A Constructivist Redesign of a Graduate-level CS Course to Address Content Obsolescence and Student Motivation

Lorenzo Angeli
lorenzo.angeli@unitn.it
Università degli Studi di Trento
Povo, TN, Italy

Juan Josè Jara Laconich
juan.jaralaconich@unitn.it
Università degli Studi di Trento
Povo, TN, Italy

Maurizio Marchese
maurizio.marchese@unitn.it
Università degli Studi di Trento
Povo, TN, Italy

ABSTRACT

The last decade has seen a rising popularity of active learning methodologies in Computer Science (CS), empowering students and developing their soft skills as well as their technical knowledge. In parallel, the speed of technological obsolescence also increased, creating challenges for teachers to keep their course content fresh and up to date. In this paper, we present a constructivist redesign of a Graduate-level laboratory course in Web Service Design and Engineering that leverages latent pockets of student knowledge to tackle these challenges through Learning by Teaching (LbT). We illustrate how such redesign was planned, deployed and evaluated, highlighting the guiding role of teachers in the process and discussing how this approach was able to solve the problem of keeping content updated while broadening both content and tools students were exposed to. Furthermore, we will discuss how the additional motivation stemming from their empowerment allowed students not only to perform more work compared to a lecture-based implementation, but also to perceive it in the end as a lesser load.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education**; **Software engineering education**; • **Applied computing** → *Collaborative learning*;

KEYWORDS

active learning; learning by teaching; actor-network theory; pedagogy; constructivism; graduate instruction

ACM Reference Format:
Lorenzo Angeli, Juan Josè Jara Laconich, and Maurizio Marchese. 2020. A Constructivist Redesign of a Graduate-level CS Course to Address Content Obsolescence and Student Motivation. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11–14, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3328778.3366910

1 INTRODUCTION

Constructivist education has enjoyed relatively high popularity in the past decades[4], with its applications mostly focused towards younger learners [28]. In CS in particular, constructivist pedagogies have been deployed successfully, applying models such as challenge-based learning [27][29], team-based learning [25][18], and, in a constructionist derivation, FabLearn [17].

A parallel phenomenon to this pedagogical evolution, however, has been an accelerating speed in the evolution — and therefore also obsolescence — of the technologies used in CS. The World-Wide Web and its tools, which represent the context of our course, have been no exception to this. As these technologies evolve, Higher Education (HE) Institutions need to update their curricula to make sure that what they are teaching is current, relevant, and useful for students when they enter the jobs market.

Some technologies and programming languages have remained in the last years relatively stable in popularity [7], and have become the backbone of many HE courses. Nonetheless, industry constantly asks for candidates to be trained in different technologies [31], and not necessarily the aforementioned "evergreens". Because of this, it makes sense for HE Institutions to expose their students to these trending technologies, though constantly updating courses to use flavour-of-the-year technologies would be cumbersome and impractical. Frequently changing programming languages taught in courses to follow trends might also create issues where — because of fast technological obsolescence — what students learn obsolesces quickly, lowering the value of the educational experience.

Nonetheless, the rise of online learning platforms such as Codecademy[1], but also edX and Coursera, shows that students and young professionals do have an interest in keeping up to date with trends, and are even willing to use their free time for this purpose.

In this paper, we will discuss how we redesigned a programming laboratory course on "Web Service Design and Engineering" through a constructivist framework. The course did not change its core content, overarching structure or Learning Objectives (LOs), but instead implemented a form of Active Learning, giving students more freedom in implementing their final projects, while keeping the same formal requirements as previous years. Starting from a reflection inspired by Actor–Network Theory (ANT), we sought to detach the course from its originally-adopted programming language and ground it instead on students' expertise and interests, with the goal of making the experience more updated and engaging.

In practice, students worked in teams to teach to their peers a laboratory class, and then formed different teams to work on the projects. During both phases, they were able to propose whatever tool or programming language they deemed suitable or interesting for the tasks they wanted to tackle. In labs, they were also encouraged to propose additional content that they thought would be relevant for the course and, in projects, they were asked to propose themes that would be useful in their own workflows and daily lives.

[1] https://www.codecademy.com/

SIGCSE '20, March 11–14, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6793-6/20/03.
https://doi.org/10.1145/3328778.3366910

In the following sections, we will first provide a brief overview of related work, focusing in particular on the theoretical grounding of ANT and, in the practice, on active learning and LbT experiences; we will then describe our implementation and the setup for measuring the outcomes of this redesign; we will later illustrate the main results, and finally provide conclusions and outline opportunities for further exploration of the outcomes of this experience.

2 RELATED WORK

This study is positioned at the intersection of two different segments of the broader topic of constructivist pedagogies. Epistemologically, we draw the grounding of our study from the understanding of Actor–Network Theory in education, while methodologically, we adopt the practices of Learning by Teaching.

ANT as a tool can be used to describe how the relationships between human actors are affected and shaped by non–humans (and their interrelations) [21]. One such key relationship is that where humans treat technical artifacts as "black boxes" which, when deeply nested and accumulated, require substantial effort to be unpacked [19]. In ANT, black–boxed non–human actors are also attributed with "agency", as they become able to concretely affect relationships between humans [30], as if the actions they allow or disallow represent the actions of these non–humans. This form of agency, it can be argued, can be traced back to the designer's intent embedded in technological artifacts [20].

CS classes are naturally full of tangible and intangible technological artifacts or, in ANT terms, designed non–humans. The idea of applying ANT to the field of education has been most thoroughly explored by Fenwick and Edwards [12]. Of particular interest for our discussion is that in the views of Fenwick and Edwards, the curriculum is reified (and thus black–boxed) holistically, along with the non–humans it employs as part of its design. In the context of a programming course, programming languages become one of the key tools enabling the process of translation [6] of theory into learned practice. In this sense, technological tools in a programming course are embedded — or rather, in–scribed [2] — in the course, and become part of the black box.

Our action aims at carrying out a process of de–scription [1] of languages and tools from the course, including in its stead tools brought by the students temselves. This, we hypothesised, could reduce the effort required by the students to unpack the black boxes, if nothing else because tools are now familiar. The way we enable this process is by letting students teach parts of the course, namely the lab sessions. Empowering the students in this way puts them on the same side as the teachers in defining the relationships in the class' actor–network and enables them to truly enrol the technologies to serve their learning.

While Active Learning (AL) has been a popular concept lately, its penetration in the STEM education field, and in particular in CS, is still relatively low [14]. Learning by Teaching is a form of AL originally developed for the teaching of foreign languages, which has most recently been the focus of the research activity of Grzega [16]. LbT has been called an effective way for students not only to more effectively retain course content [15], but also to develop soft skills such as communication [5].

Other forms of AL have been successfully used in universities [13], especially with the goal of improving engagement. In CS, the most common examples are forms of *Learning by Doing (LbD)* such as Problem–Based Learning (PBL). Nonetheless, a study by Okita [26] shows that LbT has lasting effect on student learning, even sometimes outlasting those of LbD. In parallel, it has been noted that students of CS are increasingly using their free time to develop personal projects and learn new tools [24], and are drawing strong motivation from these endeavours [23].

This last phenomenon implies that the classroom space increasingly contains pockets of latent or tacit knowledge which are not tapped by lecturing. Other contexts such as those of makerspaces have proven to be well-suited for these purposes, with studies showing how the mode of learning (formal, non-formal, informal) affects knowledge generation and transmission [10], but this remains a relatively inexplored topic in HE classroom teaching.

Transmission of formal and tacit knowledge, nonetheless, are radically different matters [33]. In this sense, as it is typical for AL processes, the role of the teacher as a guiding mentor and a facilitator is key [5]. To illustrate this in ANT terms, we can say that the teacher can use pedagogy (LbT) to enrol students and their knowledge to assist the process of de–scription of the non–humans normally embedded in courses, and achieve our design goals.

3 IMPLEMENTATION

When we started redesigning the course, we set one main overarching goal: we wanted to see if a radical empowerment of students could be used to increase the relevance of the course's content while promoting more active engagement at the same time. We also defined two ancillary goals: first, we expected that the new course structure would lend itself naturally to the development of student soft skills; second, we wanted to guide students toward creating content packages that would be reusable by future cohorts. These two ancillary goals were not addressed in this first run, and instead we wanted to observe if students also saw these opportunities.

3.1 Context

Our course involved a total of 24 students, 19 attending and 5 non-attending. The experience we report here focuses on the attending students, since they are the ones that were tasked to deliver the lab session to their peers. Non-attending students were asked to deliver the same material that went into the preparation of a lab session in a mock presentation to the teachers during their exam, but we will not discuss their case, since it lacks the fundamental student-to-student peer instruction.

The course took place twice a week in lectures of two hours each in a university in Northern Italy from September to December of 2018, and was entirely taught in English. 13 of the attending students were Italian, and 6 were international students. The teaching team was composed of one professor, a PhD student tasked with developing the lab methodology and mentoring for the labs and projects, and a post-doc researcher that was tasked mostly with technical mentoring for the labs and projects. The course lasted a total of 23 sessions of 2 hours each, of which 6 were dedicated to theory; 7 to the labs; 3 to plenary mentoring sessions for the labs; 4

to plenary mentoring sessions for the projects, and 3 to the course introduction, wrap-up and in itinere testing on the theory part.

The course's theory lectures were divided in four blocks: (1) a high-level introduction; (2) data representation, marshalling and exchange (XML and JSON); (3) service engineering techniques (REST and an introduction to SOAP); (4) designing and deploying service architectures. The labs sessions were partly pre-determined mirroring theory lectures (XML; JSON; Testing; writing REST services) and partly new topics proposed by students and validated with the teaching team (virtualization and microservices[2]; automated Documentation; Authentication and Authorization). For all labs, students were able to propose not only the flow of the session (number and difficulty of exercises, instruction style, coaching method etc.) but also, centrally, they were free to choose any programming languages or tools that they deemed were the most appropriate or interesting, as long as they provided adequate background information to their peers to be able to follow the session. Similarly, for final projects, students only had a document with loose guidelines, and were asked to propose their own topic and architecture, as long as it satisfied a number of given technical constraints.

For this article, we will focus on an analysis of the labs and how their workflow was established. This is both because labs were the centrepiece of the course and because theory was kept with no difference from the previous implementation, and projects kept the same guidelines, except that students were also asked with ideation and choosing their implementation tools.

3.2 Lab Workflow

On the third lecture, students were introduced to the general workflow of the labs (which we will briefly present here) and the predetermined lab topics.

Students were given two weeks to autonomously form groups of 2–3 students and select or propose lab topics, each team bidding on up to two topics they would be interested in preparing. Two weeks later, the bidding phase was concluded in a plenary session where teams negotiated together the final distribution of lab topics, and lab slots were assigned on the course's calendar, starting two more weeks from that date.

From here, each team went through the same pipeline:

(1) Drafting session
- *Format*: Informal discussion in office hours after class time.
- Students propose a high-level session outline to the teachers.
- Exchange of immediate feedback on broad changes that need to be made.
- Scheduling of a meeting for the first mentoring session.
- *Timing*: At least two weeks before final lab time.

(2) First mentoring session
- *Format*: Ad-hoc scheduled meeting of 1 hour.
- Students present a more detailed overview of the lab session.
- High-level presentation of theory, exercises, tools, etc.
- Discussion on session content, mode of delivery and presentation content.
- If major revisions are needed, a second mentoring session is scheduled. If only incremental improvement is necessary, the Dry Run is scheduled instead.
- *Timing*: Two weeks before final lab time.

(3) Second mentoring session (optional)
- *Format*: Ad-hoc scheduled meeting of 1 hour.
- Same mode as the first mentoring session, but focusing on any necessary improvements.
- At the end of the session, the Dry Run is scheduled.
- *Timing*: Between two and one weeks before final lab time.

(4) Dry run
- *Format*: Ad-hoc scheduled meeting of 1 hour.
- Students perform a mock run of the session in front of the teachers.
- All environments need to be ready and working, slides need to be in a final draft state, exercises are skipped to expedite the dry run and only discussed.
- Fine-grained feedback from the teachers both on technical content and on presentation/delivery modes.
- *Timing*: At least three days before final lab time.

(5) Lab package delivery
- *Format*: e-mail
- Delivery of an e-mail containing the final "lab package" (see below).
- Circulation of the lab package by the teachers.
- Students participating to the session are expected to download the lab package and install the environment to be ready at the start of the session.
- *Timing*: At least 24 hours before their sessions.

Each lab package was designed to be used in the class, but also as study material to be used at home by absent peers, non-attending students, those needing to revise and, ideally, also future cohorts. Each team had to deliver an "environment" which was either made of a VirtualBox VM containing all the necessary tools to perform the exercises in the sessions or, when the lab only used web-based tools, a document with links to those tools.

Each lab also required a set of slides that students would use to support their in-class presentation, which could possibly be made different between slides for attending and non-attending students. Slides should contain all theoretical/background information needed to follow the session — focusing especially on content not covered during the lectures — and a set of exercises, of which one — longer — designed as a take-at-home exercise. Finally, since students had the option to introduce new technologies that could be unfamiliar to their peers, they were asked to deliver a one-page "cheat sheet" summarizing all necessary syntax and key commands used in the session. These lab packages were published on the course website as soon as they had been received from the teams.

During the lab, students conducted their session with no intervention by the lecturers, which also followed the labs as if they were students. Teams were given full freedom on how to teach their class, also deciding who in the team would speak and present, and how to support their peers.

Immediately after the class, all attending students were asked to fill an anonymous questionnaire on Google Forms, which served as our main way to evaluate the success of the labs. We will present

[2]Introducing Docker as the main tool and use case.

the questionnaire in more detail in section 3.3. In parallel, each teacher also gave an evaluation of the lab for the purpose of grading. Dimensions evaluated by the teachers were the quality of (i) background information; (ii) materials; and (iii) exercises. Each dimension was evaluated on a 1–10 scale, and complemented with comments. In the week after their session, teams were finally asked to reflect on their own experience through a non-evaluated one-page document that we call an After–Action Report (AAR). This will also be illustrated in 3.3.

3.3 Evaluation Tools

To evaluate the success of our redesign, we relied on three main tools: Student Evaluation of Teaching Questionnaires (SET Questionnaires), Lab Questionnaires (LQs) and AARs.

The first, SET Questionnaires, are the standard anonymous evaluation of teaching questionnaires that all students have to fill at the end of a course in Italy. The SET Questionnaire has been developed by the Italian National Agency for the Evaluation of University and Research (ANVUR) following the European ESG standard [11]. While similar instruments have been in the past criticised, especially for not being able to accurately evaluate faculty's teaching effectiveness [32], we decided to still use them for two reasons: first, they represent our only source of constant historical data, since they have been gathered since the course's inception in its previous design; second, we are using SET Questionnaires not as a tool to directly evaluate our intervention, but to discuss their most literal interpretation, namely student *perceptions*. A sample SET Questionnaire with the same questions as the one used in this course can be found at [8][9].

The second tool, the Lab Questionnaire, has been developed *ad–hoc* for this course. It is a Google Form document with 10 mandatory questions using a 1–5 Likert Scale, with 5 being high, plus two optional open questions. The questions evaluated dimensions such as engagement, quality of the presentation, quality of materials, difficulty progression, use of class time, and perceived usefulness. A Likert question also asked students to self-evaluate previous knowledge on the lab's topic to filter out potential biases. The two open questions simply asked what was the most valuable aspect of the lab and what were points of improvement.

Questionnaires were delivered in anonymous form, using Google Forms' authentication to ensure that each student would only submit one response, and a narrow opening window to reduce chances of tampering and external influences. Students were encouraged to fill LQs at the end of each lab session they attended, and their presence was tracked in-person. Cross-checking the number of questionnaire responses with the number and testimony of present students allowed to preserve anonymity and avoid pollution of the data from external actors. Each student had to attend at least 5 out of 7 lab sessions (including theirs) to be considered attending.

Finally, the last tool was a written one-page group document, called the AAR. Here, we asked students to tell us: (i) what they think worked; (ii) what did not work; (iii) what they learned from the experience; and (iv) what they would change if they were to do the session again. The goal of this tool was to have a written trace of the perspective of the student-lecturers so that, in case a session would be controversial, we would be able to see both sides. Lastly, the AARs also gave us a way to gather insights on the teams' own sensemaking of the process they participated in.

4 RESULTS

To analyse the results of our intervention, we will address separately the three main evaluation tools that were used: the first, the SET Questionnaires, will serve as a longitudinal tool to evaluate how student perceived the course across the years; the second, the LQs, allows us to observe more closely perceptions tied to each session,; the third, the AARs, will be used to gather additional insights on critical aspects that might have arisen, and to draw general observations on how the students perceived their "role reversal" from listeners to leaders in the class.

It should be noticed that, for the longitudinal element, the course maintained the same lecture materials and project structure, with only minimal changes from year to year. Lab sessions were roughly equal in amount as the revised implementation, but used a completely different format. Previously, each session was a 2–hours tutorial where the Teaching Assistant (TA) would guide students through a set of exercises done in Java using the Eclipse IDE, along with a number of other tools. Attendance to the laboratories was not tracked, and labs were divided in three blocks (data processing and marshalling; REST; SOAP) with a short assignment to be delivered two weeks after the end of each block.

4.1 SET Questionnaires

SET Questionnaires have been gathered since the beginning of the previous implementation of the course, which was run four times, from 2014 to 2017.

As far as general appreciation of the course is concerned, the course was generally well-received, with only 2016 being particularly critical. In the four years, student satisfaction is at 93.5%, 95.2%, 68.0%, and 92.3%.

The most critical metric in SET Questionnaires, however, is the one related to perceived load of the course. Students feeling that the load of the course is balanced with the number of awarded credits is 61.3% in 2014, 76.2% in 2015, 52.0% in 2016 and 84.6% in 2017. This is consistently the lowest score the course receives in SET Questionnaires.

Since 2016, students also started suggesting to improve the quality of teaching materials. While 2014 and 2015 had all students satisfied about the quality of the teaching material, 2016 and 2017 report 80% and 84.6% of the students being satisfied, despite the material remaining the same. In 2016, many students also report in the open feedback section of the SET Questionnaires difficulties in configuring and using the lab and project environment.

The 2018 implementation seems to have solved these issues. In the 2018 SET Questionnaires, all students report being satisfied with the course, all students thought the load was balanced with the number of awarded credits, and that the teaching material was adequate. The only critical remark received in SET Questionnaires is in the open feedback section, where one student states that "student-led lab lectures need to be more consistent [sic]".

4.2 Lab Questionnaires

In LQs, the highest measures obtained are those of the support materials, Virtual Machines (VMs), and exercises, averaging 4.57, 4.56 and 4.53 out of 5 on the Likert scale. The lowest is the one for engagement, which is however still at 3.99 on average.

The measure of "previous knowledge on the topic" also allows us to identify topics where students feel stronger or weaker. Here, JSON and XML seem to be the topics where students were most familiar (3.36 and 2.82 respectively), and Virtualization was the least familiar topic (1.91).

The Virtualization lab was also, overall, the lowest-rated session, while still having an average of 3.98 across the measured metrics. Exercises have been particularly critical, with students noting in the open questions that the progression was at times too flat and at times too fast. The best-rated lab was instead the one on Authentication, especially on the engagement metric (4.50), with only a remark to give more time for exercises. These evaluations, and all overall evaluations given by students, do not significantly differ from those given independently by the course lecturers.

Reading the answers to the open questions, students clearly appreciated the variety of technologies and tools that were introduced, with many students stating that they were seeing them for the first time. Most requests for improvements, on the other hand, focus on the need to have a clear difficulty progression in exercises and, in general, stronger guidance during exercises.

Labs that introduced technologies that were unfamiliar to the class raised division: the Virtualization, REST and Documentation lab in particular relied more heavily on specific tools, and answers to open questions show that students always noticed this, but were divided in stating whether this was overall positive or negative.

4.3 After–Action Reports

In AARs, all groups evaluated positively the experience of teaching to peers. Three groups also explicitly mention feeling like the experience was a good way to hone their presentation skills and learn the nuances that go into preparing lectures. All groups also made remarks correlating the pacing and progression of the exercises they proposed with how the class appeared to receive their lecture.

We also observed during the preparation sessions that teams introducing new technologies usually did so following the lead of one group member that was already interested in the chosen technology. In AARs, nonetheless, all teams that introduced new technologies report appreciating the opportunity of having to learn a new tool in enough depth to be able to explain it to their peers.

Three groups also say that, if it were possible, they would have preferred to have more time in each lab session to go more in depth.

Finally, the Authentication lab team reflects on their (relatively heavy) use of comedy in the form of "memes" as a mean to engage the class. In their AAR, the team states that they tried to use them to "express an opinion and improve participation", and that *"the small number of meme [sic.] reduced the impact of the lesson"*, something which LQ data seems to disprove.

5 DISCUSSION

Different insights can be gathered, which we will present in four parts: (i) SET Questionnaires; (ii) LQs; (iii) AARs and (iv) general.

SET Questionnaires: the measure on perceived course load is particularly interesting. From a strictly quantitative perspective, the redesigned implementation did not reduce student load, and indeed, the redesigned course did not remove any topic covered in lectures or labs in the previous implementation. Project requirements also remained the same as the previous years, with the only difference that students were tasked to propose their own project rather than developing one assigned by the teachers.

With this in mind, it could be argued that the load on the student actually *increased*. Before the redesign, labs were used to exemplify theory and to learn the tools for the project. After, they acquired a bigger role, and required extra effort from the students to ideate, prepare teaching materials, and execute. Similarly, projects, which before were a pre-assigned exercise, required extra effort in ideation, which was moved from the teachers to the students and mentors.

These results can be read under the light of the fact that, in a way, the previous implementation ended up tying the subject-matter of the course (web services) to its implementation language, Java. In labs, the redesigned course flow definitely represented an increase in load both because of the added content (Virtualization introduced Docker, Automated Documentation and Authentication were not covered previously) and of the increase in tools and languages that the students were exposed to. In projects, the decision of letting students to be free to choose the implementation language and tools allowed them to reduce their load in a customised way, as students were encouraged to use the tools that they were the most familiar with, focusing on making sense of the content of the course rather than of the tools previously used by the course (Java and Eclipse).

When redesigning the course, we did not expect that students would perceive a decrease in load compared to the previous formula. If anything, we actually expected students to find the new implementation to be heavier, and that this would be a weakness of our model that we would need to address in the future as a point of improvement. We knew, from discussing with students of the previous years, that the Java and Eclipse environment was perceived as cumbersome and something that added extra complexity. What we did not expect was that removing this constraint and turning the choice of tools in the hands of the students would create so much slack in terms of perceived load. We also think that the added motivation given by the higher degree of empowerment afforded to the students helped them in tackling the extra challenge, but we do not have measures to help us back this claim.

Lab Questionnaires: labs have been overall evaluated positively. Some open questions stem from the LQ's outcomes, though. The highest measures — those related to class materials — are also those where, by nature, students have the least expertise to judge whether material is of actually good quality or not, since they are likely seeing that content for the first time. The lowest measure being engagement — while still being high — also suggests us that further exploration on our course model should attempt to use different measurement tools, since engagement is by its nature a more subjective matter, and thus something that students would be able to reliably report on. Nonetheless, the answers to the open questions, when given, have been constructive and shown attentiveness and insight, which makes us think that the questionnaire was taken seriously by students.

The introduction of new technologies seems to create a fine line between generating insight and confusion in students, and exercises appear to be the key in keeping the class engaged in the process. The Virtualization and REST sessions both introduced new technologies, but the comments on the Virtualization lab suggest that a non-smooth progression in the exercises made the class unable to draw strong conclusions on the lab content and the presented technology.

After–Action Reports: all students express a positive evaluation of the overhauled experience. Because of this, we think that the LbT represents a high added value of the course also from the perspective of the students, and could have a bigger place in the course's LOs as well as in its promotion in the department. The pacing of exercises also seems to be a main concern for groups reflecting on their own sessions, and observations in the AARs generally reflect those present in the open comments of LQs. The observations on soft skills made by the students also suggest us that students are aware of the importance of soft skills both in general and in the course. In general, the AARs seem to be an insightful yet somewhat superficial tool. A more structured template to guide reflection might help in making the obtained information more relevant, since the perspective of the student-lecturers is important in gathering a full picture of each lab session.

General: We provided an example of how a laboratory programming course can de-scribe programming languages from its core content. This has the net effect of emphasising theory and programming paradigms, reframing the technologies used in the laboratories as instantiations of the paradigm rather than as self-standing techniques. By leveraging the silent technical knowledge present in the classroom through the students, the course remains relevant for a longer time, and obsolescence becomes a matter of programming paradigms rather than one of programming languages.

We also linked motivation with the perceived reduction in load that the students experienced. This, however, can become potentially problematic. Students report a lesser load compared to the previous implementation, but they are objectively required to perform more demanding tasks. Motivation in this sense can become a double-edged sword: it has the potential to reduce the perceived load, but motivated students might be led to situations of burnout [22]. To get more insight on the reduced perception of load, we can once again use the lens of ANT to look at the course as a whole. The design choices embedded in programming languages [20] affect how the students carry out their task, shaping how they think, and how subparts of a larger project interact, affecting also division of work. Students choosing their tools empowers them to reshape the relationships in the actor-network which the course represents, but this is a source of extra complexity and potential additional stress if tools are chosen unwisely.

6 CONCLUSIONS AND FUTURE WORK

In the previous sections, we illustrated how the course was redesigned from a formula based on traditional lectures to a structure that embraces a constructivist approach, empowering students to become co-owners of the classroom. We now wish to draw some conclusions from three main points of view: (i) LOs and skills; (ii) motivation and load; and (iii) scalability.

LOs and skills: the overhauled course implements the same core LOs, and asks the students to develop the same hard competences. A potentially relevant opportunity and field of future work, however, is that of using the course to explicitly address the development of soft skills such as communication, presentation, teamwork and leadership. From a "hard" skills perspective, instead, the assumption that it is desirable to have a more general view of a programming paradigm rather than specific knowledge of currently trending languages comes from informal discussion with students and companies. A systematic survey of whether this shift represents an added value in the market for potential employers or not also represents an opportunity for future research.

Motivation and load: we discussed how the perception of a heavy load was reduced in the overhauled implementation, though the load put on students was objectively higher. Empowered students surely enjoy the process more, but teachers need to pay extra attention to the consequences of the choices that students make. Different tools and styles of teaching the lab, to name two key choices, both affect how much time and energy students will need to invest in order to make sure that the laboratory will be successful.

Teachers need to have a strong ethical compass and develop a fine sense of empathy to push the students toward greater achievement while ensuring they are not overworking. This is particularly true for projects: if teachers misjudge the level of load of a project when students set their own tasks and decide their own tools, motivation might soon vanish. These reflections, here just stated as hypotheses and more general observations, are not formally explored in this article, and represent opportunities for further research.

Scalability: as this is a key challenge for many pedagogies alternative to lectures, we feel the need to draw conclusions in this sense. We think that our methodology would have worked better with a bigger class — around 30 students. This would have ensured that sessions were led by teams of around 4 students, giving capacity to the team leading the session to address questions and assist students experiencing difficulties. Surely, our approach can not scale indefinitely, since adding extra lab sessions also implies extra work from the teachers. A possible solution could be to split the class in two, so that labs are done in parallel by two teams to two subsets of the class, or to adopt a framework similar to micro-classes [3].

Another scalability concern stems from the idea that lab sessions should represent a package that can be reused by future cohorts as a source of extra exercises or more content. If this were to be adopted, we think there could be issues not only of incremental accretion of content (leading to heavier load), but also issues in taking questions about the material and evaluating it. In this sense, a framework of peer evaluation could represent a solution.

In conclusion, we think that our experiment — while at a small scale, with substantial room for improvement, and many untapped resources — represents a good example of how teachers can take concrete steps to lead students not as top-down managers, but as mentors and even, in the learning of new tools, as co-learners. As technology evolves and information spreads and decentralises, opportunities to create knowledge in the classroom space follow a similar path. If the role of the teacher is that of guiding students toward knowledge, we ought to teach our students not to look far away, but to look at their peers, and discover opportunities which are surprisingly close and humanly rich.

ACKNOWLEDGMENTS

The authors wish to thank all students of the "Introduction to Service Design and Engineering" course in University of Trento in cohort 2018–2019 for enthusiastically taking part in the course's redesigned experience.

REFERENCES

[1] Madeleine Akrich. 1992. The De-Scription of Technical Objects. In *Shaping Technology/Building Society: Studies in Sociotechnical Change*, Wiebe E. Bijker and John Law (Eds.). https://pedropeixotoferreira.files.wordpress.com/2014/03/akrich-the-de-scription-of-technical-objects.pdf

[2] Madeleine Akrich and Bruno Latour. 1992. *A summary of a convenient vocabulary for the semiotics of human and nonhuman assemblies.* The MIT Press.

[3] Christine Alvarado, Mia Minnes, and Leo Porter. 2017. Micro-Classes: A Structure for Improving Student Experience in Large Classes. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE '17.* ACM Press, Seattle, Washington, USA, 21–26. https://doi.org/10.1145/3017680.3017727

[4] Roya Jafari Amineh and Hanieh Davatgari Asl. 2015. Review of Constructivism and Social Constructivism. *Journal of Social Sciences, Literature and Languages* 1, 1 (2015), 9–16.

[5] Safiye Aslan. 2015. Is Learning by Teaching Effective in Gaining 21st Century Skills? The Views of Pre-Service Science Teachers. *Educational Sciences: Theory & Practice* 15, 6 (Dec. 2015). https://doi.org/10.12738/estp.2016.1.0019

[6] Michel Callon. 1984. Some elements of a sociology of translation: domestication of the scallops and the fishermen of St Brieuc Bay. *The Sociological Review* 32, S1 (1984), 196–233. http://onlinelibrary.wiley.com/doi/10.1111/j.1467-954X.1984.tb00113.x/full

[7] Pierre Carbonnelle. 2019. PYPL PopularitY of Programming Language index. http://pypl.github.io/PYPL.html

[8] Agenzia Nazionale di Valutazione del Sistema Universitario e della Ricerca (ANVUR) (Italy). 2013. Proposta operativa per l'avvio delle procedure di rilevamento dell'opinione degli studenti per l'A.A. 2013-2014. http://www.anvur.it/attachments/article/26/RIlevazioneOpinioneStudDef_06_11_13.pdf

[9] Agenzia Nazionale di Valutazione del Sistema Universitario e della Ricerca (ANVUR) (Italy). 2019. Rilevazione Opinioni Studenti – ANVUR – Agenzia Nazionale di Valutazione del Sistema Universitario e della Ricerca. https://www.anvur.it/attivita/ava/opinioni-studenti/

[10] Árni Már Einarsson and Morten Hertzum. 2019. Scaffolding of Learning in Library Makerspaces. In *Proceedings of the FabLearn Europe 2019 conference on ZZZ - FabLearn Europe '19.* ACM Press, Oulu, Finland, 1–8. https://doi.org/10.1145/3335055.3335062

[11] European Students' Union (ESU) (Belgium), European University Association (EUA) (Belgium), European Association of Institutions in Higher Education (EURASHE) (Belgium), and European Association for Quality Assurance in Higher Education (ENQA) (Belgium). 2015. *Standards and Guidelines for Quality Assurance in the European Higher Education Area (ESG).* European Students' Union. 20 Rue de la Sablonniere, 1000 Bruxelles, Belgium. Tel: +32-2-502-23-62; Fax: +32-2-706-48-26; e-mail: secretariat@esu-online.org; Web site: http://www.esu-online.org. OCLC: 7927262920.

[12] Tara Fenwick and Richard Edwards. 2010. *Actor-Network Theory in Education.* Routledge. https://doi.org/10.4324/9780203849088

[13] Scott Freeman, David Haak, and Mary Pat Wenderoth. 2011. Increased Course Structure Improves Performance in Introductory Biology. *CBE—Life Sciences Education* 10, 2 (June 2011), 175–186. https://doi.org/10.1187/cbe.10-08-0105

[14] Scott Grissom, Renée Mccauley, and Laurie Murphy. 2017. How Student Centered is the Computer Science Classroom? A Survey of College Faculty. *ACM Transactions on Computing Education* 18, 1 (Nov. 2017), 1–27. https://doi.org/10.1145/3143200

[15] Joachim Grzega. 2005. Learning By Teaching: The Didactic Model LdL in University Classes. (2005). http://www.joachim-grzega.de/ldl-engl.pdf

[16] Joachim Grzega and Marion Schöner. 2008. The didactic model LdL (Lernen durch Lehren) as a way of preparing students for communication in a knowledge society. *Journal of Education for Teaching* 34, 3 (Aug. 2008), 167–175. https://doi.org/10.1080/02607470802212157

[17] Mikkel Hjorth, Ole Sejer Iversen, Rachel Charlotte Smith, Kasper Skov Christensen, and Paulo Blikstein. 2015. *Digital Technology and design processes: Report on a FabLab@School survey among Danish youth.* Technical Report. Aarhus University Library. https://doi.org/10.7146/aul.12.11

[18] Michael S. Kirkpatrick. 2017. Student Perspectives of Team-Based Learning in a CS Course: Summary of Qualitative Findings. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE '17.* ACM Press, Seattle, Washington, USA, 327–332. https://doi.org/10.1145/3017680.3017699

[19] Bruno Latour. 1987. *Science in action: How to follow scientists and engineers through society.* Harvard university press.

[20] Bruno Latour. 1992. "Where are the missing masses?". In *"Where Are the Missing Masses? The Sociology of a Few Mundane Artifacts".*

[21] Bruno Latour and others. 2005. *Reassembling the social: An introduction to actor-network-theory.* Oxford university press.

[22] Shu-Hui Lin and Yun-Chen Huang. 2014. Life stress and academic burnout. *Active Learning in Higher Education* 15, 1 (March 2014), 77–90. https://doi.org/10.1177/1469787413514651

[23] Robert McCartney, Jonas Boustedt, Anna Eckerdal, Kate Sanders, Lynda Thomas, and Carol Zander. 2016. Why Computing Students Learn on Their Own: Motivation for Self-Directed Learning of Computing. *ACM Transactions on Computing Education* 16, 1 (Jan. 2016), 1–18. https://doi.org/10.1145/2747008

[24] Robert McCartney, Anna Eckerdal, Jan Erik Moström, Kate Sanders, Lynda Thomas, and Carol Zander. 2010. Computing students learning computing informally. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research - Koli Calling '10.* ACM Press, Berlin, Germany, 43–48. https://doi.org/10.1145/1930464.1930470

[25] Larry K Michaelsen, Arletta Bauman Knight, and L Dee Fink. 2004. Team-based learning: A transformative use of small groups in college teaching. (2004).

[26] Sandra Y Okita and Daniel L Schwartz. 2006. When observation beats doing: Learning by Teaching. (2006), 7.

[27] Timothy K. O'Mahony, Nancy J. Vye, John D. Bransford, Elizabeth A. Sanders, Reed Stevens, Richard D. Stephens, Michael C. Richey, Kuen Y. Lin, and Moe K. Soleiman. 2012. A Comparison of Lecture-Based and Challenge-Based Learning in a Workplace Setting: Course Designs, Patterns of Interactivity, and Learning Outcomes. *Journal of the Learning Sciences* 21, 1 (Jan. 2012), 182–206. https://doi.org/10.1080/10508406.2011.611775

[28] Jean Piaget. 2005. *The psychology of intelligence.* Routledge.

[29] Alan R. Santos, Afonso Sales, Paulo Fernandes, and Mark Nichols. 2015. Combining Challenge-Based Learning and Scrum Framework for Mobile Application Development. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '15.* ACM Press, Vilnius, Lithuania, 189–194. https://doi.org/10.1145/2729094.2742602

[30] Edwin Sayes. 2014. Actor–Network Theory and methodology: Just what does it mean to say that nonhumans have agency? *Social Studies of Science* 44, 1 (Feb. 2014), 134–149. https://doi.org/10.1177/0306312713511867

[31] TIOBE. 2019. Programming Languages Definition. https://www.tiobe.com/tiobe-index//programming-languages-definition/

[32] Bob Uttl, Carmela A. White, and Daniela Wong Gonzalez. 2017. Meta-analysis of faculty's teaching effectiveness: Student evaluation of teaching ratings and student learning are not related. *Studies in Educational Evaluation* 54 (Sept. 2017), 22–42. https://doi.org/10.1016/j.stueduc.2016.08.007

[33] Georg Von Krogh, Kazuo Ichijo, Ikujiro Nonaka, and others. 2000. *Enabling knowledge creation: How to unlock the mystery of tacit knowledge and release the power of innovation.* Oxford University Press on Demand.