Peer Assessment in Experiential Learning

Assessing Tacit and Explicit Skills in Agile Software Engineering Capstone Projects

Fabian Fagerholm, Arto Vihavainen
Department of Computer Science, University of Helsinki
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014, Finland
fabian.fagerholm@helsinki.fi, arto.vihavainen@cs.helsinki.fi

***Abstract*—To prepare students for real-life software engineering projects, many higher-education institutions offer courses that simulate working life to varying degrees. As software engineering requires not only technical, but also inter- and intrapersonal skills, these skills should also be assessed. Assessing soft skills is challenging, especially when project-based and experiential learning are the primary pedagogical approaches. Previous work suggests that including students in the assessment process can yield a more complete picture of student performance. This paper presents experiences with developing and using a peer assessment framework that provides a 360-degree view on students' project performance. Our framework has been explicitly constructed to accommodate and evaluate tacit skills that are relevant in agile software development. The framework has been evaluated with 18 bachelors- and 11 masters-level capstone projects, totaling 176 students working in self-organized teams. We found that the framework eases teacher workload and allows a more thorough assessment of students' skills. We suggest including self- and peer assessment into software capstone projects alongside other, more traditional schemes like productivity metrics, and discuss challenges and opportunities in defining learning goals for tacit and social skills.**

Keywords*—Peer assessment; assessment metrics; self-assessment; case study; capstone project; experiential learning; project-based learning; tacit skills; teamwork; computer science education; agile software engineering.

I. Introduction

It is well known that there exists a gap between what engineering students learn and what is expected from them as they graduate [1], [2], [3]. The expectation gap [4] is especially visible in software engineering education, where practices learned while studying may even have to be unlearned later [5].

Among the expected skills are the abilities to read social cues, regulate emotional expression, and to engage in constructive dialogue with project stakeholders to discover tacit knowledge. These so-called *soft skills* are particularly important in software engineering projects that rely more on informal communication than document-driven and plan-based approaches. They are important for building and maintaining cohesion in development teams, working with external teams, and for involving other project participants and stakeholders in the software development process.

The gaps in communication and teamwork skills of new engineers were discussed already in the 1990s [6], and the most relevant skills required from entry-level IT personnel still include personal attributes such as problem solving, critical and creative thinking, and team skills and communication [7]. Teamwork and communication is also emphasized in the emerging agile methodologies, where interaction between individuals is valued over processes and tools [8]. Ultimately, students should grow into members of their communities of practice [9], adopting the tacit skills required to function in the field. This is a notable challenge for higher education.

Teamwork is usually practiced in several projects in higher education. Perhaps the most notable project is the capstone project, which is often the culmination of a degree program. Capstone projects are typically made for real customers in as realistic settings as possible, given the constraints of the educational institution. Capstone projects provide an opportunity to assess higher-order cognitive dimensions of learning as well as affective and skill-based dimensions [10]. Even if the end product of a project is the most valuable deliverable for a customer, the whole project can be a continuous and valuable learning process for the students.

As students direct their activities based on the given assessment criteria [11], the assessment design plays a key role in what students will focus on. In a software engineering capstone project, the assessed skills and knowledge should contain: (1) elementary software engineering related skills such as requirements analysis, design, development, and validation; (2) tool related skills such as the use of a version control system, development tools, and process management tools; and (3) process related skills such as process knowledge and how a selected process is followed. How the students utilize and benefit from these skills in a teamwork setting is moderated by several tacit, soft, and social skills.

In this paper, we present ongoing work on an assessment framework that can be used as a decision support tool to assess tacit skills together with explicit skills in capstone project environments. The framework has been built to help focus students' attention to important team-related aspects, and to help teachers assess student performance in capstone projects.

This paper is structured as follows. In Section II, we give an overview of our educational context and the learning objectives of our capstone courses, as well as discuss our motives to develop a new framework. In Section III, we describe related work on assessment of project-based education as well as self- and peer assessment, and in Section IV, we describe the framework. The evaluation of the framework is discussed in Section V through a multiple case study, and finally, in Section VI, we conclude this paper and outline future work.

II. Background

Computer Science studies at the University of Helsinki are divided into a three-year bachelor's degree, and a two-year higher master's degree. The bachelor's degree is a comprehensive computer science degree, which prepares the students for both working life and future studies. There is no "specialization track" within the bachelor's studies: every student takes courses on e.g. math, software engineering, distributed systems, as well as algorithms and machine learning.

If the students choose to pursue a master's degree, they have a variety of specialization tracks to choose from. Our focus is on the software engineering specialization track, in which students deepen their understanding on e.g. software processes and quality, agile methodologies and coaching, as well as software architecture.

A. Capstone Projects

Both the bachelor's and master's degrees contain a capstone project. The bachelor's degree studies culminate in either a 7- or a 14-week *Software Engineering Project*, during which the students work in 4–5-person teams on a project from e.g. an industry partner or a research group. The 7-week version is a full-time project, where the students are collocated at one of our labs, while the 14-week version is a part-time project, and can be partially distributed. Although the students are mentored by staff, they handle all project aspects in a self-organized manner, including project management and setting customer expectations.

The capstone project for the software engineering specialization track is the *Software Factory Project* [12], [13], which simulates a teamwork environment in contemporary software development organizations. Its design aims to shift responsibility of all aspects of project operation to the student team, in order to ensure that students are exposed to the realities of software development.

The Software Factory Project is similar to the bachelor's-level capstone project, but with some characteristics that make it more challenging. The project usually begins with more ill-defined goals. Part of the purpose is to discover, together with the customer, even which software the project is to produce, and how the software can bring value to the customer and end user. Some of the projects also operate in a distributed environment together with other Software Factory nodes at separate universities.

B. Motivation and Learning Objectives

The motivation for assessing teamwork and tacit skills arose initially during the design and development of the Software Factory Project. The tacit skills are of particular importance in such courses, and thus were set as important learning objectives of the project. One of the challenges was to design an approach to assess not only project deliverables and productivity, but also performance in terms of tacit and team skills. The lessons learned in the Software Factory were incorporated, together with results from a multi-year improvement effort [14], into the Software Engineering Project.

The main learning objective for both capstone projects was originally defined as "*the ability to become a member of a software development team, function as part of it, contribute to its development, and work as part of it towards its current mission or purpose*". In addition, the software engineering project has a specific learning objective rubric, which outlines the principal themes in the course and what is required for each level of student assessment. The rubric follows the principles of constructive alignment [15], and outlines software development related skills, as well as management and tool usage.

The effort that students put on learning is heavily determined by the assessment criteria [16], [17]. However, neither the rubric nor the main learning objective supported assessing soft skills. As the main learning objective of the capstone projects consists of a set of distinct sub-objectives, assessment requires that each of them is identified and assessed independently, preferably using a small number of traits that cover the knowledge and skills required to do well in each part of an activity [17].

Our framework is based on the cognitive domain of Bloom's revised taxonomy [18], [19], which provides guidelines for agreeing on assessment and learning objectives for a course. It outlines six levels (*remembering*, *understanding*, *applying*, *analyzing*, *evaluating*, and *creating*), which are ordered from simpler to more complex; the original idea was that mastering a "higher" category requires the mastery of the previous categories.

Based on the six levels defined by the cognitive domain in Bloom's revised taxonomy, we outlined the following team skills that each participant focuses on: *presence*, *activity*, *eagerness*, *devotion*, *contribution*, and *expert maturity*. In addition, the bachelor's-level software engineering projects put additional focus on the participant behavior and its influence on *process* and *result*. A more comprehensive description of these team skills is given in Section IV.

III. Related work

Several studies have examined self- and peer assessment of teamwork in regular courses, and assessment of teamwork in projects, including capstone-like courses. Here, we briefly present some of the issues examined and results found.

One of the most fundamental questions regarding assessment is that of its purpose. Naturally, assessment can serve multiple purposes simultaneously; it can help rank students with respect to their performance, allowing selection to be made in different stages of an educational system, and it can provide important feedback to students regarding their study performance. When assessment is tied to specific learning objectives, students' activities can be directed towards activities that build knowledge and skills that are deemed relevant.

Assessment can be used on the systemic level to evaluate learning programs in terms of how well they support achievement of learning outcomes [20]. The nature of capstone projects as comprehensive experiences means that they allow assessing a wide range of abilities; they are indicative of learning program strengths and weaknesses. Analysis of capstone project outcomes can provide valuable insights for improving learning programs, and thus, improving student learning. Payne et al. suggest assessing student readiness for capstone courses in order to gather feedback on both the presence of necessary background knowledge, skills, and dispositions, and the ability to apply them to capstone courses [10]. They outline critical concepts and skills that students must be taught to assure their success in capstone courses, noting that educators and researchers should set up continuous feedback frameworks that could be used to transfer knowledge to core-course faculty on the level of preparation the students believe they have for the upcoming capstone experience.

A pertinent question is how to actually assess capstone projects: what is to be assessed and how? One approach is to map project deliverables and artifacts to general and specific learning outcomes and rubrics, and then assess the deliverables with respect to the rubric, as proposed by Murray et al. [20]. As an example, Murray et al. describe the goals of information systems capstone projects: students should be able to i) understand that projects require collaboration as well as individual effort, ii) participate as contributing members of a development team, iii) apply teamwork skills in development and implementation of a system, iv) demonstrate acknowledgment of and respect for the team members, and v) identify the qualities needed to be an an effective leader, and explain the roles of leadership and teamwork in system development and implementation. The artifacts used to evaluate these outcomes include individual reports, peer evaluations, and weekly status forms.

Self- and peer assessments appears to be viewed favorably by many teachers and researchers in terms of how well it includes students into the assessment process. For instance, Fellenz finds that peer evaluation can improve the quality of the students' experience and increase their engagement in the learning task [21]. However, a particular concern in assessing teamwork skills is the accuracy of assessment. Through a review of assessment literature, Van Duzer and McMartin [22] identified two primary types of bias as especially relevant for self-assessment and peer evaluation: self-enhancement, where one's own performance is evaluated as unreasonably optimistic, and downward comparison, a general tendency for positive self-bias and negative other-bias. Similar results are reported in many works. For example, Ryan et al. compared peer and self-evaluations of class participation against those of professors [23]. They found that faculty grades tended to be higher than peer grades, and that self-evaluation grades were typically higher than faculty grades. This study used a forced ranking system for students to rank each other while faculty did not use forced ranking.

Van Duzer and McMartin suggest some approaches to reduce self-enhancement and downward comparison biases [22]. Using language shared by respondents and testers in assessment criteria helps to reduce misinterpretation and thus improves the validity of the assessment process. Correlating self-assessments with scores by multiple raters allows evaluation of instrument reliability. Designing questions so that they rate past performance, not expected future performance, improves reliability by reducing the effect of downward comparison. Finally, asking respondents to make comparisons with an explicit group of known individuals rather than an abstract group when social comparisons are required, also improves reliability. Qualitative analysis while developing the instrument is necessary to understand the meaning of the assessment to participants. Van Duzer and McMartin developed a process with both quantitative and qualitative parts for improving and tailoring teamwork skill assessment in specific environments. They found a dramatic improvement in sensitivity when applying the process to their own instrument.

A number of approaches and frameworks for self- and peer assessment have been described in the literature. Willey and Freeman report on a tool that facilitates formative assessment via self- and peer assessment [24]. They report that formative feedback encouraged development of teamwork skills, and also discouraged free-riding and sabotage, thus promoting academic honesty. They argue that while self- and peer assessment is often implemented as summative assessment, even better outcomes may be achieved by using them as formative assessment. They observe that the administrative burden of applying self- and peer assessment can often outweigh the perceived benefit. Furthermore, they observe that feedback is often given long after the assessable work has been completed, which means that students' attention may already have shifted to other tasks.

Beyerlein et al. [25] describe an assessment framework for capstone design courses. Their framework is based on a conceptual model of knowledge representation and expertise development. They strive to examine students' performance and growth from several perspectives. They examine growth in personal knowledge and skills applied in problem solving. They examine professional development through goal-driven initiative, competence in problem-solving, integrity and professionalism, and ongoing reflection. Also, they examine team processes and dynamics as well as productivity by determining whether team resources are used strategically, and decisions made add real value to the project. They also examine how well students are able to formulate solution requirements, consider stakeholder needs, and formalize these into specifications. Finally, they evaluate deliverables in terms of desired functionality, economic benefits, feasibility of implementation, and favorable impact on society.

Another concern is students' motivation to rate their peers. Friedman et al. found that students who provided categorical ratings (multiple scores on different categories or dimensions) multiple times during a course experienced the lowest motivation to rate their peers, while students who provided holistic ratings (a single score) multiple times reported the highest motivation [26]. We may hypothesize that respondent fatigue plays a role here: a small number of items is less likely to feel overwhelming. The type of item may also be important: describing a particular behavior and asking the respondent to indicate its frequency is usually recommended – an approach used, e.g., in rating a system developed by Clark et al. [27].

Finally, also related to practical concerns, is the burden of manual work in collecting and analyzing self- and peer assessment data. Naturally, online questionnaires and semi-automated analysis tools can remove much of this manual work. Some reports exist on complete systems for self- and peer assessment management. The SPARK system, described by Freeman and McKenzie [28], emphasizes fairness in group work assessment and reduced administrative burden through automation. Similarly, the CATME system, described by Ohland et al. [29], provides automation to reduce teacher workload, but places greater emphasis on using behavioral anchors in the assessment itself. The SMARTER system [30] extends CATME and attempts to link educational research with teaching faculty actions to enhance learning of teamwork skills.

IV. Framework for Assessing Tacit Skills

Our Framework for Assessing Tacit Skills consists of a questionnaire whose items can be used (e.g. by weighted averaging) to provide assessment decision support for teachers. The framework enables assessment of tacit skills through nine indicators, used for both self- and peer assessment. We categorized the indicators to represent six different tacit skills, and decided that the assessment should impose as little overhead as possible for all participants and thus should be implemented as a short online questionnaire.

The framework factors, questionnaire items, and scales are shown in Table I. The questionnaire, which is filled in by the students, project coach, and the client, allows rating each student based on the questionnaire items. Once the questionnaire has been answered, the answers are exported for further data-analysis, where a set of scripts is used to e.g. suggest overall grades based on a given weighting, or to indicate students that have been free-riding.

The questionnaire is structured along six factors, beginning from basic factors and progressing towards higher levels of involvement and skills. The first factor, presence, is a prerequisite for becoming a member of the development team. The activity factor implies that a person is not only present, but also actively involved in the project. Eagerness reflects the attitude that the person takes towards the project: is the person not only active but also taking initiative and displaying a positive desire to get things done. Devotion reflects a deeper level of commitment: the person not only takes the initiative but actually invests effort into carrying out planned tasks. Contribution reflects actual impact on the project, whether in the form of code, documentation, or other deliverables, or in the form of project management, customer communication, or support tasks. Finally, expert maturity reflects an overall assessment of how the person performed in their role. We purposefully chose to leave the definition of this factor quite open and broad, in order to allow each individual to assess it according to the specific conditions of each particular project.

While we appreciate the objectivity and wide coverage of the approach described by Murray et al. [20] and other similarly detailed assessment schemes, we suspect that both students and teachers can quickly be overwhelmed by the amount of effort required to produce and analyze the assessment artifacts, resulting in both less effort being available for project work and formative assessment and guidance. It also feels counter to the philosophy of Agile software development methodology to employ a heavy-weight assessment framework – after all, agile projects purposely do not define artifacts to be produced until there is a proven need to produce them.

Three main criteria were defined for the framework. First, the framework should ease teacher workload. The framework should function as a support tool for teachers during assessment, and it should support assessment of project-based courses even when teachers cannot constantly observe students' activities. Second, it should allow systematic assessment of students' skills; each factor in the framework can be thought of as building on top of the previous factors. Finally, it should be easy to detect attempted misuse of the framework, so that teachers can be confident that they may use the results as valid decision support information.

V. Evaluation

The framework has been evaluated iteratively during its development. It was first evaluated in several projects in the Software Factory, and later also in the Software Engineering Project. In this section, we report on the evaluation procedures and present the most relevant evaluation results. We then discuss the validity and limitations of our evaluation and present results from evaluating the framework from a teacher perspective.

As noted, the motive for assessing tacit skills arose during the design of the Software Factory. We first conducted a pilot project in spring 2010 with 11 students, during which the framework dimensions were developed. Then, the framework was deployed to 11 consecutive Software Factory projects with a total of 77 students. The evaluation of the framework in the Software Engineering Project started in fall 2011, after which a total of 18 projects with a total of 88 students have been both evaluated by the framework and given their evaluation of the framework. Since the latter project is mandatory for all bachelor's-level students, we wanted to gain reasonable confidence that the framework worked well before deploying it there. As part of that deployment, we found that the Software Engineering Project students did not perceive teamwork-related skills as important. For example, competitive situations arose where several strong individuals attempted to pull the project in their desired direction. For this reason, factors regarding individual behavior in relation to the group were added.

Our evaluation strategy is laid out as follows. Ultimately, the objective is to find out whether the framework is suitable for the purpose of influencing learning of teamwork skills through assessment. However, before actually determining its effect on learning, we want to understand whether the framework is otherwise suitable for use in capstone projects. This includes evaluating the accuracy of assessment and utility of the framework as a decision support tool: does the framework adequately guard against biases such as self-enhancement and downward comparison, does it adequately reflect rater's understanding of the factors, and does it produce results that are in line with teachers' expert evaluations, taking into account the rich, qualitative observational data obtained when guiding students in the capstone projects?

To perform this evaluation, we proceed as follows. We check the association between self- and peer ratings to determine whether a bias is visible (see Table II). Peer ratings should help dampen bias in self-ratings. We check association between the different rating factors. There should be discernible differences between the factors both in self- and peer ratings; they should not have perfect correlation. However, there should be some association between the factors that are in fact conceptually related.

Table II shows correlations between self- and peer assessments in both the Software Factory (SF) and the Software Engineering Project (SP). Most of these correlations are as expected: there is a large degree of correlation but there are differences in the gradings. However, some correlations stand out from the others. In SF, there is quite low correlation on eagerness, and self-ratings tend to be higher (mean: 0.863) than peer ratings (mean: 0.743). In SP, self-ratings tend less towards the highest grade (mean: 0.788) and peer ratings are similar in distribution (mean: 0.786). In our interpretation, students

TABLE I. Framework factors, questionnaire items, and scales.

Factor	Questionnaire item	Scale
Presence	How many days per week did you work on this project? How many hours did you spend on the entire project in total? (Round to nearest hour.) How much was each team member present? Also rate your own presence.	1 – Was not present at all 2 – Was sometimes present 3 – Was moderately present 4 – Was nearly always present 5 – Was always present 0 – I don't know
Activity	How actively did each team member participate in the project? Also rate your own activity.	1 – Was not active at all 2 – Was somewhat inactive 3 – Was moderately active 4 – Was quite active 5 – Was very active 0 – I don't know
Eagerness	Eagerness: a positive feeling of wanting to push ahead with something. How eager was each team member to participate in the course? Also rate your own eagerness.	1 – Was not eager at all 2 – Was a little eager 3 – Was moderately eager 4 – Was quite eager 5 – Was very eager 0 – I don't know
Devotion	Devotion: commitment to some purpose; "the devotion of his time and wealth to our project" How devoted was each team member to the course? Also rate your own devotion.	1 – Was not devoted at all 2 – Was a little devoted 3 – Was moderately devoted 4 – Was quite devoted 5 – Was very devoted 0 – I don't know
Contribution	How much did each team member contribute to the deliverables (code, documentation, tests, bugs, plans, or anything else that the project produced)? Also rate your own productivity.	1 – Did not contribute at all 2 – Contributed a little 3 – Contributed moderately 4 – Contributed quite much 5 – Contributed very much 0 – I don't know
Expert Maturity	Each team member has acted as a software development expert with some specific focus area. How mature was each team member in their expert role? Also rate your own maturity.	1 – Very low expert maturity 2 – Low expert maturity 3 – Neutral expert maturity 4 – Some expert maturity 5 – High expert maturity 0 – I don't know
Process (only BSc project)	Group dynamics: each member can influence the team spirit and the end result with their social behavior. How did the group behavior of each member influence the sensed meaningfulness of the project work?	1 – Influenced negatively 2 – Did not influence 3 – Influenced a little 4 – Influenced quite much 5 – Influenced very much 0 – I don't know
Result (only BSc project)	How did the group behavior of each member influence the end quality of the project work?	1 – Influenced negatively 2 – Did not influence 3 – Influenced a little 4 – Influenced quite much 5 – Influenced very much 0 – I don't know

TABLE II. Correlations between self- and peer ratings on different framework dimensions in Software Factory (SF) and Software Engineering Project (SP), with corresponding p-values.

Dimension	Correlation between self- and peer rating	p-value
Presence (SF)	0.492	< 0.001
Presence (SP)	0.457	< 0.001
Activity (SF)	0.531	< 0.001
Activity (SP)	0.544	< 0.001
Eagerness (SF)	0.279	0.017
Eagerness (SP)	0.473	< 0.001
Devotion (SF)	0.433	< 0.001
Devotion (SP)	0.333	0.002
Contribution (SF)	0.582	< 0.001
Contribution (SP)	0.376	< 0.001
Expert maturity (SF)	0.461	< 0.001
Expert maturity (SP)	0.207	0.062
Contribution to meaningfulness (SP)	0.487	< 0.001
Contribution to quality (SP)	0.370	0.002

in SP could be less inclined to penalize each other, perhaps because their level of experience is lower and the course is mandatory – they may not want to give low ratings to each other on eagerness given that situation.

On devotion and contribution, the trend is similar: in SF, the correlation is stronger than in SP. In the SP data, high peer ratings were more common than in the SF data. In SF, roughly one third of students rated their peers at average expert maturity, while more than two thirds of SP students assigned each other the two highest scores. This may indicate that the competitiveness among students in the SF is higher. We observe that this information allows the teacher to assess the amount of bias in responses and that there appears to be agreement on the meaning of the dimensions.

Next, we consider the association between the variables. In the self-evaluation scores, presence correlates somewhat with activity and eagerness but less with devotion, contribution, and least with expert maturity. This could indicate that students do see these factors as separate. Activity correlates quite strongly with eagerness, devotion, and expert maturity. Devotion correlates strongly with contribution and expert maturity. Contribution correlates most strongly with expert maturity.

In the peer evaluation scores, all factors are moderately to strongly correlated. In SF, the strongest (≥ 0.9) correlations are i) activity with eagerness (0.930), devotion (0.932), and contribution (0.943); ii) eagerness with devotion (0.912) and contribution (0.911); iii) devotion with contribution (0.939); and iv) contribution with expert maturity (0.906; $p < 0.001$ in all cases). In SP, the correlations are smaller but still quite strong. The order of strength is roughly the same. We interpret these results as supporting the intended structure of the factors.

In SP, the two added factors had moderate to low correlation between self- and peer evaluation. On contribution to meaningfulness, self- and peer evaluations had a moderate correlation (0.487), while on contribution to quality, the correlation was lower (0.370). In the latter, there may be bias toward thinking that one's own contribution is the most important, and therefore one rates the others lower.

A. *Validity*

The validity of the framework is limited by the fact that it uses a questionnaire-based approach. Respondents are asked to recall their own behavior and that of their teammates, and this recall may not be perfect. However, more fundamentally, the validity is ultimately relative to context in which the instrument is deployed. The purpose of the framework is to function as a decision support tool, and teacher judgment should be used to determine the final assessment. As MacLellan notes, validity concerns not the assessment instrument used or the resulting scores as such, but rather the inferences which are derived from them [31].

To lend more validity to such inferences, the framework should provide a way to detect whether the data may be biased or incorrect. The most common reason besides unintentional bias is students' attempts to artificially influence their grades. We found some cases of attempted subversion, where a small number of students systematically rated themselves with the highest scores and others with the lowest scores. These cases were easily detected using simple, semi-automatically produced outlier analysis.

B. *Teacher satisfaction*

In our context, we have evaluated the framework with three different teachers. While the results of this evaluation are experiential and cannot be generalized, we find it important to report on these experiences to enable other teachers to determine whether our approach is of value in their context.

Our first finding relates to the goal of easing the teacher's workload. We found the framework to be non-intrusive and supporting formative assessment and feedback during the capstone courses. The framework required no extra effort during the courses, and the teachers were able to devote their time to in-situ instruction. At the end of the course, some administrative effort was needed to administer the on-line questionnaire, collect the results, and perform the required data analysis. However, since many of these tasks were automated or semi-automated, teachers could focus on the intellectual side of summative assessment: interpretation of the numeric results and comparison of them to other assessment sources, including notes taken during the course.

One of the teachers voiced concerns regarding fairness and comparability between students and projects. However, we found that when used as a decision-support tool for assessment, the framework did not introduce any fairness concerns. This was also reflected in students' attitudes – all students were given the same opportunity to grade themselves and each other, and the teacher validated the results so that unfair biases were accounted for in the final grade. Cross-project comparability is still an issue, however, but it is not unique to this framework. Each capstone project is inherently different, and maintaining the level of realism often desired in such projects means that comparisons of performance are difficult.

VI. Conclusions and Future Work

In this article, we have described our tacit skills assessment framework, which is an easy-to-use decision support utility for evaluating students' teamwork proficiency. The framework has been evaluated with data from 18 bachelor's and 11 master's-level capstone projects, where it has been found to provide reasonable support for teachers in evaluating tacit, social, and teamwork skills. We found that the framework guarded against rater bias, that its dimensions were well understood, and that it matched teachers' expert ratings. Our results are relevant in the context of project-based courses emphasizing experiential learning and agile methodologies.

We suggest including self- and peer assessment into software capstone projects. However, although technically possible, one should not base assessment of students in capstone projects only on the values provided by the self- and peer ratings. We suggest using additional criteria that takes into account several other data sources, such as version control system commits and their quality. In addition, feedback on the overall project can be obtained from the customer as well as a possible team lead or coach. Aggregating scores into a final grade requires experimentation and the inclusion of teacher judgment.

In case participants display behavior that is not seen as beneficial for the team, additional assessment criteria can be added to the framework due to it's small size. As an example, a few participants in our current bachelor's level software engineering projects have displayed a tendency for "safety seeking", where individuals have avoided working on tasks that require learning new tools and practices. Additional incentives for moving away from the comfort zone have been introduced via a new assessment criteria "How well did the participant handle tasks that required learning new tools and practices?".

We are currently considering a replication study to evaluate the framework in a Software Factory in another country, as well as evaluating approaches to make the framework easier to use. Other possible directions include formative assessment support, and determining the association between framework factors and objectively measurable metrics such as code metrics.

References

[1] R. Martin, B. Maytham, J. Case, and D. Fraser, "Engineering graduates' perceptions of how well they were prepared for work in industry," *European Journal of Engineering Education*, vol. 30, no. 2, pp. 167–180, 2005.

[2] R. L. Meier, M. R. Williams, and M. A. Humphreys, "Refocusing our efforts: Assessing non-technical competency gaps," *Journal of Engineering Education*, vol. 89, no. 3, pp. 377–385, 2000.

[3] M. Natishan, L. Schmidt, and P. Mead, "Student focus group results on student team performance issues," *Journal of Engineering Education*, vol. 89, no. 3, pp. 269–272, 2000.

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

[5] A. Begel and B. Simon, "Struggles of new college graduates in their first software development job," in *ACM SIGCSE Bulletin*, vol. 40, no. 1. ACM, 2008, pp. 226–230.

[6] P. J. Denning, "Educating a new engineer," *Communications of the ACM*, vol. 35, no. 12, pp. 82–97, 1992.

[7] M. E. McMurtrey, J. P. Downey, S. M. Zeltmann, and W. H. Friedman, "Critical skill sets of entry-level it professionals: An empirical examination of perceptions from field personnel," *Journal of Information Technology Education*, vol. 7, pp. 101–120, 2008.

[8] M. Fowler and J. Highsmith, "The agile manifesto," *Software Development*, vol. 9, no. 8, pp. 28–35, 2001.

[9] E. Wenger, *Communities of Practice: Learning, Meaning, and Identity*, ser. Learning in Doing. Cambridge University Press, 1998.

[10] S. L. Payne, J. Flynn, and J. M. Whitfield, "Capstone Business Course Assessment: Exploring Student Readiness Perspectives." *Journal of Education for Business*, vol. 83, no. 3, pp. 141–146, 2008.

[11] J. Biggs and C. Tang, *Teaching for quality learning at university*. Open university press, 2011.

[12] F. Fagerholm, N. Oza, and J. Münch, "A Platform for Teaching Applied Distributed Software Development: The Ongoing Journey of the Helsinki Software Factory," *Collaborative Teaching of Globally Distributed Software Development*, 2013.

[13] P. Abrahamsson, P. Kettunen, and F. Fagerholm, "The set-up of a software engineering research infrastructure of the 2010s," in *Proceedings of the 11th International Conference on Product Focused Software*, ser. PROFES '10. New York, NY, USA: ACM, 2010, pp. 112–114.

[14] M. Luukkainen, A. Vihavainen, and T. Vikberg, "Three years of design-based research to reform a software engineering curriculum," in *Proceedings of the 13th annual conference on Information technology education*. ACM, 2012, pp. 209–214.

[15] J. Biggs, "Enhancing teaching through constructive alignment," *Higher education*, vol. 32, no. 3, pp. 347–364, 1996.

[16] J. Biggs and C. Tang, *Teaching for Quality Learning at University*, ser. SRHE and Open University Press Imprint. McGraw-Hill Education, 2011.

[17] J. R. Frederiksen and A. Collins, "A systems approach to educational testing," *Educational researcher*, vol. 18, no. 9, pp. 27–32, 1989.

[18] B. S. Bloom, M. Engelhart, E. J. Furst, W. H. Hill, and D. R. Krathwohl, "Taxonomy of educational objectives: Handbook i: Cognitive domain," *New York: David McKay*, vol. 19, p. 56, 1956.

[19] L. W. Anderson, D. R. Krathwohl, and B. S. Bloom, *A taxonomy for learning, teaching, and assessing*. Longman, 2005.

[20] M. Murray, J. Pérez, and M. Guimaraes, "A Model for Using a Capstone Experience as One Method of Assessment of an Information Systems Degree Program," *Journal of Information Systems Education*, vol. 19, no. 2, pp. 197–208, 2008.

[21] M. R. Fellenz, "Toward Fairness in Assessing Student Groupwork: a Protocol for Peer Evaluation of Individual Contributions," *Journal of Management Education*, vol. 30, no. 4, pp. 570–591, 2006.

[22] E. Van Duzer and F. McMartin, "Methods to improve the validity and sensitivity of a self/peer assessment instrument," *Education, IEEE Transactions on*, vol. 43, no. 2, pp. 153–158, 2000.

[23] G. Ryan, L. Marshall, K. Porter, and H. Jia, "Peer, professor and self-evaluation of class participation," *Active Learning in Higher Education*, vol. 8, no. 1, pp. 49–61, 2007.

[24] K. Willey and M. Freeman, "Completing the Learning Cycle: The Role of Formative Feedback When Using Self and Peer Assessment to Improve Teamwork and Engagement." Auckland, NZ: Australasian Association for Engineering Education, 2006.

[25] S. Beyerlein, D. Davis, M. Trevisan, P. Thompson, and O. Harrison, "Assessment framework for capstone design courses," Chicago, IL, 2006.

[26] B. A. Friedman, P. L. Cox, and L. E. Maher, "An Expectancy Theory Motivation Approach to Peer Assessment," *Journal of Management Education*, vol. 32, no. 5, pp. 580–612, 2008.

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

[28] M. Freeman and J. McKenzie, "SPARK, a confidential web–based template for self and peer assessment of student teamwork: benefits of evaluating across different subjects," *British Journal of Educational Technology*, vol. 33, no. 5, pp. 551–569, 2002.

[29] M. W. Ohland, M. Loughry, R. Carter, L. Bullard, R. Felder, C. Finelli, R. Layton, and D. Schmucker, "The comprehensive assessment of team member effectiveness (catme): A new peer evaluation instrument," in *Proceedings of the 2006 ASEE Annual Conference*, 2006.

[30] M. Ohland, R. Layton, M. Loughry, H. Pomeranz, D. Woehr, and E. Salas, "Smarter teamwork: System for the management, assessment, research, training, education, and remediation of teamwork," 2010.

[31] E. Maclellan, "How convincing is alternative assessment for use in higher education?." *Assessment & Evaluation in Higher Education*, vol. 29, no. 3, pp. 311–321, 2004.