# Productivity, Turnover, and Team Stability of Agile Teams in Open-Source Software Projects

Ezequiel Scott
*Institute of Computer Science*
*University of Tartu*
Tartu, Estonia
ezequiel.scott@ut.ee

Khaled Nimr Charkie
*Institute of Computer Science*
*University of Tartu*
Tartu, Estonia
khalednchakie@gmail.com

Dietmar Pfahl
*Institute of Computer Science*
*University of Tartu*
Tartu, Estonia
dietmar.pfahl@ut.ee

***Abstract*—Productivity in software development has been studied for a long time and is still a topic of interest. Many factors, ranging from team size to music listened by developers, have been studied regarding their effect on productivity. Surprisingly, little is known about how the dynamics of open-source projects that use agile practices are related to the productivity of the developer teams. Our study aims to close this gap by analyzing the productivity of open-source projects using measures that are popular in the context of agile software development. To do this, we study data from seven open-source projects and calculated both the individual velocity of developers and focus factor of teams per iteration. First, we applied statistical process control to identify iterations with out-of-control velocity and focus factor values. Then, we studied these iterations regarding four context factors that partly characterize the dynamics of open-source projects, i.e., iteration length, turnover of developers who left, turnover of new developers, and team stability index. Our results suggest that high team stability and low turnover are strongly associated with iterations showing high velocity.**

***Index Terms*—Agile Software Development, Open Source, Productivity, Turnover, Team Performance**

## I. Introduction

Productivity in software development has been studied extensively. The first studies on this topic appeared in the '70s [1] and the continuing interest has led researchers to study a wide range of factors, such as team size [2], developers' capabilities [3], and even the music listened by developers [4]. New studies in the field are still relevant since productivity factors are recommended to be re-evaluated in modern software development contexts [5].

In today's software projects, there is a predominant use of agile practices, not only in proprietary industry projects but also in open-source projects. Synergies between agile practices and open-source project settings seem to be promising. One may even consider open-source as a variant in the multifaceted agile development paradigm [6]. Not surprisingly, the flexible and cooperative nature of open-source projects accommodates fluctuations in the number of developers, affecting the composition of teams. In other words, projects may have changing teams with different developers over time [7]. Although the impact of turnover on productivity has been studied previously [2], [8], little is known about productivity, turnover, and team stability in the context of agile software development in open-source projects.

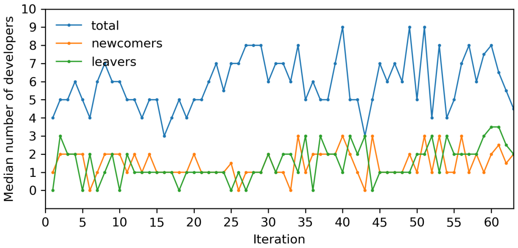

Fig. 1. Median number of developers of eight open-source projects.

Our study aims to close this gap by analyzing data collected from JIRA server instances of seven open-source projects. Our selection of projects is taken from a set of eight projects initially compiled into a dataset by Porru et al. [9], and used in further studies [10], since these projects are diverse and show evidence of the use of agile practices. Figure 1 shows the median number of developers (in blue) over time (iterations), as well as the number of newcomers (in yellow) and leavers (in green), across the projects. The fluctuations in the total number of developers are due to difference of newcomers and leavers in each iteration.

To analyze the productivity of these projects, we use software development measures as proxies. We choose measures that are popular in agile software development such as velocity and focus-factor [11], [12]. To make velocity comparable among teams, we calculated individual velocity. Then, we apply statistical process control techniques to identify the iterations that report unusual (out-of-control) values of velocity and focus factor. We studied these iterations regarding four factors that characterize part of the open-source dynamics: iteration length, turnover of developers who left (leavers), turnover of new developers (newcomers), and the team stability index. Our results show that iterations where velocity is below the lower control limit have higher turnover and less team stability than iterations where velocity is under control or above the upper control limit.

## II. Related Work

Research on productivity in software development has a long history. Wagner and Ruhe [1] point out that the topic has been addressed during several decades and classify several factors affecting productivity in software development. The authors show that, although some findings in the early studies are no longer relevant, several factors affecting productivity are still relevant, i.e., factors such as project size, the programming language, and developer experience.

Among the authors who studied factors affecting productivity, Sadowski and Zimmermann [13] report that it is worth investing in proper team communication despite the increase in team size. Moreover, activities and task switching [14], work environment [14]–[17], developers' capabilities and technical factors [3], and even the music listen by developers [4] have been studied. Rodriguez et al. [2] studied the relationship between team size and productivity based on data from the ISBSG repository. While all of the mentioned studies use data from different projects and domains, the the specific setting of agile development in open-source projects have not been taken into account. Petersen [5] recommends that productivity factors should be analyzed in varying contexts and, thus, productivity factors should be re-evaluated.

In the context of agile software development, Fatema and Sakib [18] conducted a survey in several companies to identify important productivity factors. Their study suggests that the effectiveness of an agile team relies on multiple interrelated factors such as excellent communication, leadership, adaptability, motivation, and self-management. Dingsøyr et al. [19] studied productivity in agile teams as compared to teams using non-agile development methods. They found in three out of four studies that using agile methods results in higher productivity of project teams. Downey and Shuterland [11], [20] introduced the idea of hyper-productive teams based on measures such as velocity, one of the most popular and relevant measures in agile software development [12].

In summary, much effort has been spent on identifying productivity factors in software development. However, little is known about how the dynamics of open-source projects using agile practices are related to the productivity of teams. Our study aims to close this gap.

## III. Study Design

### A. Research Objective and Research Questions

The research objective is to describe the productivity reached by agile teams in open-source projects and to identify the possible factors that explain unusual variations of productivity. We use two agile measures, velocity normalized by the number of developers and focus factor, as proxies for productivity [11]. In addition, we measure four factors that characterize the dynamics of open-source projects. These factors are our candidates for causes explaining unusual productivity variation. The four factors are iteration length, turnover of leavers, turnover of newcomers, and team stability index. We formulate two research questions:

*1) RQ1: What contextual factors are related to iterations whose velocity is out of control in open source projects?:* *Motivation:* Open-source projects have specific characteristics that make these kind of projects different from others. For example, the software built is commonly based on open collaboration, where new developers are continuously welcomed in, and, in consequence, the team size and composition becomes variable. The iteration length can also vary in open-source settings because the projects are subject to constraints different to those in other kinds of projects. For example, priorities and pressures of marketing are different in open-source projects [21]. In past studies, team size has been positively related to productivity [1], [2]. Turnover has also been related to productivity since teams with more stable structures tend to have better performance [21]. Optimal iteration length has been a topic of debate. Some authors argue that it should be variable [22] while others suggest having fixed-length iterations (time-boxing) since teams benefit from stability [23]. Having a better understanding of how context factors are related to productivity in agile software development of open-source projects can have relevant practical implications. For example, if adding new collaborators decreases productivity, then a recommendation could be that the number of new collaborators in a project should be controlled.

*Procedure:* We first calculate the velocity of each project per iteration. We normalize velocity by the number of developers, giving us values in terms of velocity per developer (individual velocity). Based on these values, we apply statistical process control to determine which iterations have unusual variation. Then, we classify the iterations of the projects into controlled and out-of-control (exceptionally high and exceptionally low). Finally, we analyze the context factors (iteration length, turnover of leavers, turnover of newcomers, and team stability index) across the groups to determine potential associations.

*2) RQ2: What contextual factors are related to iterations whose focus factor is out of control in open-source projects?:* *Motivation:* In RQ1, we investigate the relationship between velocity and the context factors. However, velocity does not reflect all the work that has been done during the iteration since it considers only completed or approved work items [11]. Therefore, we complement velocity with focus factor to get a more comprehensive understanding of productivity.

*Procedure:* We first calculate the focus factor of each project on an iteration basis. Based on the values of focus factor, we classify the productivity of the iteration as out of control or not. To do this, we apply statistical process control techniques. Then, we analyze the relationship between the iterations that had unusual variation of focus factor and the contextual factors using descriptive statistics. Finally, we supported our observations by using statistical analysis, as we did in RQ1.

### B. Productivity Measures

*1) Velocity:* Velocity is the amount of work that is completed at the end of each iteration by a specific team, and it is usually measured by adding the sizes of the completed product backlog items in the iteration. Velocity is one of the most popular and relevant measures due to its versatility of usage [11], [12], [24]. For example, it can be used for planning, estimating work, measuring work done, and improving the delivery of customer value. In this study, velocity ($V_i$) is calculated as the sum of story points of all completed issues whose last status update was made within the iteration $i$. We consider an issue as completed if its status is set to ”Done”, ”Resolved”, or ”Closed”.

*2) Work Capacity:* Work capacity was introduced by Downey and Sutherland [11] to mitigate a drawback of using velocity. Velocity is a measure that does not benefit from partially completed work since it is concerned with the size of what is delivered rather than the value of what is delivered [24]. Therefore, work capacity ensures the measurement of all the work reported during the sprint, whether it was marked as completed or not [11]. We calculate work capacity ($W_i$) as the sum of story points of all the issues that have the last status update made within the iteration $i$ regardless if they were completed or not.

*3) Focus Factor:* Focus factor measures how focused the team is towards achieving the iteration goal [11], i.e., it represents the extent to which the planned work was actually done. Focus factor is calculated on an iteration basis. It is defined as the ratio between velocity and work capacity, i.e., the focus factor ($F_i$) of an iteration $i$ is defined by Eq. 1

$$F_i = \frac{V_i}{W_i} \tag{1}$$

### C. Contextual Factors

As explained in Section III-A, we chose a set of contextual factors that characterize part of the dynamics in open-source projects and that have been found to have an effect on productivity [1], [2], [13], [25], [26]. These factors are iteration length, turnover of leavers, turnover of newcomers, and team stability index. More precisely, we define:

*1) Iteration length:* The duration of an iteration in days, calculated as elapsed time between start and end dates of the iteration.

*2) Turnover rate (leavers):* This measure is adopted from [27]. It compares the number of developers who left the team since the previous iteration to the average number of developers. The turnover rate of iteration $i$ is calculated as described by Eq. 2. $D_i$ is the set of active developers who were assigned to at least one issue in iteration $i$. $D_{i-1}$ is the set of developers who were assigned to at least one issue in the previous iteration $i-1$. Note that we perform a subtraction operation between sets (not numbers). For example, $\{a, b\} - \{a\}$ equals $\{b\}$ while $\{a\} - \{a, b\}$ equals the empty set. $|D|$ represents the cardinality of set $D$. Using this notation, the numerator $|D_{i-1} - D_i|$ represents the number of developers who left and the denominator $\frac{|D_{i-1}|+|D_i|}{2}$ the average number of developers in iterations $i$ and $i-1$.

$$\text{Turnover}_{leavers} = \frac{|D_{i-1} - D_i|}{\frac{|D_{i-1}|+|D_i|}{2}} \tag{2}$$

*3) Turnover rate (newcomers):* The turnover rate of newcomers is the number of new developers who were active in iteration $i$ and inactive in iteration $i-1$ divided by the average number of developers during the iteration. The turnover rate of newcomers at $i$ is calculated as shown in Eq. 3. Using the aforementioned notation, the numerator $|D_i - D_{i-1}|$ represents the number of new developers (newcomers), and the expression $\frac{|D_{i-1}|+|D_i|}{2}$ is the average number of developers.

$$\text{Turnover}_{newcomers} = \frac{|D_i - D_{i-1}|}{\frac{|D_{i-1}|+|D_i|}{2}} \tag{3}$$

*4) Team stability index (TSI):* Impact of team stability on performance has been investigated by Akgün and Lynn [28] based on a survey study. We aim to measure and quantitatively analyze team stability. Inspired by the Requirements Stability Index [29], we measure the stability of a team with regards to its initial composition of developers. Thus, we divide the sum of initial developers, newcomers, and leavers by the initial number of developers. Following the notation proposed above, Eq. 4 shows the formula to calculate TSI$_i$, where $D_0$ is the set of developers of the first iteration ($i = 0$). In our analyses, we report the results of TSI$^{-1}$, the inverse function of TSI, since TSI$^{-1}$ gives us a value between 0 and 1 and, therefore, is easier to interpret.

$$\begin{aligned} \text{TSI}_i &= \frac{\text{initial + newcomers + leavers}}{\text{initial}} \\ &= \frac{|D_0| + |D_i - D_{i-1}| + |D_{i-1} - D_i|}{|D_0|} \end{aligned} \tag{4}$$

### D. Process Behavior Analysis

To analyze the variation of productivity from one iteration to another, we rely on *Statistical Process Control* (SPC) techniques [30], [31]. SPC proposes the use of control charts to determine the operational limits for acceptable variation in software processes. The use of these charts has shown promising results in the context of software productivity, in particular, to detect shifts in process performance [5].

We test the projects for stability by using *Individuals and Moving Range* (I-MR) charts that establish the limits of variation of process performance. By using I-MR charts, we identify the points (iterations) where the process is out of control, and we inspect these points in order to study possible assignable causes responsible for that process instability. The equations used to set up the I-MR chars are given by Eq. 5, where $CL$ indicates the center line, $UCL$ the upper control limit, and $LCL$ the lower control limit. $\bar{x}$ refers to the median of each observation $x_i$, MR$_i$ = $|x_i - x_{i-1}|$ are the moving ranges, $\overline{\text{MR}}$ is the median moving range, and $D_3$, $D_4$, and $d_2$ are the sample size-specific anti-biasing constants. In this case, the values for our sample size are ($D_3 = 0$; $D_4 = 3.267$; $d_2 = 1.128$). We use the median instead of the mean because the latter is more susceptible to the influence of outliers.

$$\begin{aligned} UCL_{\text{MR}} &= D_4\overline{\text{MR}} & UCL_{\bar{x}} &= \bar{x} + 3\frac{\overline{\text{MR}}}{d_2} \\ CL_{\text{MR}} &= \overline{\text{MR}} & CL_{\bar{x}} &= \bar{x} \\ LCL_{\text{MR}} &= D_3\overline{\text{MR}} & LCL_{\bar{x}} &= \bar{x} - 3\frac{\overline{\text{MR}}}{d_2} \end{aligned} \tag{5}$$

Values falling outside the control limits suggest that assignable causes exist [30]. To inspect the potential causes, we adopted the zone classification proposed by Florac et al. [30] and classified the iterations according to its productivity value (i.e., individual velocity or focus factor) as defined by Eq. 6.

$$zone(value) = \begin{cases} \text{D+} & \text{if } value > UCL \\ \text{D-} & \text{if } value < LCL \\ \text{N} & \text{otherwise} \end{cases} \tag{6}$$

Once the iterations were labeled in zones, we analyzed the contextual factors of the zones using descriptive statistics. We did the analysis in two steps; first, we analyzed the projects separately, and then all the projects together. At aggregated level, we supported our observations by conducting non-parametric statistical analysis based on Kruskal-Wallis H test, followed by post-hoc tests to determine what groups (zones) differ in their medians. The results are presented in Section IV, where we reported the main findings along with their p-values.

### E. Data Collection

We collected data from eight different open-source projects using the popular JIRA issue tracker to manage their software development. The dataset consist of issues, their change log, and iteration dates extracted directly from the JIRA server instances of the projects. The dataset includes the following projects: Appcelerator Studio (APSTUD), DNN Platform (DNN), Apache Mesos (MESOS), MuleSoft (MULE), NEXUS, Titanium SDK/CLI (TIMOB), Accelerator studio (TISTUD), and Spring XD (XD). All the projects are open-source, their issue trackers are publicly accessible, and they have been used in several related studies [9], [10]. The projects also evidence the application of different agile practices such as the use of *Epics* and *User Stories* to manage requirements, the use of *Story Points* to represent the size (or imposed workload) of issues, and the use of specific fields to store information about the iterations. Full details on these projects are given in Appendix A[1], Table II.

### F. Data Cleaning

We applied several steps to pre-process and clean the dataset. First, we fixed the missing starting dates of some iterations. All projects have a field in which the information about the iteration is stored. This information includes the start and end dates. However, three projects (APSTUD, TIMOB, TISTUD) have recorded only the end date of the iterations. In these cases, we consider the start date of an iteration as the day after the end date of the previous iteration.

After applying this fix, we removed all issues that had incomplete information, such as issues having empty iteration dates or story points. We also removed all the issues that were assigned 0 story points, as we consider it as incorrect information. We also removed all issues whose issue types were not related to software development activities such as "Documentation", "Wish", "New Feature", "Patch Submission", and "Technical Debt".

After conducting the aforementioned steps, we calculated the productivity measures and applied an additional cleaning step in which we removed iterations that had null velocity or work capacity. Keeping these iterations of apparent inactivity would have introduced unnecessary noise into our subsequent analyses.

Since the SPC technique that we applied relies on the normality assumptions, we checked whether the values of productivity (i.e., individual velocity and focus factor) are normally distributed. Appendix B[1] gives more details about this process. As a result, project DNN was removed from the dataset since it does not fulfill the required prerequisite.

## IV. Results

### A. Study Population

After cleaning, our resulting dataset consists of 7 projects with 6887 issue reports and 387 iterations in total. The total number of issues per project varies between 168 and 2102, and the total number of iterations per project ranges from 22 to 93. In total, there are 198 unique developers in the dataset, ranging from 9 to 65 unique developers per project. The median duration of the projects is 831 days. Table I summarizes these values.

TABLE I
Descriptive statistics of the cleaned dataset

| Project | Devs | Issues | Iterations | Start | End | Duration |
|---|---|---|---|---|---|---|
| APSTUD | 9 | 355 | 24 | 27.01.2012 | 17.01.2014 | 721 |
| MESOS | 65 | 1091 | 63 | 15.05.2014 | 11.05.2016 | 726 |
| MULE | 32 | 831 | 93 | 20.02.2013 | 12.05.2016 | 1176 |
| NEXUS | 16 | 612 | 63 | 12.09.2013 | 10.05.2016 | 970 |
| TIMOB | 30 | 168 | 22 | 03.12.2011 | 14.03.2014 | 831 |
| TISTUD | 15 | 1728 | 57 | 31.01.2012 | 24.04.2014 | 814 |
| XD | 31 | 2102 | 65 | 06.05.2013 | 26.02.2016 | 1025 |
| Total | 198 | 6887 | 387 | – | – | 6263 |
| Median | 30 | 831 | 63 | – | – | 831 |
| Std | 18.58 | 711.59 | 24.92 | – | – | 168.81 |

Appendix A[1], Table III, shows the full details of the descriptive statistics of the the contextual factors used, i.e., iteration length (days), turnover of leavers, turnover of newcomers, and team stability index. One sees that the projects have iteration lengths ranging from 3 to 56 days, with standard deviations ranging from 1.50 to 11.70 days indicating that all projects have changing iteration lengths over time. The largest variation is shown by TIMOB and the smallest by MESOS.

Table III also shows the descriptive statistics of the calculated measures related to productivity, namely, velocity (story points), work capacity (story points), focus factor, and individual velocity (story points per person). Two projects (NEXUS and TIMOB) show a low average velocity (between 8 and 18 story points) in comparison with the others, which show average velocities around 100 story points. TISTUD has the highest median velocity (130 story points) and MESOS has several outliers that reach a maximum value of 800 story

[1] The appendix can be found on our repository https://bit.ly/2Vum1ts

points in one iteration. The projects show similar patterns in the work capacity values since work capacity and velocity are related measures. The projects are diverse in terms of their team stability as shown by the values of TSI$^{-1}$. Overall, TISTUD is the most stable project, followed by APSTUD, whereas TIMOB, followed by MULE, are the less stable ones.

### *B. Analysis Results*

In this section, we present our results per research question.

*1) RQ1: Individual Velocity:* To answer RQ1, turnover of leavers, turnover of newcomers, iteration length, and TSI were analyzed regarding the individual velocity. The analysis is done for each project separately.

We set up I-charts for the individual velocity of each project. Figure 2 depicts the I-charts, where the x-axis indicates the iteration number and the y-axis the individual velocity. Five out of seven projects (MESOS, MULE, NEXUS, TIMOB, and XD) violate the normality assumption that is required to apply SPC, so we applied a logarithmic transformation to the individual velocity which led to a normal distribution in these cases. In Figure 2, the dashed red lines indicate the control limits (CL, UCL, and LCL), and the red dots highlight the iterations that were above/below of the control limits.

We divided the iterations into groups according to their corresponding zones (D+, D-, and N) and compared the values of the contextual factors of the groups. Figure 3 shows the boxplots for each factor grouped by zone. The results related to individual velocity are on the left hand sides of each boxplot. We observed the following patterns:

**Turnover rate (leavers).** The iterations with low individual velocity (zone D-) show a higher turnover rate of leavers than the iterations under control (zone N) and iterations with high individual velocity (zone D+). This pattern occurs in 3 out of 5 projects (APSTUD, MESOS, and XD) that had zone D- iterations. Only one project (TIMOB) shows no differences between the rate of zones N and D-. When taking all seven projects together, the iterations in D+ zones show significantly lower turnover rates than iterations in D- ($p = .002$) and N zones ($p = .007$).

**Turnover rate (newcomers).** In five out of seven projects (APSTUD, MESOS, MULE, TIMOB, and XD), the iterations with low individual velocity (zone D-) show a higher turnover rate of newcomers than iterations in N and D+ zones. This observation is corroborated when the analysis is conducted on all seven projects together, showing statistically significant differences between zone D- and zones D+ ($p = .042$). The two projects that do not exhibit this pattern (NEXUS and TISTUD) did not have any iterations in zone D-. Regarding the comparison between the iterations in zones D+ and N, there is no clearly observable pattern since the turnover seems to be higher in N than D+ in three projects (MESOS, MULE, and NEXUS), almost identical in one project (XD), and lower in another project (APSTUD). This is consistent with the results from the statistical analysis, which indicates that there are no statistically significant differences between the turnover rates of newcomers in the D+ and N zones ($p = .36$) when the data of the seven projects is analyzed at aggregated level.

**Iteration length.** The iterations in zone D+ of three projects (APSTUD, MULE, and XD) are larger than iterations in zone D-. The remaining projects have low variation of iteration length making it difficult to see differences between the zones. However, when the projects are analyzed at aggregated level, the statistical results show that iterations with high individual velocity (zone D+) are significantly longer ($p = .008$) than iterations with low individual velocity (zone D-).

**Team stability index (TSI)** In five out of seven projects (APSTUD, MESOS, MULE, NEXUS, and XD), the iterations with high individual velocity (zone D+) show a higher TSI$^{-1}$ than iterations in N. These differences are stronger in APSTUD and MULE. When analyzing all the projects together, the iterations in zone D+ show significantly higher values of TSI$^{-1}$ than the iterations under control (zone N) ($p = .017$) and low velocity (zone D-) ($p = .013$). In addition, iterations in zone D+ mostly remain in values close to 0.8, which indicate stability in the team. Only XD shows a different behaviour since it shows almost similar values of TSI in both D+ and D- zones (TSI$^{-1} \approx .6$).

*2) RQ2: Focus factor:* To answer RQ2, the chosen set of contextual variables were analyzed regarding the focus factor. As in RQ1, the analysis is based on SPC and is done for each project separately.

Figure 4 depicts the I-charts for the variable focus factor, where the x-axis indicates the iteration number and the y-axis the focus factor. The dashed red lines indicate the control limits, and the red dots highlight the iterations outside of the control limits. We tested the distribution of the focus factor of each project, and the results indicate that the focus factor is normally distributed; in consequence, the projects satisfy the normality assumption that is required to apply SPC.

We divided the iterations into groups according to the observed values in the I-chart (zones D+, D-, and N). Then, we compare the zones regarding the values of contextual factors. Figure 3 shows the boxplots for each factor grouped by zone. The results related to focus factor are on the right hand sides of each boxplot. We observed the following patterns:

**Turnover (leavers).** In two projects (NEXUS and TIMOB), there are slight differences in the turnover rate of leavers between zones D+ and N iterations. However, the rest of the projects show no evident differences. This observation is supported by the results of the Kruskal-Wallis H test ($H = 1.27, p = 0.52$), which indicates that there are no significant differences in the medians of the turnover among the zones.

**Turnover (newcomers).** Six out of seven projects (APSTUD, MESOS, MULE, NEXUS, TISTUD, and XD) have iterations in zone N with a median turnover rate of newcomers between 0 and 1. However, there are no significant differences in the medians of the turnover among the zones as it is supported by the results of the Kruskal-Wallis H test ($H = 4.27, p = 0.19$).

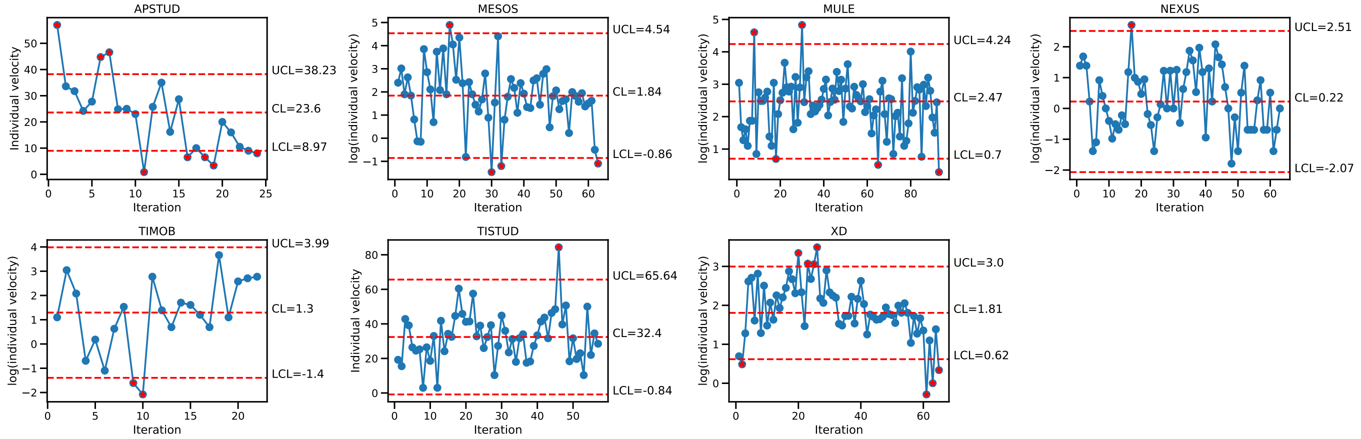

Fig. 2. Project-specific charts for individual velocity (story points per person). Dashed lines indicate the upper control limit (UCL), the control line (CL), and the lower control limit (LCL). Red dots indicate the iterations that were out of control (zones D+ or D-).

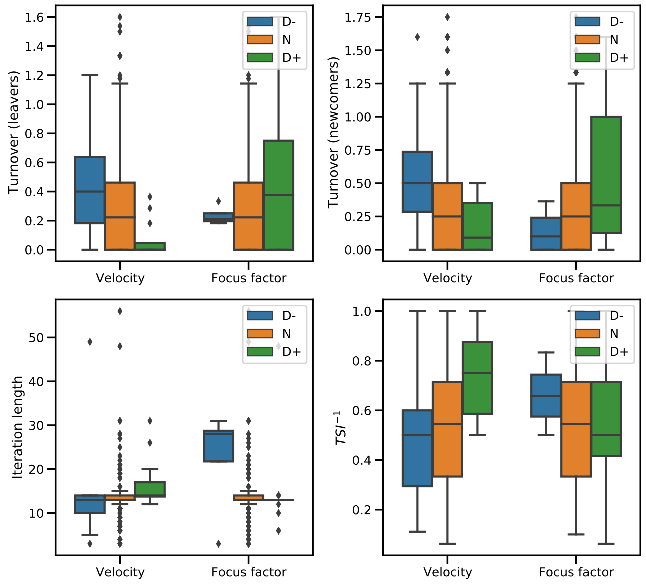

Fig. 3. Boxplots of the contextual factor values grouped by productivity measures and zones (N, D+, D-).

**Iteration length.** The low variation in iteration length of most of the projects makes it hard to identify differences. However, APSTUD and TISTUD show that the iterations in zone D- are longer than the ones in N and D+. The results from the statistical analysis at aggregated level show that, according to the results of the Kruskal-Wallis H test $(H = 4.57, p = 0.10)$, the lengths of iterations in zone D- (low focus factor) are significantly longer than in zone D+ (high focus factor).

**Team stability index (TSI)** There is no clear observable pattern when TSI$^{-1}$ is analyzed for each project separately. NEXUS and TIMOB seem to have lower values in zones D+ than in zone N while MESOS seems to have similar median values in those zones. When D- and N zones are compared, APSTUD shows larger values in zone D+ than in zone N while TISTUD shows the opposite. This indicates that there are no significant differences in the medians of TSI$^{-1}$ between zones. This is supported by the results of the Kruskal-Wallis H test $(H = 1.38, p = 0.50)$.

## V. Discussion

### A. *Interpretation of results*

In our study, we collected data of two productivity measures, i.e., (individual) velocity and focus factor, and analyzed how outliers, i.e., data points in zone D- (under-performers) and in zone D+ (over-performers), are associated with four context factors, i.e., turnover (leavers), turnover (newcomers), iteration length, and team stability (TSI$^{-1}$).

Our results indicate that iterations where velocity was below the lower threshold in the control chart (zone D-) are associated with a higher median turnover (leavers) than those between control limits (zone N) and those in zone N are associated with a higher turnover (leavers) than those above the upper control limit (zone D+). The same pattern can be observed for the association with context factor turnover (newcomers). In addition, the association with team stability points in exactly the same direction for each of the three zones, i.e., the higher (TSI$^{-1}$), the higher the velocity. Since team stability can be considered an aggregate measure of the two turnover measures, the consistency in the association patterns are not surprising. We interpret the observed patterns as a clear recommendation to control fluctuation in development teams of open source projects, if high productivity is a goal.

When looking at the association patterns between the second productivity measure, focus factor, our results suggest that an above the upper control limit (zone D+) focus factor is associated with high turnover (of both leavers and newcomers) and low team stability while for iterations classified in zone D- show the opposite association, and iterations between the control limits (zone N) take the middle position. This is somewhat surprising. A high focus factor indicates that many

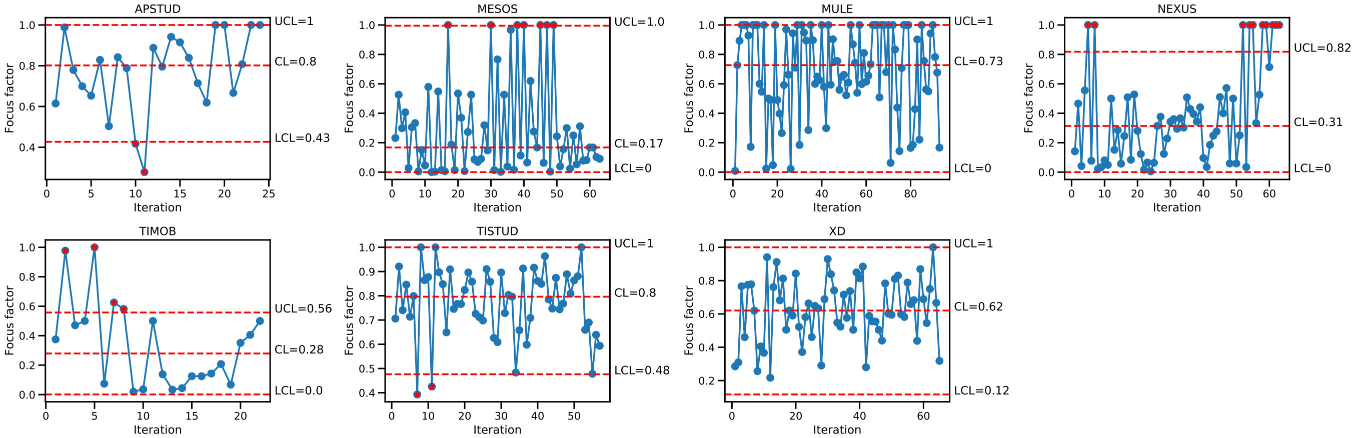

Fig. 4. Project-specific charts for focus factor. Dashed lines indicate the upper control limit (UCL), the control line (CL), and the lower control limit (LCL). Red dots indicate the iterations that were out of control (zones D+ or D-).

of the work items assigned to an iteration (work capacity) are actually completed (velocity) when the iteration ends. It is difficult understand why high team stability (and the related context factors) is associated with high velocity but low focus factor. Perhaps there exists some intrinsic motivation for stable teams to work on as many work items as possible (trusting that they manage to complete them) while less stable teams are more focused on finishing up work instead of starting to work on the next work item. However, speculation might not be important when recalling that– different to the results for velocity –our analysis did most of the time not indicate statistically significant variation of the discussed context factors with regards to the grouping of iterations based on the measure focus factor.

The third context factor, iteration length, does not show strong associations with any of the zones D+, N, and D- of productivity measure velocity, and zones N and D+ of productivity measure focus factor. Only iterations in zone D- of measure focus factor seem to be associated with longer iterations. Again, this is slightly counter-intuitive as one expects that more available time gives more opportunity to finish up work items that were planned for an iteration. It is possible, though, that our way of calculation work capacity retrospectively is not a fully correct conceptualization of the situation in open source projects where iterations are not necessarily planned as strictly as in closed source project within companies and, instead of working on planned set of work items developers set out to work any open issue at any time and when the observed time span is longer there is a bigger chance to work on more items (without finishing them up) as when the time span is shorter and more discipline enforced.

### B. Limitations

There are several limitations to be considered when interpreting the reported results. First, the study design follows a data-driven approach that does not support causal inference. Our results are mainly based on statistical analysis that, although it allowed us to identify statistically significant relationships, these results are not proof of causality; controlled experiments are required to prove causality, and this study can set the basis for future controlled experiments.

The measures used in our study could introduce several threats to construct validity. First of all, there is no general agreement on how to measure productivity. We tried to mitigate this threat by choosing measures that are well-known and often used in agile software development projects, such as velocity which is understood as a proxy of productivity [11], [12]. In addition, the removal of iterations with velocity or work capacity equal to zero could impact on the results since such iterations could be interpreted as low-performance iterations. However, we interpret iterations with no work done or planned as cases that are hard to explain and would yield misleading results if included. Finally, the formulas we used for calculating turnover do not always accurately reflect the team dynamics within iterations, especially for long iterations, as changes might happen within an iteration. However, we think that, overall, team changes within an iteration are rare and thus ignoring them does not bias our results.

Another threat is related to the degree of agility applied by the teams. We mitigated this threat by manually inspecting the data and studying the information available from the official websites of the projects. In particular, we manually analyzed the issue description and summary fields to identify properly formulated user stories (i.e., issues following the usual syntax structure ”*As a [role], I want [feature] so that [benefit]*” [23]), the presence of epics, iterations, and story points. We took the presence of these data items in the dataset as indicators agility. Also, the team sizes are typical for agile projects.

The method used to identify iterations that were out of control introduces another limitation. SPC is based on the normality assumption, and the violation of this assumption could threaten the validity of the results. We mitigated this threat by applying data transformations. Our results show that we cannot detect outliers easily if the distributions are not normal. We applied a simple log transformation that allowed us to determine outliers in lognormal distributions. It is worth mentioning that if different underlying distributions are tested, different transformation methods should be applied [32].

We addressed the reliability of this study by providing an online and public repository[2] where the dataset and scripts are available. The scripts were made in interactive Jupyter notebooks, where the results can be analyzed. This way, other researchers can reproduce and replicate the current study. Given the relatively small number of projects analyzed, we cannot claim generalizability of our results to all open source projects using the agile development paradigm.

## VI. Conclusion

Productivity in software development is a topic that has been extensively discussed, and it still worth to be studied. Analyzing productivity is a challenging task since it depends on a variety of factors as well as the specific development context. In fact, factors that have been associated with productivity are recommended to be evaluated in new contexts [5]. This study analyzed the productivity of seven agile software development teams in open-source projects in order to determine factors of importance. We used velocity and focus factor as proxies to measure productivity in agile teams. In addition, we analyzed productivity with regard to factors that are related to open-source dynamics, namely turnover of leavers, turnover of newcomers, team stability index, and iteration length. The analysis was conducted on 6887 issues taken from seven different JIRA projects. Our results suggest that high team stability and low turnover are associated with iterations showing high velocity.

## Acknowledgment

This work was supported by the Estonian Center of Excellence in ICT research (EXCITE), ERF project TK148 IT, and by the team grant PRG 887 of the Estonian Research Council.

## References

[1] S. Wagner and M. Ruhe, "A systematic review of productivity factors in software development," *Proc. 2nd Int. Workshop on Software Productivity Analysis and Cost Estimation. Technical Report ISCAS-SKLCS-08-08, Chinese Academy of Sciences*, 2008.

[2] D. Rodríguez, M. Sicilia, E. García, and R. Harrison, "Empirical findings on team size and productivity in software development," *J. of Syst. and Soft.*, vol. 85, no. 3, pp. 562–570, 2012.

[3] A. Trendowicz and J. Münch, "Factors influencing software development productivity—state-of-the-art and industrial experiences," *Advances in Computers*, vol. 77, pp. 185–241, 2009.

[4] L. E. Barton, G. Candan, T. Fritz, T. Zimmermann, and G. C. Murphy, "The sound of software development: Music listening among software engineers," *IEEE Software*, 2019.

[5] K. Petersen, "Measuring and predicting software productivity: A systematic map and review," *Inf. & Soft. Tech.*, vol. 53, no. 4, pp. 317–343, 2011.

[6] J. Warsta and P. Abrahamsson, "Is open source software development essentially an agile method," in *Proc. of the 3rd Workshop on Open Source Soft. Eng.*, pp. 143–147, 2003.

[7] G. Robles and J. M. Gonzalez-Barahona, "Contributor turnover in libre software projects," in *IFIP Int. Conf. on Open Source Syst.*, pp. 273–286, Springer, 2006.

[8] M. Foucault, M. Palyart, X. Blanc, G. C. Murphy, and J.-R. Falleri, "Impact of developer turnover on quality in open-source software," in *Proc. of the 2015 10th Joint Meeting on Foundations of Soft. Eng.*, pp. 829–841, 2015.

[9] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, "Estimating story points from issue reports," in *Proc. of the 12th Int. Conf. on Predictive Models and Data Analytics in Soft. Eng.*, pp. 1–10, 2016.

[10] E. Scott and D. Pfahl, "Using developers' features to estimate story points," in *Proc. of the 2018 Int. Conf. on Soft. and Syst. Process*, pp. 106–110, 2018.

[11] S. Downey and J. Sutherland, "Scrum metrics for hyperproductive teams: how they fly like fighter aircraft," in *2013 46th Hawaii Int. Conf. on Syst. Sciences*, pp. 4870–4878, IEEE, 2013.

[12] E. Kupiainen, M. V. Mäntylä, and J. Itkonen, "Using metrics in agile and lean software development – a systematic literature review of industrial studies," *Inf. & Soft. Tech.*, vol. 62, pp. 143–163, 2015.

[13] C. Sadowski and T. Zimmermann, *Rethinking Productivity in Software Engineering*. Springer, 2019.

[14] A. N. Meyer, L. E. Barton, G. C. Murphy, T. Zimmermann, and T. Fritz, "The work life of developers: Activities, switches and perceived productivity," *IEEE Trans. Soft. Eng.*, vol. 43, no. 12, pp. 1178–1193, 2017.

[15] A. N. Meyer, G. C. Murphy, T. Zimmermann, and T. Fritz, "Design recommendations for self-monitoring in the workplace: Studies in software development," *Proc. of the ACM on Human-Comp. Interaction*, vol. 1, pp. 1–24, 2017.

[16] B. W. Boehm, M. H. Penedo, E. D. Stuckle, R. D. Williams, and A. B. Pyster, "A software development environment for improving productivity," *Computer*, no. 6, pp. 30–44, 1984.

[17] B. Johnson, T. Zimmermann, and C. Bird, "The effect of work environments on productivity and satisfaction of software engineers," *IEEE Trans. Soft. Eng.*, 2019.

[18] I. Fatema and K. Sakib, "Factors influencing productivity of agile software development teamwork: A qualitative system dynamics approach," in *2017 24th Asia-Pacific Software Eng. Conf.*, pp. 737–742, IEEE, 2017.

[19] T. Dingsøyr, T. E. Fægri, T. Dybå, B. Haugset, and Y. Lindsjørn, "Team performance in software development: research results versus agile principles," *IEEE Soft.*, vol. 33, no. 4, pp. 106–110, 2016.

[20] J. Sutherland, S. Downey, and B. Granvik, "Shock therapy: A bootstrap for hyper-productive scrum," in *Agile Conf.*, pp. 69–73, IEEE, 2009.

[21] E. D. Canedo and G. A. Santos, "Factors affecting software development productivity: An empirical study," in *Proceedings of the XXXIII Brazilian Symp. on Soft. Eng.*, pp. 307–316, 2019.

[22] D. Leffingwell, "Mastering the iteration: An agile white paper," *Rally Software Development Corporation*, 2007.

[23] M. Cohn, *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[24] K. S. Rubin, *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley, 2012.

[25] C. Melo, D. S. Cruzes, F. Kon, and R. Conradi, "Agile team perceptions of productivity factors," in *Agile Conf.*, pp. 57–66, IEEE, 2011.

[26] B. Boehm and R. Turner, "People factors in software management: lessons from comparing agile and plan-driven methods," *Crosstalk-The J. of Defense Soft. Eng.*, 2003.

[27] M. Sanja and L. Eftimov, "Calculating the cost for employee turnover in the it industry in macedonia by using a web calculator," *J. of Human Resource Management*, 2016.

[28] A. E. Akgün and G. S. Lynn, "Antecedents and consequences of team stability on new product development performance," *J. of Eng. and Tech. Management*, vol. 19, no. 3-4, pp. 263–286, 2002.

[29] S. Anderson and M. Felici, "Quantitative aspects of requirements evolution," in *Proc. 26th Annual Int. Computer Soft. and Applications*, pp. 27–32, IEEE, 2002.

[30] W. A. Florac and A. D. Carleton, *Measuring the software process: statistical process control for software process improvement*. Addison-Wesley Professional, 1999.

[31] D. J. Wheeler, D. S. Chambers, *et al.*, *Understanding statistical process control*. SPC Press, 1992.

[32] T. R. Willemain and G. C. Runger, "Designing control charts using an empirical reference distribution," *J. of Quality Tech.*, vol. 28, no. 1, pp. 31–38, 1996.

[2] https://github.com/ezequielscott/SEAA2020-productivity