6-26-2017

Understanding the impact of strategic team formation in early programming education

Tony A. Lowe
Purdue University, lowe46@purdue.edu

Sean B. Brophy
Purdue University - Main Campus

Understanding the impact of strategic team formation in early programming education

Abstract

This evidence based research looks at the impact of a team-based instruction on learning to program in a first year engineering course designed under the Bauhaus studio model. Each team is formulated with a "more knowledgeable other" [1], or for this paper the "ringer" based on self-reported prior learning. The ringer is intended to support the team through early programming challenges. In addition to the professor and teaching assistants, having a peer mentor can yield higher satisfaction and confidence in learners [2]. Our analysis evaluates learning outcomes as student progress through the term, comparing performance based on the performance and prior knowledge reported by the ringer. The major research questions investigate the role of the ringer in the success of the team, as well looking to see if teams that include a low performing student have any common characteristics. Findings include data from 2013, 2014, and 2015 with trends apparent in each of the years across major topics.

This study shows that the formulation of teams around a carefully selected more knowledgeable other can improve the learning of the entire team. In general, ringer score correlates to an increase in the rest of the team's average. The ringer score only supports learning to a certain degree where if the gap in score is too larger compared to the rest of the team, lower performing members can suffer. In general the formation of teams using prior programming experience seems to do no harm and even possibly improve learning outcomes, and the data may also suggest additional improvements on the use of teams.

Introduction

Students learning to program tend to persist and perform better when they are paired with their peers to complete coursework. Students fail in programming classes at rates starting at 20% [3], [4] up to 50% [5]. Pair Programming suggests grouping a student with a peer, employing the "two heads are better than one" philosophy shown to improve the output of projects [6] and perhaps learning outcomes [2]. Students placed in teams may also gain the benefits of peer programming, while also providing more authentic industry working conditions and supporting ABET student outcome (d), working in multidisciplinary teams [7]. This paper looks at how using teams in the Bauhaus studio model impacts student outcomes within a programming-centric Honor First Year Engineering (HFYE) course at a large Midwestern research University. We will start by looking at how teams are formulated and move into the general theoretical frameworks behind peer learning. Based on these frameworks we propose two research questions on how well the teaming paradigm supports student learning.

Curricular and Team Construction

Engineers with programming skills can be radically more productive than their counterparts. The ability to automate mundane calculations and create simulations and models can both open up the creative process as well as accelerating daily work [8], [9]. Most engineers will not have the time to dedicate to fully learning to code, but basic concepts of Computational Thinking (CT) [10] as well as Computer Science (CS) can be taught to engineers alongside Engineering Design, teamwork and problem solving challenges. This is a core pedagogy driver behind the HFYE course of the study. Students are taught programming and given course credit in Computer Science while being introduced to Engineering practices and processes taught in many first year curricula.

The instructional design team structured the entire HFYE course around the Bauhaus Studio Model, creating a classroom that is active, team-driven and engaging. The Studio model formulates learning through the use of team driven projects [11], [12]. Part of the original Bauhaus concept was to bring together craftsman and designers from multiple disciplines to share ideas and to work collaboratively. The Bauhaus model translates well into HFYE as students are destined for all Engineering disciplines, bringing a wide variety of prior learning. HFYE's course objectives look beyond specific domain knowledge instead to fundamental engineering skills such as teamwork, problem solving and communication. How the Bauhaus model builds approaches coursework naturally facilitates these skills.

Students learn best in a team when the team is carefully constructed. The best learning teams can be formulated intentionally using empirical data collected from the students. Teams should be balanced across a variety of factor such as gender and diversity, but should also include more strategic factors [13]. The teams for this study were formulated using the CATME tool used by Layton et al., using further customized criteria. One of the factors Layton et al. notes as important is student scheduling. Scheduling is deemed to be less of an issue in HFYE as, being members of the Honors College, students literally live and attend classes under the same roof and generally have a common schedule. Instead, teams are formulated around prior programming experience first and then balanced for diversity and other demographic factors.

Teams matched around skill level are predicted to both balance capabilities and improve learning. The course is fundamentally an engineering course that solves problems using coding, not a coding course. The first goal is to teach and practice the engineering design process. Projects are designed to stretch students, so by planting at least one team member who has signified they have strong prior programming experience the team should be more able to successfully tackle complex problems. This chosen student, which we will call the "ringer", is the first placed on each team. The team as a whole is both expected to contribute to the design or projects and the coding, but the ringer may best be positioned to implement the "tricky parts" the most advanced code. This team formulation allows more challenging design projects in general but, as we will discuss in the next section, should foster greater learning across the entire team.

Team Learning Theoretical Frameworks

Peer learning is a widely used in general education but strongly advocated in several programming pedagogies. We will touch on three methodologies including the Bauhaus Studio Model, Pair Programming, Peer-led team learning (PLTL). These three frameworks provide insight how the HFYE pedagogy was designed, how it can help students learn and potential improvements for future consideration.

The Bauhaus philosophy is rooted in the experiential learning of John Dewey [14], looking to challenge and motivate students through active learning. Not all engineering students may consider programming to be an essential skill, and thus may not be intrinsically motivated to dedicate time to such a finicky and abstract skill. Engineering and programming share common design skills, so programming challenges can be wrapped in engineering design problems challenging students to learn both simultaneously. Learning to code, though perhaps distasteful for some "lies in the direction of the agent's own growth, and is, therefore, imperiously demanded, if the agent is to be himself." [15, p. 13] Dewey is making the point that some facts are perhaps not attractive to learn on their own, but in this case by framing the skill of coding within the larger context of engineering design, students learn *both* design *and* coding while participating in an active learning team on a realistic project. The studio model provides a way of engendering intrinsic motivation for programming within the context of engineering work.

Beyond being motivated to learn coding, students need support in learning a programming language. An interesting problem does not guarantee a student fully engages in coding when they hit the steep learning curve comes with a programming language. Pair Programming creates partners who work collectively to overcome hurdles in coding. Industry uses Pair Programming to improve code quality [6] while in the classroom it is found to improve satisfaction and motivation of students [2]. Within HYFE, true Pair Programming activities are used occasionally in the classroom and the model is extended to the team as a whole working with 3-4 students. By seeding each team with a ringer, novice programmers are not only supported by the instructional team, but also by a "more knowledgeable other" [1] inside each team. According to Vygotsky's theory, individuals learn best within their Zone of Proximal Development, so teammates are more likely to be successful when their actual skill levels are compatible [16], [17]. For this reason, teams are constructed so the ringer's prior experience does not too far exceed those of their peers. Pair Programming is being deployed in HFYE to teams, using Vygotsky as a guide toward careful formulation of teams by their skill levels.

Peer-Led Team Learning provides students with a team working environment in which students work together under the guide of a facilitator. The core of most PLTL experiences includes 1.) small teams which meet regularly, 2.) tie-ins to course materials 3.) a trained facilitator for each team, 4.) appropriate and challenging problems and 5.) a proper space to facilitate group discussion [18]. The facilitator is not required to be a subject matter expert (some of the best facilitator are not!) and is not given answers to any of the challenges, but is to encourage the team and guide towards learning. The teams act as a network of support in

collaborative learning. Students participating in PLTL activities are shown to perform better than their peers [19], [20]. PLTL was not used as a foundational framework for formulating HFYE, but in includes a source for future enhancements to be described later.

Research Questions

The theoretical frameworks for team learning have demonstrated improved student experience, but not always learning outcomes. There is little or no literature on how the Bauhaus model impacts learning, and while pair programming can be shown to improve retention and performance on individual assignments, evidence shows no impact on each individual's long-term learning [21], [22]. Retention of students is certainly valuable, but HFYE is a blended course without this core retention issue. For us it is important to understand how team formation impacts learn CT and CS concepts as well as overall grades. To better understand the student experiences and outcomes, we are looking to answer the following questions:

1. How does the relative skill level of the "ringer" impact the team's learning?
2. Are there any common characteristics of teams which include a lower performing student?

Methods

We have conducted a retrospective corollary study on student outcomes based on the grade received. The pedagogical design of the class was created using empirically researched practices, as well as considerations facilitating educational research, but was not designed specifically for this study. This section will outline the nature of the participants, the available data, the general approach of our analysis, as well as limitations of this methodology.

Research Participants

The participants in this study are from a Freshman Honors First Year Engineering class at a large Midwestern Research University between 2013 and 2015 (three terms total). The course focuses on early concepts of engineering and design. The Honors sections grant Computer Science credit, adding on extensive programming activities.

Students are assigned into teams of four and complete in-class homework and project challenges with their team. Teams are assigned using a survey (discussed later) in order to balance out multiple individual characteristics such as gender mix and self-reported efficacy and prior learning. The exact 'formula' by which the team assignments are made varies slightly in year, but generally uses the same categories of data later discussed in Table 1. The methodology for forming team attempts to pick a 'ringer' for each team, based on self-reported self-efficacy in programming. The ringer is chosen based on the reported programming skills, but is balanced across the demographic factors mentioned earlier as well as ensuring a balance of experienced, somewhat experienced and novice programmers. The formula for calculating the ringer was generated by one of the instructional staff and contains the following:

Self-Efficacy = {General Programing Rating}/ 2 +*

*2/7 x Average {(Sequencing**), (Conditionals and loops**), (Complex problems**)}*

*Ranked on a sale of 1-4, 1 low and 4 high ability
** Ranked on a scale of 1-7, 1 low and 7 high ability

Students rank themselves in their perceived *General Programming Rating* as well as specific categories of programming including the ability to code *Sequences*, *Conditionals and Loops* and generally *Complex Problems*. The final ranking is based on the formula above, which is primarily used to choose the ringers.

Research Data

This study leverages two sources of data: a background survey given the first week of class and the student's grades (quizzes, homework and exams). The survey is a tool to assign teams including information shown in Table 1. All data is self-reported and thus perception-based, with some students not completing the survey or providing a full response. In some cases statistical analysis must excludes students/teams that do not have full survey data as it would be impossible to categorize their prior knowledge. This brings variability to some team data as the team may have been formulated with a student who has high self-efficacy and prior learning, but was not chosen as a ringer by the CATME tool. We chose to eliminate many teams from the analysis that included members with incomplete data to ensure a clearer picture at the cost of a lower team count.

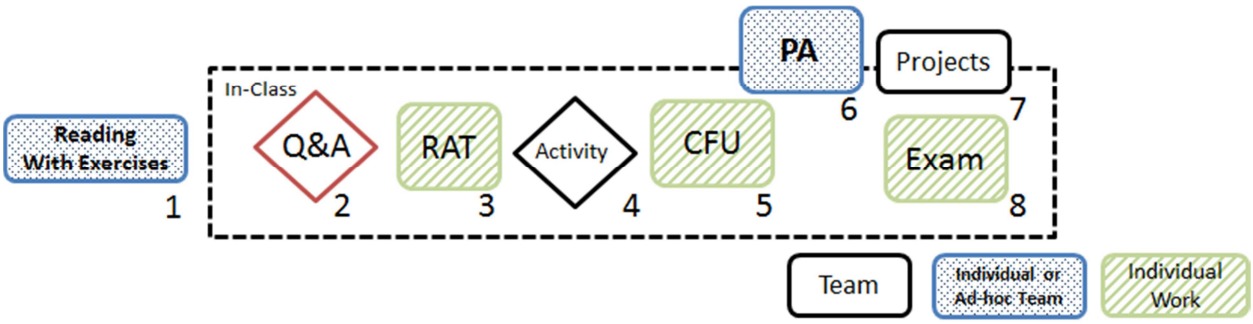
Table 1 Background survey categories

Survey Category	Description
Expected Outcomes	Anticipated GPA, Grade in class
Workload	Number of hours spent on this and other classes as well as paid workload
Programming skills	A self-ranking of general programming skills (the ability to program sequences, decisions, loops, etc)
Specific Languages	The number of courses and their self-rating in languages such as Basic, C, Python, Java, Ruby, Swift, Matlab…)
Non-coding Learning	Prior coursework and grades in Math, Physics, Chemistry, as well as specialized topics like Statistics, Drafting, Manufacturing…
Team experiences	Whether the student has been asked to work in a team, of what size and nature and how they perceive that experience.

Student outcomes include robust data set in the form of exams, in-class assignments and homework. This study is focusing on Computational Thinking aspects of this class, thus all reported grades are filtered to assignments that reflect CT and/or CS topics, unless otherwise stated. An example of topics omitted include questions about the general engineering design process, aspects of teamwork, or work that is not deemed to use or be a direct precursor to CT concepts (e.g. statistics). The pedagogical approach used a semi-flipped classroom wherein

students are expected to engage in the materials and come to class prepared. The typical sequence of assessment is shown in Figure 1 and as follows.

Figure 1 Pedagogical overview of HFYE



1. Reading – The course is supported by an online textbook which includes programming exercises. Problems are assigned from the text book weekly.
2. Q&A – Each class starts with a question and answer session based on the readings to focus the class session.
3. Readiness Assessment Test (RAT) - Students take this initial quiz to assess their self-guided learning and set expectations for the class session.
4. In-class Activities – The team is challenged to complete in-class activities on programming, engineering, technical, and professional skills.
5. Check for Understanding (CFU) –The last 20 minutes of class are used for students to work individually on problem targeting the main topic of the day's activities. This performance task involves generating and submitting individualized code, but the grade may sometimes be assessed as a team rather using the team high grade, low grade, or average as well as individual scores.
6. Post Activities (PA) – Each individual is expected to complete their own program, but is allowed to leverage their team or other resources to assist.
7. Projects – These are completed by the team and cannot reflect individual learning and thus are not considered in this study.
8. Exams – The course consists of two midterm exams and one final exam. The exam asks questions on a variety of topics, but as stated, only those containing CT or CS concepts are included here. The included questions include both conceptual questions as well as practical programming work.

The order presented here is typically the order students would experience the assessment. The RAT provides a 'pretest' of learning, the CFU drives further understand using peer instruction, which is practiced as part of the PAs. After several cycles of RAT/CFU/PA a test is given to provide a summative assessment before transitioning to a new topic area. Since we have multiple assessments around pedagogical interventions, we can see some impact of each of the different approaches.

Data Analysis and Limitations

The course objectives are not limited to CT/CS topics, so exclusions were taken from the data. Exercises and exam question not related to CT/CS concepts are removed from analysis. Thus when we talk about 'failing' or 'low performing' students, this may not mean they failed the class, but simply that they are falling behind their peers on the CT/CS ideas being explored in this paper. It is possible that a student in the D/F category does well overall the class! It is improbable, but possible a student performing well in CS/CT concepts failed the class. Our objective here is conceptual understanding in CT/CS alone, thus we are not tracking actual grades or withdrawals. The count of students/teams in the data may be less than the full registration, as we only included students will a full comparable data set in the analysis.

The research questions guided our initial look at data analysis, but we understood there was an opportunity for data mining for additional insights so a software platform was built to facilitate specific data mining efforts. Given that the course varies in content and approach each year, the software framework aids in normalizing the data to a common model for data analysis. This common model can then be explored using specific extracts of data that can be exported or reported upon directly. This technique allows for quick access to structured data, enabling analysis of emergent questions and insights. The dynamic nature of the data analysis can present limitations. Specifically mining within a single data set may lead to external validity issues if the pattern only applies to a given cohort. We curbed this risk by limiting data mining activities to a single cohort year (2015) and then analyzed other years to see if the pattern is confirmable.

The nature of the students may have limitations on the generalizability of our findings. Being accepted to the Honors College, this cohort is already cultivated from general college bound population. This study might be looking at 'the best of the best' or at least the "most motivated" amongst college students. It is possible that the data set is already looking at such a limited ban of student profiles, that the benefits or lack thereof seen in the data would not appear, or have a greater impact in a different population. As an example, perhaps teamwork is more effective here as most students in HFYE have prior teaming experience (100% of students in 2015). Or perhaps the benefits of peer learning underrepresented as all of our students have shown academic success in Math (92% report a 4.0 in 2015). We will attempt to consider some of these options as part of the discussion, though the generalizability of the data may be limited and require further studies to show greater range of impact.

Results

Overall Data Context

The amount of data and resulting statistics is vast, so to aid in context we will start with an overview of the students and teams. The data set does not contain specific demographics (age, gender, race, etc.) so we cannot report past describing "general college students in HFYE". Table 2 shows descriptive statistics describing the nature of the team and student performance.

The overall self-efficacy of each class (Row 3) varies year-by-year hovering just above the lowest level (1) of no professed skill. The ringer's average self-efficacy (Row 4) is statistically stable across years with the ringers on average are close to a 3 on the 4 point scale. The formula for deriving teams seems to keep the 'self-efficacy gap' consistent across years (Row 5).

The team performance varies across the years, but this does not impact our analysis. For instance, the teams in 2014 performed significantly lower than those in 2013 and 2015 (rows 6 and 7). The ringer's scores also vary across years (rows 8 and 9) as does the gap between the ringer's score and that of their teammates on average (row 10). We are not comparing years but trends within a year to see if the trend is repeated across years, so the variance of raw scores across the years does not impact our analysis. Overall the number of students receiving 55% or less (which would be failing the course) in CT/CS assessments is very low (row 11), and in line with the fail rates for the course in general, but not aligning CT/CS failure to course failure.

Table 2 Descriptive statistics by year

		2013	2014	2015	Cross-Year ANOVA
1	N (Teams/Students)	66/263	50/206	65 272	-
2	Worked in Teams (All/Ringers)	97%(100%)	98% (98%)	95% (97%)	-
3	Average Self-Efficacy (SE)	1.62 (σ=0.26)	1.55 (σ=0.22)	1.69 (σ=0.23)	**p=0.011**
4	SE Ringer	2.75 (σ=0.68)	2.68 (σ=0.66)	2.86 (σ=0.56)	p=0.34
5	Ringer SE Delta	1.54 (σ=0.65)	1.53 (σ=0.72)	1.6 (σ=0.62)	p=0.81
6	Team Percentage*	76.7% (σ=0.04)	66.5% (σ=0.04)	74.1% (σ=0.05)	**p=0.0**
7	Team Score*	418.5 (σ=23.8)	381 (σ=18.6)	399.5 (σ=26.6)	-
8	Ringer Percent*	78.1% (σ=0.05)	68.7% (σ=0.06)	78.9% (σ=0.07)	**p=0.0**
9	Ringer Score*	427.6 (σ=30.0)	395.9 (σ=34.5)	426.5 (σ=34.6)	-
10	Ringer Score Delta	10.4 (σ=30.75)	20.01 (σ=33.42)	37.1 (σ=42.6)	**p=0.0**
11	Grade of D/F**	2.3%	3%	2.3%	**p=0.004**

*Scores are for CT/CS topics only, not full course grades

** D/F grade is in CT/CS scores only, not actual DWF

Relationships between the ringer and the team

The first research question investigates the relationship between the ringer and the performance of their team. At the macro level there is no strong linear regression correlation between the ringer's self-efficacy as shown in Table 3 (row 1). The belief in a ringer's ability does not seem to translate into improved performance. In two of the years, however, the ringer's average score shows a statistically significant positive relationship to the average score of the rest of the team (row 2). When the ringer scores better, so does the rest of the team. For instance in 2013 for each 50 points the ringer score improved, the rest of the team averaged 15.6 points better. Neither the ringer self-efficacy nor their score had any significant effect on the deviation of the scores within the team. The ringer may help the rest of the team's average, but does not seem to reduce the variation of scores within the rest of the team members (rows 3 and 4).

Table 3 Ringer relationship to the team's performance

		2013		2014		2015	
	Relationship	R^2	p	R^2	p	R^2	p
1	Ringer SE to Team Average Score*	-	0.45	-	0.89	-	0.997
2	Ringer Score to Team Average Score*	0.15	**0.001**	0.13	**0.009**	0.03	0.14
3	Ringer SE to Team Score Deviation*	-	0.76	-	0.8	-	0.88
4	Ringer Score to Team Score Deviation*	-	0.47	-	0.64	-	0.69
5	Ringer Score Gap to Team Average*	0.18	**0.000**	0.08	**0.04**	0.36	**0.000**

* Not including the Ringer's score

Teams with low scoring members

The next research question seeks characteristics of teams which includes a member who falls behind in CT/CS assessments (no team had more than one such student in any year). For this analysis the teams who included a low scoring member were compared to the rest using ANOVA to look for statistically significant differences. Neither the ringer, nor the team's average self-efficacy show any variance (Table 4 rows 1 and 2). The initial perception of programming ability does not seem to indicate coming struggles. Teams containing a failing member have a lower average than their counterparts (row 4) and the deviation of scores within the team (row 6) is larger. Yet having a failing member does not seem to significantly hurt the performance (row 5) or the deviation (row 7) of the remainder of the non-failing team members.

Table 4 Analysis of teams with D/F members

		2013		2014		2015	
	Relationship	f	p	f	p	f	p
1	Ringer SE by D/F occurrence	-	0.16	-	0.62	-	0.95
2	Team SE by D/F occurrence	-	0.24	-	0.77	-	0.95
3	Ringer Score by D/F occurrence	-	0.77	7.27	**0.01***	-	0.51
4	Full Team Average by D/F occurrences	5.46	**0.02**	45.6	**0.000**	22.5	**0.000**
5	Rest of Team Average by D/F occurrence	-	0.94	9.13	**0.004***	-	0.42
6	Full Team Deviation by D/F occurrence	35.2	**0.000**	26.4	**0.000**	54.8	**0.000**
7	Rest of Team Deviation by D/F occurrence	-	0.93	-	0.62	-	0.41

*One 2014 ringer was also D/F, likely skewing this statistic

Discussion

The theoretical basis of how teams are formulated in HFYE seems supported by the results of the analysis. The data cannot compare student performance in team versus non-team settings, but the dramatically low rates of failure (Table 2 row 11) and the general improvement of learning outcomes (Table 3 row 2) suggest the teaming does no harm yet puts student in more authentic working environments which also supports ABET student outcome (d). Teamwork is typically already common for most students and is not a statistical predictor of performance or D/F rates. In fact, only one student who scored D/F in all three years (n=741) also reported having never worked on a team. Our analysis is not comparing the breadth or depth of learning,

but does seem to show that for the materials and approach chosen teaming does not show harm and may show great benefit.

Research Question 1: Ringer Impact

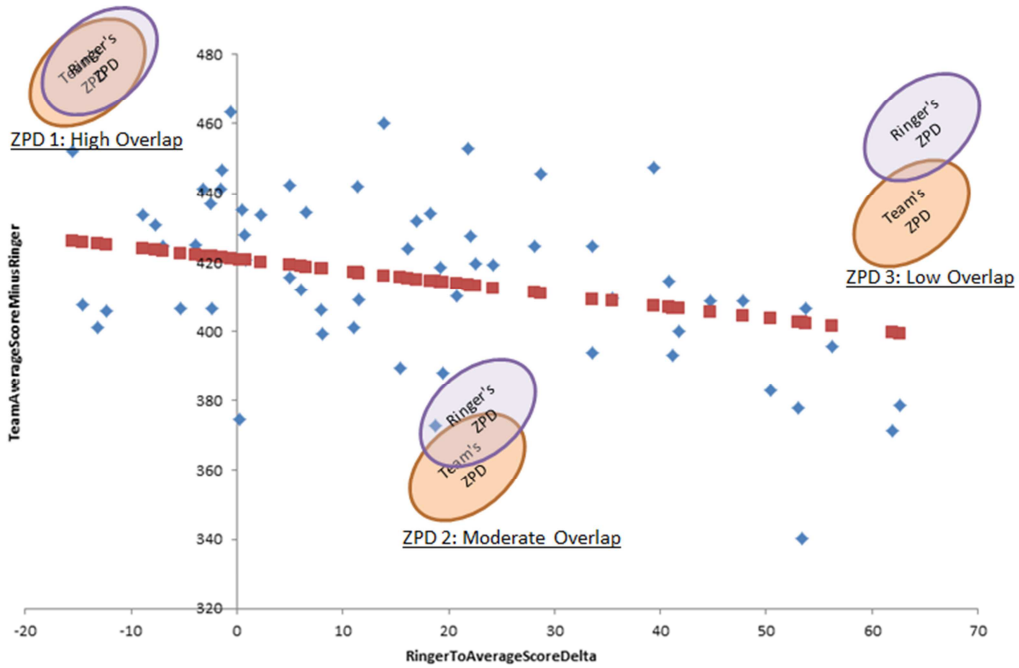
The method by which ringers are selected and teams are formed seems to be beneficial. The self-efficacy reported by the ringer is not perhaps the best indicator of actual team performance (Table 3 row 1) but when the ringer does perform well in the class the team seems to benefit as well (Table 3 row 2), if only at a fraction of the gains of the ringer. We cannot pinpoint the root cause of this improvement, but looking at Vygotsky's theory of the "more knowledgeable other" (MKO), it suggests the ringer may be acting in this capacity. Learning to program involves both general problem solving as well as very specific understanding of syntax and logic. The ringer may support the entire team as they unpack the engineering based problems of the course and then stages of the programming task. The data suggests the ringer may be acting as a MKO to their peers, but not without limitations.

Neither the ringer's self-efficacy nor their actual score seems to impact the variation of the team. One hope of using teams rather than pairs is that the ringer could stabilize the variation of learning across three peers. It would be uncommon for half of any students to possess significant prior CT/CS experience, so using teams of four not only creates a natural team size, but also requires half as many ringers. The skill of the ringer does not show reduction in the variability of the scores however (Table 3 rows 3 and 4). This may be due to what Vygotsky suggests is the "zone of proximal development" (ZPD) within which a student is capable of comprehending new material. For instance, a student may be struggling to comprehend loops, where the ringer has long mastered that skill and forgotten what it was like to struggle with loops. If a student's and the ringer's ZPD do not overlap enough, then the student will be missing out on the benefits of working with an MKO as the ringer cannot relate. On the flip side, if the team's ZPDs overlap too much, the ringer can only stretch the team so far. Vygotsky might suggest that a team must be formed so that the ZPD of each member overlaps that of at least some of the other members.

Using the ZPD model, teams must be carefully matched in order to gain the maximum success. We can see reflections of this in the data by looking in more detail at the gap between the ringer's score and the team average compared against the team's score average not including the ringer (Table 3 row 5). In all years there is a significant negative correlation. Figure 2 shows 2013 data plotted where the x axis shows the gap between the ringer score and the team average (for negative values, the ringer's score was lower than the team; the more positive the value the greater the gap between the ringer and the team) and the y axis shows the team's average score excluding the ringer's score. The rest of the team loses 15 points for each 50 points the gap grows, supporting the idea that gap in ZPD between the ringer and the team. An overlapping in ZPD (Shown as ZPD1 in Figure 2) means the ringer scores the same or lower as the rest of the team. When the ZPD is moderate (ZPD 2 in Figure 2) the ringer does slightly better while still supporting the team, but as the ZPD grows so does the gap (ZPD 3 in Figure 2) and the rest of

the team's average drops. The team's score might grow with the ringer's ability, as suggested in Table 3 row 2, but that growth would likely tail off as the gap between ZPDs grows too large. This brings up interesting questions and suggestions covered in the conclusion.

Figure 2 ZPD analysis for ringers

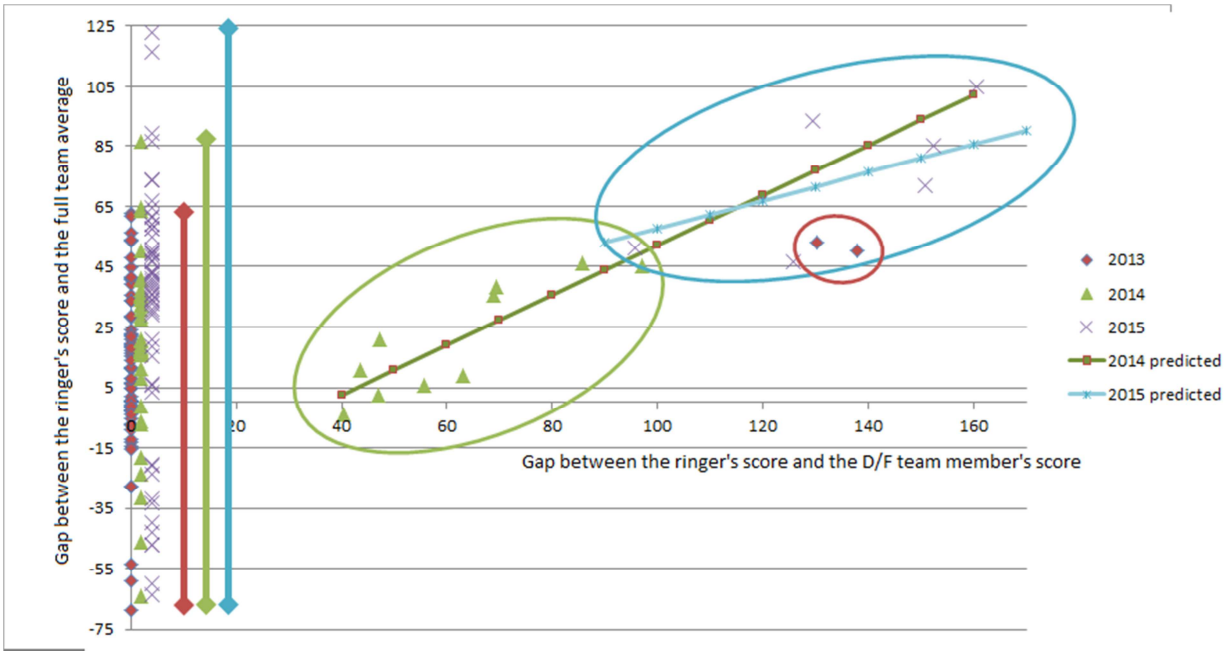


Research Question 2: D/F Teams

The ringer does not seem to impact the student fail rate. When a member of the team falls behind, the average and deviation is noticeable (Table 4 rows 4 and 6) but does not seem to be impacting the rest of the team (Table 4 rows 5 and 7). The rest of the team's perceived or actual skill does not seem to help or hurt students receiving D/F grades in CT. The exception in the statistics is for 2014, which seems to be caused by the failure of a ringer, thus skewing the significance. The very low fail rate (n=18 across 3 years) may make it difficult to generalize if the constitution of the team impacts the fail rate from this study though.

Vygotsky's theory would imply that mismatch within the respective ZPDs could lead to the ringer being an unsuccessful MKO for the D/F students. If the ringer's skill level is too low, they may be too consumed with their own learning, while if their skill is too high they may not be able to 'look back' at the basic concepts in order to help. We can look for a potential gap/overlap by comparing the gap between the ringer's score and the D/F student against the ringer's score and the full team's average as shown in Figure 3. The three bars on the left represent the range of the gaps for each year between the ringer and the rest of the team. Teams which do not include a failing member are graphed at the X value of 0. All other values show the size of the gap between the ringer and the failing team member.

Figure 3 Analysis of the ZPD between the ringer and D/F students



The teams which contain a D/F student all contain a ringer which is more likely to be higher in score than their team. Only one team contains a ringer who scored lower than the team average, and the majority scored much better. Thus we can reject the idea that any team included a ringer unable to provide some level of expertise in CT/CS concepts. It would seem nobody failed from the lack of an MKO on the team. It seems more possible that some of the ringers were unable to 'reach back' to help the D/F students due to an insufficient overlap of ZPDs. The trend shows that the gap between the ringer and the D/F student grows faster than the gap between the ringer and the rest team as a whole. In 2014 a ringer who scored 20 points higher than the D/F team member would only have scored 16.6 points higher than the rest of the team and in 2015 it would only be 9.4! This seems to confirm the theory that when the ringer's ZPD does not overlap with one of the teammates they are less able to help them.

Conclusion

Research Findings

The use of teams organized around prior experience in programming seems to have a positive impact on learning outcomes. We were able to look back at three terms of student survey and outcome data to test theoretical approaches and mine for data to support the research questions. Our first research question looked at the impact of including a ringer on each team. The inclusion of the ringer seems to have a mild to moderately positive impact on the ability to program, in that ringers with a higher score tend to lead to teams that score higher as well.

The second research question looked for any trends in teams including a member with lower scores. One concern could be a 'weaker' ringer could harm a student with less prior knowledge, but in fact we seem to show that students who scored lower in CT/CS assessments were among teams of better scoring ringers. It is also important to note, that individuals are not seemingly harmed when they are placed on teams with individuals who do not succeed. We have no way of discerning the causes of low performance, but also did not find any trends in teams either causing lower score or impacting the rest of the team's scores.

Implications and Future Research

The Bauhaus studio model can be successfully deployed to teach engineering *and* programming at the first year level. A ringer can act as a more knowledgeable other and we can see benefits within learning outcomes that are not always apparent in pair programming studies. These finding must be taken with the caveat that the nature of this classroom is unique, in that all of the students are members of the Honors College and thus likely already come in with better prior success in school and potentially are more motivated. That being said there are some recommendations which could be used to enhance the teaming model even further.

The use of self-reported efficiency alone is not the most reliable measure. While generally the ringers do perform well, there are cases where their self-efficacy is overstated, or they simply do not live up to the predicted score. It would seem that a tradeoff must be made between forming the teams quickly versus getting an accurate assessment of who are the best ringers. Perhaps the inclusion of a pre-test in addition to or instead of a self-report survey could better identify possible ringers. By either delaying the team formation, or getting proven performance metrics for selecting ringers, teams can be better balanced and maximize the diversity of skills.

One oversight may be a trust that the team members understand how to learn as a team. At no time in HFYE are the ringers explicitly identified as such, though the team is given training on how to act as a team as part of the course content. Looking at the literature from the Peer-led team learning methodology, the team can be formed with "a leader who serves as a facilitator, but not as a content expert" [23, p. 1440]. By training teams specifically in how to work collaboratively and ensure all team members are learning at the same pace, the facilitator could be a team member at any level of prior learning, and ringers are not expected to always be the expert. This is not to say that the HFYE would ever exactly implement a PLTL methodology, but certainly the principles of facilitated team problem solving could be presented to form a community of learning where each member is expected to grow and participate in collaborative work.

The findings in this study hint at possible future studies. Within this HFYE curriculum the instructional team can now track changes in pedagogy and approach against a baseline of data to see how new interventions impact student learning. For instructors not using a studio model of learning, they could baseline their student performance data and perhaps begin to

introduce elements of the studio model, Peer Programming or even PLTL to see how it impacts student perceptions and most important, learning outcomes. There is a wealth of literature to show benefits to the retention and outlook of students who engage with their peers as part of Pair Programming but less evidence of learning outcomes. The research on PLTL shows promising data, but many of the studies are potentially limited because they primarily engage self-selecting students in extra-curricular study groups, and thus may include be selecting motivated students who rather than simply effects of PLTL. One of the additional hopes of this paper is to inspire empirical research, perhaps even retrospective views of courses such as this study, to further the understanding of peer learning in the classroom.

References

[1] L. Vygotsky, *Thought and language*. Cambridge, MA: MIT Press, 1962.

[2] N. Salleh, E. Mendes, and J. Grundy, "Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review," *IEEE Trans. Softw. Eng.*, vol. 37, no. 4, pp. 509–525, 2011.

[3] L. Porter, C. Bailey-Lee, and B. Simon, "Halving Fail Rates using Peer Instruction : A Study of Four Computer Science Courses," *Proceeding 44th Tech. Symp. Comput. Sci. Educ. (SIGCSE '13)*, pp. 177–182, 2013.

[4] D. Teague, "Neo-Piagetian theory and the novice programmer," 2014.

[5] A. B. Woszczynski and C. Guthr, "Per sonality and Pr ogr amming," vol. 16, no. 3, pp. 293–300, 2000.

[6] A. Cockburn and L. Williams, "The Costs and Benefits of Pair Programming," *Extrem. Program. Examined*, pp. 223–243, 2001.

[7] ABEEK, "Criteria for Accrediting Engineering Program," *Accredit. Board Eng. Educ. Korea*, pp. 1–8, 2005.

[8] W. J. Rasdorf, "Computer Programming in the Civil Engineering Curriculum," *J. Prof. Issues Eng.*, vol. 111, no. 4, pp. 141–148, 1985.

[9] P. Guo, "Why scientists and engineers must learn programming," *Communications of the ACM*, 2013. [Online]. Available: http://cacm.acm.org/blogs/blog-cacm/166115-why-scientists-and-engineers-must-learn-programming/fulltext. [Accessed: 02-Jul-2017].

[10] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[11] K. Lynch, A. Carbone, D. Arnott, and P. Jamieson, "Studio-Based Approach to Teaching Information Technology," vol. 8, no. April 2013, pp. 75–79, 2002.

[12] M. Woodley and S. N. Kamin, "Programming Studio : A Course for Improving Programming Skills in Undergraduates," 1971.

[13] R. A. Layton, M. L. Loughry, M. W. Ohland, and G. D. Ricco, "Design and validation of

a web-based system for assigning members to teams using instructor-specified criteria," *Adv. Eng. Educ.*, vol. 2, no. 1, pp. 1–28, 2010.

[14] J. Dewey, *Experience & education*. New York, NY: Touchstone, 1938.

[15] J. Dewey, *Interest and effort in education*. Houghton Mifflin, 1913.

[16] N. Katira, L. Williams, and J. Osborne, "Towards increasing the compatibility of student pair programmers," *Proceedings. 27th Int. Conf. Softw. Eng. 2005. ICSE 2005.*, pp. 3–4, 2005.

[17] L. Williams, L. Layman, J. Osborne, and N. Katira, "Examining the compatibility of student pair programmers," *Proc. - Agil. Conf. 2006*, vol. 2006, pp. 411–420, 2006.

[18] D. K. Gosser, "The Journal of Peer-led Team Learning," vol. 14, no. 1, 2011.

[19] M. C. Loui, B. A. Robbins, S. D. Johnson, and N. Venkatesan, "Assessment of Peer-Led Team Learning in an Engineering Course for Freshman," *Int. J. Eng. Educ.*, vol. 29, no. 6, pp. 1440–1455, 2013.

[20] J. R. Reisel, M. R. Jablonski, E. Munson, and H. Hosseini, "Peer-led team learning in mathematics courses for freshmen engineering and computer science students," *J. STEM Educ.*, vol. 15, no. 2, pp. 7–16, 2014.

[21] L. Madeyski, "Is external code quality correlated with programming experience or feelgood factor?," *7th Int. Conf. Extrem. Program. Agil. Process. Softw. Eng. (XP 2006)*, vol. 4044 LNCS, pp. 65–74, 2006.

[22] C. McDowell, L. Werner, H. Bullock, and J. Fernald, "The effects of pair-programming on performance in an introductory programming course," *ACM SIGCSE Bull.*, vol. 34, no. 1, p. 38, 2002.

[23] M. C. Loui and B. A. Robbins, "Work-in-progress - Assessment of peer-led team learning in an engineering course for freshmen," *Proc. - Front. Educ. Conf. FIE*, vol. 29, no. 6, pp. 1440–1455, 2008.