Relation of Individual Time Management Practices and Time Management of Teams

Tapio Auvinen*, Nickolas Falkner†, Arto Hellas‡, Petri Ihantola§, Ville Karavirta¶ and Otto Seppälä‡
*tapio.auvinen@vxt-research.com, VXT Research, Finland
†nickolas.falkner@adelaide.edu.au, The University of Adelaide, Australia
‡{arto.hellas, otto.seppala}@aalto.fi, Aalto University, Finland
§petri.ihantola@helsinki.fi, University of Helsinki, Finland
¶ville@villekaravirta.com

***Abstract*—*Full research paper*—Team configuration, work practices, and communication have a considerable impact on the outcomes of student software projects. This study observes 150 college students who first individually solve exercises and then carry out a class project in teams of three. All projects had the same requirements. We analyzed how students' behavior on individual pre-project exercises predict team project outcomes, investigated how students' time management practices affected other team members, and analyzed how students divided their work among peers. Our results indicate that teams consisting of only low-performing students were the most dysfunctional in terms of workload balance, whereas teams with both low- and high-performing students performed almost as well as teams consisting of only high-performing students. This suggests that teams should combine students of varying skill levels rather than allowing teams with only low performers or letting students to form teams without constraints. We also observed that individual students' poor time management practices impair their teammates' time management. This underlines the importance of encouraging good time management practices. Most teams reported that they divided tasks in a way that is beneficial for the acquisition of technical skills rather than collaboration and communication skills. Only a few teams assigned tasks so that students would have worked only on tasks they already knew and thus felt most comfortable to work with.**

***Index Terms*—student project, time management, team configuration**

I. Introduction

Studies on the skills that recent graduates lack when entering the industry continue to highlight the importance of teamwork and communication [1]–[4]. While the gap between industry expectations and university education is often explained by universities emphasizing knowledge and skills that stay throughout the careers and act as a foundation to learning new knowledge [5], many universities seek to bridge the gap by giving more attention to collaboration skills (see e.g. [6]–[8]). This need for skills beyond technical expertise has also been acknowledged on a higher level, which has resulted in the explicit inclusion of professional skills such as teamwork and communication to computing curricula [9].

In existing curricula, professional skills are often practiced in projects that students perform together. These projects are typically either part of a course, or a separate project course such as a capstone project, where students get to apply their knowledge and skills in a real-world setting [10], [11]. As students' experiences from teamwork affect their attitudes toward future projects [12], it is important that the experiences from the project courses are constructive. When instructors design and create these opportunities, care needs to be taken for multiple reasons. To increase students' motivation, the projects should be realistic and meaningful [13], the applied teamwork practices need to be focused to avoid learning of bad habits [2], and the assessment of the projects needs to be designed in a fashion that directs the students' actions towards the desired goals [14]. In addition, instructors should pay attention to ensure that the teams are well formed, and that the students have sufficient knowledge of the tools and practices needed in the project [6].

While numerous streams of research on student projects exist, e.g. assessing teamwork [15], [16], factors that influence project outcomes [17], [18], and practices instructors can use to improve students' collaboration skills [7], [8], [19], little focus has been given to students' behavior in individual tasks, i.e. course work, and how that behavior is visible when a student becomes a part of a team. As research has suggested that student teams benefit from being balanced skill-wise [6], and that certain personality traits and their mix influence project outcomes [20], [21], insight on students' behavior and how it affects others can provide guidelines and hints for instructors constructing student teams. Studies have also established the different roles that students can occupy within teams, where these roles are easier to observe and quantify than subjective measures of personality [22]. Instructors need concrete guidance to assist them in supporting and developing teams, leading to a better outcome and experience for students.

While there exists a number of studies that suggest taking personality and other factors into account when forming teams, the analyses may require cumbersome arrangements such as surveys to collect data about student characteristics. Ironically, self-reported measures of personality traits may also be affected by a range of self-reporting biases, including those caused by the same personality traits we wish to identify [23]. Thus, the present study explores potential of building teams based on only transparently collected data from tools that students use as a natural part of their coursework. Moreover, we will look at the ways how student teams behave in terms of how they distribute work among team-members.

II. Related Work

A. Students in Software Projects

Challenges that students face are not only related to the challenges that software engineering teams in the industry face, but include challenges that are related to the life at the University or College. It is often the case that students are working on multiple courses, many of the courses typically strenuous and attention demanding. Thus, allocating sufficient time to the project is at times an issue.

Some students choose to reduce their own workload and consequently increase the time that they have at hand by free riding in group projects [24], [25], i.e. by contributing little or not at all to the collaborative work, but still acting outwards as if being a contributing member in the project. This behavior can lead to poor overall project performance, and can also result in poor learning experiences even for the hardworking students, as they may reduce their own effort to avoid being taken advantage of [19], [26]. Other students cope with the time-pressure by scheduling their work in various ways; research suggests that students' time-management skills are related to academic performance [27]–[29].

A number of strategies has been suggested to avoid free riding, including the use of multiple evaluations that start early in the course with a specific criteria [30], continuous evaluation and time logging [31], and pair programming [24]. Activities such as time logging can also be beneficial for time-management skills, as it raises the students' awareness of their own time usage. Other approaches include the active rotation of team roles among the group, to maintain novelty but to also avoid the ability of students to hide in less active roles. The roles identified in [22] have a range of categorisations but a simple division is between those that initiate activity and those that support - none are truly passive, but the clear identification of what a given role does can give valuable cues to students who have formed a belief that teams are made up of people doing things and people doing nothing. Rotation of these roles has been shown to increase engagement and enhance student retention when used in small programming tasks in the classroom [32].

Students have also been observed to be reluctant to work in teams and prefer individual work [19]. Waite et al. suggest that the situation is partly caused by a culture where students are required to demonstrate proficiency in individual exercises and collaboration is discouraged. They argue that teamwork skills cannot be taught simply by introducing team projects but a wider cultural shift is necessary. Waite et al. propose using the *conversational classroom* strategy [33] to demonstrate the advantages of collaboration to students and to improve their collaboration skills, introducing group decision making exercises because indecisiveness has been identified as a bottleneck in student projects, and decreasing the weight of individual assignments in assessment to avoid teaching students to value individual work over teamwork.

Another challenge is the variance in technical skills. While this is also an existing challenge in the industry, one can argue that it is more challenging in student software engineering projects, as the skill levels of students are (understandably) typically lower than those of the graduates working in the industry. In addition, the timespan of projects is typically shorter than of the projects in the industry, even weeks instead of months or years, and thus, little time exists for learning course- or project-specific tools and technologies. Small projects also require much shorter, or possibly no, design phase, reducing the natural group formation that occurs during discussion and moving more activity to the relatively isolated activity of programming. While low technical competence can make it challenging for a student to contribute to a group project, numerous suggestions to improve the situation exist. For example, a teacher can provide well-defined small scale projects that students need to work on before taking on a larger open-ended project [34], providing first the basic necessary skills before students start to work on the project themselves.

Wiggberg [35] identifies four key features that are important for the success of student collaboration projects. First, there must exist a mechanism for allocating work among the students because students themselves are not good at choosing roles that maximise their learning outcomes. Second, a connection to an external stakeholder such as an industry partner creates pressure to finish the project successfully. Third, whether the assessment is focused on the end result or the process affects students' behavior. If the focus is on the result, students may choose strategies that optimize the quality of the resulting artifact and thus the grade over strategies the maximize learning. For example, students may assign each task to the member with most experience of the task even though they would learn most from tasks that they are least experienced with. Fourth, the level of freedom in the assignment has a substantial impact on its content and aim. Wiggberg suggests that assessment should be focused on the process instead of the end result.

B. Mining Student Software Projects and Individual exercises

Our objective of understanding students on a programming course transparently (in order to facilitate group formamation) is methodologically related to repository mining and identifying students' behavior on individual programming exercises [36]. Repository mining has been used to, e.g., identify indicators that could be used to predict students' grades [37] and for identifying usage patterns and understanding what types of teams inhibit these explicit patterns [38]. While Mierle et al. [37] did not find any significant correlation between version control system behavior and students' grades, Kay et al. [38] suggest that good teams are more responsive and react faster to issues found in their systems.

Behavior on individual programming exercises has been studied from multiple angles. For example, Jadud proposed identifying students at risk of dropping out from introductory programming by quantifying the compilation errors that students encounter in an IDE [39]. A more recent study by Edwards et al. investigated the study behaviors of novice programmers [40]. In their dataset, Edwards et al. found

that students who received both high and low marks from programming assignments typically received the high marks from work that they started and ended well before deadline, while the low marks came typically from assignments that the same students started relatively close to the deadline. That is, remaining time to deadline influences results from programming assignments.

Other studies have also found that starting to work early correlates with higher success. Fenwick et al. [41] analyzed students working with the BlueJ programming IDE. Their analysis showed that students who start working early achieve better results. Falkner and Falkner [42] analysed over 220,000 student submissions on multiple courses, and concluded that students who submit their first piece of work late have a higher risk to be late throughout their career. Submitting early also correlated positively with the students' GPA. Further studies where similar correlation has been found have been carried out by [43] and [44].

III. Research Questions and Data Collection

The previous research has highlighted the importance of temporal factors in understanding learners and their behavior. The purpose of the present study is to explore the degree to which team configurations and team members' working preferences influence teamwork, and how students divide their work in the web development projects. The research questions answered in this study include:

1) How do different configurations of high and low performing students affect project work?
2) How does students' time management behavior affect their team members?
3) How do teams divide their work in web development projects?

A. Context

The study was conducted in a Web Software Development course (CSE-C3210) at Aalto University university during Fall 2013-Spring 2014 (the course spans two semesters). This 5 ECTS[1] course is offered in both the bachelor's and master's level curricula. The course was targeted for students with good programming experience but no skills related to web development. The main goal of the course was to teach how to develop full stack web applications with frameworks such as Django for back-end and libraries such as JQuery in the front-end.

Theoretical background is provided during the fist half of the course, during which the students also work on individual, automatically assessed programming assignments. These assignments focus on introducing technologies such as HTML, CSS and JQuery, and generic principles used in web frameworks such as routing, ORM and MVC.

In the middle of the course, students take an exam. After that, during the second half of the course, students work on a team project where they create a web application in three person teams. In this study, the teams were self-selected. The topic of the project was a photo album, where a user can add and edit images, captions and layouts; order and pay albums through a third party payment service, and various optional features (e.g. third party authentication and Flickr[2] integration). All teams provided a project plan in the very beginning of the project. At the end of the project, the teams had to have a production ready system running on the Heroku[3] cloud platform. The project had no other intermediate deliverables or deadlines. For version control in the project, all groups were instructed to use a private GitHub repository accessible also to the course staff. Apart from Django, Github and Heroku, the students were free to select communication and development tools of their own liking. Both the individual assignments and the exam contribute to 20% to the final grade, while the remaining 60% comes from the project.

B. Participants

Data from a total of 150 students, forming a total of 50 groups, was collected for this study. 180 students registered to the course, 150 started the course project and gave their consent, from where 125 passed the project. The students who attended the course had diverse backgrounds ranging from seasoned programmers accustomed to teamwork to novices with only little software engineering and programming experience. From the respondents, 13% were freshmen, 30% 2nd or 3rd year students, 35% 4th or 5th year students and the remaining 20% were at least 6th year students. As illustrated in Table I, most students felt that they had programming experience before the project, although their web software development programming experience was more scattered. In addition, teamworking skills were equally scattered. The main reason for having such diverse backgrounds especially in subject related skills is that at the time of data collection, the course was new, and thus attracted students at multiple stages of their studies.

C. Measures

We used multiple sources of data for our analysis: 1) individual (pre-project) assignments submitted into an automatic assessment system which stored all solution attempts, submission times, and the resulting grades, 2) student interviews during project demonstrations, 3) course survey sent to all students after the course, and 4) project repositories in GitHub version control and collaboration platform, including commits, issues, and pull requests. The course survey sent out at the end of the course included questions such as 1) how did you divide work and why, 2) with whom did you work the most, and 3) had you worked with your teammates before this project.

IV. Analysis and Results

The results of this study are presented in three parts. First, in Section IV-A, we consider team configurations and their effect on project outcomes, then, in Section IV-B, we discuss

[1]European Credit Transfer System

[2]https://www.flickr.com/

[3]https://www.heroku.com/

TABLE I
STUDENTS' SELF-REPORTED PRIOR PROGRAMMING AND TEAMWORK EXPERIENCE. 1=STRONGLY DISAGREE ... 4=STRONGLY AGREE

	1	2	3	4
I had programming experience	-	1 (3%)	7 (18%)	30 (79%)
I had web software development experience	7 (18%)	11 (29%)	10 (26%)	10(26%)
I had experience from teamwork in software projects	3 (8%)	8 (21%)	9 (24%)	18 (47%)

how students' time management behavior affects their team members, and finally in Section IV-C, we discuss how students divided their work.

A. Team configurations

To answer Research Question 1, i.e. "How do different configurations of high and low performing students affect project work?" we analyzed how combinations of students with different performance levels affect the outcomes of the project by (1) dividing students into high and low-performing ones (based on their behavior in the individual assignments), and then comparing how groups with only high performing, only low performing, or mixed groups perform in terms of (2) final marks of the project and (3) how balanced the workload was between the team members. That is, if a group contains both high and low performing students, do the high performing ones carry out all of the work while the low-performing ones do nothing.

First, to categorize students as high or low performing, we analyzed their performance in the individual assignments. Due to the fact that students can resubmit their work in the automated assessment any number of times, the students' mean total points from the individual assignments are a poor indicator of performance and have no correlation with the final points of the project (Spearman's ρ=0.02, p=0.909). That is, students are able to reach high points in the assignments if they choose to work on them long enough. The number of attempts could be used as an indicator of how much difficulty a student is experiencing in the assignments, however, it is not statistically significantly correlated with the final score of the project either (Spearman's ρ=-0.22 p=0.145). On the other hand, the mean points from the very first attempt for each assignment has a statistically significant correlation with the final score of the project (Spearman's ρ=0.48, p$<$0.001). Thus, we use the average points over the first submissions to all assignments as an indicator of students' performance.

We assigned students into high- and low-performing categories based on the median of their average first attempt scores (66.2 % of the maximum points). Based on students' categories and the types of students within each team, the teams were further divided into three categories. Teams in the *low* category consist only of low-performing students, the *mixed* category has teams with both low and high performing students, and the *high* category has only high performing students.

The project work was investigated from three viewpoints. First, the quality of the end product of the project was estimated using the teams' final project score that was also used to calculate the project grades. Second, to measure how evenly the students divided their workload, we analyzed the differences between GitHub commit counts of the team members as well as the differences between the number of edited lines of code (loc). Workload *imbalance (commits)* is defined as the ratio between the highest and lowest number of commits between the students whereas *imbalance (loc)* is defined as the corresponding ratio of code lines. For example an imbalance of 3.0 would mean that the most active student made three times more commits than the least active one. Third, we measured how far the GitHub activities of the project, i.e. commits, opening of issues, and commenting on the issues, were from the project deadline on average. This measure is later referred to as *earliness*.

TABLE II
MEAN PROPERTIES OF TEAMS WITH ONLY LOW-PERFORMING STUDENTS, MIXED, AND ONLY HIGH-PERFORMING STUDENTS.

	low	mixed	high	F	p
project points	1161.67	1230.58	1303	2.460	0.098
imbalance (commits)	5.65	2.87	2.45	4.043	**0.025**
imbalance (loc)	47.04	13.16	16.68	3.019	0.060
earliness (days)	14.98	11.57	10.95	0.949	0.395
N (teams)	9	26	10		

Analysis of the relationship between the team configuration and the extracted properties are shown in Table II and Figure 1. Although the teams with high-performing students reach the highest points, no statistically significant difference is observed between the groups' final project points (one-way ANOVA, p=0.098). Similarly, no statistically significant difference exists between the groups' *earliness* values (measured in days from the deadline) (one-way ANOVA, p=0.365). However, the imbalance of commits differs significantly between the groups (one-way ANOVA, p=0.025) being the highest for the *low* group. A similar phenomenon, although not statistically significant (p=0.060) can also be observed in the *loc* based measure.

B. Effect of Students' Time Management Behavior on Other Team Members

To answer Research Question 2, i.e. "How does students' time management behavior affect their team members?", we analyzed how student's tendency to work early or late affects teammates' tendencies when moving from individual assignments to the project.

Figure 2 illustrates the correlation between the difference of a student's and his/her teammates' earliness in the individual assignments (x-axis, denoted as *earliness_diff*), vs. the change in the teammates' earliness from the individual assignments to the project (y-axis, denoted as *earliness_delta*). The vari-

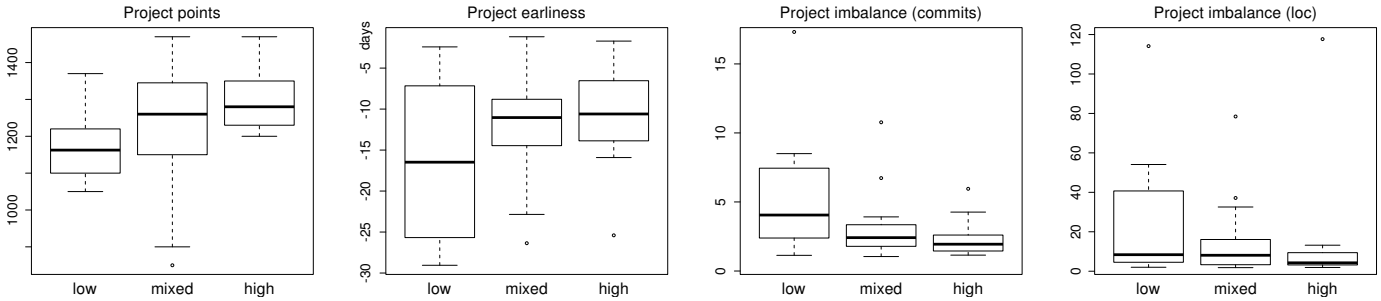

Fig. 1. Properties of teams with only low-performing students, mixed, and only high-performing students.

ables are normalized so that they have zero mean and unit variance, in order to account for the different time scales of the individual assignments and the project. A statistically significant correlation exists between the variables (Spearman's ρ=0.49, p<0.001). This can be interpreted so that if a student submits individual assignments later than the teammates, the teammates tend to start committing later when moving on to the project. Vice versa, an early-submitting student causes the teammates to start working earlier in the project.

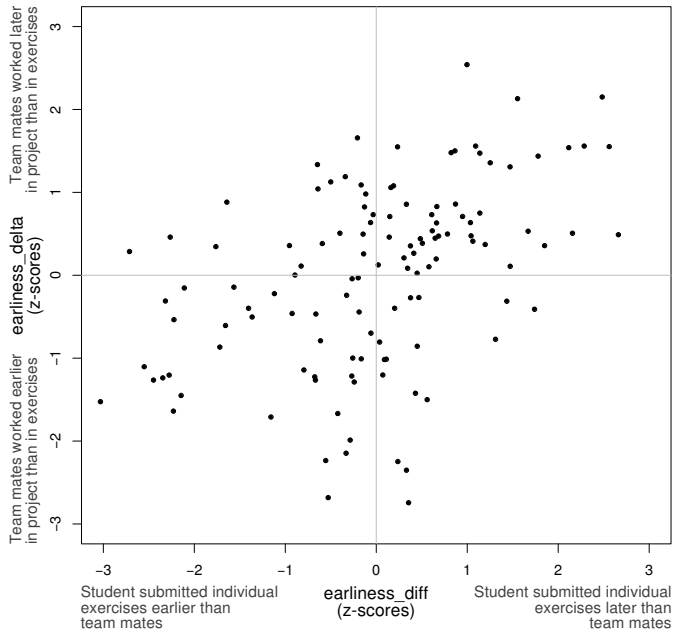

Fig. 2. The effect of student's time management practices on teammates.

To analyze whether the effect size is the same in both opposites, i.e. if a late submitter affects the teammates as much as an early submitter, we compare the opposite quartiles of the *earliness_diff* metric. Students in quartile q_1 are those whose relative earliness in the individual assignments are, on average, in the earliest 25%, and q_4 contains those in the latest 25% (earliness_diff thresholds; q_1 <=-0.63, q_4 >=0.67).

As we are studying whether an early submitter causes an equally large improvement as a late submitter causes a decline, we study the absolute values of the *earliness_delta*. In q_1, the mean change in teammate's earliness when moving from individual assignments to the project is 0.47, while in q_4, the mean change is 0.91 standard deviations. A Student's two tailed t-test confirms that the populations in the quartiles q_1 and q_4 are different (t=-2.1571, df=59.781, p=0.03), and thus, late-submitting students affect their team members more than the early submitters. This agrees with earlier research regarding the difficulties in motivating students to engage with group work: procrastinating members may create roadblocks as well as decrease the overall motivation.

C. Division of work

To answer Research Question 3, i.e. "How do students divide their work in web development projects?", we (1) analyzed students' answers to free form text questions in the survey, and (2) interviewed students during the project demonstrations. From the students who commented on the task division, a major proportion (86%) told that in most cases, students divided the work based on expected features, while the rest did not point out any particular strategy.

When working on a feature, a student would work on a vertical slice containing elements from both the backend and frontend. Some students explained this division of work from a learning perspective: *"We wanted that everyone would learn something related to all of the components used"*, *"It gives each one a complete view of the architecture"*, *"Felt natural and everybody got to work with back and front end"*, while other students had a project management standpoint: *"Organizing work into bitesize features helped to organize work between group members and enabled several features to be developed at the same time without major fear of code conflicts"*, *"Some tasks must be divided so that people don't do the same features unknowingly of each other. Some features were given completely to one person and some were shared to some extent, because it seemed to be convenient and made sense."* A similar, but slightly different reason was that working feature-wise allowed team members in some teams to work independently with little inter-team communication: *"There was not much communication between group members. We mostly worked alone, so dividing tasks like this worked nicely."* One team also noted that the difference in the difficulty of the features allowed assigning more complicated tasks to the more experienced members.

While the majority of the respondents pointed out that they divided work based on features, a few groups reported on dividing their work based on whether the required functionality resided in the backend or the frontend, capitalizing on the existing knowledge of the team members: *"We did what we knew the best."*

A few groups also pointed out that they did not have a clear strategy for task division. Based on the comments, it seems that these groups often worked face-to-face with everyone working together on one feature or that the more experienced teammates worked on every feature alongside their companions. *"We always worked all together, so we did some of the tasks together and then we just did what was needed to do next."*

V. Discussion

The discussion is structured around the research questions. We first discuss each research question, and then, discuss the limitations of our work.

A. RQ1: How do different configurations of high and low performing students affect project outcomes?

Our results indicate that teams that contain only poorly performing students tend to work in an imbalanced manner and perform poorly also as a team. This result replicates an earlier finding, where the equal participation of team members has been identified as one of the factors that influence project scores [6]. It is slightly surprising that mixed groups with both low and high performing students seem to be more balanced. It could have been expected that high performing students would have taken a central role in the groups leading to imbalance, while homogeneous groups would have a more balanced workload. However, we may be seeing students moving into different group roles that are valuable, even if performed at a lower level, as identified in [22]. This result implies that it would be beneficial to make sure that there are no groups with only low performing students. One option would be to assign students to groups automatically based on their measured performance in individual assignments. On the other hand, self-selected groups may be more motivating for students [10].

Another surprising finding is that teams consisting of only high performing students tend to start their work later than mixed or low performing groups. This contradicts earlier findings where high performance has been linked with starting to work early [40], [41]. A possible explanation is that the high-performing students had prior web development experience, were able to estimate the required workload, and were confident to start working later. Such rationalizations, although not linked with performance, have been reported by [19].

B. RQ2: How do students' time management behaviors affect their team members?

The results in Section IV-B indicate that students' time management behavior affects their teammates' work. It is not surprising that there is a correlation between the difference in teammates' earliness and the change in the their earliness when moving to the project, because the time management behavior must be synchronized at least to some extent. It would be unusual if in a project one member had finished their part before another member has started.

The synchronization of time management practices is in line with Tuckman's model of group development [45]. The model states that teams go through four stages of development: forming, storming, norming and performing. After an initial conflict due to the team members' different working practices (the storming stage), they must come to an agreement of shared practices (the norming stage) before they can start performing.

An interesting observation is that late submitters seem to affect their teammates more than early submitters, i.e. bad habits are apparently more contagious than good habits. The asymmetry can likely be explained by dependencies between the features in the software. An early-working student can finish their tasks before others, giving them freedom to work at their own pace, thus not forcing them to change their habits. In contrast, a late-working student forces others to wait if their tasks are dependent. As stated by Waite [19], procrastination is harmful for team efforts because starting late leaves less time for discussion. Important tasks that require coordination, such as architectural decisions, are likely to suffer from procrastination. Thus, time management appears to be a bottleneck in team projects because badly behaving individuals handicap their team members as well.

It has been consistently shown in the literature that starting to work early is associated with higher academic achievement [28], [29]. For one reason or another, the effect is not visible in the results regarding Research Question 1, where the high-performing students started to work later. It is, however, easy to argument that starting to work earlier is a better practice than starting to work late because it leaves students with time to overcome unforeseen problems. Furthermore, there are indications that dividing practice over a longer period of time rather than cramming in a hurry leads to better learning outcomes [41], [46]. Our results suggest that students are more likely to slip into poor time management practices than improve their habits. Thus, it seems that teaching or encouraging better time management practices would be beneficial for the success of team projects.

C. RQ3: How do students divide their work in web development projects?

While universities emphasize long-term gains such as learning to learn and skills that remain throughout life, many companies expect students to have skills that they could apply directly after graduation [5]. While universities often align parts of their curricula to match industry expectations, industry job postings often stress technical skills and fail to highlight the importance of soft skills [47].

Real life projects[4] are often such that teamwork is required. They are either of a scale that cannot be handled alone in

[4]With real life projects we refer to project in general and not restricted to projects used for educational purposes.

a sensible fashion in the given time frame, or, if they could, having a single developer work on a project creates a situation where an illness or some other factor that creates an absence would incapacitate the project. A real life team also has people with specialized skills. For example, in the context of a large web development project, there are domain experts, user experience designers, UI designers, back-end programmers, front-end experts etc, some of whom may only participate in the project for a short time.

A university programming project shares some of these features of real life projects. The workload and time frame of a course project can be designed so that collaboration becomes desirable. One key difference to real life projects is that in a university project the team members are supposed to learn new things whereas the goal of industry projects is to bring additional value to the customer. Because of this, the work division is also likely to be different. Our course being an introduction to web programming, we want students to gain experience from all aspects of the project, something less often seen in real projects.

When using a web framework, the students can work on individual program features with fairly little co-ordination. This work is further supported by the intelligence built in version control tools which can often automatically combine simultaneous edits from multiple authors. This was clearly reflected in the students' answers about work division. Feature-wise work division that is possible in web software development allows both a learning experience where everyone could work on every layer of the program and a possibility to work as a team with fairly little conflict management. While enhancing productivity, the loose coupling and tight cohesion provided by a framework and the work division adopted by many teams may, however, work against learning how to collaborate by reducing the need for communication with other team members.

The majority of students reported dividing work so that each student works on a specific feature and implements all of the necessary parts in the backend and frontend. Waite et al. [19] observed a similar preference in student projects, but they reported the main reason to be aversion of teamwork. Interestingly, many of our students reported learning-related reasons, such as wanting to give every team member an opportunity to learn all aspects of the framework. However, some students also noted that dividing work this way reduced the need for collaboration.

One way to enforce students to practice all aspects of the project while forcing them to communicate and collaborate would be to not allow students to freely choose how to divide tasks. The teacher might assign one student to work on the frontend and one on the backend of feature A, and then swap for feature B. Not allowing students to choose their roles is also suggested by Barker [48] who argues that students do not necessarily choose the roles that maximize their learning outcomes but roles that they are most comfortable with or have the most prior experience on. Pair programming could provide another approach that facilitates the acquisition of collaboration skills while allowing students to practice all aspects of the project. Pair programming has been observed to improve students' satisfaction towards collaboration and in some cases to lead to better grades when compared to students programming alone [49]. Only a few groups reported preferring to work physically together which suggests that changes in course arrangements are necessary if pair programming is the desired working method. Luckily, educationally the least effective working method, each student working on what they already know best, was rarely reported.

We assume that many of these observations can be generalized even outside of web software development. Regardless of the field, there are effective practices to divide work so that individuals are not hampering each others work. As discussed earlier, these practices have pros and cons for learning as effective teamwork does not necessarily imply ideal learning experience.

D. Validity concerns

Two project courses are rarely identical with each other. There is almost an endless list of factors that can be arranged differently [10]. Still, replicating our study in another context with automatically assessed individual assignments followed by a project work in small teams would be extremely interesting. In this section, we have listed a few obvious limitations related to the generalizability and validity of our results, as well as concerns likely to affect future attempts to replicate the study.

First, as is typical for studies that study educational settings, it is possible that our results are context dependent. For example, in the country where this study was conducted, it is quite typical that many students enter workforce as e.g. junior software developers already during their studies. Thus, it is possible that the programming experience of the course participants is higher than it would be in other contexts and that there are larger differences between the skills of the students. Naturally, the chosen pedagogy and teaching practices in earlier courses play a role here as well, as do other courses that the students are enrolled in.

Second, the survey related to RQ3 was conducted as a post-course survey. Thus, the responses are from students who are asked to recall their own feelings and preferences during the project, parts of which may have already been forgotten. Furthermore, as only a subset of students answered the survey (34% of students that finished the course), it is possible that an inherent selection bias exists.

Third, in this study, we used the final project scores as a measure of success. This can cause limitations to our study as the range in the scores is relatively small. The scores also depend heavily on the teams' own goals and target levels set in internal discussions of each team. Similarly, points from the automatically assessed assignments are not a perfect measure of performance because they are affected by multiple variables such as motivation, carelessness and time spent for other courses or commitments. Other possible measures of performance, that we didn't use in this study, include e.g.

students' grade average and earned credit points averaged over study time.

Fourth, when categorizing students into high- and low-performing, we used a method based on the average of students' performance in the individual assignments. As mentioned in the previous paragraph, there are a range of reasons that could affect students' performance. Thus, there are likely students who performed well in the group project but were categorized into low performing, and vice versa.

Fifth, when analyzing students' version control activity, we have little information over the code that the students have written themselves, and what they have copied from elsewhere. For example, the use of libraries is visible in the source code commits, which may have distorted some of the results related to Github activity. Related to this, in a team setting it may happen that a colleague submits code on behalf of others (although students were explicitly instructed not to do so), or students may work as pairs or collectively. Furthermore, students may have differences in how often they commit versions to the version control system. The fact that teams had multiple commits divided to multiple days decreases the severity of this concern, however. Despite these challenges, using commits as the measure of code created can be a good starting point for estimating how much code the student has written because it shows an active contribution to the project, regardless of where the code is coming from.

VI. Conclusions

In this work, we have analyzed how different team configurations influence project work, and how students' time management behavior in individual assignments influence their team members later in the project. This study was based on data from multiple sources from 50 student teams working on a task with the same requirements; students' submissions to individual programming assignments that were done before the project, version control system logs, post-course student survey, and student interviews during project demonstrations.

Our results suggest that teams that consist of poorly performing team members share their work in a more imbalanced fashion than mixed teams or teams with only high-performing students. Thus, in team projects it may not be a good practice to let students form their own groups because it allows poorly functioning groups to be created. It might be more beneficial to use data from earlier individual assignments to form groups that never consist of only low-performing students and thus have a higher likelihood to succeed. One approach could be to enhance current automated assessment systems with the capability of automatically creating suggestions on student teams. These suggestions could be based e.g. on a heuristic that would attempt to maximize the overall average project score. However, future research is needed to verify the feasibility of these kinds of interventions.

The results indicate that student's individual time management practices prior to the project can have an influence on the teammates' time management practices during the project. It seems that teams are drawn towards the students that work later rather than those working earlier, which implies that the issue cannot be mitigated by forming groups that include at least one early-submitting student. Time management problems appear to be a pitfall in team projects. Thus, we suggest that attention is paid to teaching good time management practices or by other means persuading students to follow them. Using a version control system gives a window into the project and could allow possible interventions if we know how to recognize teams at risk.

Students employ various strategies to divide work among the team. However, most strategies have shortcomings. Each student working on a specific feature teaches them a variety of techniques but allows them to work individually and avoid collaboration and communication. On the other hand, each student specializing on a specific technique requires students to collaborate and communicate but only teaches them a narrow set of technical skills. Our results indicate that most teams divide tasks in a way that is beneficial for developing technical skills instead of collaboration skills. One way to overcome this issue might be to encourage pair programming, which students do not often choose to perform spontaneously. In addition, as students preferred dividing tasks by using the features listed in the project requirements, it might be worth designing the features so that they force the student to employ a broad set of (web software) skills.

References

[1] D. Hagan, "Employer satisfaction with ict graduates," in *Proceedings of the Sixth Australasian Conference on Computing Education - Volume 30*, ser. ACE '04. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2004, pp. 119–123.

[2] A. Begel and B. Simon, "Struggles of new college graduates in their first software development job," in *Proceedings of the 39th Technical Symposium on Computer Science Education*. New York, NY, USA: ACM, March 2008, p. 226–230. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=75113

[3] A. Radermacher and G. Walia, "Gaps between industry expectations and the abilities of graduates," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '13. New York, NY, USA: ACM, 2013, pp. 525–530.

[4] M. Exter, "Comparing educational experiences and on-the-job needs of educational software designers," in *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '14. New York, NY, USA: ACM, 2014, pp. 355–360.

[5] E. M. Trauth, D. W. Farwell, and D. Lee, "The is expectation gap: Industry expectations versus academic preparation," *MIS Q.*, vol. 17, no. 3, pp. 293–307, Sep. 1993.

[6] R. Lingard and E. Berry, "Teaching teamwork skills in software engineering based on an understanding of factors affecting group performance," in *Frontiers in Education, 2002. FIE 2002. 32nd Annual*, vol. 3, Nov 2002, pp. S3G–1–S3G–6 vol.3.

[7] J. Cushing, K. Cunningham, and G. Freeman, "Towards best practices in software teamwork," *J. Comput. Sci. Coll.*, vol. 19, no. 2, pp. 72–81, Dec. 2003.

[8] V. Pieterse, L. Thompson, L. Marshall, and D. M. Venter, "Participation patterns in student teams," in *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '12. New York, NY, USA: ACM, 2012, pp. 265–270.

[9] A. f. C. M. A. Joint Task Force on Computing Curricula and I. C. Society, *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA: ACM, 2013.

[10] T. Clear, M. Goldweber, F. H. Young, P. M. Leidig, and K. Scott, "Resources for instructors of capstone courses in computing," *SIGCSE Bull.*, vol. 33, no. 4, pp. 93–113, Dec. 2001.

[11] A. J. Dutson, R. H. Todd, S. P. Magleby, and C. D. Sorensen, "A review of literature on teaching engineering design through project-oriented capstone courses," *Journal of Engineering Education*, vol. 86, no. 1, pp. 17–28, 1997.

[12] B. C. R. Ulloa and S. G. Adams, "Attitude toward teamwork and effective teaming," *Team Performance Management*, vol. 10, no. 7/8, pp. 145–151, 2004.

[13] A. B. Frymier and G. M. Shulman, ""what's in it for me?": Increasing content relevance to enhance students' motivation," *Communication Education*, vol. 44, no. 1, pp. 40–50, 1995.

[14] J. Biggs and C. Tang, *Teaching for quality learning at university*. McGraw-Hill International, 2011.

[15] N. Clark, P. Davies, and R. Skeers, "Self and peer assessment in software engineering projects," in *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42*, ser. ACE '05. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2005, pp. 91–100.

[16] F. Fagerholm and A. Vihavainen, "Peer assessment in experiential learning assessing tacit and explicit skills in agile software engineering capstone projects," *2013 IEEE Frontiers in Education Conference (FIE)*, vol. 0, pp. 1723–1729, 2013.

[17] K. J. Chapman, M. Meuter, D. Toy, and L. Wright, "Can't we pick our own groups? the influence of group selection method on group dynamics and outcomes," *Journal of Management Education*, vol. 30, no. 4, pp. 557–569, 2006.

[18] M. Ikonen and J. Kurhila, "Discovering high-impact success factors in capstone software projects," in *Proceedings of the 10th ACM Conference on SIG-information Technology Education*, ser. SIGITE '09. New York, NY, USA: ACM, 2009, pp. 235–244.

[19] W. M. Waite, M. H. Jackson, A. Diwan, and P. M. Leonardi, "Student culture vs group work in computer science," in *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '04. New York, NY, USA: ACM, 2004, pp. 12–16.

[20] N. Gorla and Y. W. Lam, "Who should work with whom?: Building effective software project teams," *Commun. ACM*, vol. 47, no. 6, pp. 79–82, Jun. 2004.

[21] L. Capretz and F. Ahmed, "Making sense of software development and personality types," *IT Professional*, vol. 12, no. 1, pp. 6–13, Jan 2010.

[22] T. L. Dickinson and R. M. McIntyre, "A conceptual framework for teamwork measurement," in *Team performance assessment and measurement*. Psychology Press, 1997, pp. 31–56.

[23] N. M. Dowling, D. M. Bolt, S. Deng, and C. Li, "Measurement and control of bias in patient reported outcomes using multidimensional item response theory," *BMC medical research methodology*, vol. 16, no. 1, p. 63, 2016.

[24] O. Hazzan and Y. Dubinsky, "Teaching a software development methodology: The case of extreme programming," in *Proceedings of the 16th Conference on Software Engineering Education and Training*, ser. CSEET '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 176–.

[25] D. Smarkusky, R. Dempsey, J. Ludka, and F. de Quillettes, "Enhancing team knowledge: Instruction vs. experience," *SIGCSE Bull.*, vol. 37, no. 1, pp. 460–464, Feb. 2005.

[26] C. Houldsworth and B. P. Mathews, "Group composition, performance and educational attainment," *Education+ Training*, vol. 42, no. 1, pp. 40–53, 2000.

[27] B. K. Britton and A. Tesser, "Effects of time-management practices on college grades." *Journal of educational psychology*, vol. 83, no. 3, p. 405, 1991.

[28] P. Steel, "The nature of procrastination: a meta-analytic and theoretical review of quintessential self-regulatory failure." *Psychological bulletin*, vol. 133, no. 1, p. 65, 2007.

[29] N. Michinov, S. Brunot, O. L. Bohec, J. Juhel, and M. Delaval, "Procrastination, participation, and performance in online learning environments," *Computers & Education*, vol. 56, no. 1, pp. 243 – 252, 2011.

[30] C. M. Brooks and J. L. Ammons, "Free riding in group projects and the effects of timing, frequency, and specificity of criteria in peer assessments," *Journal of Education for Business*, vol. 78, no. 5, pp. 268–272, 2003.

[31] L. van der Duim, J. Andersson, and M. Sinnema, "Good practices for educational software engineering projects," in *Proceedings of the 29th international conference on Software Engineering*. IEEE Computer Society, 2007, pp. 698–707.

[32] K. Falkner and N. J. Falkner, "Supporting and structuring c̈ontributing student pedagogyïn computer science curricula," *Computer Science Education*, vol. 22, no. 4, pp. 413–443, 2012.

[33] W. M. Waite, M. H. Jackson, and A. Diwan, "The conversational classroom," in *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '03. New York, NY, USA: ACM, 2003, pp. 127–131.

[34] H. Mäenpää, S. Tarkoma, S. Varjonen, and A. Vihavainen, "Blending problem-and project-based learning in internet of things education: Case greenhouse maintenance," in *Proceedings of the 46th ACM technical symposium on computer science education*. ACM, 2015, pp. 398–403.

[35] M. Wiggberg, "Computer science project courses: Contrasting students' experiences with teachers' expectations," Ph.D. dissertation, Uppsala University, 2010.

[36] P. Ihantola, A. Vihavainen, A. Ahadi, M. Butler, J. Börstler, S. H. Edwards, E. Isohanni, A. Korhonen, A. Petersen, K. Rivers, M. A. Rubio, J. Sheard, B. Skupas, J. Spacco, C. Szabo, and D. Toll, "Educational data mining and learning analytics in programming: Literature review and case studies," in *Proceedings of the 2015 ITiCSE on Working Group Reports*, ser. ITICSE-WGR '15. New York, NY, USA: ACM, 2015, pp. 41–63.

[37] K. Mierle, K. Laven, S. Roweis, and G. Wilson, "Mining student cvs repositories for performance indicators," *SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–5, May 2005.

[38] J. Kay, N. Maisonneuve, K. Yacef, and O. Zaïane, "Mining patterns of events in students' teamwork data," in *Proceedings of the Workshop on Educational Data Mining at the 8th International Conference on Intelligent Tutoring Systems (ITS 2006)*, 2006, pp. 45–52.

[39] M. C. Jadud, "Methods and tools for exploring novice compilation behaviour," in *Proceedings of the second international workshop on Computing education research*. ACM, 2006, pp. 73–84.

[40] S. H. Edwards, J. Snyder, M. A. Pérez-Quiñones, A. Allevato, D. Kim, and B. Tretola, "Comparing effective and ineffective behaviors of student programmers," in *Proceedings of the Fifth International Workshop on Computing Education Research Workshop*, ser. ICER '09. New York, NY, USA: ACM, 2009, pp. 3–14.

[41] J. B. Fenwick, Jr., C. Norris, F. E. Barry, J. Rountree, C. J. Spicer, and S. D. Cheek, "Another look at the behaviors of novice programmers," in *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '09. New York, NY, USA: ACM, 2009, pp. 296–300.

[42] N. J. Falkner and K. E. Falkner, "A fast measure for identifying at-risk students in computer science," in *Proceedings of the Ninth Annual International Conference on International Computing Education Research*, ser. ICER '12. New York, NY, USA: ACM, 2012, pp. 55–62.

[43] K. Buffardi and S. H. Edwards, "Effective and ineffective software testing behaviors by novice programmers," in *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ser. ICER '13. New York, NY, USA: ACM, 2013, pp. 83–90.

[44] J. Spacco, D. Fossati, J. Stamper, and K. Rivers, "Towards improving programming habits to create better computer science course outcomes," in *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '13. New York, NY, USA: ACM, 2013, pp. 243–248.

[45] B. W. Tuckman, "Developmental sequence in small groups." *Psychological bulletin*, vol. 63, no. 6, pp. 384–399, 1965.

[46] J. Dunlosky, K. A. Rawson, E. J. Marsh, M. J. Nathan, and D. T. Willingham, "Improving students' learning with effective learning techniques: Promising directions from cognitive and educational psychology," *Psychological Science in the Public Interest*, vol. 14, no. 1, pp. 4–58, 2013.

[47] M. Gallivan, D. P. Truex, III, and L. Kvasny, "An analysis of the changing demand patterns for information technology professionals," in *Proceedings of the 2002 ACM SIGCPR Conference on Computer Personnel Research*, ser. SIGCPR '02. New York, NY, USA: ACM, 2002, pp. 1–13.

[48] L. J. Barker, "When do group projects widen the student experience gap?" in *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '05. New York, NY, USA: ACM, 2005, pp. 276–280.

[49] N. Salleh, E. Mendes, and J. Grundy, "Empirical studies of pair programming for cs/se teaching in higher education: A systematic literature review," *Software Engineering, IEEE Transactions on*, vol. 37, no. 4, pp. 509–525, July 2011.