Proceedings of the Federated Conference on Computer Science and Information Systems pp. 861–870
DOI: 10.15439/2019F279
ISSN 2300-5963 ACSIS, Vol. 18

Participating in an Industry Based Social Service Program: a Report of Student Perception of What They Learn and What They Need

Miguel Ehécatl Morales-Trujillo
University of Canterbury,
Computer Science and Software Engineering Department
Christchurch, New Zealand
miguel.morales@canterbury.ac.nz

Gabriel Alberto García-Mireles
Universidad de Sonora
Departamento de Matemáticas
Hermosillo, Mexico
mireles@mat.uson.mx

***Abstract*—Skills demanded by the IT industry from graduates should be aligned with the curricula of Computer Science undergraduate programs. It is well-known that theoretical knowledge undergraduate students acquire during their studies needs to be complemented with practical experience; therefore, participating in university supported real life projects is a viable option for the students to get prepared for the industry. This paper reports findings from a survey applied to students who had been involved in an industry-based program meant to fulfill their graduation requirements, including the opportunity to develop a capstone project. We gathered their perceptions regarding what they learned during their studies, what they acquired in the industry-based program and what they consider useful for their current jobs. The results show that most topics are aligned between the Bachelor's degree program and the industry needs, but there is a strong separation in the cognitive levels students achieve at each stage. The paper provides insight into the needs of Computer Science students and contributes to finding ways of increasing undergraduate student satisfaction with skills acquired at university and their application in real contexts.**

I. Introduction

An accelerated evolution of IT technologies demands software developers ready to get incorporated to IT workforce. The Bureau of Labor statistics of the United States of America estimates a 24% increase in demand for software developers in the period from 2016 to 2026, which a much faster than average growth [26]. Similarly, in other countries around the world the demand for software developers is growing in similar proportions. To obtain a desirable employment, a competent software developer is required to possess a wide variety of skills, such as managerial, engineering, team working and communication [1].

In undergraduate computer science (CS) and software engineering (SE) programs, educators provide experiences to support an adequate development of knowledge and skills in students. However, a crucial question remains open: how to educate software engineers to do their jobs efficiently and properly [25]. Current curriculum guidelines recommend that educational programs provide effective learning of SE skills and concepts by incorporating real-world elements, which could be done through capstone projects and student work experience among others [1].

In capstone projects, students focus their effort on completing a significant real project while they practice learned knowledge and skills [1]. Capstone projects represent an important learning activity in SE, given that most of the SE concepts are abstract in nature. In consequence, they are difficult to understand by undergraduate students without adequate hands-on experience [17].

However, providing a real-life experience in undergraduate programs is a difficult task [9]. Another factor to be considered is the gap between skills acquired by IT graduates and skills demanded by the industry [25]. Although it is necessary to prepare students for incorporating into the software industry, this is still a very complex endeavor that many universities are struggling with [38]. Other than theoretical knowledge, future employers are always more interested in students who are equipped with hands-on experience [41].

Despite a considerable interest to find ways of helping students to quickly become efficient software developers and blend into the IT work place, there are few empirical reports of problems that students face in capstone projects as well as of learning outcomes students perceive to have achieved after completing a capstone project [37]. Thus, this paper aims to present an experience report of how a joint effort between industry and university can increase undergraduate student satisfaction with skills learned at university and their application in a real project.

In addition, this report provides a comparison between knowledge and skills learned at university against those learned in the industry. Moreover, the results of this experience have other effects over educational aspects, such as an increase in awareness for alignment of academic courses with industrial experience by CS and SE educators.

The paper is structured as follows. Section II presents an overview of capstone projects and agile methodologies used in them. Section III describes the context of CS Bachelor's degree of a Mexican university and the nature of the industry-based student program. Section IV reports the data collection process

and data analysis. The results are discussed in Section V and the conclusions and future work are presented in Section VI.

II. Background

A. Capstone projects in CS

Capstone is defined as "the high point: crowning achievement" [24]. In a capstone project, "students solve real-life problems in the context of a large, realistic project" [37]. Working on a project helps students to develop SE skills and apply them in a realistic environment, providing them with an excellent opportunity to understand and experience various technological waves that would be present in their careers [16]. In order to improve learning outcomes, project courses can involve industry partners who provide real-world problems for students to solve, to develop both technical and soft skills, and to gain contacts to potential mentors in the industry [40].

Several papers address the use of capstone projects to enhance knowledge and skills of both CS and SE students. Vanhanen et al. [37] conducted a survey to understand the problems that arise during development of capstone projects, to gain insight into improvement in learning outcomes as perceived by students, and into customer satisfaction. They found that learning outcomes varied a lot among teams and team members since roles undertaken by students affected their level of learning. Considering common problems reported in capstone projects, the most common ones were related to testing, task management, effort estimation and technology skills [37].

In [21] students identified algorithms, programming, networks and databases, and contemporary technologies for developing software as essential technical skills in order to succeed in a capstone project. They can be seen as potential discipline knowledge gaps and become areas to better address in capstone project activities. The authors surveyed students' feedback where the majority of students reported a general skill improvement or learning new technical or project management skills during their capstone project [21].

Projects often require students to use technologies (programming languages, web frameworks etc.) that they have not been taught in their previous courses [37]. Based on observing differential learning outcomes achieved by students, Karunasekera and Bedse [17] proposed a skill based learning framework to provide objective guidance to ensure that team based projects offer a balance of management, engineering and personal skills. In case of capstone project courses, Majanoja and Vasankari [21] suggest to organize them focusing on team work, communication and problem solving skills.

It can be observed that few quantitative empirical studies report problems encountered by capstone project teams [37]. The reported problems are related to poor communication among the team members, poor leadership, failure to compromise, procrastination problems, integration testing problems, and lack of cooperation, among others [37]. Besides, such problems as lack of skills for using tools, lack of organization, lack of technology expertise, and having to combine work and study during the project are reported in [2]. Other risks are architecture complexity, quality trade-offs, personnel shortfalls, budget and schedule constraints, COTS and other independently evolving systems, customer-developer-user team cohesion, requirements volatility, user interface mismatch, process quality assurance, requirements mismatches, acquisition and contracting process mismatches [18].

Learning outcomes of capstone projects [37] include improvement in such students skills as familiarity with agile approach, programming, project planning and management, effort estimation, acquiring "a big picture" of a software development process, team work, customer interaction, and communication [20]. Improvement in learning is reported in skills related to requirements engineering, system design, modeling, programming, version control, release management, and usability engineering, among others [6]. Besides, Broman et al. report students' learning technical knowledge of a specific SE role, time management, usefulness of agile methods, and team communication and collaboration [5]. As for guidelines to discuss student experience of participation in industrial capstone projects, SWEBoK offers a set of topics, understandable to students and widely used by researchers [32].

A realistic setting of a capstone project developed for a real customer also introduces challenges for the participants. The beginning of a project course is particularly difficult, as students have to familiarize themselves with development processes and tools, get to know the project's problem domain, understand requirements, and deal with communication issues [9]. Besides, there are several organizational aspects to be considered when providing real-life experiences [38], [6], [4]. Despite the effort to provide students with real-world experiences, instructors should be aware that the level of exposure to these formative actions will be limited [1]. Instructors can only facilitate the initial stage of the process for students develop a mature understanding of the real world across their careers [1].

B. Agile methodologies in capstone projects and courses

Providing students with an iterative work by means of agile and lean methods can both fulfill the industry needs and support the design of SE courses [23]. Considering agile methodologies, Scrum is one of the most adopted and/or adapted in a capstone course design [23], [27]. Indeed, in courses based on project-based learning, Scrum is very common. It can provide control over project progress and it is able to ensure a steady pace of development [12].

Fitsilis and Lekatos [13] conducted a survey study to understand the importance of agile practices and their relevance in the industrial context while relying on the Scrum method taught in a capstone course. As a result, the majority of the participants reported a general positive experience from the capstone course; in addition, they perceived such frequently used in the industry practices as unit testing, coding standards, test-driven development (TDD) and continuous deployment as the most useful. Other researchers studied how particular estimation techniques can be introduced in a software engineering

capstone course [28]. Another research line is focused on the assessment of particular practices of continuous integration, TDD, and work tracking, among others [12].

Both, supporting tools and adaptation of agile techniques, have been explored in capstone courses and projects. Rodriguez et al. [29] propose a virtual Scrum tool that simulates a Scrum-based team room by means of the 3D virtual world metaphor. The tool can support specific configuration and progress of each student team [29]. On the other hand, Krusche et al. [19] proposed Rugby, an agile model for supporting continuous delivery of software. The authors report that the Rugby method improves coordination across multiple teams and enhances communication between developers and customers [19].

Lean approaches, such as Kanban, have also been studied in the context of Scrum-based university courses by introducing Kanban at the end of instruction period [23]. Paasivaara et al. [27], on the other hand, report an effective outcome of a Scrum based capstone course with real clients. They found that students positively change their attitudes about the importance of collaboration and communication within the teams while experiencing less than expected difficulty in learning new technologies [27].

Moreover, agility in capstone courses have been explored in relation to other topics. Fagerholm and Vihavainen [11] have studied peer assessment to provide a full view of student performance in a project. In particular, self-assessment and peer review should provide insights with regard to learning goals. In software maintenance, Weissberger et al. [39] reported an experience of using agile principles in a maintenance project developed in a capstone project.

Academic software factory is another approach used in universities to enhance the quality of education by means of capstone projects. University laboratories emulate a real working setting where teams of students implement a project for real customers and real deadlines [33]. Similarly, Fagerholm et al. [10] report another experience of developing the software factory approach. They provide a collection of patterns and anti-patterns to support the design, implementation and operation of project-based startups [10].

III. Context of the Experience

A. Computer Science Bachelor's degree

The National Autonomous University of Mexico (UNAM) is the biggest university in the country with 356,530 enrolled students in 2019 [35] and sits the #2 in Latin America (#113 university in the world) according to QS World University rankings [7]. The UNAM consists of 15 colleges that offer 127 Bachelor's degrees; CS Bachelor's degree is one of them and is taught at the College of Science.

CS Bachelor's degree is designed to be finished in four years obtaining 376 credits; it consists of 28 compulsory and 6 elective courses based on seven knowledge areas: Mathematical foundations, Discrete structures, Programming, Software engineering, Theoretical computing, Theory and practice integration, and Systems architecture. In order to graduate, CS students are required to complete 376 credits, 480 hours of social service and to develop a capstone project.

In Mexico, undergraduate students must meet the requirement of completing a social service to be able to graduate [8]. The goal of the social service program is to contribute to both academic and professional training of students. Student who have achieve 70% of undergraduate credits are eligible for social service program as they are supposed to apply their acquired knowledge and skills in an organization which is looking for social, cultural and economic development. The social service practice amounts to 480 hours that students complete in a time period of 6 to 24 months [34].

Mexican program of social service "promotes professional and human development of the student, developing an active and supportive social commitment applied to the solution of problems or needs of the country" [34]. It is similar to work experience placements required by universities around the world, however it differs from them due to its civic and social orientation. Nevertheless, there is no other alternative that would provide students with work experience previous to their graduation.

As for the capstone project, CS students are offered 10 options: writing a dissertation (thesis), participating in a research-based project, taking a research seminar, participating in a science popularization project, reaching the national stage of the ACM Programming Contest, being in the first quartile of the Graduate Record Examination for CS, obtaining an over 95 GPA (out of 100), carrying out tertiary teaching support activities (such as teaching assistant), obtaining industry experience and extending the social service.

The majority of these options are research- or teaching-based with the exception of the last two: industry experience and extended social service. In the case of the extended social service, most of the programs are research and teaching oriented. According to the information published in [36], 97 out of 272 available programs for CS students are IT related. From those, 80 are carried out within the UNAM schools and dependencies, 16 in government offices and only one is industry-based.

In addition, both options require a university professor who can supervise capstone projects based on an industry experience or an extended social service, however, there is not enough cooperation between university supervisors and industry available projects.

We identified a lack of opportunities for students to gain more industry-oriented experience both through social service or developing a capstone project. In order to improve this situation, we proposed to push for industry-based capstone projects and social service programs.

On the one hand, it would allow students to complete a social and civic service requirement as well as to obtain work experience in the current industry. On the other hand, it can lead to an increase in numbers of graduated students in CS as industry-based experience offers reasonable opportunities for developing a successful capstone project.

The current numbers of graduated students are extremely low: the terminal efficiency of CS Bachelor's degree since 1995 is 18.1%. In absolute numbers we find that 285 students graduated, 621 completed their credits but did not graduate and 672 neither graduated nor completed their credits. Only one of each five students is able to complete their credits, get through their social service and develop a capstone project. Looking at only eight last generations of students, the terminal efficiency is slightly higher: 19.7%. while only 155 out of 785 CS students graduated.

One of the reasons for the low numbers of IT-related graduates is their high level of employability, and CS students are no exception. Nowadays, there is an important demand to fill in constantly growing job offers in the IT sector. Deficit of employees with a IT profile is critical in many countries around the world; according to [25] it is growing at an exponential rate. Students are leaving the university because they are getting job offers even if they are not graduated yet.

In this context, undertaking this initiative was powered by a strong motivation to improve the exposure of CS students to relevant work experience in real projects aligning it with their graduation requirements, namely, social service and capstone project.

B. Industry based program

The industry based student program described in this paper was divided in two stages. During the first stage students worked to complete 480 hours of social service. During the second stage students could opt for continuing working but with a condition of developing a capstone project based on the work done in the first stage.

This project was carried out in a joint effort by the College of Science and a Mexican software development organization. The expected outcome of the project was an information system for public administration. The first step to link the organization and the university was to define a social service program in which the student activities would be aligned with their civic responsibilities.

Once the program was created and approved by the University, any interested student could join the program. Actually, students joined the program voluntarily at different stages of the project during the year. There were no additional restrictions to join the program out of those already imposed by the University, so all students who applied for the program were accepted.

Each student was coordinated by a professor from the College and looked for the goals of the social service to be achieved. Students were required to work in the organization during a minimum of 6 months covering the 480 hours of their social service. They also received a monthly grant during the time they participated in the project.

The organization had a full development process in place; in consequence,the students were able to carry out a wide variety of tasks in different software engineering activities. During their first month of work, they were trained and introduced into the organization, the team and the project. Tasks assigned during the initial period were related to requirements specification and testing with the intention to get them familiar with the project and system under development. Activities related to Software Configuration Management (SCM), such as version control systems and continuous integration, were specially relevant at this stage.

The organizational development process was a hybrid process based on [22] mixing agile practices with ISO/IEC 29110-5-1-2. Project management activities were assigned using boards (Kanban style) and the progress was reported through stand-ups (Scrum). Software requirements specification (SRS) was carried out first through wireframes and customer workshops and then specified as use cases. Programming tasks consisted in fixing bugs and pair programming sessions arranged on demand while continuous integration and deployment practices were in place. User acceptance tests were carried out mostly by using a Think Aloud! protocol, bughunt sessions to test the system were implemented, and a bug tracker system was used to report defects.

During the following months students rotated between different teams and carried out roles of analysts, programmers, testers and database developers. Table I enlists the tasks carried out by the students, where agile practices used for each task are provided in brackets. For standardizing purposes generic ISO/IEC 29110-5-1-2 [14] tasks definitions are used.

A student's timeline in the program was usually the following:

- **Day 1:** an introduction to the organization, team and project.
- **Day 2:** a walk-through the development process of the organization.
- **Day 3:** joining SRS and Testing teams.
- **Month 2:** joining Construction team.
- **Month 3:** joining Databases team.
- **Months 4 and 5:** joining the team of their preference.
- **Last two weeks:** writing a social service report required by the College of Science.

Students received training from the team leaders and participated in workshops obtaining knowledge in topics like:

- Version control systems with Git and continuous integration.
- Development frameworks with Grails.
- Web deployment using WebLogic Server.
- Stored procedures and packages in Oracle 12c.
- Test automation with JMeter.
- Mobile development.
- Basic accounting to deal with employment and workers' rights.

The students had regular control meetings with the project manager of the organization following a daily Scrum approach. In addition, they had regular contact with their respective team leaders and teammates.

From the College of Science side, two professors were in charge of ensuring that the purpose of the social service program was being fulfilled and the students received guidance

TABLE I
TASKS CARRIED OUT BY STUDENTS

Software Requirements
Document or update requirements specification **(User stories and Wireframes)**
Identify and consult information sources in order to get new requirements.
Verify and obtain approval of the requirements specification.
Validate that requirements specification satisfies needs and agreed upon expectations, including the user interface usability **(Think Aloud!).**
Participate in revision meetings with the customer.
Software Architectural and Detailed Design
Describe in detail the appearance and the behaviour of the UI **(Wireframes and Think Aloud!).**
Generate an architectural design, its arrangement in subsystems and components defining internal and external interfaces.
Software Construction
Construct or update software components **(CI and CD).**
Correct the defects found until successful unit test is achieved **(Pair programming).**
Perform backup according to the version control strategy.
Software Integration and Tests
Verify consistency among requirements specification, software design and test cases **(Think Aloud!).**
Design or update unit test cases.
Perform software tests and document results in the test report **(Bughunt).**
Correct defects found and perform regression tests **(Pair programming).**
Verify consistency of the software documentation with the software.
Product Delivery
Verify consistency of maintenance documentation with software configuration.

from the organization members. A non-conformity process was in place to inform any improvement opportunity or disagreement regarding the students' interaction and performance. This process was a two way communication channel that served to inform the College if students did not perform as expected or if they were absent without notifying.

Once the six months concluded and students completed the required 480 hours of their social service, they were invited to continue in the organization but developing their capstone project. The capstone project had to be related to their work in the last six months and was supervised by a professor from the College with a possibility to be co-supervised by a member of the organization. The only constraint imposed on students was to finish the capstone project in no more than six months.

This paper reports the participation of a cohort formed by 13 students. The time they spent in the program varied from 5 to 12 months. First results of the program are described in the following sections.

IV. MAPPING ACQUIRED AND REQUIRED SKILLS

As mentioned before, there is a need for aligning higher education practices to industry needs by exposing students to industrial processes [30]. It is often observed that a large gap separates software projects in industry from what can be experienced in the classroom [15]; therefore, students are unconvinced by the relevance of the material delivered in lectures. The challenge, in consequence, is to develop environments within universities that are sufficiently real to be convincing to students [31].

Our interest was to gain an insight into students' perceptions regarding the knowledge and skills they acquired during their studies, during their participation in the program, and finally in their current job. To gather data, we designed a survey based on SWEBoK areas and topics; we were interested to know to what extent the students have been exposed to them, how the students perceived the preparation they received and what they considered necessary once they joined the industry.

The results were expected to allow for evaluation of the usefulness of the social service program as well as for an appreciation of an (non)-existing alignment between what CS students perceive that have learned and what they need once they graduated.

SWEBoK establishes a baseline for the body of knowledge in the field of SE [32]. It strongly influences the manner in which curricula are defined and how academic programs are accredited. In this study, SWEBoK was chosen due to its wide acceptance as a well-known SE body of knowledge that connects industry recommended practices and the knowledge expected from a software engineering professional.

The survey consisted of two sections, the first section was an online survey and the second section was answered in a spreadsheet. The time taken to answer the whole survey ranged from one day to one week; an incentive was given for each fully completed survey.

The first section contained questions about personal involvement in the program, and their current academic and professional situation:

1) How many months were you involved in the program?
2) What activities did you carry out during your participation in the program?
3) What skills do you think you developed the most?

4) Did you develop a capstone project derived from your participation in the program?
5) Did you complete the social service program?
6) Did this program contribute or is contributing to your graduation?
7) Did this program contribute or is contributing to you getting a job?
8) Do you currently work in the IT-related industry?
9) What do you consider to be the most beneficial in your participation in the program?
10) What improvements do you consider the program needs?

In the second part, the students were presented with a table which enlisted each of the SWEBoK topics grouped by area of knowledge, and had three columns representing stages (while studying my bachelor's, while participating in the program, in my job), during which the knowledge was acquired and/or applied.

Each SWEBoK topic (rows) was presented with an example in the form of tooltips. Figure 1 shows a fragment of the survey with answers retrieved from one student. The fragment was translated from Spanish into English.

Once the collected data were analyzed through general descriptive statistics of median, mode and mean, they were used to answer the following research question:

How did participating in a real project transform the students' knowledge?

It was decided to follow an adapted Bloom's taxonomy presented in [3], particularly three lower levels of the taxonomy (Remember, Understand and Apply) were used.

In the first column, respondents selected one of the following options for each SWEBoK topic to complete the claim *"While studying my bachelor's ..."*:

1) I did not get familiar with this topic.
2) I got familiar with this topic but I did not apply it in practice **(Remember and Understand)**.
3) I got familiar with this topic and had a chance to apply it in practice **(Apply)**.

In the second column, the options to select were similar to the one in the previous question, but the claim was *"While participating in the program ..."*:

1) I did not require this topic.
2) I was already familiar with the topic, but I applied it in practice **(Apply)**.
3) I was already familiar with the topic, but I did not apply it in practice **(Remember and Understand)**.
4) I got familiar with the topic and I had the chance to apply it in practice **(Remember, Understand and Apply)**.

The second question of interest was:

What software engineering topics/areas are regarded as most useful in the industry by the students?

In this case, a 5-point Likert scale was used. In the third column, respondents selected one of the following options to complete claims corresponding to *"In my job, <SWEBoK topic> is ..."*:

1) Not useful.
2) Slightly useful.
3) Useful.
4) Very useful.
5) Essential.

Besides it was of interest to know what positions they held in the industry once they finished their participation in the project. This information was collected via email.

V. Results and Discussion

A. First section of the survey: Industry-based program

A total of 13 students participated in the program and 12 completed it successfully; Table II shows a summary of the students who participated in the program. The only student who did not complete the program accepted a job offer in the IT industry and left the program. This is one of main reasons of why students do not graduate and there is very little the universities can do to minimize its impact.

The first part of the survey was responded by 10 students. Eight out of 10 students reported that their participation in the program contributed to their graduation and nine out of 10 reported that it contributed to getting a job after the program. The following statements are examples of the students' opinions regarding their experience in the program:

"Having a real-world experience that goes beyond a social service gave me a better position."

"The experience I have got was well valued by the recruiter."

"I met people with a lot experience in software development, I learned a lot from them."

"I interacted with real customers and teams, the experience I have got was real-life."

Six students provided information regarding the position they occupied after the project:

- Database Developer
- SOA Java Developer
- Software Test Engineer
- Data Scientist Jr.
- Software Engineer
- SECaaS and DBaaS Engineer

Two of them worked in an organization that followed Scrum as its main methodology, allowing one of them to become a Scrum Master.

Last but not the least, four students concluded their capstone projects as based on their work during the program. Given the proportion of students who completed their social service (92.3%) and those who graduated (30.8%) we consider this an improvement for the historical data of graduated CS bachelors in the UNAM.

B. Second section of the survey: SWEBoK areas and topics

Nine students completed the SWEBoK mapping. The first question to answer is:

How did participating in a real project transform the students' knowledge?

On a more general level, Table III displays an association perceived by students between each SWEBoK area

	0: I did not get familiar with this topic. **1:** I got familiar with this topic but I did not apply it in practice. **2:** I got familiar with this topic and had a chance to apply it in practice.	**0:** I did not require this topic. **1:** I was already familiar with the topic, but I applied it in practice. **2:** I was already familiar with the topic, but I did not apply it in practice. **3:** I got familiar with the topic and I had the chance to apply it in practice.	**1:** Not useful. **2:** Slightly useful. **3:** Useful. **4:** Very useful. **5:** Essential.
1. Software Requirements	**While studying my bachelor's ...**	**While participating in the program ...**	**In my job ...**
1.1. Software Requirements Fundamentals	2	1	3
1.2. Requirements Process	2	1	3
1.3. Requirements Elicitation	1	3	5
1.4. Requirements Analysis	2	1	2
1.5. Requirements Specification	2	1	4
1.6. Requirements Validation	1	3	5
1.7. Practical Considerations	1	3	4
1.8. Software Requirements Tools	2	1	5

Fig. 1. A survey fragment

TABLE II
PARTICIPANTS' DETAILS

ID	# Months	Roles carried out	Completed the program?	Developed a capstone project?	Got employed?
S1	5	Analyst and DB developer	No	No	Yes
S2	12	Analyst, Programmer and Tester	Yes	Yes	Yes
S3	8	Analyst and Programmer	Yes	Yes	Yes
S4	6	Analyst and Tester	Yes	No	Yes
S5	6	Analyst, Tester and DB developer	Yes	No	Yes
S6	7	Analyst and Tester	Yes	No	Yes
S7	6	Analyst and Tester	Yes	No	Yes
S8	7	Analyst and DB developer	Yes	No	Yes
S9	9	Analyst, Programmer and Tester	Yes	No	Yes
S10	11	Analyst, Programmer and Tester	Yes	Yes	Yes
S11	10	Analyst, Programmer and Tester	Yes	Yes	Yes
S12	9	Analyst and Tester	Yes	No	Yes
S13	6	Analyst and Tester	Yes	No	Yes

and cognitive levels. It is observed that, except for Software Maintenance, all the areas required in the program had been taught at the BSc (students got familiar with them) and most of them reached the next cognitive level (students applied them in practice).

It can be noted that Software Maintenance was perceived as an area not familiar to students during the BSc; however, by the end of the program, students perceived a shift from not knowing it to applying it in practice. Software Engineering Economics is another area perceived as unknown by students, and this area of knowledge was not required during the program either.

Students reported that after participating in the program, cognitive levels improved in nine areas and remained at the same level in three. These results allow to conclude that, in this particular case, real experience in a controlled environment offered an important advantage for the student development.

Going down on a more detailed level, we found an increase in cognitive levels associated to SWEBoK topics as 45 topics were perceived as improved and for 25 their cognitive levels remained the same. Only topics required during the project were analyzed (70 out of 102 SWEBoK topics).

In particular, three topics changed their states from Unknown to Apply, as well as from Unknown to Remember and Understand. Their breakdown is shown in Table IV. An important message is the lack of Software Maintenance and Software Configuration Management in the curricula. The gap between students' knowledge acquired during their studies is large as compared to the importance of this area for their job in the industry. This could be caused by the lack of opportunities to work in long projects and over an existent software product, where it is possible to show the effects of good/bad maintenance practices or SCM strategies.

The second question to answer was:

What software engineering topics/areas are regarded as most useful in the industry by the students?

The data showed that Software Engineering Professional Practice is the only area with a median of 5 (essential), while eight other areas obtained a 4 (very useful). It is worth mentioning that the least useful, according to the student perception, turned out to be Engineering Foundations. The analysis consisted in calculating the median of topics per each area; see Table V for the results.

A more granular analysis performed on the topic level showed that nine topics got a median of five (see Table VI).

Students reported to know all of those topics from their university background, however, only when participating in the project they had an opportunity to apply three of them in

TABLE III
COGNITIVE LEVELS OF SWEBOK AREAS ACCORDING TO STUDENT PERCEPTIONS

SWEBoK area	Cognitive level at the end of BSc	Cognitive level at the end the program
1: Software Requirements	1. Remember and 2. Understand	3. Apply
2: Software Design	1. Remember and 2. Understand	3. Apply
3: Software Construction	3. Apply	3. Apply
4: Software Testing	1. Remember and 2. Understand	3. Apply
5: Software Maintenance	0. Unknown	3. Apply
6: Software Configuration Management	1. Remember and 2. Understand	3. Apply
7: Software Engineering Management	1. Remember and 2. Understand	3. Apply
8: Software Engineering Process	1. Remember and 2. Understand	3. Apply
9: Software Engineering Models and Methods	3. Apply	3. Apply
10: Software Quality	1. Remember and 2. Understand	3. Apply
11: Software Engineering Professional Practice	1. Remember and 2. Understand	3. Apply
12: Software Engineering Economics	0. Unknown	0. Not required
13: Computing Foundations	3. Apply	3. Apply
14: Mathematical Foundations	3. Apply	0. Not required
15: Engineering Foundations	1. Remember and 2. Understand	0. Not required

TABLE IV
SWEBOK TOPICS WITH IMPROVED COGNITIVE LEVELS

SWEBoK topics
Unknown $\longmapsto$ Remember and Understand
10.4. Software Quality Tools
Unknown $\longmapsto$ Apply
5.5. Software Maintenance Tools
6.1. Management of the SCM Process
6.5. Software Configuration Auditing

TABLE V
SWEBOK AREAS SORTED BY USEFULNESS IN THE INDUSTRY

SWEBoK area	Median
11: Software Engineering Professional Practice	5
1: Software Requirements	4
2: Software Design	4
3: Software Construction	4
4: Software Testing	4
6: Software Configuration Management	4
7: Software Engineering Management	4
8: Software Engineering Process	4
13: Computing Foundations	4
5: Software Maintenance	3
9: Software Engineering Models and Methods	3
10: Software Quality	3
14: Mathematical Foundations	3
12: Software Engineering Economics	2
15: Engineering Foundations	1

TABLE VI
SWEBOK TOPICS CONSIDERED TO BE ESSENTIAL IN THE WORK PLACE

SWEBoK topics
3.3. Practical Considerations
3.4. Construction Technologies
6.6. Software Release Management and Delivery
11.2 Group Dynamics and Psychology
11.3. Communication Skills
13.3. Programming Fundamentals
13.4. Programming Language Basics
13.12. Database Basics and Data Management
14.2. Basic Logic

TABLE VII
SWEBOK TOPICS THAT STUDENTS KNEW BUT HAD NEVER APPLIED BEFORE THE PROGRAM

SWEBoK topics
6.6 Software Release Management and Delivery
11.2 Group Dynamics and Psychology
11.3 Communication Skills

practice (see Table VII).

In the context of software development, Software Release Management and Delivery constitutes a fundamental area of knowledge for every practitioner. On the other hand, Group Dynamics and Psychology together with Communication Skills are abilities strongly required for teamwork. It was a favorable outcome that students were presented with an opportunity to applied these topics during the project.

The most developed during the project areas are shown in Table VIII while the least developed ones are presented in Table IX.

TABLE VIII
SWEBOK AREAS THAT STUDENTS DEVELOPED THE MOST

SWEBoK topics
1: Software Requirements
2: Software Design
3: Software Construction
4: Software Testing
5: Software Maintenance
6: Software Configuration Management
7: Software Engineering Management
8: Software Engineering Process
9: Software Engineering Models and Methods
10: Software Quality
11: Software Engineering Professional Practice
13: Computing Foundations

TABLE IX
SWEBOK AREAS THAT STUDENTS DEVELOPED THE LEAST

SWEBoK topics
12: Software engineering economics
14: Mathematical foundations
15: Engineering foundations

C. Limitations and Threats to Validity

Construct validity: the data collection, particularly the survey on SWEBoK areas and topics, included examples of each topic presented to students for clarification. Also, in order to properly define the cognitive level of each SWEBoK topic, an adaptation of the Bloom's taxonomy was used. Although the insights of this experience are based on a limited number of sources, the data were obtained directly from the main stakeholders, namely the CS students. However, the results, in order to be generalized, require more generations of students to join similar programs.

External validity: it is worth to mention that the social service is specific for the Mexican context, however, it is representative of the country. Besides, we consider this program to be an initial step towards demonstrating the usefulness of running industry-based programs within university contexts.

Internal validity: the students joined the program voluntarily, and the information about the program was disseminated in the same way as the rest of the programs. A distinctive feature of this program was the grant offered to the students, which is not common to the majority of the programs; therefore, it could be a factor for the students to choose the program.

The findings are based on the perception of students and could be improved by applying specific assessments at certain phases of the program. Nevertheless, an important advantage of this study relies on the fact that we traced student data from studying a bachelor's degree till joining the IT workforce, which provides a deeper insight into the problem.

VI. CONCLUSIONS

In Mexico, university graduation requirements consist of developing a capstone project in the form of a thesis and its oral defense, completing credits of the BSc and 480 hours of social service.

As an alternative for providing students with relevant working experience, the UNAM College of Science put in place a social service program run in conjunction with a software development organization. In this program students had the opportunity to cover their graduation requirements while being immerse in a real project. In this first experience, 13 students participated representing around 11% of the CS students of a generation (114 students).

A total of 12 students completed their social service, which was the primary goal of the program, and 4 students finished their capstone projects and graduated (30.8% of the students who joined the program).

Despite differences across countries, there is definitely a growing interest in establishing a university-industry collaboration in order to promote well-prepared graduates, where these initiatives are welcomed by the students [30]. Teachers in charge of capstone project courses could benefit from a better understanding of what kind of problems students typically encounter in capstone projects [37].

We hope that this type of experiences will help to increase academic success, improve students' skills, reducing numbers of dropouts. It is worth to mention that all the students who participated in the program were employed in the IT sector after finishing the program.

Finding of the study showed that area 12: Software Engineering Economics was neither required during the project nor covered by the curricula according to the participants' perception. On the other hand, two important areas are perceived as not covered by the curricula but were required during the program: 5: Software Maintenance and 6: Software Configuration Management.

Finally, there is a clear cut division between cognitive levels developed by the students: *remember* and *understand* during their studies, *apply* when working. It was an expected outcome; however, it reinforces the idea of looking for alternatives for providing students with opportunities to apply their knowledge in practice in order to obtain an integral education.

As future work it is expected to run more programs involving IT companies in order to demonstrate the usefulness of aligning social service with relevant work experience, thus helping students to achieve their academic goals and to successfully join the industry. Besides, it is expected to explore potentials of this approach in other universities in the country. In parallel, an in-depth analysis of the current curricula should be carried out with the aim of including such missing topics as Software Maintenance and Software Configuration Management.

ACKNOWLEDGMENT

The authors would like to thank Mauricio Morgado and Ricardo Cruz for leading the project, the professors Guadalupe

Ibargüengoitia and Hanna Oktaba for coordinating the capstone projects. Also, thanks to the CS students that participated in this initiative for their contribution: Gerardo González, Adolfo Marín, Aarón Guerrero, Alan Gutiérrez, Jhovan Gallardo, Julio Chávez, Rafael Robles, Diana Góngora, Marco Estrada, Edson Servín, Edgar Tapia and Rodrigo Casiano.

REFERENCES

[1] ACM-IEEE. Software engineering curriculum guidelines, 2014.
[2] Tero Ahtee and Timo Poranen. Risks in students' software projects. In *22nd Conference on Software Engineering Education and Training*, 154–157. IEEE, 2009.
[3] Lorin Anderson, David Krathwohl, Peter Airasian, Kathleen Cruikshank, Richard Mayer, Paul Pintrich, James Raths, and Merlin Wittrock. A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives, Abridged edition, Longman, 2001.
[4] Barry Boehm and Daniel Port. Educating software engineering students to manage risk. In *23rd International Conference on Software Engineering*, 591–600, IEEE, 2001.
[5] David Broman, Kristian Sandahl, and Mohamed Abu Baker. The company approach to software engineering project courses. *IEEE Transactions on Education*, 55(4):445–452, 2012.
[6] Bernd Brügge, Stephan Krusche, and Lukas Alperowitz. Software engineering project courses with industrial clients. *TOCE*, 15:17:1–17:31, 2015.
[7] QS Crimson. QS World university rankings, 2019.
[8] Diario Oficial de la Federación (Official Journal of the Federation). Reglamento para la prestación del servicio social de los estudiantes de las instituciones de educación superior en la República Mexicana (Regulation for the social service provision of the students of the tertiary education institutions in Mexico), 1981.
[9] Dora Dzvonyar and Bernd Bruegge. Reaching steady state in software engineering project courses. In *Combined Workshops of the German Software Engineering Conference (SE 2018)*, 8–11, 2018.
[10] Fabian Fagerholm, Arto Hellas, Matti Luukkainen, Kati Kyllönen, Sezin Yaman, and Hanna Mäenpää. Designing and implementing an environment for software start-up education: Patterns and anti-patterns. *Journal of Systems and Software*, 146:1 – 13, 2018. https://doi.org/10.1016/j.jss.2018.08.060
[11] Fabian Fagerholm and Arto Vihavainen. Peer assessment in experiential learning assessing tacit and explicit skills in agile software engineering capstone projects. In *2013 IEEE Frontiers in Education Conference (FIE)*, 1723–1729. IEEE, 2013. https://doi.org/10.1109/FIE.2013.6685132
[12] Vinícius Gomes Ferreira and Edna Dias Canedo. Design sprint in classroom: exploring new active learning tools for project-based learning approach. *Journal of Ambient Intelligence and Humanized Computing*, 1–22, 2019. https://doi.org/10.1007/s12652-019-01285-3
[13] Panos Fitsilis and Alex Lekatos. Teaching software project management using agile paradigm. In *21st Pan-Hellenic Conference on Informatics*, 47:1–47:6, ACM, 2017. http://doi.acm.org/10.1145/3139367.3139413
[14] ISO/IEC TR 29110-5-1-2:2011 software engineering – lifecycle profiles for very small entities (VSEs) – part 5-1-2: Management and engineering guide: Generic profile group: Basic profile, ISO, 2011.
[15] M.J.I.M. Genuchten, van and D.R. Vogel. Getting real in the classroom. *Computer*, 40(10):106–108, 2007.
[16] Carlo Ghezzi and Dino Mandrioli. The challenges of software engineering education. In *27th International Conference on Software Engineering*, 637–638, ACM, 2005. http://doi.acm.org/10.1145/1062455.1062578
[17] Shanika Karunasekera and Kunal Bedse. Preparing software engineering graduates for an industry career. In *20th Conference on Software Engineering Education and Training (CSEET'07)*, 97–106. IEEE, 2007.
[18] Supannika Koolmanojwong and Barry W. Boehm. A look at software engineering risks in a team project course. *26th International Conference on Software Engineering Education and Training (CSEET)*, 21–30, 2013.
[19] Stephan Krusche, Lukas Alperowitz, Bernd Bruegge, and Martin O Wagner. Rugby: an agile process model based on continuous delivery. *RCoSE*, 14:42–50, 2014.
[20] Viljan Mahnic. A capstone course on agile software development using scrum. *IEEE Transactions on Education*, 55:99–106, 2012.
[21] Anne-Maarit Majanoja and Timo Vasankari. Reflections on teaching software engineering capstone course. In *10th International Conference on Computer Supported Education*, volume 2 of *CSEDU 2018*, 68–77, 2018.
[22] Erick Matla-Cruz, Miguel Morales-Trujillo, and David Velázquez-Portilla. Disciplinando equipos pequeños con prácticas ágiles (agile practices and small teams discipline). *Difu100cia*, 8(2):28–33, 2014.
[23] Christoph Matthies. Scrum2kanban: integrating kanban and scrum in a university software engineering capstone course. In *2nd International Workshop on Software Engineering Education for Millennials*, 48–55, ACM, 2018.
[24] Merriam-Webster.com. Capstone, 2019.
[25] Ana M Moreno, Maria-Isabel Sanchez-Segura, Fuensanta Medina-Dominguez, and Laura Carvajal. Balancing software engineering education and industrial needs. *Journal of Systems and Software*, 85(7):1607–1620, 2012.
[26] Bureau of Labor Statistics. Occupational Outlook Handbook, 2019.
[27] Maria Paasivaara, Dragoş Vodă, Ville T Heikkilä, Jari Vanhanen, and Casper Lassenius. How does participating in a capstone project with industrial customers affect student attitudes? In *40th International Conference on Software Engineering: Software Engineering Education and Training*, 49–57, ACM, 2018.
[28] Marko Poženel and Viljan Mahnič. Studying agile software estimation techniques: the design of an empirical study with students. *Global Journal of Engineering Education*, 18(2), 2016.
[29] Guillermo Rodriguez, Álvaro Soria, and Marcelo Campo. Virtual Scrum: A teaching aid to introduce undergraduate software engineering students to Scrum. *Computer Applications in Engineering Education*, 23(1):147–156, 2015.
[30] Manuel Rodríguez, Mario Vázquez, Hariklia Tsalapatas, Carlos de Carvalho, Triinu Jesmin, and Olivier Heidmann. Introducing lean and agile methodologies into engineering higher education: The cases of Greece, Portugal, Spain and Estonia. In *IEEE Global Engineering Education Conference*, 720–729, 2018. https://doi.org/10.1109/EDUCON.2018.8363302
[31] Robbie Simpson and Tim Storer. Experimenting with realism in software engineering team projects: an experience report. In *30th Conference on Software Engineering Education and Training (CSEET)*, 87–96. IEEE, 2017.
[32] IEEE Computer Society, Pierre Bourque, and Richard E. Fairley. *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*. IEEE Computer Society Press, 3rd edition, 2014.
[33] Davide Taibi, Valentina Lenarduzzi, Muhammad Ahmad, Kari Liukkunen, Maria Lunesu, Martina Matta, Fabian Fagerholm, Jürgen Münch, Sami Pietinen, Markku Tukiainen, Carlos Fernández, Juan Garbajosa, and Kari Systä. "Free" innovation environments: Lessons learned from the software factory initiatives. In *International Conference on Software Engineering Advances IARIA*, 2015.
[34] DGOSE UNAM. Dirección General de Orientación y Atención Educativa (General Directorate of Educational Orientation and Attention), 2017.
[35] DGPL UNAM. Portal de estadística universitaria (Statistics web portal of the University), 2019.
[36] SIASS UNAM. Sistema de Información Automatizada de Servicio Social (Social Service Information System), 2019.
[37] Jari Vanhanen, Timo OA Lehtinen, and Casper Lassenius. Software engineering problems and their relationship to perceived learning and customer satisfaction on a software capstone project. *Journal of Systems and Software*, 137:50–66, 2018.
[38] Elaine Venson, Rejane Figueiredo, Wander Silva, and Luiz Ribeiro. Academy-industry collaboration and the effects of the involvement of undergraduate students in real world activities. In *Frontiers in Education Conference (FIE)*, 1–8, IEEE, 2016.
[39] Ira Weissberger, Abrar Qureshi, Assad Chowhan, Ethan Collins, and Dakota Gallimore. Incorporating software maintenance in a senior capstone project. *International Journal of Cyber Society and Education*, 8(1):31–38, 2015. http://dx.doi.org/10.7903/ijcse.1238
[40] Claes Wohlin and Björn Regnell. Achieving industrial relevance in software engineering education. In *12th Conference on Software Engineering Education and Training (Cat. No. PR00131)*, 16–25, IEEE, 1999.
[41] Murat Yilmaz, Faris Serdar Tasel, Ulas Gulec, and Ugur Sopaoglu. Towards a process management life-cycle model for graduation projects in computer engineering. *PLOS ONE*, 13(11):1–17, 11 2018. https://doi.org/10.1371/journal.pone.0208012