An Interdisciplinary Approach to Software Engineering Teaching: An Experience Report

Gláucia Braga e Silva[1], **Daniel Mendes Barbosa**[1], **Fabrício A. Silva**[1]

[1]Institute of Exact and Technological Sciences - Federal University of Viçosa (UFV)
Florestal – MG – Brazil

{glaucia, danielmendes, fabricio.asilva}@ufv.br

Abstract. *This work presents the report of an interdisciplinary approach for teaching Software Engineering that involves four related disciplines: Software Engineering II, Software Architecture, Database Systems, and Object-Oriented Programming. The approach was conducted through an specific methodology in which students were grouped together and had to assume roles and responsibilities in the scope of each discipline. In addition, computational tools were used to support collaborative tasks and evaluation and monitoring mechanisms were also included. This interdisciplinary approach was adopted in a Computer Science major course during 2015 and 2016. The results reveal positive impacts in motivation, as well as in learning aspects of the involved students.*

1. Introduction

Due to the increasing demand for software applications in industry, there are many challenges of teaching software engineering [Zeidmane and Cernajeva 2011]. The industry requires more and more professionals with multi-functional skills and capable of working in multidisciplinary environments. Although over the last 45 years the practice in Software Engineering has made significant progress, there are significant gaps between the teaching and the needs of the software industry [Bass 2016, Moreno et al. 2012]. The teaching of Software Engineering is a hard task because there are a lot of competences and abilities to be taught in order to prepare a professional which combines technical knowledge, experience, and ability to interact with clients [Jazayeri 2004, Teel et al. 2012]. According to Nurkkala and Brandle [Nurkkala and Brandle 2011], traditional approaches of Software Engineering teaching present the following problems: no real products; short duration that causes an artificial time constraint and requires projects with low complexity; high turnover of the students; low projects complexity; no software maintenance; and lack of interaction with real customer. Furthermore, because of the large amount of theoretical concepts, Software Engineering is often considered by students as a boring subject [Teel et al. 2012].

In this context, educators must be continuously engaged in the creation of new teaching strategies that include practices compatible with the current software industry trends and that motivate students to appreciate the importance of this discipline for their careers. Some of these issues can be addressed by using interdisciplinarity to teach Software Engineering since it is a key topic in computing education [Teel et al. 2012] and it must be taught in an integrated way. Interdisciplinary teaching involves the interactions between two or more academic disciplines with a common goal. In this context, educators can bring to the classroom an environment that engages students and helps them to

develop knowledge, insights, problem solving skills and self-confidence. The use of interdisciplinary teaching contributes to a more complete and integrated academic formation of the students who will be better prepared and qualified for the industry.

This work presents an experience report of an interdisciplinary approach for teaching Software Engineering in a Computer Science major course of our University, during 2015 and 2016. The approach was conducted through an specific methodology designed to develop a software product around the teaching of four related disciplines: Software Engineering II, Software Architecture, Database Systems, and Object-Oriented Programming. The approach was proposed with the objective of bringing industry-related scenarios to the academy, allowing students to experience team work, share their knowledge and learn by experimentation [Ghezzi and Mandrioli 2005]. The approach adopts a problem-based learning process in which students can construct and acquire knowledge, enhance group collaboration and communication while develop a software project. In addition, technical capabilities, such as project management, requirements tracking, configuration management, software quality and test and collaboration tools can be experienced in a hands-on manner [Teel et al. 2012]. Our approach has been used to engage and empower students' learning in an undergraduate computer science course. In this paper, we summarize our experiences and lessons learned. The results reveal positive impacts in motivational as well as in learning aspects of the involved students.

The paper is organized as follows: section 2 discusses some related works. Section 3 describes the proposed approach, details its methodology, and presents the results of two consecutive years of its adoption. Finally, in section 4 we conclude the report.

2. Related Works

This section presents studies that address new strategies to teach Software Engineering which explore interdisciplinarity, experiential learning and non-technical skills. Marsicano et al. [Marsicano et al. 2016] present a teaching method that integrates the disciplines Requirements Engineering and Process modeling in an undergraduate course. The method was analyzed in two ways: grades and feedback on technical report analysis. Schaetter et al. [Schaetter et al. 2009] describe a multidisciplinary approach to teach Software Engineering based on teaching and learning methods. Through interdisciplinary projects, students are trained in software projects under realistic conditions. The results pointed that this approach has had significantly enhanced the students employability, according to feedback from their industrial partners. Chen and Chong [Chen and Chong 2011] present an study which introduces a meetings-flow approach to help in the instruction of student teamwork and to formalize stakeholder participation. The authors conducted a quantitative investigation which assessed the project and examined the numerical benefits that the approach brought to the project development. In addition, the study conducted group interviews to discuss the qualitative and educational effects of the approach. Bareiss and Griss [Bareiss and Griss 2008] discuss the use of teaching methods, coaching, and feedback in the Carnegie Mellon University and reports the positive impacts in the students formation such as competitive advantage and salary increases. The methods are based on student-centered learning where students are encouraged to discover knowledge themselves and to learn by doing and they are evaluated based on what they produce. Letouze et al [Letouze et al. 2016] propose a Problem-Based Learning approach to develop a web system for managing academic projects. In their ap-

proach, role-play scenarios were used in order to prepare students and to improve abilities within a role as development team member. Our approach is also based on interdisciplinarity since it involves the integrated teaching of Software Engineering in four disciplines in a Computer Science major course. However, we also propose a methodology to structure the approach in stages, allowing its replication in other academic institutions that offer, in the same academic semester, disciplines in the areas of Software Engineering, Database Systems and Programming. The proposed approach is not restricted to the application of Software Engineering concepts, but it aims to explore the interactions between the various roles of a software process, the collaborative production of artifacts and the use of computational tools, simulating situations commonly found in the real scenarios of the job market. Furthermore, the new realities of teaching Software Engineering [Jazayeri 2004] are covered by the proposed approach, as it tackles current issues such as software evolution, software quality, tools and environments. In addition, the approach provides a proper environment to develop non-technical skills such as communication and ability to work as a team.

3. Structuring of the Interdisciplinary Approach and Experience Report

This section presents an experience report of an interdisciplinary approach for Software Engineering teaching in the Computer Science undergraduate course at Federal University of Viçosa (UFV) - campus Florestal. The approach involved the integrated teaching of four related disciplines present in the course curriculum. All disciplines are offered each even semester, as shown in Table 1, and they are strongly related since their contents are complementary. It is important to note that Software Engineering I is a previous discipline, offered on the fifth semester of the course, and is responsible for the fundamentals of the area. However this discipline is not considered in our approach.

Table 1. Involved Disciplines

Discipline	Initials	Semester	Scope
Software Engineering II	SE	sixth	Process Management and Test
Object-Oriented Programming	OOP	fourth	Coding according to the O.O. paradigm
Database Systems	DB	sixth	Modeling, Design and Queries
Software Architecture	SA	eighth	Specification, Design and Code Integration

The scope of each discipline within the approach was defined based on their technical scope, as recommended by the course analytical program, but explored under an interdisciplinary perspective. Table 2 illustrates the interdisciplinary relations involved.

Table 3 illustrates the relation of students for each discipline (SE, OOP, DB and SA) in both editions (2015-2 and 2016-2) as well as the intersections of students enrolled in more than one discipline. In both editions, there were students who attended only one of these four disciplines, as well as others who attended two, three or four of them simultaneously. In 2015-2, 38 students and 2 instructors (1 professor was responsible for 3 disciplines) participated of the approach. In 2016-2, this number increased to 56 students and 3 instructors (1 professor was responsible for 2 disciplines).

To guide the interdisciplinary approach, we propose a methodology comprised by three stages: planning, execution and control, and closing (Figure 1). The planning

Table 2. Interdisciplinary Relations

From	To	Activities and Artifacts
SE	OOP	schedule, configuration management and software patterns
	DB	schedule, configuration management, quality (DB models and SQL scripts)
	SA	schedule, configuration management, GUI design, quality, tests (UML models and DB models)
SA	OOP	model specifications (UML use cases and class diagrams)
	DB	model specifications (UML class diagrams)
	SE	model specifications (UML use cases and class diagrams) and integrated code (Model-View-Controller architecture)
OOP	DB	specification of database queries
	SA	code of Model and Controller layers (to be integrated)
DB	SE	DB models, DB SQL scripts (creation and queries) to be tested
	SA	DB models and DB SQL queries (to be integrated).

Table 3. Students per discipline

Year	OOP	DB	SE	SA	$OOP \cap DB$	$OOP \cap SE$	$DB \cap SE$	$SA \cap DB$	$OOP \cap DB \cap SE$	Total
2015-2	20	10	10	5	2	1	5	-	1	38
2016-2	29	25	20	8	10	4	13	1	2	56

stage is responsible for the following definitions: case study, roles and responsibilities, and computational tools. This stage is also responsible for the development of the project schedule. In the execution and control stage, the schedule is controlled through the evaluation of the work products and the continuous feedback from the participants. At the end, in the closing stage, the approach is evaluated and the lessons learned are registered.

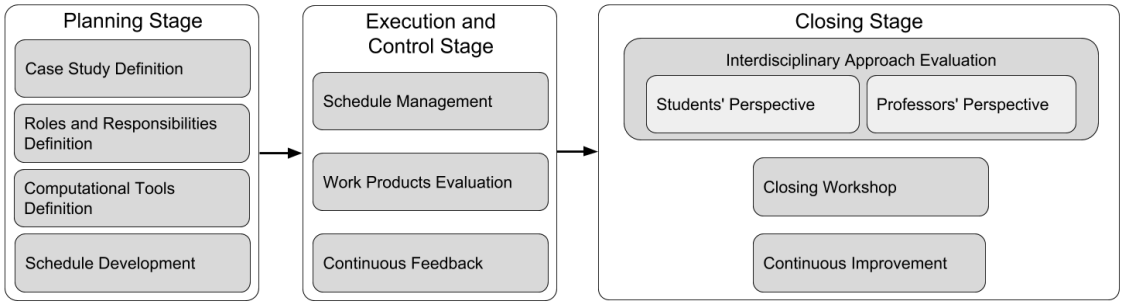


Figure 1. Methodology of the Interdisciplinary Approach

The following sections detail the proposed methodology stages and describe the results of both editions.

3.1. Planning Stage

The four management activities of this stage are conducted by the professors with the support of some students of SE and SA disciplines. The first activity of the planning stage is the *case study definition*. In order to cover the curricular content of the involved disciplines, this case study should address the development of a software product using the

object orientation paradigm, data persistence in relational databases, design patterns and frameworks. An academic system to control attendance at the university and a prototype web-based system to control denunciations of focus of *Aedes Aegypti* were the choices in the 2015-2 and 2016-2 editions, respectively.

The *roles and responsibilities definition* was based on the technical nature of each discipline, the activities planned in each one and the knowledge level of the students involved, with students from the more advanced periods (SA and SE) taking on more heterogeneous roles and of greater responsibility in the project. Because of that and considering that OOP and DB disciplines are offered in the 4th and 6th periods respectively, the activities of these two disciplines were uniform and so all the students of each one were divided into teams and each team performed the same tasks. Table 4 presents the defined roles and the corresponding responsibilities and work products for each discipline.

Table 4. Roles and Responsibilities

Role	Responsibility	Work Products	Discipline
Project Manager	Schedule and Workflow Control	Project Schedule	SE
Configuration Manager	Change and Version Control	SCM Plan	SE
GUI Designer	Design of Graphical User Interfaces	GUI prototypes	SE
Quality Analyst	Quality Assurance (models, codes, ...)	Quality Assurance Plan	SE
Tester	Software Tests	Test Plan, Test Cases, Test Reports	SE
Discipline Leader	Support to students and professors	Communication Plan, Meetings Reports	SE
Software Architect	Software specification and Design	Use Case, Classes and Components	SA
	Revision of database models	Improved Models	SA
Senior programmer	Revision of codes	Improved Codes	SA
	Coding of View layer	Boundary Classes	SA
	Integration code between layers (MVC)	Integrated code	SA
O.O. Programmer	Coding of Model layer	Entity Classes	OOP
	Coding of Controller Layer	Controller Classes	OOP
	Specification of database queries	Queries demands report	OOP
DB Designer	DB Logical Modeling	DB Logical Model	DB
	DB Physical Modeling	DB Physical Model	DB
DB Analyst	Database creation and data insertion	SQL Scripts	DB
	Development of SQL Queries	SQL Queries	DB

As shown in Table 4, in order to correct some workflow and communication problems, four SE students took on the role of leaders from each of the four disciplines to improve the teamwork. The discipline leaders were responsible for supporting the students in the execution of their tasks and also the professors in the monitoring of performed tasks and produced results. Compared with the 2015-2 edition, we noted that the performance of the discipline leaders in 2016-2 edition was central to improve teamwork and especially the relationships between disciplines. This is a very important aspect because students working in teams to complete software tasks is an effective method to learn necessary teamwork skills required in software industry[Shuto et al. 2016].

As the case study involves a software product development, the *Planning Stage* includes the *definition of computational tools* to support both process and collaborative work. Several tools were used during the project execution (Figure 2) and helped the work conduction by the professors in management and didactic support contexts. In addition, other tools supported the students in carrying out their tasks in the software process scope.

In the didactic support context, theoretical contents and task specifications of each

Context	Computational Support	Tools	
		2015-2	2016-2
Didactic Support	Virtual Learning Environment	System 1 (@OurUniversity)	System 1 (@OurUniversity)
	Control of Attendance and Grades	System 2 (@OurUniversity)	System 2 (@OurUniversity)
	Feedback	Google Forms, Padlet	Google Forms
Project Management	Schedule Management	Gantt Project	GanttPro
	Workflow Management	-	Trello
	Communication Management	Gmail, Hotmail, GoogleGroups	Gmail, Hotmail
Software Development	UML Modeling	Astah Community	Astah Community
	Database Modeling	MySQL Workbench	MySQL Workbench
	Design of Graphical User Interfaces	Moqups	Moqups
	Database Management	MySQL Server	MySQL Server
	Code development (IDE)	NetBeans	NetBeans
	Frameworks	JSF, PrimeFaces	JSF, Hibernate, SpringSecurity, ...
	Web Server or Application Server	Apache Tomcat	Apache Tomcat
	Unit Tests	-	JUnit
	Funcional Tests	Selenium	Selenium, Sikuli
	Security Tests	-	ZAP, IronWASP, SqlMap
	Configuration Management (Issue Tracking)	Mantis Bug Tracker	Mantis Bug Tracker
	Configuration Management (Version Control)	Subversion	Git

Figure 2. Support of Computational Tools

of the disciplines were available to students through a virtual learning environment. The students could view their grades in each of the participating disciplines through the official grading system. In the context of software development, tools were used for UML modeling, data modeling and system interfaces prototyping. The resultant artifacts of that were then used as inputs for coding and testing. All produced artifacts were submitted to configuration control, using issue tracking tools and version control. Finally, in the project management context, there was a small variation of the tools used. In the 2016-2 edition, the *Trello* tool was adopted, with the purpose of supporting the discipline leaders and professors in the assignment and monitoring of tasks in each discipline. The tool was also used by the discipline leaders to improve communication with the students/teams in each discipline. It should be noted that most of the adopted tools are widely used in software development projects, either in the academy or in the software industry.

Schedule Development is the last activity of this first stage, in which we developed a schedule, defining project activities, their deadlines and interdependence relationships. It was essential to workflow management in the *Execution and Control stage*.

3.2. Execution and Control Stage

At this stage, according to the developed schedule, the deadlines and results of the planned tasks were monitored and evaluated by the professors, supported by the students with the roles of project managers and discipline leaders. The planned tasks in the project schedule were executed by the students and monitored by the professors, with the support of project managers and discipline leaders. Professors, project managers and discipline leaders (2016-2) continuously interacted throughout the semester, through meetings and exchange of emails. Throughout the project some adjustments were made to the schedule, with readjustments in dates and time lengths of tasks that had not been correctly planned initially. In the 2016-2 edition, each discipline leader created a board in the *Trello* tool with the tasks of the teams in their respective discipline, which was continuously monitored, always informing project managers of the status of the tasks performed.

From the technical point-of-view, there were many interactions between the different roles during the academic semester. In both editions, the students of the SA presented the system requirements and models to the other students, in reserved classes of DB, OOP and SE. The students attended design technical meetings, some during the classes of the involved disciplines and others outside the classroom, in person or in a virtual way. Students from the same team (DB and OOP) or role (SA and SE) were also in constant interaction, exchanging information in e-mail systems and discussion groups.

In the 2015-2 edition, professors were responsible for direct interactions with students in their respective disciplines. This caused some schedule delays, since professors have several other assignments, and in some cases such interactions with students were only possible in a class in the following week. But in the 2016-2 edition, with the participation of the discipline leaders, we observed that the information arrived more efficiently to the students of each discipline.

To control the execution of the project tasks, in each edition was defined specific *work products evaluation* for each discipline. The evaluation was carried out by the responsible professors, based on the students pre-evaluation with roles of discipline leaders (SE), senior programmers (SA) and testers (SE). In cases where teams did the same task (OOP and DB), the professor selected the best result to follow to the next tasks. At the moment of the selection, the professor then assigned the grades to each team, using an "award" strategy for the best results presented (class models, data models, source code). Professors then discussed these results with the students, in order to clarify all doubts of that task. In addition to this evaluation within the project itself, the content learned by the students could also be evaluated in other activities within each discipline, including written tests. The data stored in the repositories of the configuration control tools were also analyzed by the professors (2015-2 and 2016-2) and discipline leaders (2016-2), in order to verify the fulfillment of the deadlines of the schedule, the quality of the productions and the level of involvement of team members.

Professors, project managers and discipline leaders received *continuous feedback* from other participants throughout the semester, whether in the classroom, by e-mail or through computational support tools. In 2016-2, students reported their doubts and suggestions to the discipline leaders and so the students with this role reported to the professors who applied some adjustments throughout the semester.

3.3. Closing Stage

To evaluate the interdisciplinary approach itself, we have collected additional information at the ending of each edition. In the following we discuss the evaluation of the approach under the students' and professors' perspectives.

To evaluate the approach from a *students' perspectives*, every participating student answered a survey at the end of each edition. Each question covers a topic and contains five descriptive alternatives representing a level each: very high, high, medium, low, very low. The main results reveal that 71.4% of the students classify as *high* or *very high* their motivation to conclude the disciplines because of the interdisciplinary approach in 2015-2. We also observed an increasing of this measure from 71.4% to 76.8% in 2016-2. The survey also reveals that 89.3% and 79.1% of the students classified as *very high* or *high* the impact of the project in the learning during the editions of 2015-2 and 2016-2,

respectively, which means they considered the project as a motivator for their studies. On the other hand, some students have pointed out failures on the communication process in both editions. In 2016-2, this problem could be bigger due to increased number of participants, but we adopted *Trello* to organize the tasks and created the role of discipline leader, which was considered important to 67.4% of the students.

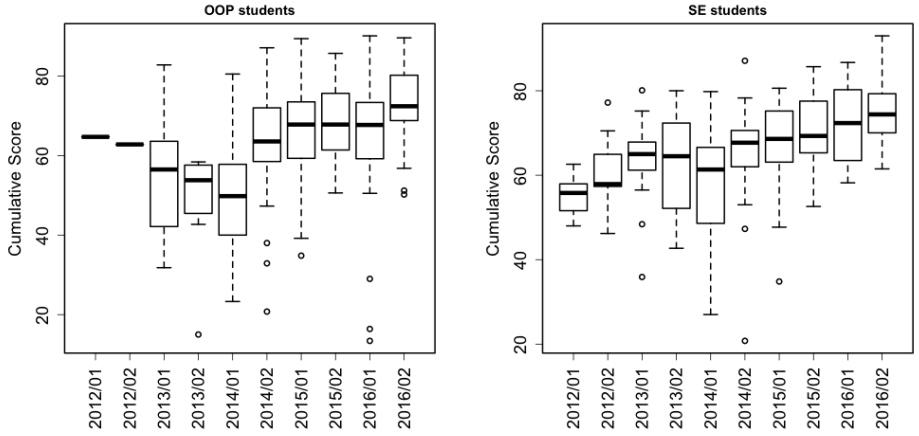
To assess how the interdisciplinary project affects the students' academic performance, from a *professors' perspective*, we analyze their grades during and after the period of the project. To this end, we consider as metric the cumulative score that is the grades' weighted average in which the weights are relative to each discipline's credits. This metric indicates well the students' performance and is officially adopted at the university where the project was conducted. Figure 3 depicts the cumulative scores for the students that were enrolled at Object-Oriented Programming discipline (Figure 3(a)) and Software Engineering II (Figure 3(b)). For both cases, it is clear the high trend in the scores, especially from 2015-2 term on, when the project was firstly adopted.

To reinforce this observation, we compute the Pearson's correlation coefficient of each student. The Pearson's coefficient varies from -1 to 1, indicating an increasing trend when positive. The assessed values were higher than 0.18 for more than half of the students that attend the Object-Oriented Programming discipline. The trend is even higher for the students that attend the Software Engineering II discipline, for whom the Pearson's coefficient was higher than 0.18 for over 75% of the students, and higher than 0.82 for more than half of them. This positive correlation reinforces that the interdisciplinary project was beneficial to improve the students' grades.

As described in Section 3.1, some students were enrolled in more than one discipline simultaneously. To assess whether this is beneficial or not, we evaluate the scores of these students separately. To avoid biased information, we ignore the student that was enrolled in the four disciplines. Figure 4 depicts the cumulative scores of the students that were enrolled in two (Figure 4(a)) or in three (Figure 4(b)) disciplines simultaneously. As can be noted, the students enrolled in three disciplines present an increasing trending in their scores, with a Pearson coefficient of more than 0.34 for over half of the students. The same observation is valid for the students that were enrolled in two disciplines simultaneously, which present a Pearson coefficient of more than 0.27 for over half of them.

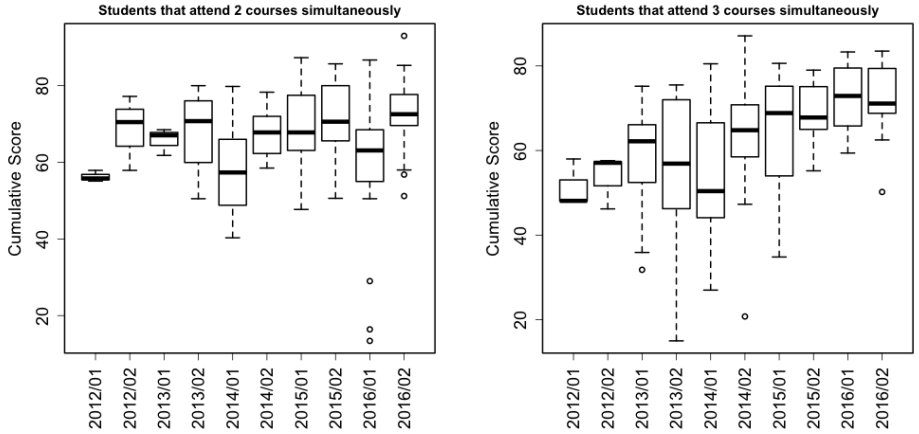
In summary, based on the cumulative scores and on the survey answers, it is possible to observe that the interdisciplinary project motivates the students to dedicate more to the studies, leading to better grades. At the end of the term, the students organized a *closing workshop* to discuss how the activities were conducted during the project. This way, the entire group get to know the overall project from different perspectives. The workshop is also used as an event where the students celebrate their accomplishments.

It is important to state that the methodology presented and adopted in this study should be adapted and improved constantly. To this end, all involved professors discuss the results and the students' feedback to detect potential improvements that would affect positively the upcoming editions. The changes adopted between the editions 2015-2 and 2016-2 in terms of roles and responsibilities (Table 4) and tools (Figure 2) are examples of results from the *continuous improvement*.



(a) OOP Students' Cumulative Scores (b) SE Students' Cumulative Scores

Figure 3. Students' cumulative scores increasing trend.



(a) Cumulative scores of students enrolled in 2 disciplines simultaneously (b) Cumulative scores of students enrolled in 3 disciplines simultaneously

Figure 4. Cumulative scores of students enrolled in 2 or 3 disciplines. It is possible to observe high trend in the grades of students enrolled in 3 disciplines.

4. Final Remarks

In this study, we present a case study of the application of an interdisciplinary project involving four software-engineering-related disciplines. The interdisciplinary approach was conducted in a way to replicate how software are developed in the industry. Thus, students could be familiar with processes, best practices and tools adopted in the industry. The survey results reveal that the project was considered as motivator to students, since they worked as a team and had to interact with other students of different disciplines. In addition, it was possible to observe a high trend in the students' grades that have attend the project, not only in the involved disciplines but in the overall computer science courses.

As future work, we plan to include other disciplines from different areas other than computer science, with the objective of integrating the computer science students with potential players in the software development process.

References

Bareiss, R. and Griss, M. L. (2008). A story-centered, learn-by-doing approach to software engineering education. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2008*, pages 221–225.

Bass, M. (2016). Software engineering education in the new world: What needs to change? In *Proceedings. 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 213–221.

Chen, C.-Y. and Chong, P. P. (2011). Software engineering education: A study on conducting collaborative senior project development. *Journal of Systems and Software*, 84(3):479 – 491.

Ghezzi, C. and Mandrioli, D. (2005). The challenges of software engineering education. In *Proceedings. International Conference on Software Engineering - ICSE 2005 Education Track*, page 115 – 127. Springer.

Jazayeri, M. (2004). The education of a software engineer. In *Proceedings. 19th International Conference on Automated Software Engineering, 2004*, pages 18–27.

Letouze, P., d. Souza, J. I. M., and Silva, V. M. D. (2016). Generating software engineers by developing web systems: A project-based learning case study. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 194–203.

Marsicano, G., Mendes, F. F., Fernandes, M. V., and de Freitas, S. A. A. (2016). An integrated approach to the requirements engineering and process modelling teaching. In *Proceedings. 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 166–174.

Moreno, A. M., Sanchez-Segura, M.-I., Medina-Dominguez, F., and Carvajal, L. (2012). Balancing software engineering education and industrial needs. *Journal of Systems and Software*, 85(7):1607 – 1620. Software Ecosystems.

Nurkkala, T. and Brandle, S. (2011). Software studio: Teaching professional software engineering. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 153–158, New York, NY, USA. ACM.

Schaetter, A., Koeglmayr, H.-G., Blankenbach, K., and Nippa, M. (2009). Interdisciplinary approach to software engineering education. *Journal of Systematics, Cybernetics and Informatics)*, 7(5):29–36.

Shuto, M., Washizaki, H., Kakehi, K., Fukazawa, Y., Yamato, S., and Okubo, M. (2016). Learning effectiveness of team discussions in various software engineering education courses. In *Proceedings. 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 227–231.

Teel, S., Schweitzer, D., and Fulton, S. (2012). Teaching undergraduate software engineering using open source development tools. *Issues in Informing Science and Information Technology (IISIT)*, 9:63 – 73.

Zeidmane, A. and Cernajeva, S. (2011). Interdisciplinary approach in engineering education. In *Proceedings. IEEE Global Engineering Education Conference (EDUCON), 2011*, pages 1096–1101.