Team Projects in Computing Education

JÜRGEN BÖRSTLER, Blekinge Institute of Technology
THOMAS B. HILBURN, Embry-Riddle Aeronautical University

Team projects are a way to expose students to conflicting project objectives and "[t]here should be a strong real-world element ... to ensure that the experience is realistic" (SE2014, [ACM/IEEE 2015b]). Team projects provide students an opportunity to put their education into practice and prepare them for their professional careers. The aim of this special issue is to collect and share evidence about the state-of-practice of team projects in computing education and to help educators in designing and running team projects. From a record number of 69 submitted abstracts, 19 were invited to submit a full paper. Finally, nine papers were accepted for publication in this and a subsequent issue. The papers presented in the present issue cover the following topics: real projects for real clients, open source projects, multidisciplinary team projects, student and team assessment, and cognitive and psychological aspects of team projects.

CCS Concepts: •**Social and professional topics** → **Computing education programs;**

Additional Key Words and Phrases: Team projects

ACM Reference Format:
Jürgen Börstler and Thomas B. Hilburn, 2015. Team Projects in Computing Education. *ACM Trans. Comput. Educ.* 15, 4, Article 1 (December 2015), 5 pages.
DOI: 0000001.0000001

1. INTRODUCTION

Modern software development projects have rapidly moved away from traditional "closed-shop models", where a single collocated team develops, from scratch, a pre-specified piece of software for a known client according to a well-defined process [Shaw 2000; Mead 2009]. Modern software is ubiquitous and typically just a part of a complex system that imposes complex inter-operability requirements based on changing platforms and technologies. Such software is often developed by distributed teams who follow different development processes and business models.

To prepare computing graduates for professional careers, their education must provide them with "real-life" experience. Simply lecturing about and discussing all the software development phases is not sufficient preparation for a professional career. There are many tasks beyond those of core software development, for which students need training: project management, team building, software estimation and planning, progress tracking, and communication. Communication, in particular, has become a critical issue, since teams in modern software development projects are often multidisciplinary and distributed over cultures and time zones. Global and distributed development also make it more difficult to set up "real projects for real clients" courses [Klappholz et al. 2009], since there are a range of practical problems, which are difficult to simulate and/or control in an educational setting.

Author's addresses: Jürgen Börstler, Software Engineering Research and Education Lab Sweden, Blekinge Institute of Technology, Campus Gräsvik, 371 79 Karlskrona, Sweden; Thomas B. Hilburn, Embry-Riddle Aeronautical University, Daytona Beach, FL 32114-3900, USA.

DOI: 0000001.0000001

There is a large body of knowledge on issues and challenges pertaining to team project courses in computing education ([Crnković et al. 2012; Ellis et al. 2009; Fincher et al. 2001; Hilburn and Humphrey 2002; Wikstrand and Börstler 2006] and problem- and project-based learning in general ([Helle et al. 2006; Hmelo-Silver 2004]). Furthermore, there are accreditation regulations and international curriculum initiatives, like ABET ([ABET 2015]), CDIO ([CDIO 2011]) and curriculum guidelines from, for example ACM/IEEE ([ACM/IEEE 2015a]) that drive educational institutions to integrate the teaching of non-technical skills into their engineering and computing curricula.

This and the subsequent issue present the experience and research of educators, which we believe can help those who want to improve the delivery of team project courses, and which supports the overall goal of preparing graduates for professional practice.

2. KEY AREAS OF INTEREST

The aim of this special issue is to share evidence-based practices that have been applied to team projects in computing education. The goal is twofold. On one hand, this will help educators to improve the state-of-practice in computing education. On the other hand, it will bring forward research on various aspects related to the teaching and learning about team-projects.

The call for papers for this special issue required an initial submission of an abstract. In total, a record number of 69 abstracts were submitted, from all over the world: Australia, Canada, China, Egypt, England, Estonia, Finland, Germany, Greece, India, Israel, Italy, Jamaica, Macedonia, Netherlands, Norway, Poland, South Africa, Spain, Sweden, and USA.

The editors reviewed the abstracts submitted and invited nineteen to submit a full paper. A team of twenty-five reviewers reviewed the submitted papers, and based on the reviews, nine were finally accepted for publication in this special issue. Papers in the following categories were accepted:

— multidisciplinary team projects,
— studio-based approaches,
— cognitive and psychological aspects,
— student and team assessment,
— team building and team dynamics,
— collaborative learning and methods and tools to support team project courses,
— real projects for real clients, and
— open source projects.

3. ARTICLES IN THIS ISSUE

The articles in this issue and a subsequent one range over the categories listed above. What follows is a short description of each article.

3.1. Real projects for real clients

In the first paper of this special issue, “Software Engineering Project Courses with Industrial Clients”, Bruegge, Krusche and Alperowitz report on a course that focuses on executable prototypes and informal models as a key enablers for the communication between students and real clients. Students follow a process, called Rugby, that integrates elements from Scrum and the Unified Process. In multiple feedback cycles, students use executable prototypes, instead of formal documents and reports, to get feedback from their clients. This makes it easier to respond to changing requirements. To support early communication, the authors use Tornado, a light-weight scenario-based design approach, that emphasizes informal models.

Despite their focus on informal models and executable prototypes, the authors emphasize that it is important to teach the transition between informal modeling and more formal modeling. In their course, the students finally have to deliver consistent and complete artifacts to their clients.

3.2. Open source projects

In "Team Project Experiences in Humanitarian Free and Open Source Software (HFOSS)," Ellis, Hislop, Jackson, and Postner discuss the potential and challenges of student participation in Free and Open Source Software (FOSS) projects within the context of undergraduate computing degree programs. More specifically, this work focuses on Humanitarian Free and Open Source Software (HFOSS) projects. HFOSS is open source created to benefit the human condition in some manner and can range from disaster management to microfinance to applications that help monitor elections.

The authors study of student participation in HFOSS indicate that there are significant potential benefits from student participation, but that significant issues need to be addressed to achieve these benefits. The challenges include a fairly steep learning curve for students and faculty, and scheduling and other coordination issues between the project and academic courses.

3.3. Multidisciplinary team projects

In "Team Building in Multidiscipline and Multicourse Projects," Pastel, Seigel, Zhang, and Mayer emphasize that experience working in multidisciplinary teams is important both to prepare computer science students for industry and to improve their communication with teammates from disciplines other than their own. The authors describe the activities and results from collaborations between three courses: an undergraduate user interface design and implementation course, a course about usability and technical communication, and a graduate course about user interface evaluation and usability testing.

Students from the undergraduate courses form multidisciplinary teams to design and implement Android apps for collecting environmental information, while the graduate students consult with the teams by evaluating and user testing the apps. The authors compare the collaboration within the teams and the coordination with the scientist clients across two years of activities. The paper offers recommendations for improving the collaboration within teams and the coordination with clients in multidisciplinary course projects.

3.4. Student and team assessment

In "Assessing Large Project Courses: Activities and Lessons Learned," Vasilevskaya, Broman, and Sandahl stress the importance of large project courses in giving students a realistic software engineering experience. The paper discusses the challenge of assessing such projects: how to assess individuals fairly, and still maintain the focus of the project as a group activity; extensive teacher involvement is necessary for objective assessment, but may affect the way students work; and continuous feedback to students can enhance learning, but can be hard to combine with a fair, accurate assessment.

The authors present an assessment model, developed over a seven year period, which is a collection of ten assessment activities, each covering different aspects of individual and group work. They designed and executed an experiment to collect assessment data, and use the data to analyze the results, discuss findings, and summarize lessons learned.

3.5. Cognitive and psychological aspects

In "Media, Mood, and Meetings: Related to Project Success?," Schneider, Liskin, Paulsen, and Kauffeld argue that software engineering project courses should take into consideration the way modern software is often developed: use of global teams distributed over time zones and cultures, in which team members may feel isolated and distant from each other and may in turn adversely impact the success of the project.

The paper describes a study of 20 student teams over a period of four months. The teams were free to schedule team meetings when and where they wanted; and they could use any communication media they chose. This authors investigated and analyzed the relationships between media, mood and meeting styles in the team, and evaluated how they were related to project success. Statistical correlations and nonparametric test are used for quantitative empirical analysis. The results led to a number of recommendations for course organizers and for software engineers in general.

4. PREVIEW OF NEXT ISSUE

The upcoming second issue on Team Projects will also vary over a range of topics: Studio-based approaches, Collaborative learning & Methods and tools to support team project courses, Team building and team dynamics, and Cognitive and psychological aspects. We believe they complement and extend the experience, research and recommendation contained in this issue.

5. DIRECTIONS FOR FUTURE RESEARCH

The papers in this issue motivate and support research in the following areas:

— The development of tools, environments, and processes to deal with recurring tasks, in order to improve the delivery of multi-project, multi-customer courses with a large number of students (100+).
— The use of open source software to enhance student learning about software development and to support student career choice and preparation.
— The development and analysis of techniques for improving the communication and collaboration within multidisciplinary teams, and between clients and teams.
— The design, implementation, and analysis of assessment models for team projects, especially for large teams.
— How professional projects are influenced by the relationships between media usage, mood, goals, and project success.

ACKNOWLEDGMENTS

We would like to thank the reviewers who supported us in selecting a number of interesting and high quality contributions. Without their expertise and time developing this special issue would have been an impossible task. We would also like to thank the editors of TOCE, Josh Tenenberg and Robert McCartney for their guidance and support in developing this special issue.

REFERENCES

ABET. 2015. Accreditation Criteria and Supporting Documents. (2015). http://www.abet.org/accreditation/accreditation-criteria/, last visited June 22, 2015.

ACM/IEEE. 2015a. Curricula Recommendations. (2015). http://www.acm.org/education/curricula-recommendations, last visited June 22, 2015.

Joint Task Force on Computing Curricula ACM/IEEE. 2015b. Software Engineering 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. (2015). Draft report pending approval.

CDIO. 2011. The CDIO Syllabus v2.0. (2011). http://www.cdio.org/framework-benefits/cdio-syllabus/, last visited June 22, 2015.

Ivica Crnković, Ivana Bosnić, and Mario Žagar. 2012. Ten tips to succeed in global software engineering education. In *Proceedings of the 34th International Conference on Software Engineering*. 1225–1234.

Heidi J.C. Ellis, Steven A. Demurjian, and J. Fernando Naveda. 2009. *Software Engineering: Effective Teaching and Learning Approaches and Practices: Effective Teaching and Learning Approaches and Practices*. IGI Global, Hershey, NY, USA.

Sally Fincher, Marian Petre, and Martyn Clark. 2001. *Computer science project work: principles and pragmatics*. Springer Science & Business Media, London, UK.

Laura Helle, Päivi Tynjälä, and Erkki Olkinuora. 2006. Project-based learning in post-secondary education – theory, practice and rubber sling shots. *Higher Education* 51, 2 (2006), 287–314.

Thomas B. Hilburn and Watts S. Humphrey. 2002. Teaching Teamwork. *IEEE Software* 19, 5 (2002), 72–77. DOI:http://dx.doi.org/10.1109/MS.2002.1032857

Cindy E Hmelo-Silver. 2004. Problem-based learning: What and how do students learn? *Educational psychology review* 16, 3 (2004), 235–266.

David Klappholz, Vicki L Almstrum, Ken Modesit, Cherr Owen, Allen Johnson, and SJ Condly. 2009. A framework for success in real projects for real clients courses. See Ellis et al. [2009], 156–189.

Nancy R Mead. 2009. Software engineering education: How far weve come and how far we have to go. *Journal of Systems and Software* 82, 4 (2009), 571–575.

Mary Shaw. 2000. Software engineering education: a roadmap. In *Proceedings of the 22nd International Conference on Software Engineering – The Future of Software Engineering*. 371–380.

Greger Wikstrand and Jürgen Börstler. 2006. Success Factors for Team Project Courses. In *Proceedings of the 19th Conference on Software Engineering Education and Training*. 95–102.