Sustainable Bytes: Embedding Sustainability Awareness in Computer Science Education

Michael Wahler
Institute of Computer Science
Zurich University of Applied Sciences (ZHAW)
Winterthur, Switzerland
ORCID 0009-0006-5301-6315

Josef Spillner
Institute of Computer Science
Zurich University of Applied Sciences (ZHAW)
Winterthur, Switzerland
ORCID 0000-0002-5312-5996

***Abstract*—Industry and academia recognize the importance of computer scientists having sustainability awareness and knowledge. While bachelor's programs in computer science provide a comprehensive education, the topic of sustainability is typically only briefly covered in a few elective modules. Consequently, most computer science graduates lack awareness and fundamental knowledge of software sustainability. How can we ensure that all graduates receive basic sustainability training without increasing their workload or removing other courses from the curriculum?**

To achieve this goal, we have developed the novel concept of *Sustainable Bytes*, a voluntary online course whose sections (the *bytes*) are linked from the educational material of existing mandatory courses. In this experience report, we present the goals of the course, the key decisions made during the design process, and an overview of its contents. Evaluation and usage statistics from the first instance of the course reveal that only 26 % of first-semester students accessed it and 7 % completed it. Consequently, we share lessons learned to help us and other educators increase student participation in the future.

***Index Terms*—sustainability, computer science, software engineering, education.**

I. Introduction

Computer science (CS) students across universities receive a broad and sound knowledge of computer science and software engineering. However, the curricula hardly cover the integration of sustainability concepts in software engineering education, which is an important part of the CS curriculum. Yet, software engineering (SE) processes should meet sustainability criteria across different dimensions [1], [2]. Some of these criteria may conflict with one another, necessitating trade-offs. Therefore, software engineers must possess sustainability awareness and knowledge to navigate these challenges effectively. As an example, optimizing a computationally heavy algorithm may result in a lower energy consumption (having a positive effect on the *environmental* dimension of sustainability), but can lead to an increased mental load of the programmers (negatively impacting the *individual* dimension) or increased maintenance cost (negatively impacting the *economic* dimension).

Therefore, there is a growing understanding among educators that contemporary CS education should cover sustainability concepts [3], [4]. In the industry, there is a need for technical personnel to acquire sustainability knowledge through in-house or external training and collaborations [5]. Today, however, there is often little knowledge about sustainability in the industry, except for technical aspects such as *performance* and *maintainability* [6], [7].

Our university, Zurich University of Applied Sciences, is one of the largest in Switzerland with strong and established CS programs. Our CS curriculum[1] provides students with a wealth of theoretical knowledge and practical skills, which allows them to build high-quality software systems that meet customer requirements. Yet, most of our CS students obtain their degree without any exposure to the topic of sustainability in software engineering. Around one third of our students participate in elective modules that cover some aspects of sustainability, but these modules do not provide a comprehensive view of all dimensions of sustainability. Therefore, there is a need for a course on sustainable software and software engineering and exposing all our students to it. The content of this course should cover the theoretical aspects of these skills (e.g., awareness and knowledge of sustainability, influence of quality criteria on sustainability) and the practical aspects (e.g., increasing maintainability by decoupling software components).

Therefore, we have designed and implemented such a course (called *Sustainable Bytes*) at the start of the autumn semester 2024 and made it available to the 204 first-year CS students. At the end of the semester, we analyzed usage statistics and conducted a survey among these students. In this experience report on the first implementation of the course, we first present the background of our CS program and basic terms of sustainability in Section II. Then, we introduce the goals and design of our course (Section III), followed by the course contents (Section IV). In Section V, we evaluate our course based on the usage statistics and the survey. In Section VI, we present valuable lessons learned and discuss how future iterations of the course can be improved. We present related work on sustainability in (CS) education and on the industry needs for such education in Section VII. We conclude our report in Section VIII.

Partially funded by the ZHAW Sustainable Impact Program

[1]https://www.zhaw.ch/en/engineering/study/bachelors-degree-programmes/computer-science

II. Background

Our university offers a bachelor's degree program in Computer Science. We claim that its educational content is representative of many CS programs, aligned with educational best practices such as ACM's curricula recommendations that cover societal topics like ethics and accessibility, but not sustainability [8]. The program comprises 180 ECTS credits and requires 6 or 8 semesters for full-time and part-time students, respectively. During the first two years, our students learn foundational concepts and methods of computer science with a focus on a current, comprehensive, practice-based, and sustainable education. In the last year, our students choose from a selection of compulsory elective modules, which provide in-depth knowledge on selected topics. In their final semester, they also write their bachelor's thesis.

Compulsory module lessons are conducted in face-to-face classes of 20–30 students, allowing close interaction with lecturers who assist with theoretical and practical advice. The lessons for compulsory *elective* modules usually take place in slightly larger classes of 20–50 students. Teaching materials and organizational information for each module are provided in the learning management system Moodle[2], with one course per module and semester.

Despite its emphasis on sustainable education—meaning that the students can apply what they have learned for the rest of their lives—the mandatory modules of the program do not *explicitly* cover sustainability aspects in CS or SE as a mandatory part, despite teaching many sustainable practices such as agile software development or communication skills for team projects. There are, however, two elective modules that incorporate sustainability aspects in the in curriculum. First, the module "Serverless and Cloud Application Development " (SCAD, taught by the second author), the students learn about energy consumption of cloud solutions along with cloud-native sustainability [9] and must account for it in their practice project. Second, the module "Advanced Software Engineering" (ASE, taught by the first author and Marcela Ruiz) covers the different dimensions of sustainability introduced by Duboc et al. [10] and teaches the students to relate to them in requirements engineering and software design.

While teaching their elective modules, the authors observed that students are curious and eager to learn about sustainability in computer science. They take the topic seriously, quickly grasp key concepts, and apply them to their work. Many of them choose topics around sustainability and the United Nations' 17 Sustainable Development Goals (SDGs) [11] for their semester projects when given a free choice of topic. However, only around one third of our CS students attend one of these two elective modules on average, which means that around two thirds of our students obtain their degree with virtually no knowledge of sustainability in computer science. This is something that we want to change by introducing *Sustainable Bytes*.

[2]https://moodle.org

A. Sustainable Software

Sustainability is often used synonymously with environmental protection such as green software design and coding [12], carbon footprint of software applications and services at runtime [13], [14] and the battery efficiency of devices [15]. These are important learning units, but they often remain on a technical level, leaving a gap between technology and socio-technical education.

In our work, we therefore use the broader definition of sustainable development of the UN's Brundtland report [16], which sets the goal of meeting "the needs of the present without compromising the ability of future generations to meet their own needs". We apply this notion both to the software being developed and the software engineering process itself.

Furthermore, in the context of CS and SE, sustainability has a broader focus than environmental protection. We follow the definition of Duboc et al., who present five dimensions of software system sustainability [10]. Originally developed for requirements engineering, this framework is relevant and suitable for the other phases in software engineering too [1], [17]. These five dimensions comprise *environmental* aspects (e.g., the energy consumption of the software during its creation (development) and operation, *social* aspects (e.g., contributing to equal opportunities by making software accessible to a diverse set of users), *technical* aspects (e.g., maintainability and adaptability of the software), *economical* aspects (e.g., operating costs of the software compared to its benefits), and *individual* aspects (e.g., improving developer's quality of life by automating repetitive tasks).

III. Goals and Design of the Course

Software engineering directly and indirectly impacts the environment, society, and individuals. To understand, assess, and optimize this impact, our CS students need *additional* knowledge about sustainability in software engineering. Therefore, we defined these goals for the new *Sustainable Bytes* course:

Goal 1 Create sustainability awareness in *all* our computer science students.
Goal 2 Provide educational material on sustainable software and software engineering.
Goal 3 Enable our students to assess their own knowledge of the course material.

Before designing the course, we discussed with the study program director and the module coordinators for the software engineering modules how such a course could be integrated into the current curriculum. The main requirement was that the course must not incur additional obligatory workload to the students. The reason is that the curriculum of our bachelor's degree program is designed for 180 ECTS credits (see Section II). Adding an obligatory course would entail removing an existing course from the program.

When designing the course, we followed the *constructive alignment* principle [18], which starts with setting the intended learning outcomes. We suggest three intended learning outcomes (ILO) according to the previously set goals:

ILO1 The students know five different dimensions of software sustainability (according to the Sustainability Awareness Framework [10]) and the different orders of effect of sustainability (levels 1-3).

ILO2 The students know which aspects of software engineering have an impact on the sustainability of digital products and services.

ILO3 The students can examine a software engineering process for sustainability criteria and improve it for greater sustainability.

Having defined the intended learning outcomes, we subsequently define the teaching and learning activities (Section III-A) and the examinations (Section III-B).

A. *Teaching and Learning Activities*

As mentioned in Section II, the bachelor's degree program our university primarily focuses on face-to-face classes. However, there is an important reason that speaks against designing our course as a face-to-face class. Since our course can only be offered as a voluntary course, we would ask students to attend an on-site lecture for which they do not receive any ECTS credits. Therefore, only a highly motivated and curious fraction of students would attend the course, which contradicts Goal 1. Furthermore, it is likely that in this scenario, those students who are already interested in sustainability would attend a face-to-face class and we would preach to the converted.

Therefore, we decided to design the course as an online course, which allows students to access the course material at a convenient time. The downside of this decision is the lack of direct contact between students and lecturers, and among students. Consequently, students learn in isolation, necessitating digital means for asking questions and tracking progress.

For deciding on the format of the online course, we considered several options. Important factors were the location of the teaching material (in a central location or distributed across other courses), the accessibility for different user groups, the possibilities of making the course interactive, and the longevity of the course and its teaching materials over a longer time horizon. The options that were considered are summarized in Table I.

Based on these factors, we decided to build a new dedicated Moodle course on sustainable software engineering, using Moodle's features of integrating different types of content and various ways of establishing interaction with its users. This brings us closer to achieving Goal 2 (providing educational material) and Goal 3 (self-assessment of students' knowledge). However, building yet another Moodle course takes us away from Goal 1 (reaching all students): how can we make our students aware of the new course and encourage them to register for it? This gave us the idea of enriching the Moodle courses of compulsory modules with pointers to relevant sections of the new Moodle course. We call these pointers *Sustainable Bytes*.

We selected to add *Sustainable Bytes* to four compulsory modules (called *base modules* in this paper) in the first two semesters of the bachelor's degree program. The *Introduction to Programming 1&2* modules (abbreviated as PROG) cover programming concepts such as object-oriented and functional programming, clean code, software testing, and concurrent programming. The *Software Projects 1&2* modules (abbreviated as PM) introduce the students to project work in software engineering. In these modules, the students develop software in teams. Besides sharpening their technical and project management skills, the students learn about written and oral communication and basic requirements analysis. This wide range of topics covers several aspects in different dimensions of sustainability.

B. *Examinations*

The constructive alignment principle [18] suggests designing examinations after the learning outcomes and the course contents have been defined. For a voluntary course, we do not see a need to provide examinations. However, we wanted to allow students to assess their knowledge and be able to repeat relevant parts of the course. To provide such self-assessments, we decided to use the *quiz* feature of Moodle. Moodle quizzes can be configured flexibly with different types of questions, such as multiple-choice questions or cloze-text tests.

It is obvious that students learn more if they not only read the contents of the online course, but also reflect on it, e.g., by completing the quizzes. This requires additional effort. Since we cannot grant ECTS credits for the course at the moment, we decided to add an element of gamification to the course. Therefore, we designed virtual badges that are awarded to students for completing the quizzes. These badges can be collected on the Moodle platform across all courses. When they complete all quizzes, they are awarded an additional badge, *Sustainability Champion* (see Figure 1).

Fig. 1. Virtual Badge "Sustainability Champion".

IV. Course Contents

The Moodle page of the course consists of four chapters: an introduction to sustainability, an explanation of the five dimensions of sustainability for software, an investigation of the relation of first-year software engineering topics with sustainability, and the quizzes.

Before the actual content starts, there is a motivation for the course and usage instructions that help students navigate through the content of the course.

TABLE I
OPTIONS CONSIDERED FOR ONLINE COURSE DESIGN

Course type	*Teaching material*	*Access*	*Interactivity*	*Longevity*
Existing Moodle courses of mandatory modules	spread across several courses	restricted to course participants	good - forums and quizzes	courses may close after they end
New dedicated Moodle course	in one place	anyone at institution	good - forums and quizzes	can be open indefinitely
Static website	in one place	anyone at institution and beyond	n/a	can be open indefinitely
Interactive website	in one place	anyone at institution and beyond	very flexible	can be open indefinitely, but maintenance effort for interactive elements

Fig. 2 gives a visual impression of the course navigation structure. The module-specific acronym references (e.g. PM1) are known to the students and are specific to our computer science program whereas the general navigation approach could be re-used in other institutions and curricula.

A. Introduction to Sustainability

Assuming that most students only have a marginal understanding of sustainability (at least in the context of software and software engineering), the first chapter of the course introduces the concept of sustainability, emphasizing its importance in various domains, including natural resources, species preservation, and human well-being based on the Brundtland Commission's 1987 report [16]. The chapter highlights the significant impact of digitalization on society and the environment, noting the high energy consumption of data centers and the pressure on software engineers as examples. It stresses the responsibility of future software engineers to consider sustainability in their work, providing tools and methods to create sustainable digital solutions.

The chapter also categorizes the impacts on sustainability of software engineering into three levels (following Hilty et al. [19]): first-order (direct impacts like energy-efficient coding), second-order (indirect benefits such as long-term cost savings and improved productivity), and third-order (broad societal and environmental changes). It connects software engineering to the United Nations' SDGs, illustrating how digital tools can contribute to goals such as reducing poverty, improving health, and promoting clean energy. The chapter concludes by emphasizing the need for both sustainable software products and SE processes to ensure long-term positive impacts on sustainability.

At the end of the chapter, the students find additional links to literature on selected advanced topics. These links are presented with a guide that suggests a reading order and provides a short summary for each linked document.

B. Dimensions of Sustainability

The second chapter introduces the five dimensions of sustainability from Duboc et al. [10]. To this end, it provides a description of each dimension with examples and pointers to the respective topics in the base modules, which are explained in a separate chapter. In this chapter, we also integrated our existing material from the modules ASE and SCAD in a section "Additional Literature and Links".

C. Topics from the Base Modules

In this chapter of our course, we present selected topics from the base modules (PROG and PM) and explain their relation to sustainability dimensions. We also link to the sustainability criteria defined in [1] where applicable.

We started by iterating through all topics covered by the base modules and investigated their relation to sustainable software engineering practices. Out of all topics, only a few are not easily relatable to sustainability such as file I/O, iterating over object collections, or debugging practices. For the remaining topics, we systematically examined their impact on each sustainability dimension and provided examples. Where applicable, we added literature references for further study.

The *programming* courses start with an introduction to object-oriented programming and cover advanced topics such as refactoring, testing, and concurrent programming. The *project* courses cover team communication, collaboration, and development tools.

1) Class Design and Refactoring: Students learn about design by responsibility, coupling, and cohesion, which improve code readability, comprehensibility, maintainability, and extensibility. These factors are crucial for the *technical* dimension of sustainability. Readability and comprehensibility also reduce the mental load on software engineers, positively influencing the *individual* dimension of sustainability. Regular refactoring is essential for maintaining clean code [20].

2) Clean Code: We teach that clean code is fundamental for enhancing the SE process and ensuring long-term project success. Clean code reduces developers' mental load and stress, positively impacting the *individual* dimension of sustainability. It also facilitates scalability and adaptability, crucial for the *technical* dimension of sustainability [21].

3) Software Testing: Software testing positively impacts the *economic* dimension of sustainability by ensuring reliability, efficiency, and maintainability. Rigorous testing identifies and fixes defects early, reducing costly post-deployment issues and maintenance costs. Comprehensive testing extends software

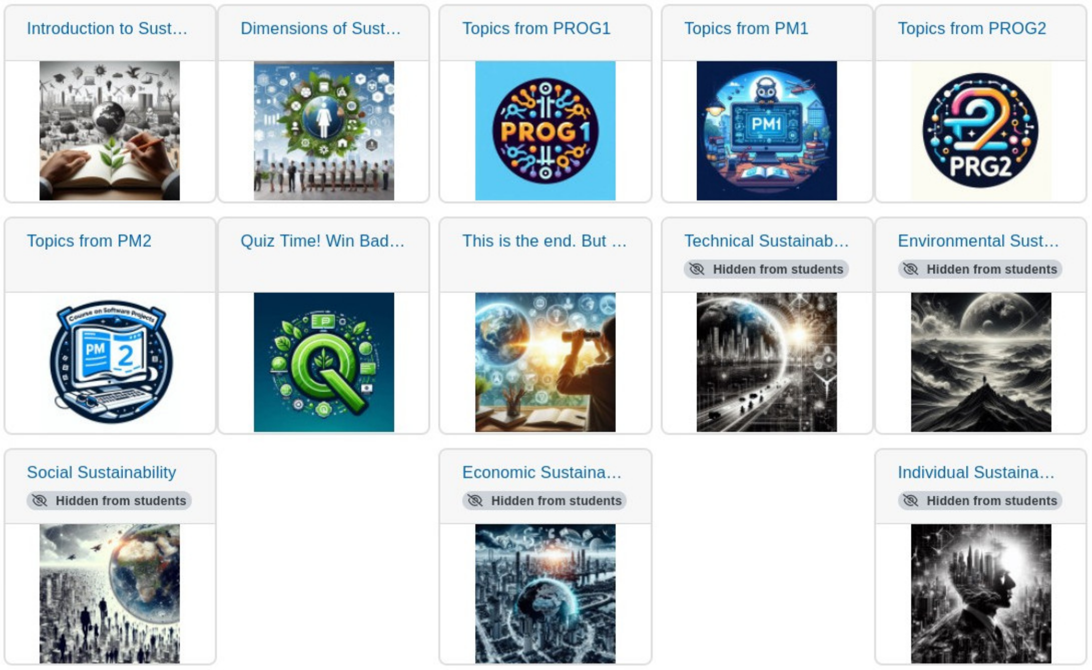


Fig. 2. Appearance of the course content as tiles in Moodle.

lifespan and reduces technical debt, leading to long-term cost savings and higher customer satisfaction. However, running tests requires hardware and electricity, impacting the *environmental* dimension of sustainability. Test managers must ensure no unnecessary tests are run [22].

4) Graphical User Interfaces: The *social* dimension of sustainability is impacted by GUIs. A sustainable GUI considers diverse user needs, promoting inclusiveness by supporting multiple languages, accommodating cultural differences, and ensuring compatibility with assistive technologies. As an example, having a multicultural engineering team helps with creating inclusive interfaces [23], [24].

5) Concurrent Programming: Proper thread synchronization optimizes resource utilization and reduces energy consumption, positively impacting the *environmental* dimension of sustainability. Efficient synchronization minimizes delays and conflicts, leading to better CPU usage and lower energy requirements. This also benefits the *economic* dimension by reducing energy costs, especially in cloud environments. Frameworks are available to help developers optimize energy consumption [25].

6) Team Reflection and Retrospective: In PM, students write reflections and hold retrospectives after each project, impacting the *social* dimension of sustainability by fostering continuous improvement and collaboration. This enhances communication, mutual respect, and team morale, reducing turnover rates and building a cohesive, motivated team [26].

7) Decision Protocol: Writing a decision protocol promotes transparency, accountability, and inclusiveness in decision-making, addressing the *social* dimension of sustainability. It ensures all team members have a voice, mitigating misunderstandings and fostering trust and respect [27].

8) Issue Tracking: In PM, students learn to use GitHub's issue tracking feature, which promotes transparency and collaboration, positively affecting the *social* dimension of sustainability. It also supports *economic* sustainability by improving project management and resource allocation, and *technical* sustainability by enhancing code quality and maintainability [28].

9) AI-supported Programming: AI tools like GitHub Copilot and ChatGPT are used for generating code, tests, and documentation. We teach students the advantages and risks of these tools and guide them in applying them carefully. AI-generated code can impact the *technical* dimension of sustainability positively or negatively, depending on its quality. AI tools can reduce developers' mental load, enhancing the *individual* dimension of sustainability, but over-reliance can lead to skill erosion. The environmental impact of AI, due to energy consumption, is also a concern. Students need sustainability awareness to assess the pros and cons of AI in programming [10], [29].

10) Project Management: GitHub's project management features enhance the *economic* sustainability of software projects by improving collaboration, productivity, and resource optimization. Features such as Issues, Project Boards, and Pull Requests centralize task tracking and workflow management, reducing coordination time and errors. Automation with GitHub Actions increases efficiency by handling repetitive tasks, allowing developers to focus on complex work, leading to faster project completion and lower labor costs. Integration with various development tools ensures effective resource use, reducing operational costs and waste.

Using collaboration tools such as GitHub also promotes transparency and traceability with version control and documentation, helping teams to quickly identify and address issues, improving *social* sustainability [27], [28]. Comprehensive documentation and knowledge sharing reduce the learning curve for new team members and ensure consistent project quality, contributing to cost-effective and efficient software projects.

11) Project Outline and Elevator Pitch: In the second semester, students must create a project idea, document it, and pitch it to investors. A well-developed project outline and elevator pitch ensure clear technical requirements and solutions, reducing technical debt and future problems, thus enhancing *technical* sustainability.

A precise project outline helps plan the budget realistically, avoiding unexpected costs, impacting *economic* sustainability. However, overly detailed planning can incur high initial costs and may lack flexibility to adapt to market changes.

A transparent project outline and elevator pitch promote stakeholder involvement and confidence, positively influencing *social* sustainability. Clear goals and structure increase team motivation and engagement, leading to better participation from all members [30].

D. Interactive Elements

The course provides two types of interactive elements: a forum and quizzes. The forum is titled “Discussion and Feedback” and contains two discussion threads from the start. The first thread, “Feedback”, which asks participants what they liked, what they do not agree with, and about what topics they want to hear more. The second thread, “Issue Tracking”, encourages participants to report problems with the structure or content of the course such as broken hyperlinks.

There are 6 quizzes that can be used by the course participants to self-assess their knowledge. The first quiz covers the general concept of sustainability (see Section IV-A). The other quizzes cover one dimension of sustainability each. Each quiz contains between 6 and 16 questions, most of which are multiple choice questions except for a few cloze-text tests.

V. Course Logistics and Evaluation

The *Sustainable Bytes* course was ready in Moodle at the start of autumn semester 2024. When the course opened, pointers from the course contents of the base modules had already been in place, linking the topics of PROG and PM with sustainability topics. We also added a short introduction to sustainability in the Moodle courses of the base modules.

When students follow one of the links to *Sustainable Bytes*, they are asked to enroll in the course once. This can be done easily by clicking an enrollment hyperlink. We decided to not automatically enroll all students of the base modules to obtain a better picture of the students’ interest in our course.

To further advertise *Sustainable Bytes*, the first author of this paper visited all parallel instances of the PROG course after one third of the semester and gave a short pitch of *Sustainable Bytes* to encourage students to participate.

A. Usage Statistics

The Moodle platform provides us with detailed usage statistics for the course: at the end of the first semester, 54 (out of 204) first-year students registered for the *Sustainable Bytes* course on Moodle, which corresponds to 26 % of all first-year students. Out of these 54 students, 16 students started the first quiz, 5 students completed it. All 6 quizzes were completed by 4 students (7 % of the students who registered). The forum was not used by the students.

B. Survey and Results

In the last week of the semester, we sent an invitation to a survey to all participants of the base modules to inquire about their motivation for our course. The students were not given time to respond to the survey during class hours. Instead, they needed to respond in their own time. Therefore, we kept the survey short and added an early exit for those students who did not register with the *Sustainable Bytes* course. In total, 26 students responded to our survey.

The first two questions concern the pointers that were added to the courses of the base modules. 23 students (88 %) noticed the links from the base modules to *Sustainable Bytes*, whereas 3 (12 %) did not. Out of those who *did* notice them, 9 students (39 %) followed them, whereas 14 (61 %) did not.

We asked those participants who noticed the links, but did not follow them, for their reasons. Students were allowed to enter multiple reasons. 12 students responded that they did not have time for voluntary content. Four students responded that the benefits of *Sustainable Bytes* were unclear to them. No students responded that they were not interested in sustainability in general. Two responses in the “Other” category were ”At the beginning I planned to do it, but in the end, I really didn’t have the time” and ”Due to my previous knowledge, I wasn’t very interested.” These results are shown in Table II.

TABLE II
Reasons for not enrolling in the course

Reason	*Number of responses*
The benefits are unclear	4
No time for voluntary content	12
Not interested in sustainability	0
Other	2

We asked those participants who *did* enroll in *Sustainable Bytes* about their main takeaways. To this end, we provided four statements from which they could select all they deemed applicable. We also provided a 5th answer (“Other”), which allowed students to enter additional information. 6 students answered that they learned “which aspects of software engineering have an impact on the sustainability of software”. 5 students said that they learned “how to analyze and improve a software engineering process with regard to sustainability.”

2 students each learned that "sustainability is not just about energy consumption, but encompasses 5 dimensions" and that "sustainability has three different levels of effect ("orders of effect")." These results are summarized in Table III.

TABLE III
MAIN TAKEAWAYS OF THE STUDENTS

Takeaway	*# of responses*
Sustainability is not just about energy consumption, but encompasses 5 dimensions	2
Sustainability has three different levels of effect ("orders of effect")	2
Which aspects of software engineering have an impact on the sustainability of software	6
How to analyze and improve a software engineering process with regard to sustainability	5
Other	0

We then asked a few general questions about the setup of the course. The results of these questions are shown in Table IV. From these results, we gain valuable insights:

1) Whereas most of our compulsory courses (such as PROG and PM) are taught in German, our course is in English. This was not an issue for the students. Also, most students appreciate that the course material is implemented as a dedicated course on Moodle. As a result, we will continue this course in English, which allows us to open the course for other universities in other countries or language regions of Switzerland.
2) The students need a stronger motivation for completing the course, e.g., in the form of points being awarded for the grades in the base modules. This is understandable because their curriculum is already tightly packed with activities required for obtaining a degree.
3) Only 22 % of the students felt motivated by their teachers to attend the course. This can be attributed to the facts that a) we did not give a thorough introduction to the course to our colleagues and b) the teachers putting their focus on the exam-relevant educational material.

We need to further investigate the results for the statement "*I read through the content, but didn't have the time/motivation for the quizzes.*" According to the responses, not having time or motivation for the quizzes was *not* an issue, which seems contradictory to our other observations.

We gave the participants of the survey the opportunity to enter feedback as free text at the end of the survey. Six participants responded. Some comments emphasize aspects that were covered by the questions before (integration of the course contents in the base modules, awards points/credits, complicated structure of the course). Two comments emphasize the importance of sustainability as a topic, and they praise that we offer a course on it. One comment criticizes that there is too much text in our course and suggests loosening up the course by using other media such as video. One comment criticizes the graphical quality of the virtual badges.

C. Threats to Validity

In conducting our survey, two potential threats to validity were identified.

a) External validity: One potential threat is sample representativeness. Although we invited all first-year students to the survey, we cannot exclude that most of the survey participants are those students who showed an interest in sustainability and our course. Since our survey was anonymous, we cannot correlate the data.

b) Statistical Validity: Due to the low participation in our survey, the sample size is not large enough to draw conclusions for the entire cohort of students.

VI. LESSONS LEARNED

In this section, we present our lessons learned based on the results of the usage statistics and the responses to our survey.

A. Use a bigger carrot

Students must be economical with their time. With a demanding schedule, we need to increase the motivation of our students further by embedding *Sustainable Bytes* more deeply into the curriculum.

The biggest motivator would be granting ECTS credits for the course. However, this would require replacing an existing mandatory module from the curriculum, which is currently not possible. There are two alternatives. 1) We could include the course in the students' transcripts of records, even if no ECTS credits are granted. 2) The students could earn points for this course that will contribute to their grade in a base module. In some base modules, this is already done today by awarding points for successfully completing practical assignments.

While developing this course, we discussed with several colleagues of other departments at our university. In these discussions, we have learned that one department requires its students to discuss sustainability aspects in their bachelor theses. We could follow this model and make it mandatory for our students to address one or more dimensions of sustainability in their bachelor theses. The *Sustainable Bytes* course can then be offered as introduction and inspiration.

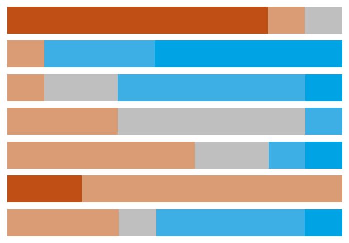
B. Teach the teachers

The low participation rate teaches us that for a voluntary course such as ours to receive wider attention, we need to advertise its value better. This is challenging because the concept of sustainability is not yet widely established in the software engineering community. We believe that an important step is to establish a notion of importance and urgency for sustainability among the educators at our university. If most of our educators emphasize the importance of the topic, students will be more inclined to attend the course.

By teaching the teachers, we can also achieve that more lecturers discuss sustainability aspects in their courses. This would allow us to spread the effort for reaching our first goal, *creating sustainability awareness in all our computer science students*, across many courses throughout the study program.

TABLE IV
OTHER FEEDBACK ON THE COURSE

I would have used the course more if it had been offered in German.	88%	0%
I would have used the course more if I had received points for the base modules.	11%	89%
I find the course well structured.	11%	67%
The contents of the course do not go far/deep enough for me.	33%	11%
The teachers motivated me to use the Moodle of the course.	56%	22%
I read through the content, but didn't have the time/motivation for the quizzes.	99%	0%
The content on sustainability should not be in a separate Moodle course, but directly in the Moodle courses of the base modules.	33%	55%



■ Very important ■ Important ■ Moderately important ■ Slightly important ■ Not important at all

Note: The number on the left of each bar chart is the sum of the percentages for the categories "Slightly important" and "Not important at all". The number on the right is the sum for "Very important" and "Important", analogously.

C. Make the course more fun

Our survey has shown that while the contents of the course are considered attractive by our students, its visual representation is not. The long text sections in the Moodle course should be broken up with exciting images and videos. Also, the virtual badges were considered as not looking valuable. Having them redesigned by a professional designer could increase the motivation of some students to win the badges.

To make the course more attractive, we envision to enhance it with practical assignments as suggest by Peters et al. [31]. This will show the practical relevance of the educational material to the students and give them more ways to interact with the course.

VII. RELATED WORK

This section discusses related work on the intersection of sustainability topics and computer science education, then widen towards education in general, and finally complement with industry-inspired views and findings.

A. Sustainability in Computer Science Education

The need to integrate sustainability into CS curricula was identified as early as 2010. Mann et al. [3] highlight barriers to this integration and propose the CE4s framework to help educators. Our approach addresses two barriers not covered by CE4s: *"(1) limited knowledge of sustainability [...] among computing educators, (2) the belief that computing may not significantly impact social and/or environmental sustainability."*

Penzenstadler et al. [32] propose a three-stage approach: a seminar, a lecture series, and integrating sustainability into software engineering lectures. Our approach covers the third phase.

Torre et al. [33] report that sustainability in curricula often focuses on energy efficiency, neglecting other dimensions. They recommend raising awareness among educators, including sustainability in all software engineering courses, and involving students in creating teaching materials. Our approach follows these recommendations by providing digital teaching materials and incorporating sustainability into first-year programming lectures.

Swacha et al. [34] use game-based learning to teach sustainability in computer science. They developed *Eco JSity*, a game where students solve programming puzzles to prevent pollution. Our work focuses on spreading awareness of the relationship between software engineering and sustainability, and adds gamification aspects in the form of quizzes and virtual badges. Using a game-based learning approach is an interesting option for the future, but would require a tight integration into our base modules.

Saraiva et al. [4] emphasize the importance of incorporating environmental aspects into the CS curriculum. They suggest introducing green thinking early and often. Our approach introduces sustainability concepts in mandatory first-year BSc modules and deepens these topics in elective third-year modules.

Eriksson et al. [35] explore using eco-anxiety to teach sustainability. They integrated a research project on emotions into an introductory course on sustainability and ICT, concluding that emotions are crucial in sustainability education. They provide eight suggestions for educators. While we have not yet addressed these suggestions in our course material, we are considering integrating them in order to increase the students' motivation for our course.

Oyedeji et al. [17] investigate students' perspectives on the Sustainability Awareness Framework (SusAF [10]) in two master's courses. Surveys conducted before and after introducing SusAF show that students' knowledge of sustainability expanded beyond the environmental dimension. SusAF is confirmed as an effective guide, which motivated us to base our course on its sustainability concepts.

Peters et al. [31] conduct a literature review on sustainability in computing education. They distinguish between incremental and transformative approaches. Our approach is incremental, adding content to existing courses. The report suggests focusing on topic-specific and cross-cutting competencies and gaining awareness, with practical experience as a potential enhancement.

Summary and gap analysis: Our *Sustainable Bytes* approach is aligned with the aim of many scholars to enhance CS education and in particular the SE track with sustainability topics. The concept is unique by combining the broad view on sustainability with the targeted access from SE lectures. As our results show, participation need to be increased and options exist in the form of gamification and emotions.

B. Sustainability in Other Fields of Education

We have also found relevant work on sustainability education in other disciplines than computer science.

Azapagic et al. [36] surveyed engineering students' sustainability knowledge in 2005, finding it limited to the environmental dimension and "not satisfactory." They suggest an integrated approach to teaching sustainable development and redesigned their university's engineering curriculum accordingly. While we have chosen an integrated approach, we have not yet managed to change our university's curriculum.

Morris et al. [37] emphasize the importance of complementing engineering courses with design courses to enhance students' self-reflection and critical analysis, particularly in sustainable design. Students are encouraged to assess their design choices considering environmental, social, and economic impacts. For CS students, design is a common activity in software engineering. Therefore, our course links the software and system design elements in the base modules to the relevant sections of our course.

Yarime et al. [38] advocate for collaboration between academic and private stakeholders across disciplines to establish sustainability science, including education. Our approach could benefit from partnerships with data center operators or software engineering companies. In the future, our course could be complemented with relevant case studies from such partnerships.

Leifler et al. [39] report on an interdisciplinary approach combining sustainability education with ethics and scientific writing in a master's course. A survey showed students adopted a broader view of sustainability and recognized its relevance to their careers. We are currently evaluating how we could integrate such an approach in our course. Scientific writing and communication skills are already taught at bachelor's level at our university, but adding sustainability would require replacing an existing topic.

O'Neill et al. [40] describe incorporating sustainability into existing courses by shifting assignments from a technology-centered to a community-centered view, addressing real-world problems. Their results indicate that students apply their sustainability knowledge in decision-making after attending these courses. This is difficult to achieve in our base modules because they take place in the first year of studies where tackling real-world problems would exceed the skills of most students.

Summary and gap analysis: While we lack a deeper integration into our university's curriculum, *Sustainable Bytes* is a portable concept and can be reimplemented in similar ways in many educational institutions. The link to industry needs and viewpoints remains a critical gap especially in applied sciences education that is measured by its utility in workforce education. Video clips and other interactive links to industry could fix this issue while increasing attractiveness of the material.

C. Industry Needs

Several studies highlight the software engineering industry's recognition of the need for sustainability awareness, knowledge, and practices, which are currently lacking. This underscores the importance of educating future software engineers on sustainability.

Heldal et al. [5] explore the industry's perspective on sustainability education for software engineers. Their findings indicate a need for technical personnel to acquire sustainability knowledge through in-house or external training and collaborations. They provide a comprehensive list of topics for CS curricula, including *core sustainability concepts* and *system thinking*, which our work addresses. Future course versions should cover the remaining topics.

Karita et al. conducted a survey among software engineering professionals in 97 companies [6], [7], revealing low sustainability knowledge, except for technical aspects like *performance* and *maintainability*. The survey participants acknowledged the benefits of sustainable practices for both their companies and society, highlighting the need for CS graduates to have solid sustainability knowledge.

Wahler et al. [1] report on a sustainability assessment of an industry partner's software engineering process. They found a lack of sustainability awareness and knowledge, leading the partner to integrate sustainability into their engineering principles and values, update use case templates, plan sustainability training for employees, and conduct regular sustainability assessments.

David proposes SusDevOps [41] as framework to embed sustainability into software delivery. It hooks into the monitoring phase of DevOps, aligns the monitoring results with sustainability goals and KPIs, and prioritises them before entering the replanning. However, SusDevOps has not been validated yet in industrial practice, and techniques to teach the framework to students are not yet documented.

Summary and gap analysis: Industrial recognition of sustainability is becoming mainstream. With initiatives such as the Green Software Foundation, digital souvereignty proposals like EuroStack and increasing transparency of workplace conditions and stress levels for software engineers, many directions exist but few can be taught to students in a condensed form. Selecting and mapping emerging industrial standards to *Sustainable Bytes* along with impactful examples becomes an important housekeeping duty to sustain our approach and to make students want to take the course from the beginning to the end.

VIII. Conclusions

We have presented *Sustainable Bytes*, a novel online course on sustainability in computer science that is linked with the contents of mandatory first-year modules of our bachelor's program in computer science. The course covers different dimensions of sustainability and provides means for the students to self-assess their learning progress. We have set ourselves three goals for our course. We missed Goal 1, *creating sustainability awareness in all our computer science students*. As shown, only 26 % of first-year students registered to our course. On the other hand, we have created sustainability awareness in up to 26 % of our first-year students. Also, we have learned important lessons (see Section VI), which should help us to reach more participation in future iterations.

We achieved Goal 2, *providing educational material on sustainable software and software engineering*. As shown, students can find the pointers to our course in the teaching material of the base modules, and they positively assess the structure and contents of our course. We have created a Moodle course that can be used and extended for many years.

We achieved Goal 3, *enabling our students to assess their own knowledge of the course material*. We have added quizzes for our students to self-assess their knowledge. From the usage statistics, we can see that students can find the quizzes and can solve them.

Despite missing the first goal, our course evaluation showed that the contents of the course are valuable. Also, we have learned important lessons how we might improve the course and motivate more students to learn about sustainability in the future.

A. Future Work

We are planning to continuously improve the course and its integration into the computer science curriculum of our university according to our lessons learned. As a first step, we want to spread sustainability awareness among our fellow lecturers as outlined in Section VI-B. This activity could be started with a half-day workshop moderated by the authors and supported by the materials in our course.

We seek for collaboration with lecturers at other universities to share teaching materials and experiences. Since our course material is in English, it can be easily reused for a wide range of students.

An online course such as ours might also be useful in an industrial context. As companies also face sustainability challenges [5], there is a growing need to educate practitioners as well. We are convinced that with some adaptations, the contents of our course are also relevant for experienced software engineering professionals. To this end, the course material should be extended with topics that are currently not covered in our base modules such as the implications of the choice of programming language (we only introduce Java), architectural frameworks and libraries (we use only basic frameworks such as JavaFX), or performance metrics such as latency (we teach them in other modules). Also, the references from our course to our curriculum (see Figure 2) need to be replaced with references to common software engineering practices such as requirements engineering or software testing.

The use of generative AI in software engineering has several strong effects on sustainability, which can be both positive (e.g., less repetitive tasks, quick "peer" feedback) and negative (e.g., increased energy usage, possible violations of intellectual property). This is currently a very active field of research. We plan to harvest the research results and augment our course with considerations on AI-supported software engineering.

Acknowledgment

We thank our fellow lecturers who supported our work with feedback and promoting the need for sustainability knowledge to their classes: Patrick Feisthammel, Christof Marti, Marion Mürner, Arjan Mooij, and Marcela Ruiz.

We thank the reviewers for their detailed feedback and clear intent to improve the paper.

Microsoft Copilot was used to enhance the language of some sections of this paper. The suggestions provided by Copilot were critically reviewed and incorporated by the authors.

References

[1] M. Wahler, N. Seyff, and M. S. Soriano Ramirez, "Exploring assessment criteria for sustainable software engineering processes," in *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Society*, ser. ICSE-SEIS'24. New York, NY, USA: Association for Computing Machinery, 2024, p. 107–117. [Online]. Available: https://doi.org/10.1145/3639475.3640109

[2] A. C. Moises de Souza, "Social sustainability approaches for a sustainable software product," *SIGSOFT Softw. Eng. Notes*, vol. 48, no. 1, p. 38–43, Jan. 2023. [Online]. Available: https://doi.org/10.1145/3573074.3573085

[3] S. Mann, L. Muller, J. Davis, C. Roda, and A. Young, "Computing and sustainability: evaluating resources for educators," *ACM SIGCSE Bulletin*, vol. 41, no. 4, pp. 144–155, 2010.

[4] J. de Sousa Saraiva, Z. Zong, and R. Pereira, "Bringing green software to computer science curriculum: Perspectives from researchers and educators," *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235474009

[5] R. Heldal, N.-T. Nguyen, A. Moreira, P. Lago, L. Duboc, S. Betz, V. C. Coroamă, B. Penzenstadler, J. Porras, R. Capilla, I. Brooks, S. Oyedeji, and C. C. Venters, "Sustainability competencies and skills in software engineering: An industry perspective," *J. Syst. Softw.*, vol. 211, p. 111978, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258427108

[6] L. Karita, B. C. Mourão, and I. do Carmo Machado, "Software industry awareness on green and sustainable software engineering: a state-of-the-practice survey," *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:202728886

[7] L. Karita, B. C. Mourão, L. A. Martins, L. R. Soares, and I. do Carmo Machado, "Software industry awareness on sustainable software engineering: a Brazilian perspective," *J. Softw. Eng. Res. Dev.*, vol. 9, pp. 2:1–2:15, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:234753303

[8] A. N. Kumar, R. K. Raj, S. G. Aly, M. D. Anderson, B. A. Becker, R. L. Blumenthal, E. Eaton, S. L. Epstein, M. Goldweber, P. Jalote, D. Lea, M. Oudshoorn, M. Pias, S. Reiser, C. Servin, R. Simha, T. Winters, and Q. Xiang, *Computer Science Curricula 2023*. New York, NY, USA: Association for Computing Machinery, 2024.

[9] M. Vitali, P. Schmiedmayer, and V. Bootz, "Enriching cloud-native applications with sustainability features," in *IEEE International Conference on Cloud Engineering, IC2E 2023, Boston, MA, USA, September 25-29, 2023*. IEEE, 2023, pp. 21–31. [Online]. Available: https://doi.org/10.1109/IC2E59103.2023.00011

[10] L. Duboc, B. Penzenstadler, J. Porras, S. Akinli Kocak, S. Betz, R. Chitchyan, O. Leifler, N. Seyff, and C. C. Venters, "Requirements engineering for sustainability: an awareness framework for designing software systems for a better tomorrow," *Requirements Engineering*, vol. 25, no. 4, pp. 469–492, 2020.

[11] D. o. E. United Nations and S. A. S. Development, "Transforming our world: the 2030 agenda for sustainable development," p. 16301, 2015. [Online]. Available: https://sdgs.un.org/2030agenda

[12] M. Freed, S. Bielinska, C. Buckley, A. Coptu, M. Yilmaz, R. Messnarz, and P. M. Clarke, "An investigation of green software engineering," in *Systems, Software and Services Process Improvement - 30th European Conference, EuroSPI 2023, Grenoble, France, August 30 - September 1, 2023, Proceedings, Part I*, ser. Communications in Computer and Information Science, M. Yilmaz, P. M. Clarke, A. Riel, and R. Messnarz, Eds., vol. 1890. Springer, 2023, pp. 124–137. [Online]. Available: https://doi.org/10.1007/978-3-031-42307-9_10

[13] S. Forti, J. Soldani, and A. Brogi, "Carbon-aware software services," in *Service-Oriented and Cloud Computing - 11th IFIP WG 6.12 European Conference, ESOCC 2025, Bolzano, Italy, February 20-21, 2025, Proceedings*, ser. Lecture Notes in Computer Science, C. Pahl, A. Janes, T. Cerný, V. Lenarduzzi, and M. Esposito, Eds., vol. 15547. Springer, 2025, pp. 65–80. [Online]. Available: https://doi.org/10.1007/978-3-031-84617-5_6

[14] H. E. Jadi, M. Vitali, and S. Andolina, "Balancing environmental sustainability and user experience: Preliminary insights from green search engines," in *Proceedings of the 2024 International Conference on Advanced Visual Interfaces, AVI 2024, Arenzano, Genoa, Italy, June 3-7, 2024*, C. Conati, G. Volpe, and I. Torre, Eds. ACM, 2024, pp. 89:1–89:3. [Online]. Available: https://doi.org/10.1145/3656650.3656742

[15] K. Geissdoerfer and M. Zimmerling, "Riotee: An open-source hardware and software platform for the battery-free Internet of Things," in *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems, SenSys 2024, Hangzhou, China, November 4-7, 2024*, J. Liu, Y. Shu, J. Chen, Y. He, and R. Tan, Eds. ACM, 2024, pp. 198–210. [Online]. Available: https://doi.org/10.1145/3666025.3699332

[16] W. C. on Environment and Development, *Our Common Future*, ser. Oxford paperbacks. Oxford University Press, 1987.

[17] S. Oyedeji, M. O. Adisa, L. Abdullai, and J. Porras, "Application of sustainability awareness framework in software engineering courses: Perspectives from ICT students." in *ICT4S (Doctoral Symposium, Demos, Posters)*, 2023, pp. 122–133.

[18] J. Biggs and C. Tang, *Teaching For Quality Learning At University*, ser. SRHE and Open University Press Imprint. McGraw-Hill Education, 2011. [Online]. Available: https://books.google.ch/books?id=XhjRBrDAESkC

[19] L. M. Hilty, P. Arnfalk, L. Erdmann, J. Goodman, M. Lehmann, and P. A. Wäger, "The relevance of information and communication technologies for environmental sustainability – a prospective simulation study," *Environmental Modelling & Software*, vol. 21, no. 11, pp. 1618–1629, 2006, environmental Informatics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364815206001204

[20] S. Freire, N. Rios, B. Perez, C. Castellanos, D. Correal, R. Ramač, V. Mandić, N. Taušan, A. Pacheco, G. Lopez *et al.*, "Pitfalls and solutions for technical debt management in agile software projects," *IEEE Software*, vol. 38, no. 6, pp. 42–49, 2021.

[21] H. G. Koller, "Effects of clean code on understandability: An experiment and analysis," Master's thesis, University of Oslo, 2016.

[22] S. Rapps and E. J. Weyuker, "Selecting software test data using data flow information," *IEEE transactions on software engineering*, vol. 4, pp. 367–375, 1985.

[23] P. S. Fong, "Knowledge creation in multidisciplinary project teams: an empirical study of the processes and their dynamic interrelationships," *International journal of project management*, vol. 21, no. 7, pp. 479–486, 2003.

[24] S. Naumann, E. Kern, M. Dick, and T. Johann, "Sustainable software engineering: Process and quality models, life cycle, and social aspects," in *ICT Innovations for Sustainability*. Springer, 2015, pp. 191–205.

[25] I. Manotas, L. Pollock, and J. Clause, "Seeds: A software engineer's energy-optimization decision support framework," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 503–514.

[26] S. Ryan and R. V. O'Connor, "Acquiring and sharing tacit knowledge in software development teams: An empirical study," *Information and Software Technology*, vol. 55, no. 9, pp. 1614–1624, 2013.

[27] L. Thøger Christensen, "Corporate communication: The challenge of transparency," *Corporate communications: an international journal*, vol. 7, no. 3, pp. 162–168, 2002.

[28] D. Bertram, A. Voida, S. Greenberg, and R. Walker, "Communication, collaboration, and bugs: The social nature of issue tracking in software engineering," in *Proc. ACM Conf. Comput. Support. Coop. Work*, 2010.

[29] C. Venters, L. Lau, M. Griffiths, V. Holmes, R. Ward, C. Jay, C. Dibsdale, and J. Xu, "The blind men and the elephant: Towards an empirical evaluation framework for software sustainability," *Journal of Open Research Software*, vol. 2, no. 1, pp. 1–6, 2014.

[30] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters, "Sustainability design and software: The Karlskrona manifesto," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2. IEEE, 2015, pp. 467–476.

[31] A.-K. Peters, R. Capilla, V. C. Coroamă, R. Heldal, P. Lago, O. Leifler, A. Moreira, J. P. Fernandes, B. Penzenstadler, J. Porras *et al.*, "Sustainability in computing education: A systematic literature review," *ACM Transactions on Computing Education*, vol. 24, no. 1, pp. 1–53, 2024.

[32] B. Penzenstadler and A. Fleischmann, "Teach sustainability in software engineering?" *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T)*, pp. 454–458, 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:1063570

[33] D. Torre, G. Procaccianti, D. Fucci, S. Lutovac, and G. Scanniello, "On the presence of green and sustainable software engineering in higher education curricula," *2017 IEEE/ACM 1st International Workshop on Software Engineering Curricula for Millennials (SECM)*, pp. 54–60, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:2696454

[34] J. Swacha, R. Maskeliūnas, R. Damavsevivcius, A. Kulikajevas, T. Blavzauskas, K. Muszyńska, A. Miluniec, and M. K. Kowalska, "Introducing sustainable development topics into computer science education: Design and evaluation of the eco jsity game," *Sustainability*, vol. 13, p. 4244, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:234884697

[35] E. Eriksson, A.-K. Peters, D. Pargman, B. Hedin, M. Laurell-Thorslund, and S. Sjöö, "Addressing students' eco-anxiety when teaching sustainability in higher education," in *2022 International Conference on ICT for Sustainability (ICT4S)*. IEEE, 2022, pp. 88–98.

[36] A. Azapagic, S. Perdan, and D. Shallcross, "How much do engineering students know about sustainable development? the findings of an international survey and possible implications for the engineering curriculum," *European journal of engineering education*, vol. 30, no. 1, pp. 1–19, 2005.

[37] R. Morris, P. Childs, and T. Hamilton, "Sustainability by design: a reflection on the suitability of pedagogic practice in design and engineering courses in the teaching of sustainable design," *European Journal of Engineering Education*, vol. 32, no. 2, pp. 135–142, 2007.

[38] M. Yarime, G. Trencher, T. Mino, R. W. Scholz, L. Olsson, B. Ness, N. Frantzeskaki, and J. Rotmans, "Establishing sustainability science in higher education institutions: towards an integration of academic development, institutionalization, and stakeholder collaborations," *Sustainability Science*, vol. 7, pp. 101–113, 2012.

[39] O. Leifler, L. Lindblom, M. Svensson, M. Gramfält, and A. Jönsson, "Teaching sustainability, ethics and scientific writing: An integrated approach," *2020 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:227126016

[40] I. O'Neill and M. M. Gui, "Changing focus: making sustainability a major theme in existing university modules," *Discover Sustainability*, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:272332096

[41] I. David, "Susdevops: Promoting sustainability to a first principle in software delivery," 2025. [Online]. Available: https://arxiv.org/abs/2312.14843