The Company Approach to Software Engineering Project Courses

David Broman, Kristian Sandahl and Mohamed Abu Baker

Linköping University Pre-Print

N.B.: When citing this work, cite the original article.

David Broman, Kristian Sandahl and Mohamed Abu Baker, The Company Approach to Software Engineering Project Courses, 2011, Submitted to IEEE Transactions on Education.

Preprint available at: Linköping University Electronic Press
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-69483

The Company Approach to Software Engineering Project Courses

David Broman, *Member, IEEE,* Kristian Sandahl, *Member, IEEE-CS,* and Mohamed Abu Baker
Department of Computer and Information Science, Linköping University
david.broman@liu.se, kristian.sandahl@liu.se, mohamed.abubaker@liu.se

***Abstract*—Teaching larger software engineering project courses at the end of a computing curriculum is a way for students to learn some aspects of real-world jobs in industry. Such courses, often referred to as capstone courses, are effective for learning how to apply e.g., design, testing, and configuration management. However, these courses are typically performed in small teams, giving only a limited realistic perspective of problems faced when working in real companies. This paper describes an alternative approach to classic capstone projects with the aim of being realistic from an organizational, process, and communication perspective. This methodology, called the company approach, is described concerning intended learning outcomes, teaching/learning activities, and assessment tasks. The approach is implemented and evaluated in a larger master student course.**

***Index Terms*—Software Engineering, Capstone Projects, Constructive Alignment, Company Approach**

I. Introduction

PREPARING computer science (CS) and software engineering (SE) students for real-world jobs in industry is a challenging task. Feedback from industry has shown that software engineering topics such as testing, code reviews, release management, and team work are particular important from a real-world perspective [1]. To meet such demands, the ACM/IEEE-CS joint task force on computing curricula recommends to include software engineering projects in a curriculum [2]. In such a project course, students are typically working in teams and developing a larger software system for a particular customer. These project based courses that typically span over the entire last year of a curricula are often referred to as *capstone courses*. Capstone project courses have existed for many years [3] and several success stories have been reported in the literature [4], [5], [6].

However, these projects are typically performed in small development teams (three to eight students) where information is shared among the team members informally. These kind of projects can be successful when learning how to apply e.g., design, testing, and configuration management in a project setting, but give a very limited realistic perspective of the problems faced when working in a real company. For example, a small team can informally communicate the design of a system, but if there are more than 20 developers, a more systematic approach is needed. Hence, the overall pedagogic problem discussed in this paper is how a project course could be designed to enable student learning that gives an industrially realistic view from an *organizational*, *process*, and *communication* perspective.

This paper describes an alternative approach to classic capstone projects. The key components of this approach are:

- Instead of having small project teams, students are organized in *simulated companies*, each consisting of approximately 30 students/employees.
- The organization of the simulated companies faces a transition from a traditional line organization with several departments to an agile organization containing self-organized cross-functional teams.
- The student are working with constrained time budgets, i.e., project members need to time report and prioritize working tasks.

Working in large project teams [7] or organizing as virtual enterprises [8] are not new ideas. However, the main contribution of the new educational methodology presented in this article is the unique combination of larger simulated companies, organizational transformations, and constrained time budgets. This methodology is called the *company approach* to software engineering projects. More specifically, the detailed contributions of this work are as follows:

- The central ideas and concepts of the company approach are outlined based on the theory of *constructive alignment* [9] (Section II).
- The design and implementation are described for a course that is based on this methodology. The course was given two times during the years 2009 and 2010 (Section III).
- An evaluation of the company approach is performed by conducting a quantitative study during the course as well as performing a qualitative survey on the students one and a half year after the course was given (Section IV). The results are discussed in Section V.

II. The Company Approach

This section describes the general ideas and concepts of the company approach methodology in the form of a *course template*, i.e., a generic description from which a specific software engineering course can be designed. The course template is based on the framework of constructive alignment; originally invented by Biggs [9] and further developed by Biggs and Tang [10].

Constructive alignment consists of two separate aspects. The *constructive* aspect concerns the learner's view and is based on the theory of constructivism [11]. The main idea of constructivism is that the student (the learner) construct their own knowledge based on what they already know and by performing activities by them selves. Hence, from this point of view, teaching is not about transferring knowledge to the student by lecturing, but instead to empowering the student for active learning.

The *alignment* aspect relates to the teacher's performance and how a course can be designed to align the following parts:

- *Intended Learning Outcomes (ILOs)* - specific learning outcomes that a student should master after passing the course.
- *Teaching/Learning Activities (TLAs)* - different activities that should result in the intended learning outcomes.
- *Assessment Tasks (ATs)* - tasks for assessing the student's performance in relation to the intended learning outcomes.

The rest of this section presents the company approach methodology based on these three parts.

A. Intended learning outcomes

The overall goal of the company approach is that the student should gain a fundamental understanding of problems and challenges that occur in a real-world software engineering project. However, to be more concrete, the ILOs are categorized within three specific learning perspectives: *organizational* (O1), *process* (P1 and P2), and *communication* (C1 and C2). The intended learning outcomes include, but are not limited to, that the student at the end of the course should be able to:

- (O1) explain the meaning of different roles and organizational structures in a software engineering project.
- (P1) analyze the pros and cons of processes in general, and the difference between classic project management and agile methodologies in particular.
- (P2) analyze the basic fundamental constrains that are inherent in software engineering project.
- (C1) reflect on communication challenges when working in a larger heterogeneous software project.
- (C2) explain the rationales of artifacts, such as architecture document, requirements specifications, and backlogs.

The organizational learning outcome (O1) means that the student should on the one hand be able to explain the underlying meaning of classic SE roles, such as analysts, project managers, product managers, configuration managers, architects, developers, and testers. On the other hand, he/she should also be able to explain the difference of roles in an agile methodology, such as self organized teams, product owners, and scrum masters in the Scrum framework [12].

From the process perspective, learning outcome (P1) means that the student should be acquainted with different process frameworks so that he/she can analyze and draw his/her own conclusions regarding leadership and management of projects. Fundamental constraints in (P2) include e.g., calendar time (time to deadline), budget time (man hours), product functionality, and measurable quality factors.

Communication challenges (C1) aims to capture today's real-world heterogeneous aspect of engineering projects, including the diversity of cultural backgrounds, languages, and technical skills among employees/students. Learning outcome (C2) emphasizes the communication aspect of artifacts typically created in a SE project. The aim is that the student should use and create e.g., an architecture document, because they need it for precise communication between different stakeholders, not that they create it just because the teacher told them to do so.

B. Teaching/learning activities

In this course template, the aim is that the students run the whole project by them selves. The teachers should not lead the project but only guide and coach the students by asking the right questions. One of the main challenges of this kind of course design is to define this framework, so that the students have a high level of freedom to be innovative and at the same time to avoid being too loose so that the intended learning outcomes are missed.

The rest of this section outlines the ten most important teaching/learning activities of the methodology.

1) Role selection: At the beginning of the project, the students apply for a job and a role in the company by formulating a curriculum vitae (CV) and an application letter. If more than one student apply for a position (e.g., department manager), a closed election is performed by the students. Besides the chief executive officer (CEO), that is played by one of the teachers, the students elect their own leaders and key staff. Addresses (O1).

2) Company meetings: Each week the students arrange a company meeting where all employees, including the CEO, are attending. This is the main coordination and communication meeting between employees and is the only formal meeting required by the framework. Addresses (P1), (P2), (C1), and (C2).

3) Requirements elicitation: The students are not given any requirements for the project. Instead, they are told that the CEO has closed an agreement with a potential reference customer to pay for a prestudy of the product. Hence, one of the learning activities is to schedule meetings with the customer and elicit requirements. However, during the project, at least one more customer is introduced with the meaning of learning about changing requirements and development of products targeted for many customers with different priorities. Addresses (P1), (C1) and (C2).

4) Prestudy and business meetings: The prestudy is ended with a simulated business meeting where the customer decides if he/she should purchase the whole project. The students are learning about the challenges of prioritization, convincing a customer, and making commitments with regards to a limited budget and time frame. Addresses (P1), (P2), and (C1).

5) Iteration planning and reviews: At the reset of the project time after the business meeting, the students are directed to use an iterative planning style with fixed iteration

lengths. Both the requirement of having iteration planning (what tasks that should be solved in the next iteration) and iteration review (what was accomplished in the last iteration) are forced in the course framework. The rationale for this approach is to gradually improve the understanding of iterative development and regular customer feedback. Addresses (P1), (C1), and (C2).

6) Time reporting and project planning: During the whole project, all students need to perform time reporting for each hour that they work in the project. There should be a defined number of hours that each student should work, plus/minus some percentage. The overall project should be planned and followed up regarding major milestones, deliverables, and activities. The purpose is to learn about planing and working under risk with limited resources, i.e., to learn about doing "good enough". Addresses (P2).

7) Transformation of organizational structure: The students are originally introduced to a classic organizational structure, consisting of several departments, with roles such as department managers, project managers, testers, and developers. However, they are also encouraged to form cross functional teams, i.e., teams that cut across the departments and include both customer knowledge and technical expertise. The rationale for this gradual transformation of the organization is to make the students learn and reflect upon different organizational and process styles. Addresses (O1) and (P1).

8) Retrospectives and process improvements: Each iteration is followed by a retrospective meeting where the students discuss "what was good", "what went wrong", and "how can we improve it" in the next iteration. Addresses (P1) and (C1).

9) Release planning and expo: At the end of the project, the students present the product on a simulated expo, i.e., they should be given the ability to orally present the benefits with their product in a convincing and user oriented way. Addresses (P1) and (P2).

10) Self reflection and experience documentation: Finally, the students should write down self reflections on their way of working. Both positive and negative findings should be documented and analyzed from both a process and product perspective. Addresses (O1), (P1), (P2), (C1), and (C2).

C. Assessment tasks

The assessment tasks used for evaluating the performance of the intended learning outcomes are divided into two categories: the *process perspective* and the *product perspective*.

The processes perspective, which is the main focus of evaluation, concerns evaluating how students discover problems and how they solve them. Assessments are inherently mostly performed by observation (e.g., at company meetings) and by interaction (with the CEO and the customer). Another important instrument is the time report, i.e., when and how much a student has been working with different parts and their self reflection on their performance. Besides observations are also the documents of self reflection and experience report important basis for evaluation of the process perspective. Assessed learning outcomes for the processes perspective are (O1), (P1), (P2), and (C1).

The product perspective concerns both quality and functionality of the actually delivered product from a customer point of view. Moreover, internal artifacts should be evaluated from a clarity and communication perspective. Addressed learning outcomes for the product perspective are (P2) and (C2).

III. Course Design and Implementation

In 2009 and 2010 a concrete course was designed and implemented according to the company approach at Linköping University in Sweden. This section gives a short overview of the essential design decisions that were made compared to the course template presented in previous section.

A. Course overview

Each year and course had approximately 110-120 students, divided into four different companies. Each company was in turn divided into one research & development (R&D) department and one products & sales (P&S) department. The former included roles such as developers and architects, and the latter e.g., analysts, configuration manager, and testers.

The course was given during one semester, approximately 15 weeks part time, in parallel with other courses. Each student was requested to perform 160h of full time work, +/- 10 percent.

B. Students

The students formed a heterogeneous group. 81% of the students were male and 19% female. 57% of the students were Swedish students studying their fourth year of an engineering program combining computer science with management and/or media technology. The other 43% of the students were exchange or master students studying master of science in software engineering or in computer science.

C. Teachers

During the two years, different teachers and teaching assistants (TAs) were teaching the course. In the first year the first author of this paper and designer of this course played the role of CEO and supervised about planning and processes. Two TAs were supervisors, coaching students with aspects such as architecture, requirements elicitation, and testing. In the second year, four TAs were responsible for all aspects, including planning and processes. The CEO had in this year a more passive role. One TA was teaching both years and the acting customer was the same for both years.

D. Customers

This course uses a played but realistic customer. The product that all companies were developing was a simple and minimalistic *enterprise resource planning (ERP)* system, especially tailored for universities. Hence, one of the professors at the department played the customer role, but he was in fact playing himself and his usage and need for a new ERP system. After the tollgate business meeting, another customer from another department was introduced, with slightly different needs. The main rationale for choosing a played customer instead of a real company from industry was the benefit of having control of the whole framework, including requirements.

IV. Evaluation

The course has been evaluated with a quantitative study during the course, and a qualitative survey after the course.

For the quantitative study anonymous questionnaires were used to collect the data from the students. The answers were collected per company in a closed envelope, and the company IDs were randomly coded afterwards. The questionnaires were collected in December 2009, and December 2010. Each round had 4 companies, and a total number of 187 questionnaires were collected (83 % response rate). The data collected from 2009 and 2010, were analyzed together to compare two instances of the course.

For the qualitative survey we contacted the students from 2009 that have kept contact with the examiner in LinkedIn. In April 2011 a survey was made with 3 questions to the 62 students and 19 answers were received (31%). Of these, 11 were working in industry.

A. Statistical analysis of the quantitative study

The data was entered in SPSS (Originally: Statistical Package for the Social Sciences), with 7 independent variables (*round*, *age*, *sex*, *industrial experience*, *software development experience*, *curriculum*, and *result from theory exam*), and 18 dependent variables (the questions). For each of the questions, the students answered using one of six options (e.g., *Strongly Disagree*, *Disagree*, *Neutral*, *Agree*, *Strongly Agree*, and *N/A*). Hence, the data was analyzed as ordinal scale measurements.

To measure the overall impression the students aswered: "I believe that the company approach in this course - i.e., that we are organized as a simulated company - makes our experience industry relevant", showed the strongest agreement in the study, See Fig. 1. This dependent variable was then examined with all independent variables, but no difference in the strong agreement was found, not even among people with their own industrial experience. This was verified by converting the categories of support to a Likert scale and running Kruskal-Wallis one-way analysis of variance [13].

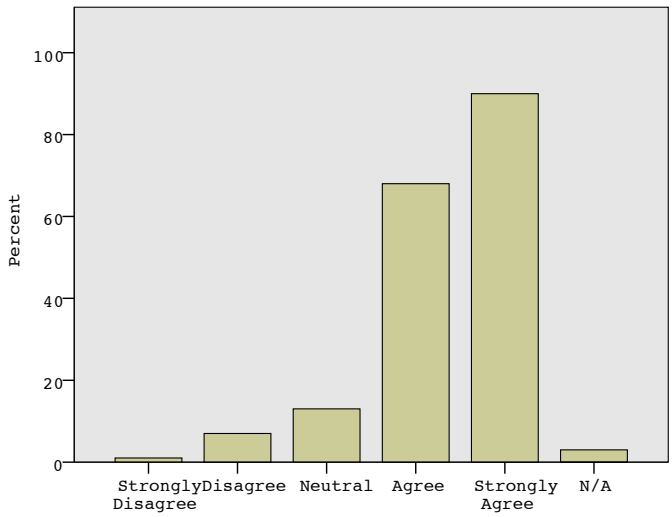

Fig. 1. The agreement of the relevance of the Company Approach.

The students in the course strongly agreed that they were working harder with numerical grading, on a 4-level scale (e.g. F/C/B/A) compared to a 2-level fail/pass grading. See statement 1 in Table 1.

The course has specified detailed criteria for the grading levels, and there was a strong agreement from the students that this influenced their way of working. See statement 2 in Table 1. Taken together, in this context grading is a key concern in project course design.

In the project the students got experience both from a line-organization as well as cross-functional teams. When asked about their preferences, students were strongly agreeing that working in cross-functional teams was more motivating. See statement 3 in Table 1.

An agreement was also observed in the preference for agile methods and that the cross-functional teams actually were self-managed, which means that the company leaders only told what they expected from the teams, not how they should work. When asked if the department manager was a coaching leader, students generally agreed but there was an observable difference in the pattern between the different companies in all 3 rounds. See statement 4 in Table 1.

B. Findings in the qualitative survey

One of the questions in the survey was about which parts of the course that are of value in professional work. The 19 answers were analyzed and we found 5 frequently occurring answers:

1) Collaborating with other people in the company and the teams was a good experience (63 % of the answers).
2) The importance of planning and managing time was practiced (53 % of the answers).
3) The setting of the course gives industry relevant experience (53 % of the answers).
4) Agile methods, such as SCRUM, were practiced. (47 % of the answers).
5) The course gave a good overview of the entire life-cycle of a software project (26 % of the answers).

C. Threats of Validity

One of the main problems with the quantitative study is that most students do not have any industry experience and might therefore not be capable of answering how realistic the approach is.

Regarding the qualitative study, the students are self-selected and the answers give only details about which parts that were appreciated by positive students.

V. Discussion

This section discusses different aspects of the company approach in relation to the results of the performed study.

A. Aligning teaching/learning activities

The proposed approach advocate a *student-centered model* of learning. In doing so, the teaching/learning activities must be aligned to the intended learning outcomes, by focusing on the

TABLE I
ANSWERS TO SELECTED QUESTIONS. EACH COLUMN INDICATE THE PERCENTAGE OF INDIVIDUAL ANSWERS FROM THE COLLECTED ANSWERS OF DECEMBER 2009 AND DECEMBER 2010. NOTATION: INVALID (I), STRONGLY DISAGREE (SD), DISAGREE (D), NEUTRAL (N), AGREE (A), STRONGLY AGREE (SA)

Statements	I	SD	D	N	A	SA	N/A
1. Numerical grading compared to two grades makes me work harder.	0	2.1	8.0	13.0	22.5	53.5	0
2. Grading criteria influences our ways of working.	3.2	0.5	4.3	11.8	38.5	41.2	0.5
3. Cross functional teams make me more motivated to do a good job.	2.7	0.5	4.8	15.5	42.2	26.2	8.0
4. Department managers have a coaching leadership style.	0.5	2.1	10.2	14.4	39.0	29.9	3.7

activities that the student performs. Hence, to quote Shuell [14, p. 429]

> *"Without taking away from the important role played by the teacher, it is helpful to remember that what the student does is actually more important in determining what is learned than what the teacher does."*

The IOLs stressing the organizational (O1) and process (P1 and P2) perspectives are addressed in ten different learning activities. Central to all these activities are that they are performed by the students, i.e., the teacher has only a coaching and observing role. Most notable, a deep learning approach is suggested for learning about the differences of agile processes compared to traditional project planning. Using a surface approach [15], central agile concepts such as time-boxing, cross-functional teams, and self-organization would be described and practiced without the student understanding the deeper *meaning* of the concepts. Using the organizational transformation approach where the students start using a line organization with two departments and fixed roles and then gradually transit to cross-functional teams with time-boxed iterations, the hypothesis is that students can deeper analyze pros and cons with the two approaches. For example, from the qualitative study (Table I) clear agreement can be seen that cross functional teams tend to make the students more motivated. Without having a line organization to compare with, the students would not gain such an insight.

In the survey (performed one and a half year after finishing the course) the students were asked what they learned in the course. The main answers concerned the need of processes and management of time. Several of our learning activities, such as time reporting, iteration planning, review, and changing requirements directly relate to both this observation and IOLs (P1) and (P2). Direct feedback from students shows that time reporting is not always appreciated during the course. However, a tendency is shown in the survey result that it is still a very effective way of emphasizing the problem of getting good professional results within a limited time budget.

When focusing on what the student does, the teacher uses a *guide on the side* pedagogy instead of a traditional *sage on the stage* approach [16]. One interesting question is then how dependent the course is on specific teachers? An interesting observation is that the course changed most teachers between the years 2009 and 2010, but kept very good course evaluations. On a scale 1-5 (where 5 is the best) the official anonymous student course evaluation gave an overall rating of 4.44 (year 2009) and 4.16 (year 2010), with an answer frequency of 30% and 25% respectively.

This clearly indicates that the course template is not entirely dependent on one specific teacher to be successful.

B. Aligning assessment tasks and grading

Aligning assessment tasks and grading strategies to intended learning outcomes tend to be much more difficult compared to aligning learning activities. Two of the main challenges are how to make the learning outcomes *measurable* and how to *grade teams*.

In the design and implementation of the company approach, the students were assessed by observing meetings, assessing artifacts, experience documents, as well as reflected on their interaction with the CEO and the customers. In the current implementation assessment and grading are mainly at a team level, where the individual evaluation only concerns the divergence of top students or underperforming students. Even though there are many ways to combine team and individual assessment [17], the main rationale for focusing on the team level is the importance that students work together towards a common goal. Individual assessment has a risk of reducing the incentive for collaboration.

C. Constructivism and iterative learning

Central to our approach and to the idea of constructive alignment is *constructivism*. The idea that students construct new knowledge by performing activities and building it on their previous knowledge, requires an existing model of the underlying concepts. According to Ben-Ari [11], such a model is typically absent for students learning computer science and must be explicitly taught. A similar problem seems to exist when teaching real-world software engineering projects. Unsurprisingly, the students do not have a mental model of what a project is before they have actual being part of one. Such mental models are currently taught using classic theoretical lectures, i.e., about different life-cycle models and process frameworks. However, is this enough for a student to get a deep understanding of a real SE process?

In the current implementation of the company approach a pedagogic concept called *iterative learning* is used. The approach of using iterative learning goes hand in hand with well established software engineering principles of iterative software development and has a strong connection to *formative assessment method*, i.e., assessments that are fruitful for continued student learning [18]. The concept of iterative learning described here concerns learning during a course, but can also be generalized for a curriculum [19]. This approach is

particular important when learning *functional knowledge*, i.e., knowledge that is based on understanding and performance, compared to *declarativ knowledge*, i.e., pure knowledge of "knowing about things" [10].

The former kind of deeper knowledge of SE projects, i.e., to be able to do "good enough" to meet customer expectations with a constrained budget is what here is meant by being "professional". The approach of iterative learning aims at taking the students a first step in that direction.

D. Industry relevance

How well does the company approach solve the problem of designing a software engineering project course that gives an industrially realistic view from an organizational, process, and communication perspective?

To enable student learning of functional knowledge related to company organization and processes, it is necessary to have a strong educational culture that focus on *learning how to learn from experience*. A coaching style of teaching is vital where teachers act as supervisors and teach by asking questions, not giving the answers. This coaching style of leadership ought to be implemented in the whole organizational structure, even for student elected managers. This fact is confirmed in the quantitative study (Statement 4, Table I).

Both the academic teaching environment at universities as well as real companies in industry need to fulfill their tasks within economic constraints. Limited teaching resources are one of the main challenges within higher education; especially regarding assessment and giving student feedback. In the proposed methodology, the aim of meeting these challenges has been to empower the students to run the project, where the teachers are coaches focusing on being present and available as much as possible. Empowering the students and at the same time stating clear demands on deliverables and time reporting give also a positive effect of understanding real economic constraints within companies.

VI. Conclusion

This paper describes a new methodology for running software engineering project courses, called the company approach. The main focus is to provide an industrially realistic setting from an organizational, process, and communication perspective. The quantitative survey indicated that the company approach gave industrial relevant experience. This is also partially confirmed in the qualitative survey performed on students one and a half year after they finished the course.

Acknowledgment

This research was funded by the Department of Computer and Information Science, Linköping University, Sweden.

References

[1] Interim Review Task Force, "Computer Science Curriculum 2008 - An Interim Revision of CS 2001," 2008.

[2] Joint Task Force on Computing Curricula, "Software Engineering 2004 - Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering," 2004.

[3] R. H. Todd, S. P. Magleby, C. D. Sorensen, B. R. Swan, and D. K. Anthony, "A survey of capstone engineering courses in north america," *Journal of Engineering Education*, vol. 84, pp. 165–174, 1995.

[4] A. Goold, "Providing process for projects in capstone courses," in *Proceedings of the 8th annual conference on Innovation and technology in computer science education*, ser. ITiCSE '03. New York, USA: ACM Press, 2003, pp. 26–29.

[5] S. Karunasekera and K. Bedse, "Preparing Software Engineering Graduates for an Industry Career," in *Proceedings of the 20th Conference on Software Engineering Education and Training*. IEEE Press, 2007, pp. 97–106.

[6] D. A. Umphress, D. Hendrix, and J. H. Cross, "Software process in the classroom: the capstone project experience," *IEEE Software*, vol. 19, no. 5, pp. 78–85, 2002.

[7] D. Coppit and J. M. Haddox-Schatz, "Large team projects in software engineering courses," in *Proceedings of the 36th SIGCSE technical symposium on Computer science education*, ser. SIGCSE '05. New York, USA: ACM Press, 2005, pp. 137–141.

[8] F. Meawad, "The Virtual Agile Enterprise: Making the Most of a Software Engineering Course," in *Proceedings of the 24th IEEE Conference on Software Engineering Education and Training*, 2011, pp. 324–332.

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

[10] J. Biggs and C. Tang, *Teaching for Quality Learning at University*, 3rd ed. Open University Press, 2007.

[11] M. Ben-Ari, "Constructivism in computer science education," *Journal of Computers in Mathematics and Science Teaching*, vol. 20, no. 1, pp. 45–71, 2001.

[12] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. Prentice Hall, 2001.

[13] W. H. Kruskal and W. A. Wallis, "Use of Ranks in One-Criterion Variance Analysis," *Journal of the American Statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.

[14] T. J. Shuell, "Cognitive conceptions of learning," *Review of Educational Research*, vol. 56, no. 4, pp. 411–436, 1986.

[15] F. Marton and R. Säljö, "On qualitative differnces in learning: I - outcome and process," *British Journal of Educational Psychology*, vol. 46, pp. 4–11, 1976.

[16] A. King, "From sage on the stage to guide on the side," *College Teaching*, vol. 41, no. 1, 1993.

[17] M. Lejk and M. Wyvill, "A survey of methods of deriving individual grades from group assessments," *Assessment & Evaluation in Higher Education*, vol. 21, no. 3, 1996.

[18] S. M. Brookhart, "Assessment theory for college classrooms," *New Directions for Teaching and Learning*, vol. 2004, no. 100, pp. 5–14, 2004.

[19] D. Broman, "Should Software Engineering Projects be the Backbone or the Tail of Computing Curricula?" in *Proceedings of the 23th IEEE Conference on Software Engineering Education and Training*, Pittsburgh, USA, 2010, pp. 153–156.