Nurturing Collaboration in an Undergraduate Computing Course with Robot-themed Team Training and Team Building

Michael James Scott  
Games Academy  
Falmouth University  
Cornwall, UK  
michael.scott@falmouth.ac.uk

Alcwyn Parker  
Games Academy  
Falmouth University  
Cornwall, UK  
alcywn.parker@falmouth.ac.uk

Brian McDonald  
Games Academy  
Falmouth University  
Cornwall, UK  
brian.mcdonald@falmouth.ac.uk

Gareth Lewis  
Games Academy  
Falmouth University  
Cornwall, UK  
gareth.lewis@falmouth.ac.uk

Edward J. Powley  
Games Academy  
Falmouth University  
Cornwall, UK  
edward.powley@falmouth.ac.uk

ABSTRACT

Group projects are a common feature of undergraduate degree programmes in computing. Early and sustained collaboration helps students to strive beyond introductory programming towards professional software development. However, during their first year of study, students can find teamwork challenging. To equip learners with the foundational knowledge, skills, and experience that they need to collaborate effectively so early in their studies, a 3-day Robot Olympics using Lego Mindstorms EV3 robots can be deployed. The exercise draws upon Salas’ big-five model of teamwork, making first-year students aware of coordinating mechanisms that aid in clarifying expectations and managing conflicts. These then act as lenses for reflection and feedback. Comparing a baseline cohort in 2015-16 to a cohort in 2016-17, after the introduction of the Robot Olympics, reveals a statistically significant reduction in team discord in an assessed collaborative programming project ($d = 0.76$). This suggests that the Robot Olympics made a positive contribution to the design of the first computing module. Notably, helping students to enact and reflect upon their group work and related employability skills.

CCS CONCEPTS

• Software and its engineering → Programming teams; • Social and professional topics → Software engineering education;

KEYWORDS

Collaboration, Group Projects, Teamwork, Programming, Robots

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.

CEP '19, January 9, 2019, Durham, United Kingdom
© 2019 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
ACM ISBN 978-1-4503-6631-1/19/01...
https://doi.org/10.1145/3294016.3294019

1 INTRODUCTION

Undergraduate students often experience discord when confronted with the complexities of collaboration, particularly when managing a group software development project for the first time. Obstacles include: coordinating times to meet; tracking task progress and blockers; communicating effectively; resolving conflicts; and collaborating effectively in the face of (sometimes, wildly) differing abilities [14, 16]. This not only influences satisfaction [10], but can also widen experience gaps as pressure to perform influences task allocation in ways that may deprive some group members from learning opportunities [3].

Despite these obstacles, group projects are a prominent feature of computing education. A 1997 survey of higher education institutions in the United Kingdom (UK) revealed that group projects were a key feature of provision across the sector [19]. Wiggberg also notes that “universities in the Western world largely organise computer science education in such a way that group work is an integral part of the students’ education” [37, p. 21].

Such prominence is unsurprising. Employers cite that “soft skills” are critical [2, 26]. Talent supply is as an ongoing concern in the UK, with 25% of hubs describing hiring as a “major challenge” [33]. Perhaps driven by expectations that those entering the job market should be able to “hit the ground running” [4].

This is spurring efforts to bridge work and pedagogic practice to ensure that learners develop employability skills [27]. However, educators argue that students are unlikely to master teamwork through mere ad-hoc experience [21] and so structured exercises are necessary [35]. These include: team training, team building, and coaching. However, little work examines these exercises in the context of first-year computing. This paper aims to address this gap by evaluating a robot-themed team building and training exercise deployed early into an undergraduate computing course.
2 BACKGROUND

A key motivation for embedding group projects into education is situated cognition. This is the notion that knowledge is bound to the activity and context in which it is learned [5, 6]. In other words, to learn practice that is meaningful to future employers, it should represent something that might be encountered during employment. Thus, projects aim to mirror the world of work—in some cases, doing so from the outset of a course [32].

The strategy also forms part of a wider trend in higher education (particularly, in software engineering) to develop and sustain learning communities [11]. Cooperative learning has long been considered effective [15]. It confers benefits such as knowledge sharing and mutual support, possessing synergies with other best practices in computing education (e.g., [25]).

Such approaches are also informed through insights into communities of practice. Notably, cognitive apprenticeship [7] and legitimate peripheral participation [18]. On the one hand, that modelling and articulating tacit knowledge and implicit processes is required to enact a complex skill and then improve it (i.e., observation, practice, and reflection). On the other, that newcomers first participate in low risk but nonetheless productive tasks, increasing in risk and value as they are inducted into the community and eventually work independently.

Towards this end, the design of scaffolding, drawing on the notion of the zone of proximal development [36], is pertinent. Following Hackman’s [12] observation that discordant teams are unable to perform; in the absence of explicit guidance and supervision, students can worsen the conditions that make groups productive and ultimately successful. As such, structure and support when students collaborate for the first time is valuable. Typically, educators already ensure that tasks are designed carefully and groups are kept relatively small [20]. However, teamwork research offers a rich array of constructs [23] that educators should consider when designing scaffolds.

One such set of constructs, forming the basis for the practice described in this paper, is Salas’ Big Five Model of Teamwork [30]. This model strives to condense the facets and relationships that influence team functioning into five key factors and three coordinating mechanisms, as shown in Figure 1. This model is sufficiently general to be extensible into many contexts, and handle many task-specific idiosyncrasies [9, 29].

For example, the model can be integrated into group projects in the first-year computing context through team training and team building. The former advances group members’ knowledge, skills, and attitudes by “building cohesion, managing coordination, and enhancing communication” [p. 563] in a generalisable way [9]. The latter enhances social relations, helping a group to set goals, clarify roles, develop norms for managing interpersonal relations, and practice problem solving. Meta analyses show that both team training [28] and team building [17] enhance team outcomes.

3 CONTEXT

The practice described in this paper is integrated into BSc(Hons) Computing for Games, offered in the Games Academy at Falmouth University in the UK. By the end of their first year, students are expected to work effectively in multidisciplinary groups. Many projects are shared with art and design courses where students, typically in teams of 8-12 (of whom only 2-4 may study computing), collectivly produce digital games. In preparation for multidisciplinary development practice, teamwork is heavily emphasised across the first study block of 15 weeks. Students are enrolled on an introductory programming module in which they solve problems through pair programming. They develop technical communication through exercises in their principles of computing module. They also embark upon a development principles module which covers agile project management, version control, and interpersonal skills.

The students enrolling on the course tend to arrive with a blend of academic and vocational qualifications, typically achieving 104-120 UCAS tariff points. Approximately 43% of students report that they have no formal programming experience, with 50% having a pre-university qualification in computing. Their median age is 19 and there are few women (~3%). The intake has been stable, with no statistically significant differences between the 2015-16 and 2016-17 cohorts in terms of demographic variables such as age or gender, nor any measures of academic ability such as tariff points or prior programming experience.

In 2015-16, the course team observed many bad practices as small teams of computing students completed their first assessed collaborative programming project at the end of the first study block. A peer-review exercise confirmed high levels of discord within each team. This was concerning given the cohort would soon progress into a larger multidisciplinary project. In an attempt to overcome this challenge in 2016-17, the first week of the course was redesigned to incorporate team training and building through a 3-day Robot Olympics. To evaluate success, the module otherwise remained the same and the peer-review exercise was repeated to enable a comparison of team discord across the two cohorts.

See https://www.ucas.com for a more detailed explanation of the tariff points used in the Universities and Colleges Admission System (UCAS)
Discord Metric

These are presented to students as a set of user stories form the space challenge. After which, students are sorted into programming; the robot’s capabilities; and the user stories that space-themed challenges are reviewed at an event on the final day. The setup and training user stories throughout the event, whilst the user stories as possible within the time constraint. Staff sign-off collect the asteroids). Each team is tasked with completing as many sequence of challenges in which group-made robots compete.

Students with no prior programming experience. block-based language used by 
EV3 shows that 75% of studies published prior to 2012 found robots to students build rapport [39]. Furthermore, a systematic review presents motivational affordances that capture curiosity and attention [24]. They have also been shown to be good ice-breakers in studies of transition to higher education, helping students build rapport [39]. Furthermore, a systematic review shows that 75% of studies published prior to 2012 found robots to be effective for teaching programming [22] and the visual block-based language used by EV3 is accessible to even those students with no prior programming experience.

The Lego Mindstorms Space Challenge itself comprises a sequence of challenges in which group-made robots compete. These are presented to students as a set of user stories, progressing from initial setup (e.g., build the base robot), through to simple training missions (e.g. move in a straight line), through to space-themed activities that stretch each team’s abilities (e.g., collect the asteroids). Each team is tasked with completing as many user stories as possible within the time constraint. Staff sign-off the setup and training user stories throughout the event, whilst the space-themed challenges are reviewed at an event on the final day.

There is an initial induction on: team-work strategies (e.g., communication strategies, conflict management, and mob programming); the robot’s capabilities; and the user stories that form the space challenge. After which, students are sorted into teams of three. Staff and student mentors assume a facilitator role.

Whenever a team completed a user story, or encountered a significant challenge, a retrospective was held privately with a supervising member of staff. Teams were observed throughout to highlight poor practice and/or disengagement. All of these opportunities strive to reinforce aspects of the five-factor model as described (on the next page) in Table 1.

5 ANALYSIS

Data was drawn from assessments conducted at the end of the first study block, including the peer-review exercises conducted by both cohorts. Each student rated their own engagement/contribution and that of their fellow team members (using a Likert-style response). The SPA statistic [38] was then calculated. Members of effective teams have SPA scores tending towards 1. Thus, differences from 1 can be used as a measure of discord:

\[
1 - \sqrt{\frac{\text{Total ratings for individual team member}}{\text{Average of total ratings for all team members}}}
\]

An independent t-test examined the hypothesis that those students who participated in a Robot Olympics would report a lower (mean) level of discord in their first assessed collaborative programming project compared to those students who did not have the opportunity to do the Olympics. The results suggest a statistically significant difference (\(p = .04\)). The 2016-17 cohort experienced less discord in their subsequent group project (\(\bar{x} = .164, \sigma = .134\)) compared to the 2015-16 cohort who did not do the Olympics (\(\bar{x} = .289, \sigma = .187\)). The effect size is close to "large" (\(d = 0.76\)) and can be seen more clearly in Figure 2.

6 CONCLUDING REMARKS

Due to the observational and quasi-experimental nature of the way the practice is evaluated, confounding factors cannot be ruled out. Unobserved baseline differences could have influenced the levels of discord experienced by the teams. Nevertheless, the previous cohort experienced higher levels of discord in their group project in the absence of the Robot Olympics. The effect size is close-to-large, greater than Hattie’s "hinge point" (\(d > 0.4\)) [13]. Given the theoretic grounding, no other substantial changes to curriculum or delivery, and no obvious ways in which the two cohorts differed, the difference could be attributed to the new practice.

Leveraging Salas’ five-factor teamwork model (and agile concepts) visibly pushed students through the stages of team development in an intensive way, storming for less time [34]. Explicit reference to such models in discussions with teams seemed to help them to develop back-up behaviours and adapt when faced with challenges. The team building aspect aided the development of mutual trust, shared mental models, and closed-loop communication. Supervision meetings were initially frequent, but became less so as groups normed and performed.

Overall, the Robot Olympics helped students to overcome challenges associated with teamwork. This finding is useful to educators delivering computing courses with an emphasis on software development and which, to this end, position group projects early. However, the unique contribution of the robots remains unclear. As such, future work could explore this.

---

2. https://www.dropbox.com/s/he1g22zkslahpbn/Lego-Mindstorms-EV3.pdf?dl=1

---

Figure 2: Box Plot Comparing Within-Group Discord Reported by Students in the Baseline (2015-16) and Intervention (2016-17) Cohort
Table 1: Mapping of Facets of Salas’ Model [30] to their Realization in the Robot Olympics

<table>
<thead>
<tr>
<th>Facet</th>
<th>Implementation in the Robot Olympics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team Leadership</td>
<td>Assign scrum master (managed retrospectives and stand-up meetings) and product owner (prioritised and coordinated user stories being worked on). Roles rotated each day.</td>
</tr>
<tr>
<td>Performance Monitoring</td>
<td>Review strengths and weaknesses of individual contributions and their relationship to task strategy whenever an event is attempted. Retrospective held irrespective of success or failure.</td>
</tr>
<tr>
<td>Team Orientation</td>
<td>Shared team goal, set by the team and made explicit. Everyone acknowledging to work to this joint team goal, rather than their own goals.</td>
</tr>
<tr>
<td>Back-Up Behavior</td>
<td>Pair programming guidelines provided. Explicit team contract formed with if-then clauses (e.g., for disengagement). Teams identify their key skill areas and acknowledge to use this to inform pairings.</td>
</tr>
<tr>
<td>Adaptability</td>
<td>Pair up or mob to address identified weaknesses during tasks. Match-making of stronger and weaker students during tasks, and actively seeking of help when stuck.</td>
</tr>
<tr>
<td>Closed Loop Communication</td>
<td>Reminders to sit in a pod close to each other and to communicate often. Explicit structure of update reporting using periodic stand-ups.</td>
</tr>
<tr>
<td>Mutual Trust</td>
<td>Building of trust through first-day exercise. Reinforcement in each sprint retrospective to help students feel empowered to step up to tasks and deliver.</td>
</tr>
<tr>
<td>Shared Mental Models</td>
<td>Task board and flip-chart paper provided. Note keeping reinforced. Files kept on single shared computer per team.</td>
</tr>
</tbody>
</table>

REFERENCES


[34] Tucker, B. W. Developmental sequence in small groups. Psychological bulletin 63, 6 (1965), 384.


