–494, 2003.

[21] S. Jenkins. Concerning interruptions. *Computer*, (11):116–114, 2006.

[22] P. C. Johnson. Extension of nakagawa & schielzeth's r^2_{GLMM} to random slopes models. *Methods in Ecology and Evolution*, 5(9):944–946, 2014.

[23] D. Kahneman. *Attention and effort*. Prentice-Hall, 1973.

[24] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian. The promises and perils of mining GitHub. In *MSR*, pages 92–101. ACM, 2014.

[25] M. Kersten and G. C. Murphy. Using task context to improve programmer productivity. In *FSE*, pages 1–11. ACM, 2006.

[26] A. J. Ko, R. DeLine, and G. Venolia. Information needs in collocated software development teams. In *ICSE*, pages 344–353. IEEE, 2007.

[27] F. Konietschke, L. A. Hothorn, E. Brunner, et al. Rank-based multiple test procedures and simultaneous confidence intervals. *Electronic Journal of Statistics*, 6:738–759, 2012.

[28] A. Lindbeck and D. J. Snower. Multitask learning and the reorganization of work: from Tayloristic to holistic organization. *Journal of Labor Economics*, 18(3):353–376, 2000.

[29] G. Mark, V. M. Gonzalez, and J. Harris. No task left behind? examining the nature of fragmented work. In *CHI*, pages 321–330. ACM, 2005.

[30] J. Marlow and L. Dabbish. Activity traces and signals in software developer recruitment and hiring. In *CSCW*, pages 145–156. ACM, 2013.

[31] J. Marlow, L. Dabbish, and J. Herbsleb. Impression formation in online peer production: Activity traces and personal profiles in GitHub. In *CSCW*, pages 117–128. ACM, 2013.

[32] N. McDonald and S. Goggins. Performance and participation in open source software on GitHub. In *CHI*, pages 139–144. ACM, 2013.

[33] N. McDonald and S. Goggins. Performance and participation in open source software on github. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 139–144, New York, NY, USA, 2013. ACM.

[34] A. N. Meyer, T. Fritz, G. C. Murphy, and T. Zimmermann. Software developers' perceptions of productivity. In *FSE*, pages 19–29. ACM, 2014.

[35] S. Nakagawa and H. Schielzeth. A general and simple method for obtaining r^2 from generalized linear mixed-effects models. *Methods in Ecology and Evolution*, 4(2):133–142, 2013.

[36] M. B. O'Leary, M. Mortensen, and A. W. Woolley. Multiple team membership: A theoretical model of its effects on productivity and learning for individuals and teams. *Academy of Management Review*, 36(3):461–478, 2011.

[37] C. Parnin. A cognitive neuroscience perspective on memory for programming tasks. *Programming Interest Group*, page 27, 2010.

[38] C. Parnin and R. DeLine. Evaluating cues for resuming interrupted programming tasks. In *CHI*, pages 93–102. ACM, 2010.

[39] C. Parnin and S. Rugaber. Resumption strategies for interrupted programming tasks. *Software Quality Journal*, 19(1):5–34, 2011.

[40] C. Parnin and S. Rugaber. Programmer information needs after memory failure. In *ICPC*, pages 123–132. IEEE, 2012.

[41] C. J. Parnin. *Supporting Interrupted Programming Tasks with Memory-Based Aids*. PhD thesis, Georgia Institute of Technology, 2014.

[42] J. K. Patel, C. Kapadia, and D. B. Owen. *Handbook of statistical distributions*. M. Dekker, 1976.

[43] D. Posnett, R. D'Souza, P. Devanbu, and V. Filkov. Dual ecological measures of focus in software development. In *Software Engineering (ICSE), 2013 35th International Conference on*, pages 452–461. IEEE, 2013.

[44] C. Rosen. The myth of multitasking. *The New Atlantis*, 20(Spring):105–110, 2008.

[45] P. J. Rousseeuw and C. Croux. Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88(424):1273–1283, 1993.

[46] J. S. Rubinstein, D. E. Meyer, and J. E. Evans. Executive control of cognitive processes in task switching. *Journal of Experimental Psychology: Human Perception and Performance*, 27(4):763, 2001.

[47] D. D. Salvucci, N. A. Taatgen, and J. P. Borst. Toward a unified theory of the multitasking continuum: From concurrent performance to task switching, interruption, and resumption. In *CHI*, pages 1819–1828. ACM, 2009.

[48] H. Sanchez, R. Robbes, and V. M. Gonzalez. An empirical study of work fragmentation in software evolution tasks. In *SANER*, pages 251–260. IEEE, 2015.

[49] K. H. Teigen. Yerkes-Dodson: A law for all seasons. *Theory & Psychology*, 4(4):525–547, 1994.

[50] B. Vasilescu, V. Filkov, and A. Serebrenik. Perceptions of diversity on GitHub: A user survey. In *CHASE*, pages 50–56. IEEE, 2015.

[51] B. Vasilescu, D. Posnett, B. Ray, M. G. J. van den Brand, A. Serebrenik, P. Devanbu, and V. Filkov. Gender and tenure diversity in GitHub teams. In *CHI*, pages 3789–3798. ACM, 2015.

[52] B. Vasilescu, A. Serebrenik, and V. Filkov. A data set for social diversity studies of GitHub teams. In *MSR*, pages 514–517. IEEE, 2015.

[53] Q. Xuan, P. T. Devanbu, and V. Filkov. Converging work-talk patterns in online task-oriented communities. *arXiv preprint arXiv:1404.5708*, 2014.

[54] Q. Xuan, A. Okano, P. Devanbu, and V. Filkov. Focus-shifting patterns of OSS developers and their congruence with call graphs. In *FSE*, pages 401–412. ACM, 2014.

[55] F. Zhang, F. Khomh, Y. Zou, and A. E. Hassan. An empirical study of the effect of file editing patterns on software quality. *Journal of Software: Evolution and Process*, 26(11):996–1029, 2014.

[56] A. Zika-Viktorsson, P. Sundström, and M. Engwall. Project overload: An exploratory study of work and management in multi-project settings. *International Journal of Project Management*, 24(5):385–394, 2006.

[57] L. Zou and M. W. Godfrey. An industrial case study of program artifacts viewed during maintenance tasks. In *Reverse Engineering, 2006. WCRE'06. 13th Working Conference on*, pages 71–82. IEEE, 2006.