Team Composition in Software Engineering Project Courses

Dora Dzvonyar
Chair for Applied Software Engineering
Technical University of Munich, Germany
dzvonyar@in.tum.de

Lukas Alperowitz
Chair for Applied Software Engineering
Technical University of Munich, Germany
alperowi@in.tum.de

Dominic Henze
Chair for Applied Software Engineering
Technical University of Munich, Germany
henzed@in.tum.de

Bernd Bruegge
Chair for Applied Software Engineering
Technical University of Munich, Germany
bruegge@in.tum.de

ABSTRACT

Composing well-balanced, effective development teams for software engineering project courses is important for facilitating learning, fostering student motivation as well as obtaining a successful project outcome. However, team composition is a challenging task for instructors because they have to consider a variety of possibly conflicting criteria such as practical constraints, skill distribution, or project motivation.

In this paper, we describe our process for composing development teams based on a pre-defined set of criteria that we have established from our experience conducting project courses since 2008 and constantly refined since. We reflect on these criteria by analyzing the team synergy and project satisfaction of participating students as well as their perspective on challenges in their teams in one concrete instance of a multi-project capstone course. Our findings show that lack of motivation, problems with interpersonal relationships and communication issues affect the less satisfied teams more than the others.

CCS CONCEPTS

• **Social and professional topics → Software engineering education**; • **Software and its engineering → Programming teams**; Agile software development;

KEYWORDS

Software Engineering, Project Course, Team Composition, Teamwork, Applied Education, Experiential Learning, Agile Methods

ACM Reference Format:
Dora Dzvonyar, Lukas Alperowitz, Dominic Henze, and Bernd Bruegge. 2018. Team Composition in Software Engineering Project Courses. In *SEEM'18: SEEM'18:IEEE/ACM International Workshop on Software Engineering Education for Millennials , May 27-June 3 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3194779.3194782

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SEEM'18, May 27-June 3 2018, Gothenburg, Sweden
© 2018 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-5750-0/18/05…$15.00
https://doi.org/10.1145/3194779.3194782

1 INTRODUCTION

Software Engineering is an inherently creative, intellectually complex and collaborative discipline [8, 12, 17]. Students need to apply their theoretical knowledge in practical projects to become skilled software engineers [2, 17]. Therefore, we need to teach in applied, project-based courses if we want to prepare students for their later careers [13, 34, 43]. In particular, teamwork [17, 28, 40] and involving industry partners in university courses [19, 43] are considered to be important aspects of applied software engineering education.

Team composition is regarded as a determinant of software project success, as close collaboration and good communication within the team are integral for creativity and for mastering technological challenges [18, 39]. With regard to a university context, having well-composed teams is also an important factor for learning experience [31]. Letting students form teams on their own as well as random assignment have been reported to yield suboptimal results [15, 21, 41], suggesting that a more formalized approach for team composition is needed.

However, composing project teams is a challenging task, since a large amount of criteria need to be taken into account. Students often have heterogeneous prior knowledge, and while some of these differences can be smoothed out through teaching [25, 27], it has shown to be beneficial to balance the project teams in terms of knowledge so that less experienced participants can learn from more advanced peers [29, 41]. In addition to skills and experience, instructors have to keep in mind cultural and personal differences that can impact later team performance [5, 15, 21] while often having limited knowledge about the participants of their course [36]. Although not all of these issues are connected to or caused by factors of team composition, instructors should aim to compose project teams based on a defined set of validated criteria to ensure a strong starting position for their teams.

The goal of this research is to provide instructors with a set of criteria and a process to create project teams for software engineering courses. We have developed both based on our experience in holding multi-project courses with industry partners since 2008, having conducted over 100 industry projects with more than 1200 participants. After reviewing relevant related work in team composition for software engineering project courses, we present our set of criteria as a recommendation for composing project teams. We then describe our process for composing project teams along with the information and criteria we use at each step. We critically reflect on the process using a concrete instance of a multi-project

capstone course for mobile application development. In particular, we investigate teams with low project satisfaction and synergy and discuss which of the stated reasons are connected to the team composition.

2 RELATED WORK

Early work on the (automatic) composition of teams in a beginner software engineering course without the involvement of industry partners is described in [20]. The input data for the selection of teams consists of information about previous computer science courses and grades, preferred teammates and information about time commitment. This data is used to calculate a score "reflecting the amount and quality of work each student is capable of producing" [20] in order to compose teams that are above a specified optimum.

A manual assignment process in a course for distributed software engineering involving 350 students working in around 60 teams is described in [5]. Students prioritize the projects after short presentations by the staff and submit a questionnaire with self-reported skills. Team composition is then done based on students' project priorities, aiming for a balance of knowledge and keeping in mind that skills relevant to a project should be present in the team.

In terms of a more formalized process of team composition, a decision support system for a project manager based on a list of factors affecting dynamic role allocation in software engineering is presented in [11]. However, their model was designed for industry and has not been evaluated in education. In a university context, [33] present a set of criteria for the "systematic and rigorous formation of groups" in software engineering courses based on individual characteristics and behavior with a focus on skills and personality as well as group metrics. With regard to skills, [30] describe a balance of technical and less technical majors, and [31] delay project team formation until after the first exam of the course in order to use the results to balance teams. Some researchers favor a random assignment of teams [3], while others base the process primarily on students' priorities [24, 42].

Our aim is to contribute to provide both a set of criteria and a corresponding process for instructors of software engineering project courses to contribute to closing the gap in current academic research.

3 TEAM COMPOSITION CRITERIA

In this Section, we present the criteria which are the basis of our team composition process. Some of them are evaluated at the team level, while others are taken into account at a personal level. We grouped the criteria into four categories as depicted in Figure 1 and describe each of them in this section along with relevant academic research on their impact on team performance.

3.1 Practical Constraints

These criteria have to be considered to create fair circumstances for all teams and to avoid putting certain teams at a disadvantage.

Criterion A.1: Team Size. We create teams of similar size unless a project requires more manpower due to a challenging problem or a demanding industry partner. Some studies indicate that student teams of 2-3 members perform better in terms of project scores than those of size 6-7 [31], but larger teams lead to an experience closer to that of working in industry. In any case, we recommend instructors to decide on the desired team size upfront to have a clear goal for the team composition process.

Criterion A.2: Development & Test Devices. The technologies involved in the course determine which devices are needed for development and testing. For instance, a web development project course requires only a laptop or computer, while a course on iOS app development also requires iPhones or iPads. A lack of test devices makes it difficult for the students to test their applications in the target environment and thus could prevent them from adopting state-of-the-art software engineering practices such as continuous delivery [29]. Therefore, we set a minimum threshold for the number of test and development devices in each team.

Criterion A.3: Schedule Flexibility. A project course typically requires not only a large time investment, but also a certain flexibility in terms of scheduling within the team for team meetings or work sessions. Scheduling problems have been reported to cause issues in student projects [1, 41], which is why we inquire about other commitments the students have during the semester, such as other courses with a heavy workload or excellence programs with a busy schedule. If we suspect that the student is overcommitting, we communicate clearly about the expected time investment and flexibility to ensure that they make an informed decision.

3.2 Skill Distribution

One of the biggest challenges of project courses results from the participants having very heterogeneous prior knowledge. As instructors of a project course, we can smooth out some of these differences in prior knowledge through teaching, but we believe that we achieve the best learning outcome for all course participants if we create a setting in which less experienced students can learn from their more advanced peers.

Whether in industry or in university courses, software engineers need a wide variety of skills to accomplish great work [10]. In addition to knowledge in technologies and activities directly related to software engineering such as requirements elicitation or modeling, social and personal skills like the ability to compromise or manage one's own time are important [38]. Skill-related issues have

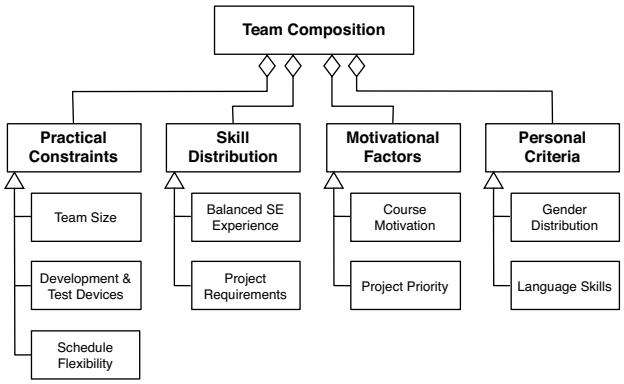

Figure 1: Team Composition Criteria

been reported to be among the most common problems in student projects, resulting in additional workload [1, 6]. Hence, we ensure the following criteria are met by the teams we compose.

Criterion B.1: Balanced Software Engineering Experience. In order to facilitate peer learning, we create balanced teams with regard to technical skills and prior knowledge in object-oriented software development, user experience design, and software modeling. This allows team members who are more skilled in one of these areas to be responsible for the most challenging tasks while guiding their less advanced teammates, creating a setting in which everyone can develop their skills [15, 30, 37]. Balanced teams create fair circumstances across the projects [15, 31] and have been reported to have a positive effect on team performance and group synergy [31, 41]. Section 4 details which skills we take into account and how we achieve a balance.

Criterion B.2: Project Requirements. In addition to a general balance in each team with regard to software engineering skills, some of the projects require experience in a particular field or technology such as computer vision or embedded programming. We aim to meet such requirements whenever this is possible. Thus, we ensure that the most critical and technologically challenging projects are adequately staffed, lowering the risk for a project team to have a heavy workload since all team members need to familiarize themselves with a completely new technology [6, 36, 41].

3.3 Motivational Factors

Motivation has been regarded as an important factor for satisfaction in the software engineering industry [16, 31]. Conversely, a lack of motivation has also shown to be a recurring challenge in student projects, leading to team members becoming inactive or dropping out, leaving behind an increased workload for their teammates [1]. Motivation-Hygiene Theory proposes a dual model: *motivators* are factors that increase a person's work satisfaction and involvement, while *hygiene factors* cause dissatisfaction if they are not met [22].

Motivation of students is influenced positively and negatively by a variety of criteria, not all of which are associated with team composition: while the grade can be a motivator for some students, others participate in the course to learn new things and to contribute to a challenging project [6]. However, teamwork in software engineering project courses is an "emotive issue" [15] and the impact of the perceived quality of team composition on motivation should not be underestimated by the instructor.

Criterion C.1: Course Motivation. Since our project courses are electives and we typically receive more applications than the number of places we can offer, we can prioritize students based on their motivation for the project course as a whole. We ask them about their reasons for choosing the course from the variety of electives offered, which gives us an impression of what they want to take away from their project and whether their motivation fits the learning outcomes of the course.

Criterion C.2: Project Priority. We let the course participants prioritize all projects offered in the course and give this criterion great importance when composing the project teams. Some projects are always more popular than others [5], which means that in most cases not everyone can get their top choice. Nevertheless, our aim is to assign every student to one of their top-prioritized projects, unless this conflicts with other team composition criteria.

3.4 Personal Criteria

A wide variety of personal factors have been examined by researchers with regard to their impact on teamwork. For instance, lack of cultural fit within a team has been identified as a source of demotivation, communication problems and conflict [6, 11, 15, 16, 35]. Compatibility of work habits and way of thinking are regarded as beneficial for teamwork [11, 23, 37, 44], but a demographic or cultural mix can also widen students' experience and lead to the acquisition of new skills [15].

Personality is another criterion that has been widely discussed. Taking personality profiles into account when composing teams has been shown to increase performance, collaboration and knowledge acquisition in student courses [9, 33]. While these criteria are interesting for future work, we do not formally include them in the team composition process at this time. However, we do take into account two other criteria for composing project teams.

Criterion D.1: Gender Distribution. The results of experiments studying gender in software engineering teams have not yielded clear results [14, 35], but research has shown that fostering collaboration across genders can break down stereotypes [19]. With a distinct lack of female students at the Department of Informatics,[1] our aim is to distribute female participants equally across the project teams.

Criterion D.2: Language Skills. We ask for the students' language skills in German, English (the official language of the course) as well as their native language to avoid situations we have experienced to be problematic. First, we do not want a project team with only one non-German-speaking member; we found that in this case the team members default to their native German language in work meetings as well as informal conversations, leaving the only foreign student isolated. Second, we try to avoid having a group of 2-3 visiting students from the same country in a team because they have shown to build a tightly coupled sub-group that is prone to isolate themselves from the rest of the team. While this criterion is not our highest priority, we regard it as a hygiene factor that can impact team communication and take it into account as long as it does not conflict with other, more important criteria.

4 TEAM COMPOSITION PROCESS

As instructors, our goal is to assign course participants to project teams with the aim of composing well-functioning teams. At the beginning of the course, the participants provide us with data about their background with regard to software engineering as well as their personal details and interests using a structured questionnaire. In terms of prior knowledge, we ask for their experience with software engineering workflows and tools as well as technologies relevant for the course. In addition to their existing knowledge,

[1] TUM Faculty for Informatics facts and numbers from 2016: http://bit.ly/2DMjWm1

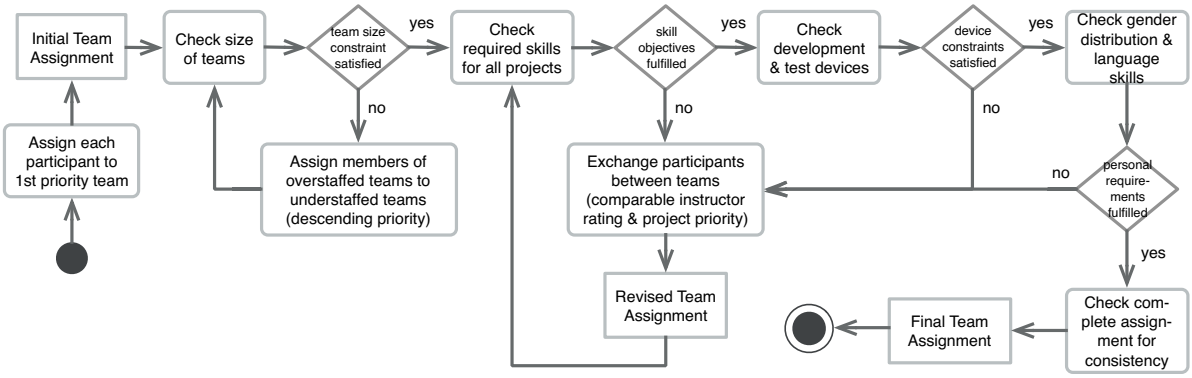

Figure 2: Overview of our Team Composition Process

students can specify whether they are particularly interested in developing their skills in each of these domains.

In order to obtain more reliable answers, we ensure the participants that the information provided will not be used to assess their contribution in any way but will be used to create balanced project teams. The questionnaire also collects data about the participants' development and test devices (Criterion A.2), language skills in German and English (Criterion D.2), their gender (Criterion D.1) as well as current program of study, which gives them the opportunity to update the information they provided when they applied for the course. Finally, students prioritize all projects based on the project descriptions and customers' presentations (Criterion C.2).

The composition of the project teams takes place with 2-4 instructors present in order to get a more complete overview of all factors through peer discussion. As a first step, the instructors consider each student individually and convert their self-reported skills and interests into a more comparable and objective rating. Despite the fact that we ask the participants to be honest about their skills so that we can create the best teams possible, students are prone to overestimating their own capabilities [6]. Therefore, we carefully assess each students' self-reported skills and verify them using experiences with the student from e.g. previous courses under our supervision. Although some educators recommend conducting interviews with students to obtain more reliable data [6], the size of our course makes this approach impractical and limits us to using second-hand knowledge.

After verifying the self-reported information with the knowledge at hand, we rate each participant on the following scale:

(1) **Novice**: The student has little hands-on knowledge in software engineering or is at the beginning of their studies.
(2) **Normal**: The student's performance and skills are average.
(3) **Advanced**: The student has above-average knowledge in software engineering or has performed remarkably well in previous courses.
(4) **Expert**: The student has considerable experience in not only software engineering, but also with developing for the technology used in the project course (e.g. they have an application in the AppStore for a course on mobile app development).

Figure 2 shows an overview of the process after assigning a rating to each participant. The instructors start with an initial assignment of the first project priority for each student (Criterion C.2). Since the priority distribution of projects varies strongly [5], this results in a distribution that is unequal in terms of team size and most other criteria. In order to equalize team size (Criterion A.1), we first evaluate which students we can move from the most popular teams to less overcrowded ones while keeping the average priority as high as possible.

We then iteratively check our criteria: We start by balancing the teams to contain an equal amount of Advanced and Expert participants while also considering projects with special requirements (Criteria B.1 and B.2). We then verify that each team has a sufficient amount of development and test devices (Criterion A.2), and we aim at having at least one female student in each team as well as minimizing language conflicts (Criteria D.1 and D.2).

In summary, our team composition process is based on the iterative refinement of the assignment based on a variety of criteria, which are sometimes ambiguous or even conflicting. The manual process takes between 3 and 5 hours. Our instructors often find it cumbersome to take all criteria into account, which is why we are currently working on algorithmically supporting the process. The fact that there are multiple instructors present makes it easier to reach an acceptable solution through discussion and a better overview of the information.

5 CASE STUDY

In this section we use a concrete instance of a project course on mobile application development from the winter semester 2016/17.

5.1 Setting

We use our multi-project course *iPraktikum* as a concrete example. The course takes place every semester with 70-80 developers in 10-12 project teams working in parallel to develop mobile applications in the context of a larger system architecture for real customers from industry. While the mobile components are developed on the iOS platform, many projects also include sensors, wearable devices or application servers running e.g. machine learning algorithms. The structure and teaching methodology of the course have been described in detail in [7, 29].

This instance of the course involved 80 students with a heterogeneous background: 46% of the participants were Bachelors students and 54% were Masters students from 5 different fields of study,

Table 1: Concrete team composition criteria in the case study

Criterion	Measure	Total	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
Team Size (A.1)	Number of developers	80	8	7	7	8	8	7	7	7	7	7	7
Development & test devices (A.2)	Amount of dev. devices	48	5	5	4	5	5	4	4	3	5	3	5
	Amount of test devices	67	4	5	7	5	7	7	9	5	6	5	7
Balanced software engineering experience (B.1)	Instructor rating "Novice"	20	1	1	2	3	3	1	1	0	3	2	3
	Instructor rating "Normal"	34	5	4	2	2	2	5	4	5	1	3	1
	Instructor rating "Advanced"	24	1	2	2	3	3	1	2	2	3	2	3
	Instructor rating "Expert"	2	1	0	1	0	0	0	0	0	0	0	0
Project priority (C.2)	Avg. priority overall	—	5.04	6.26	6.03	4.08	9.39	6.01	4.15	8.51	5.46	5.33	5.75
	Avg. priority in project team	—	1.38	1.57	1.14	1.13	4.00	1.29	1.00	4.57	1.71	2.00	1.14
Gender distribution (D.1)	Female participants	12	1	1	1	2	1	2	1	0	1	1	1
Language skills (D.2)	Non-German-speakers	26	4	2	0	3	4	2	2	4	2	2	3

including two engineering degrees without a big proportion of computer science courses in their curricula. Moreover, we had a ratio of roughly 1/3 foreign or exchange students for whom we had little or no information about their prior courses and experiences.

In order to prepare the participants for their projects, we held a week-long introductory course in which we taught them iOS development. The introductory course contained homework assignments corrected by tutors. Following this, we held a Kickoff meeting on 20 October 2016 in which the customers presented their problems to all participants. Based on these presentations as well as the project descriptions, the students prioritized the projects and provided us with the information needed for our team composition process. In addition, we collected short statements from the tutors for each participant regarding their performance in the introductory course, which we used to verify the self-reported information to create a more accurate instructor rating as described in Section 4.

We assigned the 80 developers to 11 teams. Table 1 shows the concrete values for each criterion used in the process, both for the overall set of participants as well as for the resulting project teams. Schedule flexibility (Criterion A.3) and project requirements (Criterion B.2) could not be quantified and are thus not represented in the table.

We created teams of 7-8 developers with a minimum of 4 test devices (iPhones or iPads) and 3 development devices (MacBooks) based on the average number of devices in the course overall. With regard to skills, we assigned an instructor rating to each student as described in Section 4. We then used these ratings to distribute skills equally among the project teams, balancing novice students with teammates who have advanced or expert knowledge while keeping in mind to staff projects with special requirements accordingly. Finally, we took care of distributing genders and non-German speakers so that our personal criteria were fulfilled in most teams. In T8, we broke the female participant constraint; although it would have been possible to assign at least one female to each team because 12 were registered in the course, this team already had the lowest average priority and we did not want to deteriorate it further. The team composition process took 4.2 hours and was done by 3 instructors.

Each team was led by a duo of project leader who is a doctoral candidate, as well as a team coach, a student who has taken the course before and helps the development team with regard to communication and agile practices. The iPraktikum ran for 15 weeks and finished on 2 February 2017. The results of the course, including a short description of the projects as well as recordings of the students' presentations are available online.[2]

5.2 Methodology

After the end of the course, we sent out an anonymous questionnaire to all developers as well as team coaches. The questionnaire was composed of the following parts:

(1) **Project Satisfaction** A rating of the respondent's overall experience in their project on a scale from 1 to 10,
(2) **Team Synergy** a set of questions to determine their perceived team synergy, and
(3) **Description of problems in the team** an open-ended question asking participants to describe at least one problem occurring in their project team.

The scale for 2) was adapted from the team effectiveness audit tool presented in [4]. We selected the questions from the Team Synergy section, except those who do not apply in the context of a university course, such as the team being valued by other parts of the organization. We ended up with the following 5-Point Likert items for the Team Synergy Scale:

- *There was a common sense of purpose for this team.*
- *Members were clear about their roles within the team.*
- *There was effective communication within the team.*
- *I felt valued as an individual member of the team.*
- *Morale within the team was high.*
- *There was effective and appropriate leadership within the team.*
- *All individuals performed to the best of their ability within the team.*

We sent the questionnaire after the projects were finished, but before the students received their grades to avoid influencing their answers through the final assessment. We emphasized that the questionnaire was anonymous on a team level and that we would strictly not use the provided information for grading.

5.3 Results

We received 81 responses to the questionnaire, which amounts to a response rate of 89% (minimum 50% of each team). We analyzed students' project satisfaction and team synergy scores on a team

[2] http://www1.in.tum.de/ios1617

level, followed by an analysis of the open-ended responses of all respondents.

5.3.1 Team-level analysis. As a first step, we calculated the Team Synergy Score (TSS) of each respondent on a summated scale from the 7 items, giving us a score between 5 and 35 with a median at 28 and the mean TSS at 27.9. Cronbach's alpha for the sample was 0.88, which points to good internal consistency of the items in the scale across respondents [26]. The Project Satisfaction Score (PSS) did not need further processing since it had already been collected as a single score between 1 and 10. The median PSS was 8, and the mean score was 8.1. Figures 3 and 4 visualize the mean as well as the standard deviation for both PSS and TSS by team.

Upon examining the data, we identified the teams T6 and T7 as 'problematic teams' in the scope of our case study, as they received noticeably lower averages for both PSS (5.3 and 7.0) and TSS (21.3 and 21.9, respectively). Moreover, the variance of both scores was higher than for the other teams, suggesting a low level of agreement between team members. A Kruskal-Wallis Test run on the Project Satisfaction Score yielded a significant result ($\chi^2 = 27.209, p = 0.0024$), and a post-hoc analysis was conducted to determine which of the 11 teams differed significantly from each other. We performed a Benjamini-Hochberg-adjusted Dunn Test because of its support of groups with an unequal number of observations. The results of the post-hoc analysis revealed a significant difference between both 'problematic teams' T6 and T7 to T8, which was the team with the highest PSS. In addition, T6 also differed from two other teams, namely T1 and T10. The fact that the post-hoc analysis did not yield more significant results is due to the large number of groups compared and the relatively small amount of subjects per group. We confirmed our initial observation with the two project leaders and team coaches, who agreed that there were recurring issues in their teams.

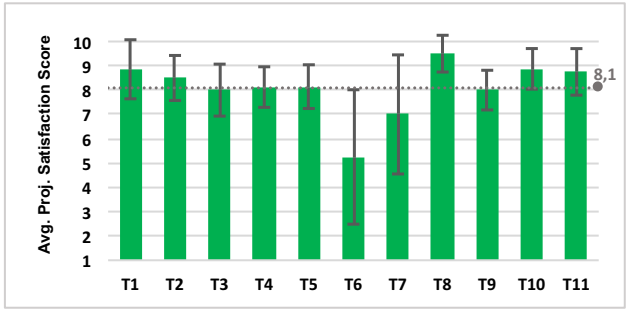

Figure 3: Average Project Satisfaction Scores (PSS) per team

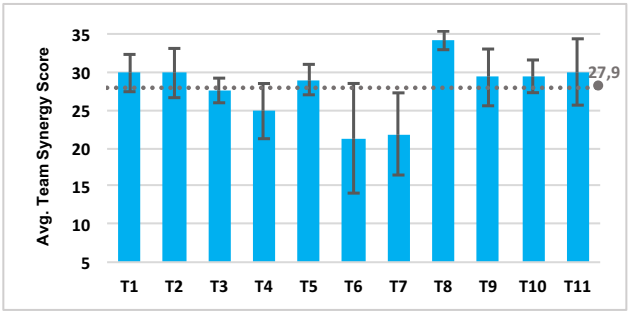

Figure 4: Average Team Synergy Scores (TSS) per team

Following this, we analyzed the distribution of project priorities from the beginning of the course. As described in Section 5.1, the students prioritized all 11 projects after the course Kickoff event. We then took these priorities into account when assigning the students to project teams with the aim of giving each participant their highest possible choice (Criterion C.2) as long as it does not conflict with other criteria we have to take into account. The average project priorities among all participants as well as the mean priority among the members of the resulting project team after composition are shown in Table 1.

Our analysis shows that T7, T4, T3 and T11 were ranked highest in terms of priorities, with an average project priority between 1.00 and 1.14 in the composed project teams. T5 and T8 were staffed with students who had prioritized those projects considerably lower, with an average priority of 4.00 and 4.57, respectively. This was unavoidable due to all course participants giving these two projects a low priority.

5.3.2 Analysis of open-ended responses. We employed a provisional coding technique [32] based on an initial set of codes we developed from our set of team composition criteria described in Section 3, and revised codes while we reviewed the responses. The final set of codes as well as their frequency in the survey responses is shown in Figure 5. The following codes are directly connected to our team composition criteria: Scheduling difficulties (Criterion A.3), inadequate skills of team members (Criteria B.1 and B.2) and lack of motivation or uneven contribution (Criteria C.1 and C.2). These codes occurred in just over one-third of the responses. 37% of responses concerned communication within the team, which was the most frequent code. These responses contained problems connected to inefficient communication about tasks, a lack of direction in meetings, or issues with the feedback and discussion culture within the team.

Following this, we examined whether the codes occur more frequently in our previously identified 'problematic teams' T6 and T7. Out of the total of 81 responses, 15 were from members of these two teams, which is comparable to the response rate of the other teams. The frequency of occurrence of each code divided into our two sub-samples is shown in Figure 5. Although the responses of T6 and T7 make up 18.5% of the overall data, they account for 25% of occurrences of issues concerning team communication, and the proportion is higher for lack of motivation or contribution and personal relationships, with 37.5% and 75%, respectively.

5.4 Discussion

The aim of our case study was to examine which problems occur in project teams and to critically reflect if and how these can be influenced by the team allocation. We identified two teams that differed from the others in their average satisfaction scores and team synergy scores. The fact that the Kruskal-Wallis Test resulted in a significant difference of these teams from some of the others despite the large number of groups compared and the small number of responses in each group lent statistical support to our initial observation. We also received confirmation from the two project leaders that there were more issues in the way those two teams worked compared to the rest of the course.

Looking at the priority the members of those teams gave the project at the beginning of the course, our results do not show a connection between mean priority of the assigned members and TSS or PSS scores: T6 and T7 did not consist of students who had ranked those projects lower; T7 was even the only team that consisted exclusively of people having given the project their top priority. Conversely, the teams T5 and T8 that had the lowest project rankings amongst their members did not end up with a lower PSS or TSS than the higher-prioritized teams; T8 even had the highest TSS of all teams! This suggests that those teams did not feel less motivated or satisfied with their project. One could draw the conclusion that the project priority does not influence students' experiences in project courses at all. However, we think that this is only partly the case.

With regard to how students prioritize projects, our experience shows that high average project priority is dependent on the following factors: the quality of the kickoff presentation by the customer, the clarity of the presented problem and the perceived achievability of a solution. The teams with lower ratings were lacking in at least one of these factors: while T8 had an interesting and clear problem, the poor presentation and questionable achievability led to an overall low priority of this project. In comparison, T5 had a high-quality customer presentation, but the problem was so vague that this also led to a low average priority among the course participants. While these factors influence participants' priorities, they have little effect on their experience in later stages of the project. We think that the project priority is a motivational hygiene factor as described in Section 3.3, and our experience shows that students react much more strongly to being assigned to a low-priority team in recent years than when we started teaching project courses. Getting assigned to a high-priority project affects the *initial motivation* of participants; it helps teams to start strong [5] and prevents early dropouts of unsatisfied students. However, we sent out the questionnaire *after the course*, which gave us insight into their experience as a whole. Further investigation is necessary to empirically validate the impact of project priority on students' experience, with a focus on the first weeks of the project. Our experience shows that project motivation is even gaining importance: Students who end up in a low-priority team show a much stronger reaction now than they did ten years ago, and we have 1-2 dropouts due to the project assignment at the beginning of each course. One could argue that Millennials put a larger emphasis on being intrinsically motivated to work on a particular problem.

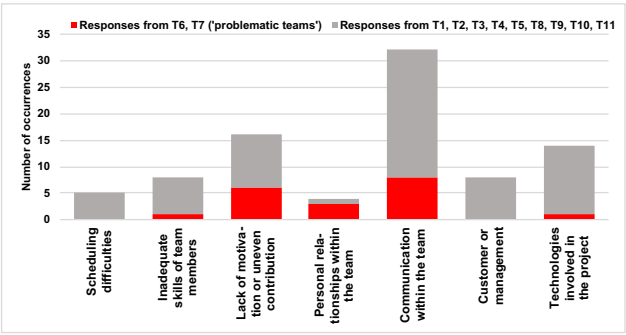

Figure 5: Comparing frequency of codes in responses from members of 'problematic teams' to the rest of participants

The analysis of the open-ended responses supports our intuition that the overall experience during the course as well as the quality of teamwork depend on a number of factors over the whole duration of the project. Not all of these factors are connected to the composition of the team, hence we cannot prevent all issues that can arise over the course of the project through the team assignment alone. However, three of the codes present in participants' responses can indeed be clearly mapped to our criteria, and others are also likely to be affected by the team composition: for instance, personal relationships within the team as well as communication depend strongly on the members of the team. Both of these codes occurred on average more frequently in our two 'problematic teams' than in the others, and it is reasonable to argue that they are in connection with the lower perceived team synergy and project satisfaction of team members.

It is also noteworthy that scheduling difficulties, inadequate skills of team members, challenges related to the customer, management problems and issues concerning technologies involved in the project did not occur more frequently in the two teams with lower TSS and PSS scores. The question whether these factors are noted by participants but have less influence on their course satisfaction than other issues remains open.

Finally, a recurring pattern in respondents' comments was the description of a problem that occurred at the beginning of the course, but the team resolved it as they got used to working together. While a structured, balanced team allocation process does not prevent all problems or leads to an overall less workload for the team members, we think that we are creating a strong starting point for the teams and establish an atmosphere in which course participants can contribute to the best of their abilities.

5.5 Validity

One of the threats to validity of this evaluation is that not all students are experienced and mature enough to identify issues in their team or to assess team synergy. Where possible, we verified the participants' perspective through conversations with the project leaders and team coaches. An additional bias is introduced by the fact that some respondents might have been influenced by their desire to impress the instructors in order to receive a better grade. We reduced this threat by stressing that the questionnaire was anonymous on a team level, but a certain bias can still remain.

The fact that we did not have a control group limits empirical support for our findings. Introducing a control group would have meant composing part of the teams at random or deliberately creating teams that do not conform to our standards, which is a risk we did not want to take considering the importance of students' learning experience and the quality of the project outcome for the industry partners. We grounded the individual criteria in academic literature and based the process on our long-standing experience conducting project courses to strengthen the basis of our approach.

6 CONCLUSION

In this paper we presented a set of criteria for composing teams for software engineering project courses, categorized into practical constraints, skill distribution, motivational factors and personal criteria. We introduced the process we use to assign participants to

project teams and described where we get the required information from. Finally, we used a concrete multi-project course on mobile application development to instantiate our team composition process and to critically reflect on the effect of our criteria on participants' experience in the course. We determined that the priorities of members of 'problematic teams' were not lower than those of the others and thus conclude that project priority is most likely a hygiene factor impacting the motivation at the beginning of the project, which is gaining importance in the Millennial Generation. We recommend instructors to take into account the priorities of course participants nevertheless, as high motivation in the beginning gives the teams a strong starting point and helps to prevent participants from dropping out. We are planning further studies to get a more accurate picture of participants' experience in the first weeks.

Our analysis of the open-ended description of issues in the projects shows that, on average, members of our two 'problematic teams' reported more problems. Especially issues centered around lack of motivation, interpersonal relationships and communication within the team occurred more frequently in those teams. Communication was the most frequent code overall, which suggests that follow-up studies are needed to further investigate how project teams communicate and how instructors can help them prevent problems in this area.

A next step in our research is the development of a semi-automatic team composition decision support system that builds on the presented criteria. The goal is to increase the manageability of the currently very complex and time-consuming process by presenting an initial, algorithmically optimal team composition to the instructors; they can then adapt this based on their intuition and knowledge, thus creating a balance between algorithmic performance and human experience.

ACKNOWLEDGMENT

The authors would like to thank everyone involved in the iPraktikum who agreed to participate in our research.

REFERENCES

[1] Tero Ahtee and Timo Poranen. 2009. Risks in Students' Software Projects. In *CSEE&T '09*. IEEE, 154–157.
[2] Victor R Basili. 1996. The role of experimentation in software engineering: past, current, and future. In *ICSE '96*. IEEE, 442–449.
[3] Cecilia Bastarrica, Daniel Perovich, and Maira Marques Samary. 2017. What can Students Get from a Software Engineering Capstone Course? *ICSE '17*.
[4] Billy Bateman, F. Colin Wilson, and David Bingham. 2002. Team effectiveness – development of an audit questionnaire. *Journal of Management Development* 21, 3 (apr 2002), 215–226.
[5] Ivana Bosnic, Igor Cavrak, Marin Orlic, and Mario Zagar. 2013. Picking the right project: Assigning student teams in a GSD course. In *CSEE&T '13*. IEEE, 149–158.
[6] Ivana Bosnić, Igor Čavrak, Marin Orlić, Mario Žagar, and Ivica Crnković. 2011. Student motivation in distributed software development projects. In *CTGDSD '11*. ACM, 31–35.
[7] Bernd Bruegge, Stephan Krusche, and Lukas Alperowitz. 2015. Software Engineering Project Courses with Industrial Clients. *ACM TOCE* 15, 4 (2015), 17.
[8] L.F. Capretz and F. Ahmed. 2010. Making Sense of Software Development and Personality Types. *IT Professional* 12, 1 (jan 2010), 6–13.
[9] L. F. Capretz. 2002. Implications of MBTI in software engineering education. *ACM SIGCSE Bulletin* 34, 4 (dec 2002), 134.
[10] Luiz Fernando Capretz. 2003. Personality types in software engineering. *International Journal of Human-Computer Studies* 58, 2 (feb 2003), 207–214.
[11] G.A. Dafoulas and L.A. Macaulay. 2001. Facilitating group formation and role allocation in software engineering groups. In *AICCSA '01*. IEEE, 352–359.
[12] Norman Fenton and James Bieman. 2014. *Software Metrics: A Rigorous and Practical Approach, Third Edition* (3rd ed.). CRC Press, Inc., Boca Raton, FL, USA.
[13] N. Fenton, S.L. Pfleeger, and R.L. Glass. 1994. Science and substance: a challenge to software engineers. *IEEE Software* 11, 4 (jul 1994), 86–95.
[14] L. Fernández-Sanz and Sanjay Misra. 2012. Analysis of cultural and gender influences on teamwork performance for software requirements analysis in multinational environments. *IET Software* 6, 3 (2012), 167.
[15] Sally Fincher, Marian Petre, and Martyn Clark (Eds.). 2001. *Computer Science Project Work*. Springer, London.
[16] A.C.C. Franca, T.B. Gouveia, P.C.F. Santos, C.A. Santana, and F.Q.B. da Silva. 2011. Motivation in software engineering: a systematic review update. In *EASE '11*. IET, 154–163.
[17] Carlo Ghezzi and Dino Mandrioli. 2006. *The Challenges of Software Engineering Education*. Springer Berlin Heidelberg, 115–127.
[18] Narasimhaiah Gorla and Yan Wah Lam. 2004. Who should work with whom? *Commun. ACM* 47, 6 (jun 2004), 79–82.
[19] J.V. Harrison. 1997. Enhancing software development project courses via industry participation. In *CSEE&T '97*. IEEE, 192–203.
[20] Sallie Henry. 1983. A project oriented course on software engineering. In *SIGCSE '83*. ACM, 57–61.
[21] Sallie Henry, Nancy Miller, Wei Li, Joseph Chase, and Todd Stevens. 1999. Using software development teams in a classroom environment. In *ACM SIGCSE Bulletin*, Vol. 31. ACM, 356–357.
[22] Frederick Herzberg. 2005. Motivation-hygiene theory. *J. Miner, Organizational Behavior I: Essential Theories of Motivation and Leadership* (2005), 61–74.
[23] H.-R. Kang, H.-D. Yang, and C. Rowley. 2006. Factors in team effectiveness: Cognitive and demographic similarities of software development team members. *Human Relations* 59, 12 (dec 2006), 1681–1710.
[24] Edward P. Katz. 2010. Software Engineering Practicum Course Experience. In *CSEE&T '10*. IEEE, 169–172.
[25] A. B. Kayes. 2005. Experiential learning in teams. *Simulation & Gaming* 36, 3 (sep 2005), 330–354.
[26] Paul Kline. 2013. *Handbook of psychological testing*. Routledge.
[27] A. Y. Kolb, D. A. Kolb, A. Passarelli, and G. Sharma. 2014. On Becoming an Experiential Educator: The Educator Role Profile. *Simulation & Gaming* 45, 2 (apr 2014), 204–234.
[28] A.J. Kornecki, I. Hirmanpour, M. Towhidnajad, R. Boyd, T. Ghiorzi, and L. Margolis. [n. d.]. Strengthening software engineering education through academic industry collaboration. In *CSEE&T '97*. IEEE, 204–211.
[29] Stephan Krusche, Lukas Alperowitz, Bernd Bruegge, and Martin O. Wagner. 2014. Rugby: an agile process model based on continuous delivery. In *RCoSE '14*. ACM, 42–50.
[30] Patricia Lago, Joost Schalken, and Hans van Vliet. 2009. Designing a Multidisciplinary Software Engineering Project. In *CSEE&T '09*. IEEE, 77–84.
[31] R. Lingard and E. Berry. 2002. Teaching teamwork skills in software engineering based on an understanding of factors affecting group performance. In *FIE '02*. IEEE, S3G–1–S3G–6.
[32] Matthew B Miles, A Michael Huberman, and Johnny Saldana. 2013. *Qualitative data analysis*. Sage.
[33] Amir Mujkanovic and Andreas Bollin. 2016. Improving learning outcomes through systematic group reformation. In *CHASE '16*. ACM, 97–103.
[34] Tom Nurkkala and Stefan Brandle. 2011. Software studio. In *SIGCSE '11*. ACM, 153.
[35] Abhoy K Ojha. 2005. Impact of team demography on knowledge sharing in software project teams. *South Asian Journal of Management* 12, 3 (2005), 67.
[36] Luis Daniel Otero, Grisselle Centeno, Alex J. Ruiz-Torres, and Carlos E. Otero. 2009. A systematic approach for resource allocation in software projects. *Computers & Industrial Engineering* 56, 4 (may 2009), 1333–1339.
[37] J.S. Reel. 1999. Critical success factors in software projects. *IEEE Software* 16, 3 (1999), 18–23.
[38] José Gamaliel Rivera-Ibarra, Josefina Rodríguez-Jacobo, and Miguel Angel Serrano-Vargas. 2010. Competency Framework for Software Engineers. In *CSEET '10*. IEEE, 33–40.
[39] Hugh Robinson and Helen Sharp. 2010. *Collaboration, Communication and Coordination in Agile Software Development Practice*. Springer Berlin Heidelberg, 93–108.
[40] Kurt Schneider, Olga Liskin, Hilko Paulsen, and Simone Kauffeld. 2015. Media, Mood, and Meetings. *ACM TOCE* 15, 4 (dec 2015), 1–33.
[41] Thomas J. Scott, Lee H. Tichenor, Ralph B. Bisland, and James H. Cross. 1994. Team dynamics in student programming projects. In *SIGCSE '94*. ACM, 111–115.
[42] Todd Sedano, Arthi Rengasamy, and Cecile Peraire. 2016. Green-Lighting Proposals for Software Engineering Team-Based Project Courses. In *CSEE&T '16*. IEEE, 175–183.
[43] C. Wohlin and B. Regnell. 1999. Achieving industrial relevance in software engineering education. In *CSEE&T '99*. IEEE, 16–25.
[44] Hee-Dong Yang, Hye-Ryun Kang, and Robert M Mason. 2008. An exploratory study on meta skills in software development teams: antecedent cooperation skills and personality for shared mental models. *European Journal of Information Systems* 17, 1 (feb 2008), 47–61.