International Journal of Engineering Education Vol. 34, No. 5, pp. 1701–1708, 2018
Printed in Great Britain

0949-149X/91 $3.00+0.00
© 2018 TEMPUS Publications.

Project and Team-Based Strategies for Teaching Software Architecture*

MELINA VIDONI, JORGE MARCELO MONTAGNA and ALDO VECCHIETTI
Institute of Design and Development, INGAR CONICET-UTN, Avellaneda 3056, Santa Fe, Argentina.
E-mail: {melinavidoni, montana, aldovec}@santafe-conicet.gov.ar

Software Architecture remains a difficult topic to teach. This is because of the problem's complexity, and the integration of interpersonal and technical skills with knowledge from different areas. This paper presents the introduction of Project-Based Learning (PBL) and Team-Based Learning (TBL) in a Software Architecture undergraduate course. The goal for students is to apply in practice the concepts learned using ATAM (Architecture Tradeoff Analysis Method) to evaluate case studies of real-world architecture. PBL is known for allowing students to experiment with realistic problems and improve their negotiation and communication skills. TBL offers a different approach to group-based activities, by using them to determine the structure of the course. This approach is applied in an optional course in a Systems Engineering degree; its results are positive, having increased students' attendance and active participation levels. A student survey also shows acceptance of the new methodology.

Keywords: project-based learning; team-based learning; software engineering; software architecture

1. Introduction

Software companies search for candidates with not only strong technical knowledge but also with excellent interpersonal skills such as the aptitude to work in multidisciplinary teams, the capability to meet deadlines, reliable communication and negotiation ability to translate their technical findings to non-technical staff. In this sense, it is a challenging matter to design a Software Architecture (SA) course capable of inducing to the students all those skills. Software Architecture is complicated to teach because of its high-level abstractions, the decisions to make, and the lack of real-world examples to work with the students [1, 2]. Reports of education projects related to Software Architecture are scarce, despite its widespread utilisation to define system domains that facilitate standardisation on derived projects [3, 4]. Therefore, there is a need to provide students with the experience of acquiring both technical know-how and interpersonal skills applicable to this area of knowledge.

Several technical topics challenge Software Architecture teaching. Some are the refinement of quality attributes, the practical applicability of extensive theoretical knowledge, and use of engaging, well-documented systems, among others [5–7]. Furthermore, students lack extensive practice of architecture topics by the time of graduation [1]. On the subject of architecture evaluation, undergraduate courses are deficient in dynamical approaches [8], without promoting higher order thinking or advanced reasoning skills [9]. Improving learning by problem-solving in teams and enhancing communication between students are essential to provide a comprehensive education [10]. It is relevant as students are used to working alone or in groups of acquaintances, mostly because of personal insecurities when collaborating with strangers [11]; also, teaming and co-creation is proven to reinforce the acquired knowledge [12]. Reducing these issues becomes critical to prepare students for real-world jobs, where team-dynamics, negotiation, and communication skills are vital.

With these issues in mind, this article presents a Software Architecture undergraduate course design and practice based on Project-Based Learning (PBL) in conjunction with Team-Based Learning (TBL). Their purpose is to promote learning technical knowledge and developing interpersonal skills.

Architecture evaluation is introduced as a Course Project, and students apply a reorganised ATAM (Architecture Tradeoff Analysis Method) to evaluate a real-world case. Teams are assembled by individually selecting a role, and activities are performed in groups, discussing their proposals through peer-review and brainstorming. The underlying methodology, task planning, and assessment of the projects are presented as a resource for other teachers using this methodology.

This paper is organised as follows. Section 2 presents the motivational and educational goals that originated this proposal and selected pedagogical approaches, while Section 3 describes the generated course syllabus, lectures organisation, and grading procedure. Section 4 presents an application experience, and Section 5 outlines the discussions. Section 6 offer conclusions and possible lines for future works.

* Accepted 4 May 2018.

2. Educational goals and proposed approaches

The lecture-based paradigm with passive students is currently the predominant approach to Software Engineering and Software Architecture education [13]. Thus, practical strategies on these topics remain shallow [9], and need to be proposed and applied. This issue is crucial for architecture evaluation, as its practice depends on teamwork, brainstorming sessions and group discussions. As a result, it requires interpersonal and technical skills that are not usually trained together in undergraduate courses. Therefore, the following educational goals are formulated:

G1. Step out of the lecture-based paradigm, to teach experimentation and collaboration based on practical educational activities [14, 15].
G2. Use real-world projects to train with realistic, complex and industry-based problems in which the benefits of adopting specifics architectural techniques are visibly appreciated [16].
G3. Integrate and apply concepts of other courses to support different approaches, application areas, and domains [17].
G4. Contribute to improving interpersonal skills such as communication, negotiation, teamwork and project management [18].

Two pedagogical approaches are combined to achieve these goals.

Project-Based Learning (PBL) leads to higher student involvement with the instructor in a less central role [18], as it implies a shift towards the facilitation of students' learning [19]. It engages them with authentic, complex challenges and problem-solving activities by using real-world examples as case studies [20] . These projects promote practical skills such as coping with incomplete or imprecise information, self-regulation and commitment, cooperation and teamwork; they often require managing interdisciplinary issues to obtain positive results [16]. Thus, PBL contributes to all educational goals, but mainly to G2 and G3.

Team-Based Learning (TBL) cares for participation in collaborative face-to-face activities, focusing on teamwork and peer-evaluation [21]. On this approach, group-based activities determine the structure of the course [8]; this means that the course project is prepared first, then deliverables and exercises are defined, and finally the class schedule is assembled in conformity to this plan. Another key point is that students cannot freely select their teammates as other teaming methodologies are enforced; this is done to stimulate critical thinking and peer-review as part of the activities and evaluation, as well as other communication skills [21] . TBL targets goals G1 and G4, and it is used to generate the course syllabus.

However, obtaining real-world case studies may limit the application of this type of course organisation. This matter can be covered by agreements between academia and software industry to gain access to more real examples other than merely illustrative ones.

3. Course description

"Architecture-Based Software Design" is an elective final year course of the Information Systems Engineering degree. Its average number of students is around 15. The course is biannual and equivalent to 96 face-hours and 120 personal-study hours. Face-hours are organised in two sessions of three hours per week, over 16 weeks (one semester). Prerequisites are successfully taking both a Software Engineering and Software Design undergraduate courses.

The learning objective of this course is to acquire applied knowledge of Software Architecture in modern applications. It includes the skills to specify and evaluate software architectures, identify and use appropriate architectural styles, deconstruct existing systems to add new capabilities, document architectures and read reports, and understand quality attributes and its tradeoffs to reduce business risks associated with the information systems. The Course Project applies all of these concepts through the formal evaluation of the architecture of an existing software system.

The course is structured so that the skills taught and learned in the classroom and labs are applied to the eventual Course Project, which must be completed at the end of the semester. The PBL/TBL approaches manage this organisation. Therefore, the course is divided into two parts, according to the academic schedule of midterm examination periods.

The first part focuses on the fundamental concepts and their analysis through small case studies. It requires nine weeks, with topics posted in three parts: (1) architectural design, business cycle, and quality concepts, (2) patterns and tactics, and (3) documentation. These are introduced through discussion in classrooms and exemplified by brief case studies in the labs.

The PBL/TBL is performed in the second part of the course, requiring seven weeks. It revolves around the Course Project, and it is designed to link together the fourth topic -software architecture evaluation- with the previous ones. It is the primary practical application of the theory of earlier units, and the one that contributes the most to the final grading. Its structure is discussed in the following subsections.

3.1 Course project

The project focuses on an architecture evaluation, as the main hands-on application of concepts. The selected evaluation method is ATAM (Architecture Tradeoff Analysis Method) [22]. However, it is not the goal of this article to discuss ATAM particularities in detail.

TBL establishes that teams should not be willingly selected by students [21]. This aims to increase the collaboration with other classmates, beyond the closest acquaintances or friends [11]. Since on industry jobs software engineers are divided into teams or groups, the learning environment should match this reality [23]. Therefore, this contributes to goal G4.

At the beginning of the introduction to the Course Project, the organisation in teams is guided by the following steps:

1. Students are offered a list of roles (see Table 1), and they select one—individually—without disclosing it to other classmates.
2. The instructor notifies students that they will be working as a team with other students with the same role.
3. Students group with their teammates. They have five minutes to introduce themselves to each other.

Regarding students' interpersonal skills, there are three vital points. First, students choose a role independently of the selection of their acquaintances. Second, it mimics real-world industry jobs by giving students a clear group goal of the project, different to other teams. Third, this leads to peer-review both inside the group and outside between roles, enforcing the need to generate compromises in the solutions proposed for the case study.

The teams become essential as the ATAM activities are organised to work on teamwork, peer-review and brainstorming activities. ATAM outputs and artefacts become the assignments and deliverables required from the student body: *Team Assignments*, created by each team with their priorities and points of view, and *Brainstorming Outputs*, produced by peer-review and agreed among groups. These constitute the assignments that are part of the grading. Table 2 summarises the activities scheduled for the Course Project.

In this Course Project, all teams work on the same project. Their *Team Assignments* differ as they need to perform the tasks from their role standpoint. For instance, the group "Database Manager" should focus on the qualities, styles, tactics, risks, and sensibilities mostly about the management of data and information on the target system. This simulates the common industry situation of having different divisions inside the Information Technology Section.

Consequently, after working on each *Team Assignment*, there is a peer-review instance in which each team exposes their results to the others and justifies their selections. The other groups discuss these results, offer insight into their decisions, and evaluate how those choices affect their goals. Adverse impacts are debated to reach consensus between teams; i.e., a given tactic by *Team A* may affect *Team B*'s priorities negatively. The aim is reviewing the artefacts to correct defects, evaluate and improve the development process, and study how each team's decision affects the others [15].

After each review, some teams may be required to correct their results, either by incorrect use of theoretical concepts or to adapt their assignment to the agreements. Following PBL/TBL ideals, the instructor only acts as a mediator, taking notes of the students' participation and outcomes, and managing the time assigned to each team. This contributes to the goal G1 by reinforcing the collaboration within the group and between the roles by adding more workshops.

As seen in Table 2, this structure includes three types of classes: *lessons*, dedicated to introducing concepts, *team workshops*, in which teams are allowed to work on their assignments and evaluation outputs, and *general workshops*, for peer-review and brainstorming among groups.

3.2 Assessment overview

Implementing PBL/TBL in the course requires a new perspective for grading.

Table 1. Roles proposed to students and used during the Course Project

Role	**Description**
Database Managers	It focuses on data sources, including development and maintenance, data input/output, and others.
Resource Managers	It centres on resource availability, fault tolerance, and strains on both software and hardware resources.
Software Developers	This role focuses on the system life cycle, maintenance, evolution and user training.
Systems Integrators	It centres on interoperability between the involved systems. It considers data and information exchange, as well as separation of concerns.
Safety & Security Managers	It manages privacy, safety, and security, prioritising requests authenticity and including evaluating the access and actions on the system.

Table 2. Schedule and tasks of the PBL/TBL Course Project

Week	Class	Class Type & Content	Assignment
1	1	Lecture: Case study introduction by the chief architect. Project goals, business rules, and limitations. ATAM procedure. Team formation.	
	2	Teams Workshop: Architectural styles identification and analysis. Generation of ATAM Questions (Level 1: quality attributes).	*Team Assignment #1:* • Styles identification. • Quality attributes Questions.
2	3	General Workshop: Peer-review of *Team Assignment #1*. Agreement on teams' priorities and choices, decisions on compromises according to tradeoffs between styles. Teams are required to update their assignments.	*Brainstorming Output #1:* • Teams' updated assignment. • Styles tradeoffs and compromises agreements.
	4	Lecture: Short debriefing on ATAM Utility Tree (Level 1). Teams Workshop: Utility Tree generation.	*Team Assignment #2:* • Utility tree. • List of identified tactics. • Situations Questions.
3	5	Teams Workshop: Utility Tree generation. Generation of ATAM Questions (Level 2: Utility Tree situations).	
	6	General Workshop: Peer-review of *Team Assignment #2*. Agreement on compromises caused by teams' tradeoffs. Generation of the first list of risks, sensibility, and tradeoffs associated with the approved tactics.	*Brainstorming Output #2:* • Teams' updated assignment. • List of risks, sensibilities and tradeoff points. • List of situations.
4	7	Lecture: Short debriefing on Situations (Level 2). General Workshop: Group brainstorming to generate ATAM situations.	
	8	General Workshop: Group brainstorming to generate ATAM situations. Lecture: Short debriefing on prioritisation of situations. Assignment of priority points to teams.	
5	9	Teams Workshop: Teams' review and prioritisation of situations (from *Brainstorming Output #2*). Improvement and reconciliation of the Utility Tree.	*Team Assignment #3:* • Situations prioritisation. List and reasoning. • Improved Utility Tree.
	10	Teams Workshop: Improvement and reconciliation of the Utility Tree.	
6	11	General Workshop: Peer-review of *Team Assignment #3*. Discussion of decisions, agreements on compromises.	
	12	Teams Workshop: Teams' revision of the Utility Tree, according to the compromises agreements.	
7	13	General Workshop: Brainstorming related to tactics and the improved Utility Trees, to refine the list of risks, sensibilities, and tradeoffs.	*Brainstorming Output #3:* • Improved list of risks, sensibilities and tradeoff points. For both styles and tactics.
	14		

First, the deliverables score is weighted and combined to obtain the final course grade. For *Team Assignments*, the weights are 10%, 25%, and 15%, while for *Brainstorming Outputs* it is 10%, 20%, and 20%. Overall, they compose the 100% of the final grade.

Second, to enforce the collaborative activities, multiple aspects are considered to be part of each student's grading. These are divided into technical and interpersonal skills. Those are graded at each *Team Assignment* and *Brainstorming Output*:

- Technical aspects include the number of elements generated (i.e., the number of questions, nodes, situations), their fit to the role, and the comprehension level of both the architecture and domain. The latter is considered during the teams' exposition of their work at peer-review stages and when elaborating the artefacts.
- Interpersonal skills include the use of time to present their decisions, individual participation and contribution to peer-reviews, and their willingness to accept peers' requests. It also covers the technical language and vocabulary used during presentations, their contribution to the brainstorming sessions and the management of project goals beyond assigned roles.

4. Practical application

The project is a Reference Architecture for the Advanced Planning Systems domain [24], which consists of a type of decision support systems, aimed to automatize the optimisation of enterprise operations, such as logistics, production planning and schedule, among others. This is a real-world case coming from an industry-academy collaboration and is thoroughly documented. It contributes to goals G2 and G3, by emphasising the unification of contents taught in other courses.

In particular, Reference Architectures are often created by software factories to develop a specific kind of system. This is because they allow high reuse of components and structures on the base of well-established practices and designs [4]; this leads to reduced analysis and design times, launching projects earlier, and ensuring a prompt return of investment for the customer.

The course resources are managed through the University online platform. *Team Assignments* are uploaded by a team delegate, while *Brainstorming Outputs* are submitted by the instructors, who record and take notes during the workshops. The resources provided to the students are lecture notes about ATAM, the formal documentation of the target architecture following the "Views & Beyond" style [3], the original list of functional requirements and quality attributes, and a glossary of terms and definitions used in the project.

4.1 Assignment highlights

A positive outcome of this course is that, in many cases, the students pro-actively proposed variations to notations, tools, and concepts, to adapt them to the case study and the specific requirements of the Reference Architecture. Then, spontaneous brainstorming sessions are performed, usually during team workshops, to reach a consensus regarding those matters.

Brainstorming Output #1 analyses the architectural styles to generate a tradeoff chart. The debate led to two conventions proposed by the students without the intervention of the instructors. Their goal is to assess the case study needs. These are:

- Because the case study is a Reference Architecture, some styles have similar risks, sensibilities, and tradeoffs. Therefore, students decided to group these styles. An example of this is *Client Server*, *Multi Tiers* and *Publish and Subscribe*.
- The instructors introduced the Harrison et al. [25] notation to represent tradeoffs. The students proposed an additional symbol $\pm$ to cover the situation of structural variations of the Reference Architecture from the case study.

The students also performed adjustments to the Utility Tree improvement process at *Team Assignment #3*. Some groups discovered that a handful of situations could be addressed differently depending on the variation points of the Reference Architecture. *Variation points* are templates documented with the architecture used to allow the designer to change it by following preplanned steps [3]. Students proposed the concept of 'partial agreements.' This implies that in a comparison of situations, the equivalence happens only on a given variation point state; equivalent nodes for their different states should be added to the Tree, to transform this into a full agreement.

It is relevant that after these debates, the students used these new concepts and applied them without any enforcement from the instructors.

4.2 Poll results and instructors reflections

At the end of the course, students are asked to complete an anonymous poll. Relevant topics addressed include reasons to choose a role, assignments difficulty, opinions on peer-review and brainstorming sessions, assessment of the documentation, among others. Regarding reasons to pick their role, more than half students answered either personal interest (56%) or experience (34%), while the rest selected other options, such as the perceived difficulty of the role in the business domain.

About the documentation, nearly two-thirds of the students (61%) deemed the provided material as sufficient but complex. They stated that this was the first time working with a complete software architecture specification on the undergraduate courses. As a result, this influenced their perception of the assignments difficulty as well (see Fig. 1).

Regarding the content and requested deadline of the assignments, most of the students agreed that they had an intermediate-to-high difficulty. Though industry jobs often demand short closing dates, the lack of active learning approaches on Software Engineering education impacts the students' ability to work at a faster pace for this project. This can be temporarily improved by adding more team workshops, but the ideal situation is including more active learning spaces throughout all courses.

Some assignments are perceived as difficult, such as *Team Assignment #2*. An explanation for this aspect is that the first deliverable is mostly straightforward with outcomes thought to improve the understanding of the case study. The other two required a more extensive and detailed knowledge of the business limitations, achieved by a throughout analysis of the provided documentation.

The possible improvements are related to the presentation of the architecture documentation. It can be introduced through the viewpoints, each at different weeks of the course, or it can also be provided at the start of the course, rather than at the beginning of the Course Project.

About the peer-review and brainstorming sessions, all answers are positive. Most of the students agree on the usefulness to integrate the teams' points

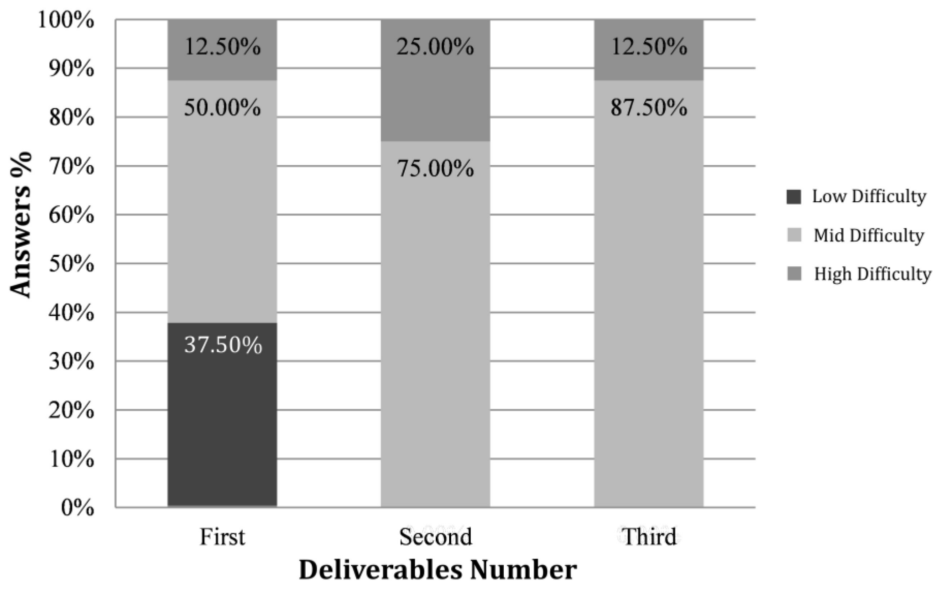

Fig. 1. Students' appraisal of the deliverables difficulty.

of view towards the primary goal (half the responses) and in the discussion of trade-offs between groups (around 40%). Other favourite answers are the utility to clarify the case study goals (20%), as well as business and methodological concepts (10%).

Finally, students can add comments regarding the process. For example, a student wrote: *"The project allowed me to understand quality attributes, and to learn to identify them. Also, now I understand which tactics to apply and their tradeoffs. I find this very useful"*. Regarding deliverables, another student stated: *"Positive points for this project are the teamwork classes that allowed to work on the deliverable and clarifying any possible doubts."*

5. Discussion

There are proposals in the academic literature regarding Software Engineering education. However, specific reports on Software Architecture courses remain scarce. Hidalgo et al. [26] propose a role-playing game to teach ATAM performing the project simulation through the game. Wang and Wu [27] use game development to explain Software Architecture, architecture evaluation, and detailed design; however, although they used ATAM for the assessment, there are no clear steps on how to apply their methodology to another course. Andrade et al. [28] applied ATAM to teach quality attribute metrics, instead of the evaluation process as an outcome itself.

This course differentiates itself from others. The PBL/TBL approach improves both technical and interpersonal skills, but to the authors' knowledge, there are no reports on using TBL for Software Architecture courses. Also, it does not require additional tools to be put into action, which simplifies its application. Although this course structure favours the understanding of quality attributes, it focuses on the evaluation steps and outcomes themselves.

As this is just a first experience, several positive aspects are detected:

- Course attendance increased from previous years. Only one student missed just one class, compared to an average of 3~6 non-attendances on each year, in the last three.
- Students proposed changes to the method and notations used, doing unplanned brainstorming and arriving at solutions having the agreement of all teams, without the intervention of instructors.
- High participation in peer-review instances, by always using the allocated time. Overall, this represented an estimated increase of 12% of the time students spent exposing their production to other classmates, based on data from the three previous years. A poll answer stated: *"I believe that peer-review was productive. This taught us not to narrow our point of view and to consider more than one standpoint"*. Another student wrote: *"There were cases in the differences between the teams became obvious. However, we were able to unify our goals by discussing the issues, even if in some cases that was a common tradeoff because the roles' goals clashed."*

It is worth noting that this is a first test of the proposed course structure, aiming to learn about it, improve and apply it in future versions.

6. Conclusions

This article presents the organisation of a Software Architecture undergraduate course, mixing PBL and TBL approaches. Its education goals are: to allow students to learn through experimentation, to expose them to real-world projects, to provide a venue to unify and apply concepts learned in other courses, and to create opportunities to improve their interpersonal skills.

These goals are achieved by focusing the practical application of architecture concepts through the Course Project: evaluating a real-world case study using ATAM. This structure can be reused and applied to other courses with similar objectives as the methodology, task planning, and assessment guides are included.

The proposed structure is applied on an undergraduate course named 'Architecture-Based Software Design' in the Information Systems Engineering degree. The case study architecture is centred on the domain of Advanced Planning Systems.

Several valuable results are obtained. Students were required to integrate knowledge and deal with uncertainty due to the business domain. The teaming process led them to work with classmates outside their circle of acquaintances, while peer-review and brainstorming allowed them to validate their results; this simulated a real-world industry job environment. Also, the complexity of the Course Project depends on the business domain of the target architecture. The target case study of this course was selected exclusively for this reason, but another one could impact on the students' perception of complexity at the final poll.

Several lessons can be learned from this experience. The PBL/TBL approach worked as expected in Software Architecture education, by allowing the students a practical approach. Working with real-world cases obtained from academy-industry agreements permitted the students to experiment a more realistic job environment.

Future works include performing more applications of this approach to different years of the course; this can be done with various case studies, to define how the business complexity affects the student perception of the project. Also, after more applications, it will be fruitful and possible to obtain qualitative information about attendances and participation rates, grading values, and other course statistics.

Acknowledgements—The authors gratefully acknowledge the financial support for the work presented in this article to Universidad Tecnológica Nacional (UTN) through PID EIUTI-FE0003974TC and PID EIUTIFE0003960TC. Also, the authors thankful Prof. Juan Carlos Ramos for his cooperation in this experiment on the 'Architecture Based Software Design' course at UTN-FRSF.

References

1. C. R. Rupakheti and S. V. Chenoweth, Teaching Software Architecture to Undergraduate Students: An Experience Report, in *IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE)*, Florence, Italy, 2015, pp. 445–454.
2. S. Angelov and P. de Beer, Designing and Applying an Approach to Software Architecting in Agile Projects in Education, *Journal of Systems and Software*, **127**, 2017, pp. 78–90.
3. P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord and J. Stafford, *Documenting Software Architecture*, 2nd ed. Pittsburg, USA: Pearson Education, Inc., 2002.
4. R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole and M. Bone, The Concept of Reference Architectures, *System Engineering*, **13**(1), 2010, pp. 14–27.
5. E. Tempero, Experiences in teaching quality attribute scenarios, in *ACE '09 Proceedings of the Eleventh Australasian Conference on Computing Education*, **95**, Wellington, New Zealand, 2009, pp. 181–188.
6. C. Costa-Soria and Jennifer Pérez, Teaching software architectures and aspect-oriented software development using open-source projects, in *Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education (ITiCSE)*, Paris, France, 2009, pp. 385–385.
7. K. Garg and V. Varma, An effective learning environment for teaching problem-solving in software architecture, in *Proceedings of the 2nd India Software Engineering Conference (ISEC)*, Pune, India, 2009, pp. 139–140.
8. S. M. Goltz, A. B. Hietapelto, R. W. Reinsch and S. K. Tyrell, Teaching Teamwork and Problem Solving Concurrently, *Journal of Management Education*, **32**(5), 2007, pp. 541–562.
9. M. J. O'Grady, Practical Problem-Based Learning in Computing Education, *ACM Transactions on Computing Education (TOCE)*, **12**(3), 2012, pp. 10:1–10:16.
10. P. Lago and H. van Vliet, Teaching a Course on Software Architecture, in *18th Conference on Software Engineering Education & Training*, Ottawa, Canada, 2005, pp. 35–42.
11. M. Kyprianidou, S. Demetriadis, T. Tsiatsos and A. Pombortsis, Group formation based on learning styles: can it improve students' teamwork?, *Educational Technology Research and Development*, **60**(1), 2012, pp. 83–110.
12. G. Ribes, M. R. Perello-Marin and O. Pantoja, Co-Creation in Undergraduate Engineering Programs: Effects of Communication and Student Participation, *International Journal of Engineering Education (IJEE)*, **34**(1), 2018, pp. 236–247.
13. S. S. Yadav and J. Xiahou, Integrated project-based learning in software engineering education, in *International Conference on Educational and Network Technology (ICENT)*, Qinhuangdao, China, 2010, pp. 34–36.
14. M. Coccoli, L. Stanganelli and P. Maresca, Computer Supported Collaborative Learning in software engineering, in *IEEE Global Engineering Education Conference (EDUCON)*, Amman, Jordan, 2011, pp. 990–995.
15. A. Baker, E. O. Navarro and A. van der Hoek, An Experimental Card Game for Teaching Software Engineering Processes," *Journal of Systems and Software*, **75**(1–2), 2005, pp. 3–16.
16. G. Hedin, L. Bendix and B. Magnusson, Teaching Extreme Programming to Large Groups of Students, *Journal of Systems and Software*, **74**(2), 2005, pp. 133–146.
17. C. Ghezzi and D. Mandrioli, The Challenges of Software Engineering Education, in *International Conference on Software Engineering (ICSE)*, vol. LNCS 4309, St. Louis, MO, USA, 2006, pp. 115–127.
18. P. J. Clarke, D. Davis, T. M. King, J. Pava and E. L. Jones, Integrating Testing into Software Engineering Courses Supported by a Collaborative Learning Environment, *ACM*

Transactions on Computing Education (TOCE), **14**(3), 2014, pp. 18:1–18:33.
19. W. Hamiza, W. Zin, A. Williams and W. Sher, Introducing PBL in Engineering Education: Challenges Lecturers and Students Confront, *International Journal of Engineering Education (IJEE)*, **33**(3), 2017.
20. J. A. Macias, Enhancing Project-Based Learning in Software Engineering Lab Teaching Through an E-Portfolio Approach, *IEEE Transactions on Education*, **55**(4), 2012, pp. 502–507.
21. L. K. Michaelsen and M. Sweet, "Team-based learning," in *New Directions for Teaching and Learning*.: Wiley Online Library, **128**(5), 2011, pp. 41–51.
22. R. Kazman, M. Klein and P. Clements, *ATAM: Method for Architecture Evaluation*, Carnegie Mellon Software Engineering Institute, Pittsburgh, USA, Final Report No. CMU/SEI-2000-TR-004, 2000.
23. S. C. dos Santos, PBL-SEE: An Authentic Assessment Model for PBL-Based Software Engineering Education, *IEEE Transactions on Education*, **60**(2), 2017, pp. 120–126.
24. M. Vidoni and A. Vecchietti, Towards a Reference Architecture for Advanced Planning Systems, in *18º International Conference on Enterprise Information Systems—ICEIS*, **1**, Roma, Italia, 2016, pp. 433–440.
25. N. B. Harrison and P. Avgeriou, Leveraging Architecture Patterns to Satisfy Quality Attributes, in *Proceedings of the First European Conference, ECSA 2007*, **4758**, Aranjuez, Spain, 2007, pp. 263–270.
26. C. Hidalgo-Montenegro and H. Astudillo, A role-playing game to teach ATAM (Architecture Trade-off Analysis Method) a simulation tool and case study, in *IEEE Conference of the Andean Council (ANDESCON)*, Bolivia, 2014.
27. A. I. Wang and B. Wu, Using Game Development to Teach Software Architecture, *International Journal of Computer Games Technology*, **2011**, 2011.
28. R. M. Andrade, R. Arakaki and J. C. Cordeiro, Teaching Software Architecture Quality Using ATAM, in *Joint International Conference on Engineering Education & International Conference on Information Technology*, Riga, Latvia, 2014, pp. 170–179.

Melina Vidoni, DEng, graduated from the Universidad Tecnológica Nacional, where she also received her PhD. She is currently postdoc fellow with a full-time scholarship under the supervision of Prof. Vecchietti and Prof. Montagna at the National Council for Technical and Scientific Research of Argentina at INGAR CONICET-UTN. Her postdoc research work focuses on integrating APS with Big Data and working with Software Engineering concepts in Operations Research.

Jorge Marcelo Montagna is PhD in Chemical Technology of the Universidad Nacional del Litoral, Santa Fe, Argentina. He is Principal Researcher of the National Council for Technical and Scientific Research of Argentina at INGAR. Also, he is Professor of Information Technology Management at the Universidad Tecnológica Nacional. His primary expertise area is Process System Engineering, working on the mathematical models for operations optimisation in Supply Chain and Scheduling.

Prof. Vecchietti, PhD, is a Chemical Engineer from the Universidad Nacional del Litoral, Santa Fe, Argentina. He also obtained the PhD in Chemical Engineering at same University. He is a Principal Researcher of the National Council for Technical and Scientific Research of Argentina at INGAR, where he also is the Institute Head. His expertise area is Process System Engineering, working on optimisation mathematical models for planning and scheduling of manufacturing and production companies and its supply chain. He has extensive experience in consulting works with private production companies. He is also Professor in the Information System Engineering Department at Universidad Tecnológica Nacional (Santa Fe, Argentina).