OULUN AMMATTIKORKEAKOULU

Mira Vorne

USER STORY

Roadmap to Learning Programming

USER STORY

Roadmap to Learning Programming

Mira Vorne
Master's thesis
Spring 2025
Modern Software and Computing Solutions
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Modern Software and Computing Solutions

Author: Mira Vorne
Title of thesis: User Story – Roadmap to Learning Programming
Supervisor: Jukka Jauhiainen
Term and year when the thesis was submitted: Spring 2025
Number of pages: 59 + 2 appendices

This research investigates the potential benefits of user story-based teaching for programming students at LAB University of Applied Sciences. The primary objective is to enhance teaching in programming to better serve engineering students. The thesis employs a research-based development model, incorporating preplanned construction tested with first-year international engineering students enrolled in an introduction to programming course. The construction is created to be used also in the advanced engineering courses. Data evaluation is conducted using qualitative methods.

The methodology follows a research-based development approach, which integrates conceptual models from previous studies, tests new constructions and hypotheses in practice, and critically evaluates conclusions to refine the constructed models. Additionally, the constructive research process is utilized, focusing on solving real-life challenges through the creation and implementation of new constructions in relevant environments. The study includes questionnaires for ICT professionals and students who participate in the testing of the construction.

Key findings indicate that students benefit from user story-based learning. This approach not only improves their professional identity, motivation, and programming skills but also garners interest from companies and organizations willing to collaborate with the students. The research leads to the development of a new construction, with ongoing improvements. The next step involves transitioning students from project-based learning to user story-based learning with customers, ICT companies, and various organizations, further enhancing their educational experience. The aim is to implement this construction across all programming courses, at every level, to benefit all students.

The students are also taught the theoretical foundations necessary to tackle complex and multidisciplinary engineering challenges of the future. The collaborative projects are designed to complement the theory. This comprehensive approach ensures that students not only gain practical skills but also develop a deep understanding of the underlying principles, preparing them to address diverse and evolving engineering problems effectively.

CONTENTS

1 INTRODUCTION 5
2 BACKGROUND 6
2.1 Teaching and learning programming 6
2.2 User stories in agile software development 17
3 RESEARCH METHOD 20
3.1 Research-based development 20
3.2 Constructive research process 22
3.3 Construction implementation 23
3.4 Data analysis 24
4 CONSTRUCTION 26
4.1 Weekly Tasks 27
4.2 Course-Long Project 27
4.3 Project Features 29
4.4 Documentation and Reporting 31
5 IMPLEMENTATION 32
5.1 Course Planning and Preparation 32
5.2 Setting Up CodeGrade 32
5.3 Project Design and Planning 32
5.4 Course Kick-off 33
5.5 Weekly Execution 33
5.6 Project Development 33
5.7 Mid-Course Review 34
5.8 Final Project Presentation 34
5.9 Final Assessment 34
5.10 Course Wrap-Up 34
6 EVALUATION 35
7 NEXT STEPS AND FUTURE STRATEGIES 51
8 CONCLUSION 56
REFERENCES 58
APPENDICES 60

1 INTRODUCTION

The rapid advancement of technology has significantly increased the demand for skilled programmers in various engineering fields. As a result, educational institutions are continually seeking new and effective methods to teach programming to the engineers of the future challenges. This thesis aims to explore and evaluate the potential benefits of user story-based teaching for programming courses at LAB University of Applied Sciences, with the goal of enhancing teaching and improving student outcomes.

User story-based teaching is a new approach that integrates practical exercises with theoretical instruction, providing students with a comprehensive learning experience. This method is relevant for all students, starting from first-year engineering students, who often face challenges in grasping abstract programming concepts. By focusing on real-life scenarios and practical applications, user story-based teaching aims to make programming more accessible and engaging for students.

The thesis employs a research-based development model, which supports the development process through the integration of conceptual models from previous studies. New constructions and hypotheses are tested in practice, allowing for continuous improvement and refinement. The constructive research process is also utilized, focusing on solving real-life challenges through the creation and implementation of new constructions in relevant environments. Data is collected through qualitative methods, including questionnaires for ICT professionals and the first-year students who participated in the new construction.

Implementing user story-based teaching is not without its challenges for educators. Organizing and overseeing various projects simultaneously requires significant effort and coordination. However, this approach makes learning more flexible and motivating for students, as it allows them to engage their studies in a more meaningful and practical way. The increased motivation and flexibility ultimately lead to better learning outcomes and a more dynamic educational experience.

2 BACKGROUND

2.1 Teaching and learning programming

Teaching and learning programming have piqued researchers' interest over the past few decades. The challenges associated with learning programming, especially for beginners, have led to high dropout rates. Consequently, educators and researchers have been exploring innovative approaches to teaching this subject. Topics of discussion include programming fundamentals, course content, student motivation, teaching methods, and learning environments. Numerous effective strategies have emerged to enhance learning and boost self-confidence. Additionally, recent studies have focused on specific aspects of teaching programming. In this overview, some of these prior studies, which have contributed to the development of the teaching method, will be examined.

Päivi Kinnunen's research sheds light on the challenges faced by students studying introductory programming courses at Helsinki University of Technology. The course was compulsory for students not majoring in computer science, for example Master of Science students in electrical engineering. Kinnunen examined the perspectives of students, teachers, and the university itself. Notably, there was a relatively high drop-out rate, which was consistent with findings from other institutes. Kinnunen found that the reasons for dropping out varied significantly and were unique to each student. Additionally, Kinnunen explored factors that students might find helpful and investigated the institution's procedures for collecting feedback. (Kinnunen 2009, 6.)

The background study conducted by Kinnunen highlights various challenges that students may encounter when studying programming; an endeavour that is by no means easy. For instance, students might struggle to grasp the purpose of studying programming or how they can apply it practically. Additionally, understanding algorithms and mastering the semantics and syntax of programming languages can prove daunting. Computational thinking, including concepts like recursion, working with arrays or loops, inheritance, and methods, can pose challenges for beginners. Students' attitudes toward learning and their chosen study approaches also play a significant role in their success. Design, testing, and debugging are areas that can overwhelm new students. Moreover, strong problem-solving skills are essential for passing programming

courses. Even if a student excels in theoretical programming knowledge, practical software development can still present difficulties. (Kinnunen 2009, 20-23, 28.)

Kinnunen realised also, the reasons to drop out are not always due to the content or teaching, but the reasons can be also family, work or health related. Sometimes time required to learn programming came to the students as a surprise or the lack of assistance available. Still, many of the students merely find the exercises too difficult. Learning skills and willingness to study hard played an essential role in passing the programming course, not to mention the self-confidence. (Kinnunen 2009, 25-27.) The social relations and learning environment seemed also to be an issue, when studying programming. Students were affected by the feeling of isolation from each other and the computer science community. (Kinnunen 2009, 29.)

In Kinnunen's literature review, teachers found the heterogeneity of students challenging, along with the industry's demands to meet future needs. They also reported that a lack of passion and discipline among students affected their success. While teachers understood the importance of students being willing to invest time and effort in self-practice, they struggled to cultivate this culture in the classroom. (Kinnunen 2009, 33-34.)

In her research, Kinnunen identified several main reasons for students dropping out of the programming course. Firstly, students found the course difficult and time-consuming. They often lacked sufficient study or time management skills. Additionally, various personal factors contributed to the students' burden, leading to their decision to drop out. Students also struggled with planning their studies and understanding the mental and time commitment required to pass the programming course. (Kinnunen 2009, 140.)

Most students reported more than one reason for dropping out of the introductory programming course. Some of these reasons were critical obstacles, while others cumulatively contributed to their decision to drop out. The students who dropped out often felt they did not receive enough help from the teacher and had difficulties understanding the course topics. Additionally, they did not perceive any consequences for dropping out, as the course was not a prerequisite for other courses (Kinnunen 2009, 190). The reasons were similar to those identified in Kinnunen's background study, such as low motivation, time management issues, and difficulties in designing code. (Kinnunen 2009, 191.)

Those who focused on motivating students achieved better results. Teachers could also support students by teaching them learning and time management skills, as well as how to search for information. It is crucial to plan the content with clear and reasonable segments and to teach using examples of good practices. Designing exercises with multiple smaller deadlines can help address time management issues. Additionally, teachers can assist students in setting personal goals. Providing feedback during the course is beneficial, and teachers should also actively seek and utilize student feedback. (Kinnunen 2009, 142, 150, 165.)

Teachers identified several challenges that students face, including logical thinking, motivation, the level of abstraction in their thinking, and the ability to apply knowledge. Both students and teachers agreed that understanding the theory and concepts of the course was difficult for students. Additionally, applying the information was considered a challenging task by both groups. Furthermore, both groups acknowledged that students have issues with time management. (Kinnunen 2009, 193.)

Kinnunen's research on educational institutions' views on teaching and learning programming required her to familiarize herself with formal documents, such as the University Act and institution-provided guidelines. She also interviewed the rector and the dean. The goal of shortening completion times and improving student recruitment was evident. There was some tension between teachers' academic freedom for high-quality research and teaching versus the institution's goals. The documents stated the goals at an abstract level and were not considered very helpful for course planning. Soft skills were not mentioned in the documents, leading to their exclusion from course content planning. Feedback collection was not systematic or well-organized at the institutional level, and the feedback channels for faculty were also considered inadequate. (Kinnunen 2009, 168, 185-186.)

Vihavainen, Paksula, and Luukkainen discuss a teaching method in their article "Extreme Apprenticeship Method in Teaching Programming for Beginners." Their model was applied to a Introductory programming course and significantly decreased the drop-out rate, demonstrating its effectiveness in teaching programming basics. Vihavainen, Paksula, and Luukkainen agree with Kinnunen that learning programming is not easy and acknowledge that researchers have been interested in this topic for a while. They recognize that there is no consensus on the best methods to teach the course. In their paper, they aim to introduce a new teaching approach as opposed to

traditional lectures, assignments, and demo sessions with model solutions. (Vihavainen, Paksula & Luukkainen, 93.)

Vihavainen, Paksula, and Luukkainen assert that students generally receive very little support in programming exercises, causing some to find them extremely difficult. This often leads to feelings of inadequacy and dropping the course. They argue that this minimally guided method is not an optimal way to learn a cognitively challenging subject. (Vihavainen, Paksula & Luukkainen, 93.)

Vihavainen, Paksula & Luukkainen point out, the traditional way of teaching programming clearly does not work. Also, motivation and comfort level are important factors in succeeding in studies. Thus, they created an extreme apprenticeship method. The method includes pair programming, the student writes code while constantly reviewed by his partner. Also, there is lot of coding involved, because that is the only way of mastering programming. The students also get constant feedback from their teacher and the teacher gets feedback in the form of student progress. Some students may need more time to learn, and students practice until they do learn. (Vihavainen, Paksula & Luukkainen, 93-95.)

The best practices of the extreme apprenticeship method include minimal lecturing and extensive coding. Lessons feature related and mandatory exercises that start as soon as possible, with assistance provided by the teacher. Goals are set low with many intermediate steps, and there are enough exercises to develop a good coding routine. Exercises are clearly structured and well-instructed, and students are encouraged to search for information on their own. The exercises are simple, but when done sequentially, they combine into a larger program. Initially, students are given simple warm-up exercises to build their confidence. (Vihavainen, Paksula & Luukkainen, 95.)

Jyrkkä and Niemi authored an article discussing effective approaches to teaching programming. They propose that students learn programming most effectively through project-based learning. Despite programming being a university subject for 50 years, there remains no universally established method for teaching it. At Oulu University of Applied Sciences, a theory and project-based approach is adopted. This method not only teaches programming but also includes project management and integrates mathematical knowledge through weekly sprint reviews, during which students present their projects. (Jyrkkä & Niemi 2020.)

Student skills and motivation vary significantly. With reduced contact teaching, students are now required to engage in self-study. The goal is to increase student motivation by having them work on their own projects. A professional guide should provide examples and demonstrate how to plan, implement, test, and debug programming. During the course, students create and implement their own projects in groups, documenting them thoroughly. This teaching method has also been shown to predict success in subsequent studies. (Jyrkkä & Niemi 2020.)

Körkkö conducted research on the challenges and potential solutions related to learning programming in his bachelor's thesis. Approaching the topic from a psychological perspective, he addresses issues such as problem-solving skills and motivation. His conclusion emphasizes that simplified solutions cannot adequately address complex programming problems. Abstract thinking and strong cognitive skills are essential for programming, and Körkkö asserts that programming can only be truly learned through practice. (Körkkö 2016, 23-24.)

Körkkö highlights a common scenario: students attending a programming course solely because it is compulsory. When students passively listen to lectures without active engagement, their motivation and self-perceived capability tend to be low. To address this, Körkkö explores the extreme apprenticeship method, suggesting that coding in pairs could enhance a sense of belonging and increase inner motivation. (Körkkö 2016, 23-24.)

Furthermore, Körkkö emphasizes the importance of high problem-solving and self-learning skills for students to succeed in future technology-related work. (Körkkö 2016, 23-24.)

Sippola conducted research on teaching methods and student motivation in programming courses. Her focus was on university programming and information technology students for her bachelor's thesis. Specifically, she explored teaching methods, pedagogy, and ways to motivate students. Sippola investigated visualization tools, themed courses, contextualization, and achievement badges. She assumed that these students already possessed an inherent interest or motivation for programming. (Sippola 2020, 1.)

Sippola observes that there is a belief that programming courses have a higher dropout rate compared to other courses. However, she notes that this claim has not been extensively researched, and there is no empirical evidence to support it. Instead, Sippola found that there is no one-size-fits-all method to motivate students. Each student has varying levels of motivation

and prior knowledge of the subject. Additionally, some students possess stronger academic skills than others. Success in programming courses depends on both the teacher and the student. Active participation from students is crucial, and they should be encouraged to delve deeper into the subject beyond mere surface-level understanding. The foundational self-confidence for advanced courses is established during the beginner-level programming course. (Sippola 2020, 2-4.)

Sippola, in her thesis, asserts that visualizing tools can aid beginner students in grasping data structure concepts in programming. These tools encompass debugger utilities, code structure modelling, and illustrating method functionality. The research indicates that visualization enhances student engagement, makes lessons more enjoyable, and improves recognition of misunderstandings. Students acquire knowledge by visually representing algorithm functionality through drawing and modelling. (Sippola 2020, 11-12.)

Sippola emphasizes the importance of teaching general and abstract problem-solving approaches. Students sometimes struggle to contextualize the subject matter, which can impact their motivation to learn. When students fail to see the relevance of a specific concept, their interest wanes. Striking the right balance is crucial. Too much contextualization can be problematic, especially in large and diverse student groups. Additionally, overly contextualized themes may not easily transfer beyond the immediate subject. Nevertheless, students appear to benefit from some level of context or a course theme. The examples of themes include game programming, robotics, or real-life data applications. (Sippola 2020, 12-13.)

According to Sippola, achievement batch is usually a graphic icon, the student gets from achieving a given goal. Batches are a form of gamifying the learning. If the batches are a public symbol of knowledge, such as one you can add to your profile, it may increase motivation in some cases. Batches have been criticized for relying on extrinsic motivation to succeed in the exercises. (Sippola 2020, 14-15.)

Sippola concludes that motivation can truly be increased by activating the students during the course. A good teacher challenges the students to actively practise with the exercises. (Sippola 2020, 16.)

In a master's thesis, Kulangara presents a platform for teaching introductory programming course. The platform facilitates LLM (Large Language Model) for feedback. According to Kulangara, the LLMs will change the programming education by helping students understand concepts and overall improving the experience of learning. (Kulangara 2023, 3, 8.)

Due to the large enrolments an introductory programming classes can lack engagement. This can cause student retention. To enhance student motivation and sense of efficacy, Kulangara suggests using learning environments. He believes both formative and summative feedback are crucial features of a successful programming course. Individually given feedback for every student during the course can be impossible for the teachers. To tackle this problem, automated assessment approaches might be an answer. With the help of LLMs, there will be new possibilities for providing feedback for the students. Also, LLMs could be used for generating new exercises and code explanations. Although, when using LLMs for generating content, the quality needs to be verified by the teacher. (Kulangara 2023, 13, 15, 17, 19.)

Kulangara created with his team a LLM based learning platform with exercises and automated feedback (Kulangara 2023, 25). The platform gives instant improvement suggestions to code submissions. It corrects the errors and guides the students for correct answers. (Kulangara 2023, 63.)

The created application can be considered as a demo version. The research model used only one LLM, GPT-4 and the target group was only six participants. Also, the students attending the study were master-level students while the target group of the application is novice students. The application was also somewhat unreliable in its answers, and it was not injection proofed. Sometimes, the application was said to give a bit too much information, which prohibited the student to actively use the problem-solving skills of his/her own. With the given specs participants found the application useful for learning. (Kulangara 2023, 64-65.)

Aleksi Lukkarinen has made a doctoral thesis about teaching event-driven programming. He recognises, that computer applications and the processes they execute do not usually operate in isolation. Instead, they continuously wait for input from users or other stakeholders to guide their actions. This means their execution is not identical every time. It depends on real-time input. This approach, known as event-driven programming (EDP), involves handling events; specific occurrences that happen at a particular time. These events trigger actions, and graphical user

interfaces (GUIs) in devices like watches, phones, and web browsers often follow this event-driven model. (Lukkarinen 2022, 1.)

According to Lukkarinen, the term “event-driven programming” (EDP) refers to creating computer software that responds to specific occurrences known as events. These technologies, including most web servers and graphical user interfaces across various devices and browser-based services, have become commonplace in modern society. As industry-developed computer applications continue to introduce new processes and automate existing ones, it is essential for software developers to have a solid understanding of EDP. Consequently, this topic should hold corresponding importance in software developers’ education. Contextualized teaching can make studying more meaningful and interesting for specific student groups. By improving engagement, such an approach may reduce the number of students dropping out of courses. (Lukkarinen 2022, 81.)

In Lukkarinen’s opinion, given the prevalence of event-driven systems, it’s crucial for software developers to grasp the complexity of event processing. As the use of computerized processes and everyday applications grows students studying software development need to acquire proficiency in related technologies. (Lukkarinen 2022, 2, 4.) Lukkarinen adds, that becoming proficient in programming, including mastering programming languages and tools, requires understanding a wide array of new concepts, terminology, and skills. One possible explanation for students’ struggles in programming courses lies in their incomplete comprehension of the content they are expected to learn. (Lukkarinen 2022, 19.)

A critical question in teaching event-driven programming is where it should fit within the computing curriculum. While some dedicated courses focus solely on EDP, other approaches integrate it into introductory programming courses. Advantages of having a dedicated EDP course after the introductory programming course include the ability to delve deeply into various EDP applications without competing against other programming fundamentals and easier comprehension of EDP after students have learned foundational concepts in earlier courses. (Lukkarinen 2022, 72.)

However, there are also potential drawbacks, such as delaying EDP knowledge until later courses might hinder its application in areas like graphical user interfaces, server programming, and low-level hardware programming, students’ motivation could start vanishing while waiting to

learn something they find interesting and if the course is elective, not all students may choose to enrol, limiting education in this important subject area. (Lukkarinen 2022, 72.)

On the other hand, an alternative hypothesis suggests that introducing fundamental EDP concepts earlier could lead to better long-term learning outcomes. This assumes correct initial understanding, followed by gradual reinforcement and expansion in subsequent courses. Conversely, an inadequate or incorrect initial grasp of EDP concepts might hinder learning in related areas. (Lukkarinen 2022, 72.)

Teemu Lehtinen has conducted a doctoral thesis about automated assessment in programming courses. In his thesis, Lehtinen suggests, after completing their initial programming course, many students often have misconceptions, struggle with program tracing, and require support with both code and programming tools. In simpler terms, a significant number of students possess limited programming knowledge by the end of their first semester according to Lehtinen. (Lehtinen 2024, 22.)

Programming involves more than just writing computer programs. Code-reading skills are essential for problem-solving and code writing. Program comprehension (PRC) refers to understanding program code, its design, execution and purpose. PRC is crucial for systematic program writing and extends beyond code editing. In software development, discussing and reviewing code with colleagues is also essential. Emerging AI tools generate code, but human assessment and discussion remain vital. Lehtinen states, that comprehending code is not guaranteed by merely writing a program. Pair programming and asking questions about code help improve understanding. (Lehtinen 2024, 15-16.)

In many introductory programming courses, students encounter numerous small program writing exercises. Writing code not only helps learners establish a routine but is also considered a realistic activity by many students. Some students systematically design and test programs using their programming knowledge. However, others begin with code they find online, copy from examples, generate using AI tools, or receive from peers. These students may fix errors based on automated feedback or advice without fully understanding their code. Those who truly comprehend how their program works learn more than those who merely aim to pass functional tests. To enhance student understanding, educators can create learning tasks that focus on

comprehending the program code produced. Asking students questions about their programs can help assess their current programming knowledge. (Lehtinen 2024, 37.)

As an example, a set of questions that could be used to find different levels of comprehension of a student could be:

- “Which are the two function parameter names in your program?”
- “How would you describe the difference between returning a value and printing a value?”
- “Describe the responsibilities of your inner loop in few words.”
- “Is function x recursive?”
- “How many parameters does function x have?” (Lehtinen 2024, 88.)

The sample questions cover a spectrum of topics, from identifying elements and their relationships in program syntax to explaining program design and execution states. Moreover, these questions are designed in a way that both their formulation and correct answers can be automated. (Lehtinen 2024, 38.)

These Question-Level Checks (QLCs) are queries related to program code written by a student. They specifically address concrete constructs or patterns within the student’s program. They are posed directly to the student by a computer and are automatically generated based on an analysis of the student’s code. The content of a QLC can vary, limited only by the designer’s creativity and technological constraints. These questions draw on knowledge that can be deduced from the program code itself, as well as other supporting materials like model solutions. Importantly, the design of QLCs aims to prompt students to answer effectively. (Lehtinen 2024, 39)

Improving students’ problem-solving process involves making them aware of their progress or ensuring completion of crucial steps. Lehtinen proposes that metacognitive support should extend to reflection. Novices’ understanding of programming is often fragile. When a novice creates a program, their comprehension of it may be elusive and challenging to articulate. Question-Level Checks (QLCs) have the potential to assist students in monitoring their comprehension. These prompts may trigger self-explanation, allowing students to silently construct deeper meanings and connections for newly acquired knowledge. (Lehtinen 2024, 41.)

Lehtinen anticipated that providing immediate automated feedback on answers to QLCs could facilitate self-regulated learning. Feedback allows learners to confirm their understanding before

moving on to new content. Therefore, when designing QLCs, it's essential to consider the ability to generate automated feedback. It's easier to support automated feedback for questions with exact answers than for those expecting open-ended responses. This makes multiple-choice questions (MCQs) or questions expecting specific numerical values appealing for developers. (Lehtinen 2024, 43.)

Lehtinen states in his thesis, Question-Level Checks (QLCs) were defined and proposed as an approach for developing automated assessments related to program comprehension. The concept involves posing personalized questions to students, targeting the structure and logic of the program code they created themselves. Additionally, the dissertation explored design decisions for an automated QLC generator and implemented such systems for three different programming languages. (Lehtinen 2024, 101.)

The practical evaluation of QLCs took place in two university courses across different countries, as well as in an open online course for lifelong learners. The results revealed that up to 20% of novice students might lack the ability to systematically understand how their own program code works, even though they had passed functional tests. Moreover, over half of novice students struggled to trace their functionally correct code. These findings align with previous research indicating that novice students require several weeks to develop the cognitive capacity to reason about program code. (Lehtinen 2024, 101.)

Performance on QLCs showed a correlation with students' programming ability and overall course success. Students weaker in QLCs tended to resort to tinkering and spent significant time without making progress. Consequently, QLCs could serve as early indicators for students who need additional support in learning programming. (Lehtinen 2024, 101.)

While state-of-the-art AI tools perform better than an average novice in answering QLCs, they still make errors for unexpected questions and programs. Despite correctly generating programs from exercise prompts, the AI's explanations sometimes contradict themselves or exhibit similarities to human errors. Additionally, students could be asked to evaluate AI-generated QLC answers, identifying errors based on the AI's explanations. Overall, automated questions about program structure and behaviour offer scalable, early practice in discussing program code—a critical skill in the programming industry. (Lehtinen 2024, 101-102.)

2.2 User stories in agile software development

Writing user stories has a well-proven process. User story is a collection of index cards or sticky notes with a sentence or a title on each. The cards are prioritized. After ordering the cards, the topics are discussed. The topics are broken down to smaller parts to figure out what to build in the next feature. (Patton & Economy 2014, chapter 1.)

Patton and Economy say this is a step from shared documents toward shared understanding. You can do this by letting the customer tell their story about their plans and goals and writing the cards as notes for the story. The cards are laid down so they can all be seen at the same time, be referred to and moved around. (Patton & Economy 2014, chapter 1.) The customers and end users of the product should take an active role in writing the user stories. (Cohn Mike 2004, part 1, chapter 1.) First the idea of the product is framed. The user story map answers questions like why the product is build or who will be the end user. The cards are organized in a sequence from left to right to form the discussed story. This type of organizing helps to notice any holes in the story. On the other hand, a card above another one indicates it is more important. Also, details are placed under the bigger entities. (Patton & Economy 2014, chapter 1.)

The customer team is responsible for writing user stories because the stories need to be written in business language rather than in technical language. This way the customer team can prioritize the stories for iterations and releases. Also, it is better, if the main visionaries of the product describe how the product should behave. (Cohn Mike 2004, part 1, chapter 1.)

Initial stories for a project are created during a story writing workshop. In the workshop, everyone brainstorms as many stories as possible. With an initial set of stories, the developers then estimate the size of each one. (Cohn Mike 2004, part 1, chapter 1.) In an ideal scenario, each story stands alone. While this is not always feasible, whenever possible, stories should be crafted to allow for development in any sequence. (Cohn Mike 2004, part 1, chapter 2.)

The customer team and developers decide on an iteration length, which can range from one to four weeks, and this length remains consistent throughout the project. By the end of each iteration, developers are expected to deliver fully functional code for part of the application. The customer team stays actively involved during the iteration, discussing the stories being developed with the developers. They also define tests and collaborate with developers to automate and run

these tests. Additionally, the customer team ensures that the project is continuously heading towards delivering the visioned product. (Cohn Mike 2004, part 1, chapter 1.)

According to Mike Cohn the main goal of user story is to describe how to product works to benefit the user or purchaser. The user stories are used as a reminder and a written description of the aims of the product. Details are worked out in the conversation and recorded in the confirmation. (Cohn Mike 2004, part 1, chapter 1.)

It is necessary to divide the features into necessary and nice-to-have features and furthermore to minimum viable product releases. This can be done visually by drawing a horizontal slices through the story line with each release in its own slice. Add cards also for the questions and concerns. (Patton & Economy 2014, chapter 2.) Ideally, stories are independent from one another. This is not always possible but to the extent it is, stories should be written so that they can be developed in any order.

There are some auxiliary questions, when writing the story map. What is the idea, we are trying to accomplish? Who is the expected customer of the product? Who is the end user? Why would they use the product? Why is this product needed? What is it used for? After answering the questions and building a story map, a prototype is sketched. The prototype is tested with the customer. (Patton & Economy 2014, chapter 3.)

The product is build using an agile development methods. Each release or iteration should include the minimum the customer can work with. Every time the product is improved with more features. The point is to discover during each iteration, if the development is going to towards the right direction. (Patton & Economy 2014, chapter 3.)

Every detail should not be written as story. It is better to discuss these details when they become relevant. You can make a few annotations on a story card based on a discussion, but the conversation is the key, not the note on the story card. (Cohn Mike 2004, part 1, chapter 1.)

The story map is built of activities and tasks. Activities form the backbone of the story map and situated across the top of the story map. Tasks are arranged into columns below the backbone. (Patton & Economy 2014, chapter 5.) After the story map the product owner and developer team meet for a rich conversation about what to build. (Patton & Economy 2014, chapter 6.)

Before each iteration, the customer team can adjust the plan as needed. As iterations progress, the development team's actual speed is better understood and can be planned better. This may require modifying the list of stories by adding or removing some. Additionally, some stories may be easier than expected, prompting the team to take on an extra story within the same iteration. Also, if some stories are more challenging than anticipated, they may need to be postponed to later iterations or removed from the release plan entirely. (Cohn Mike 2004, part 1, chapter 1.)

Acceptance testing involves ensuring that the developed stories function exactly as the customer team intended. When an iteration starts, developers begin coding while the customer team specifies the tests. Depending on their technical skills, this could range from writing tests on the back of the story card to using an automated testing tool. Tests should be created as early in the iteration as possible, or even just before the iteration begins if you can reasonably predict what will be included. (Cohn Mike 2004, part 1, chapter 1.)

3 RESEARCH METHOD

This research focuses on improving programming education for engineering students at LAB University of Applied Sciences by using a research-based development model with a preplanned construction. The construction is tested with the target group, which consists of first-year international engineering students. The data is evaluated using qualitative methods.

3.1 Research-based development

The goal of research-based development is to provide practical actions for real-life challenges based on scientific research data (Toikko & Rantanen 2009, 33-34). Therefore, the development process is scientifically justified.

In this model research supports the development process and has three features (Toikko & Rantanen 2009, 33-34):

1. The development process utilises conceptual models of previous studies.
2. New development constructions and hypotheses are based on the previous models. New constructions are tested in practice. Hypotheses can change and improve during the study.
3. Conclusions are evaluated critically, and constructed models are further refined.

As Toikko and Rantanen assert, academic research should always maintain independence and remain uninfluenced by external actors. However, in the case of research-based development, the situation becomes somewhat complex. Such research often takes place within a school environment, where the interests of the school organization, teachers, and students must be carefully considered. On a broader scale, the goal is to educate future professionals who will contribute to the local economy, which should also factor into the equation. While research-based development cannot entirely escape the influence of these interest groups, the aim remains to serve the community's needs. As Toikko and Rantanen remind readers, the objective is to uncover usable development data relevant to the community targeted by the study.

The planned construction is tested with the target group and the model is modified and refined according to the evaluation during implementation. The target group can attend in the modification by giving their feedback and suggestions. (Toikko & Rantanen 2009, 45-46.)

According to the spiral model introduced by Toikko and Rantanen, the software development process follows a cyclic pattern. Each cycle includes justification, organization, implementation, and evaluation phases. These cycles repeat in a spiral, forming a continuous development process. After each cycle, results are tested, evaluated, and then reconstructed for the subsequent cycle. Notably, the spiral model requires long-term commitment, with the initial cycle establishing the foundational construction for the entire development process. (Toikko & Rantanen 2009, 66-67.) The purpose of this study is to construct the first cycle and a model for continuous development process.

Organizing a research-based development project necessitates a well-thought-out and concrete justification. Ideally, each project should have no more than one or two goals, and the justification must clearly articulate these objectives. Additionally, it is essential to address the generalizability of the project. (Toikko & Rantanen 2009, 57.)

Organizing a development project involves planning and setting it up. If the project is solely for an employee's personal development, informing the entire community is not necessary. Typically, projects involve more than one employee, and communication should extend to the entire community. (Toikko & Rantanen 2009, 58-59.)

After the organization phase, Toikko and Rantanen introduce the implementation phase. Typically, the initial idea for development emerges during the justification or organization phase of the project. During the implementation phase, structured brainstorming, prioritization, trials, and modeling of the research-based development process occur. A key question in this phase is how to achieve the pursued goal. Ideally, implementation takes place in an authentic environment. Tentative implementation aims to describe a clear construction that can be applied and further developed. (Toikko & Rantanen 2009, 59-60.)

The subsequent phase in the development cycle concludes with the evaluation phase. The primary objective of this phase is to generate data related to the subject under development. This

data serves as the basis for decision-making and contributes to the advancement of the relevant community. (Toikko & Rantanen 2009, 61-62.)

The inclusion of the target group offers several benefits in the development process. It ensures that everyone's voices are heard, and their interests are considered. Whenever possible, collaborators should also be involved in the process (Toikko & Rantanen, 2009, pp. 89-90). One effective approach to including the target group is by testing the development construction in its natural environment. This way, the target group actively participates in the development project, providing feedback and responding to measurement questionnaires. Such an approach can also be implemented as a piloting model, where the construction is initially tested with a smaller group (Toikko & Rantanen, 2009, pp. 99-101).

3.2 Constructive research process

Lukka notes, the process of creating an ideal construction is not a simple task, but the process can be divided into seven steps (Lukka 2006, 114-118).

The first step is to find a real-life challenge to solve with the new construction. Tackling this subject with a newly invented construction should also result into theoretical contribution. (Lukka 2006, 114.)

The second step is to find a long-term research cooperation partner. The researcher and the target organization should commit to the project. It would be ideal, if the researcher was a member of the organization community. The researcher might have difficulties to apply the construction in the organization if he/she was an outsider in the community. It is also good to research the values of the target community and if they are similar with the researcher. (Lukka 2006, 114-115.)

The third step is for the researcher is to familiarize himself well with the subject at hand practically and theoretically. This is a way for the researcher to get to know to know the goals and challenges of the target organization. After this step the researcher should be familiar with the theoretical history of the subject. Lukka reminds the reader that this sets the researcher apart

from the consultants, who rarely have as profound empirical and theoretical background knowledge. (Lukka 2006, 115.)

The fourth step is to produce a creative construction, which can contribute to the organization's day-to-day life and theoretical reasoning. This is a creative process, which does not have universal instructions to it. If possible, the target community should attend to the construction creation. The creation can be an iterative process. (Lukka 2006, 116.)

According to Lukka, the fifth step is to implement the construction and test the applicability. The first applied test is one of the most important steps. The target organization should be motivated for the process and be committed to developing the construction further. Thus, the demand for objectivity is not applicable in this case according to Lukka. Even though the aim is to find a solution to a challenge, Lukka reminds us, that also failed construction can give us valuable data for the future. (Lukka 2006, 116-117.)

Lukka defines the theoretical contribution of the results and defining the scope of generalisation as the sixth step. If the results seem to be generalisable, it should be implemented, tested and analysed again in another environment. (Lukka 2006, 117-118.)

The seventh and last step of the constructive process is to analyse the theoretical contribution of the construction. This is a necessary step for any academic research. At this stage it is also necessary for the researcher to distance himself from the target group and compare the results with previous studies. The construction itself can be a significant theoretical contribution. (Lukka 2006, 118-119.)

3.3 Construction implementation

One of the types of case studies is constructive research method as Kari Lukka explains in his article. It is relevant method for researchers focusing on applied research. The constructions are generated by the researcher to solve real-life challenges. Its characteristic feature is a new construction is invented as part of the research project. The goal is also to implementation in a relevant real-life environment. With the real-life implementation of the construction its applicability. (Lukka 2006, 111-113.)

Lukka points out, the constructive research method requires also a close interaction with the researcher and the target group. Despite the strong connection to real-life, the construction model relies heavily in previous theoretical data and it also aims at providing new theoretical academic research data for the public. (Lukka 2006, 113.)

The ideal result of constructive research is to be able to solve a real-life challenge with a newly invented construction, Lukka notes. The invented construction should provide strong contribution for theoretical and applied science. (Lukka 2006, 113.)

3.4 Data analysis

The results are analysed using qualitative methods. Although Eskola & Suoranta suggest the qualitative research method to be primarily inductive and without a preliminary hypothesis, this study uses deductive method. There is a hypothesis, which is tested on the students by the researcher, who works as a lecturer in the class. The hypothesis is, using user stories as the roadmap to be learn programming can increase the diversity of student's motivation in their studies. (Eskola & Suoranta 1998, Chapter 1.)

The following data collection methods are used:

1. Questionnaires in the beginning and end of the course.
2. Questionnaires for the collaborative organisations. (Eskola & Suoranta 1998, Chapter 1.)

The study is done in the classroom with students. The researcher participates actively by teaching the class. Therefore, the researcher is an active subjective in the research herself.

Eskola and Suoranta emphasize, the researcher should be aware of his/her subjectivity and state it clearly in the research. Only then, can objectivity of the research be obtained. (Eskola & Suoranta 1998, Chapter 1.) According to the authors, pure objectivity is an illusion. Therefore, it is not realistic to even pursue complete objectivity. (Eskola & Suoranta 1998, Chapter 2.)

There is a preliminary hypothesis for this study by the teacher / researcher. The purpose of this study is to either validate or invalidate the hypothesis. Eskola and Suoranta do remind us, there

can be a working hypothesis, but the material itself, should be given a change to surprise the researcher with the results and findings. Thus, the hypothesis should not constrain the researcher. (Eskola & Suoranta 1998, Chapter 1.)

As Eskola and Suoranta state in their book, the research subjects should be selected discretionarily**,** but the quantity can be relatively small (Eskola & Suoranta 1998, Chapter 2.). In this study the research subject group is an international class of students studying the programming basics with Python. As Eskola and Suoranta mention, all the quantitative research studies are as a matter of fact, case studies. Therefore, the goal is not to make all-inclusive generalizations. Furthermore, it is not easy to decide, how much data to be analysed is enough. One used method is to analyse a small dataset and then test the results with a larger dataset. (Eskola & Suoranta 1998, Chapter 5.)

The group of students all attend the programming class. They are individually digitally asked if they are willing to attend the research to meet the ethical requirements. According to Eskola and Suoranta, there should not be a relationship between the researcher and research subject, such as the relationship between a teacher and a student. Ethical questions are being considered thoroughly. This study is made purely for improving teaching methods; therefore, the ethical questions can be met by anonymizing the material. Furthermore, there is no personal information shared in the study. Pupil's identity or grades are not visible in the outcome. Also, the school is aware of the study. (Eskola & Suoranta 1998, Chapter 2.)

4 CONSTRUCTION

This user story-based learning project is designed to enhance programming education for students at LAB University of Applied Sciences. The project combines weekly tasks with a comprehensive course-long project, providing a mix of theoretical knowledge and practical application. The aim is to make programming more accessible and engaging, while also preparing students to tackle complex engineering challenges.

The construction was implemented in introductory programming lessons worth 5 credits. These lessons are part of the Introduction to Industrial ICT Engineering course, which is worth 15 credits. The students were first-year Industrial Information Technology (IIT) engineering students.

The IIT program is conducted in English and includes both Finnish and international students. The content of the Industrial Information Technology engineering program is depicted in the following image.

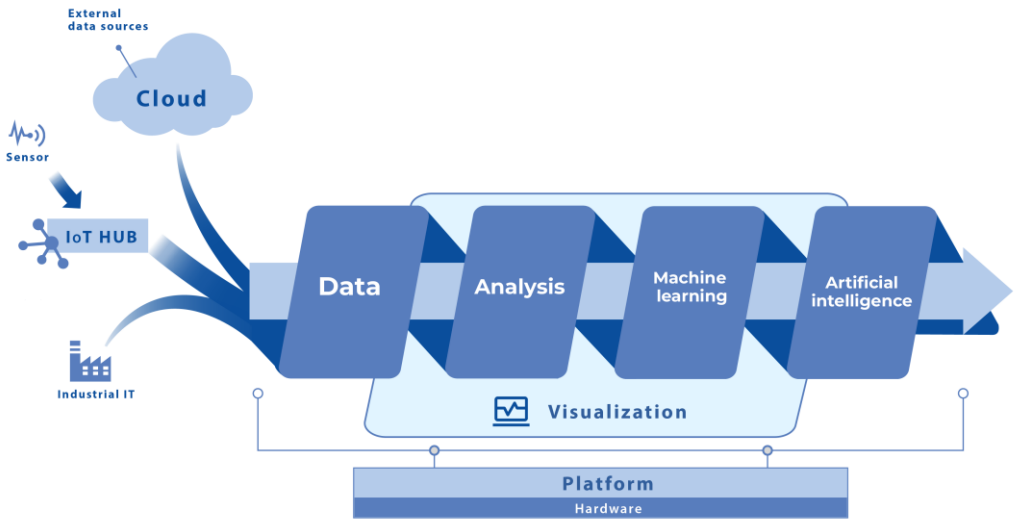

FIGURE 1: LAB University of Applied Sciences, Technology department, Content of ICT education

4.1 Weekly Tasks

The course has previously been implemented by giving students weekly assignments about the theory lessons. This part of the course was kept as it was and organized by the responsible teacher. Only the number of tasks was reduced to fit the project. The new construction implements 10/14 weeks of tasks, each having 5 sub-tasks.

Each week, students are assigned five tasks which are directly related to the weekly theory lessons. These tasks are designed to reinforce the concepts taught in class and provide hands-on practice.

The tasks are graded automatically using CodeGrade (https://www.codegrade.com/), an application that evaluates the code submitted by students. This ensures timely and consistent feedback.

Students are reminded of soft deadlines throughout the week to encourage steady progress. The hard deadline for all tasks is at the end of the course, allowing flexibility for students to manage their time effectively.

CodeGrade not only grades the tasks but also provides feedback on the code, helping students understand their mistakes and learn from them.

The introductory programming course is graded as pass or fail. Students receive numerical grades for the weekly tasks, but these are only for the students themselves to track their progress.

4.2 Course-Long Project

The central project involves developing a Python console application which interacts with an API. The embedded programming and backend of the project are provided for the students for their first introductory programming course. Raspberry pi uses Python to collect data from the PIR sensor and sending it to a service called ThingSpeak (https://thingspeak.mathworks.com/) to create an API for the project. Also, the temperature of the CPU is sent to ThingSpeak. In the

upcoming courses students learn to do the electronics, backend, database, and frontend with data analysis by themselves.

An example of the hardware:

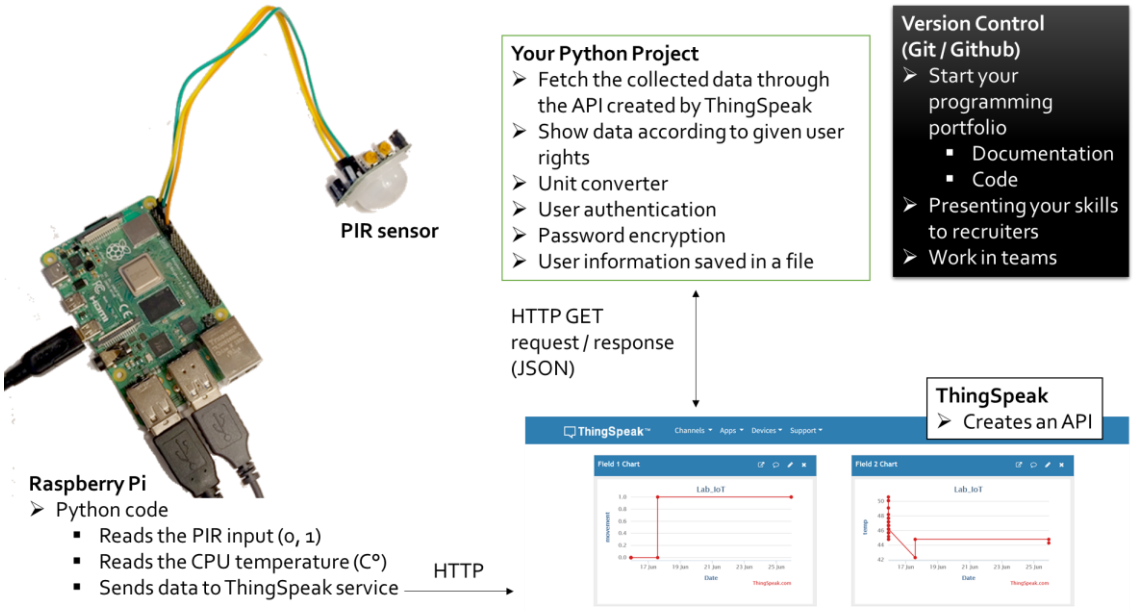

FIGURE 2. Teaching material for the course

An example API:

https://api.thingspeak.com/channels/2578404/feeds.json?api_key=XSXF6WH7DAECB6S1

An example of the JSON data:


```
{
 "channel": {
  "id": 2578404,
  "name": "Lab_IoT",
  "latitude": "0.0",
  "longitude": "0.0",
  "field1": "movement",
  "field2": "temp",
  "created_at": "2024-06-15T07:49:03Z",
  "updated_at": "2024-06-15T18:37:45Z",
  "last_entry_id": 131
 },
 "feeds": [
  {
   "created_at": "2024-06-15T18:47:33Z",
   "entry_id": 32,
   "field1": "0",
   "field2": "49.6"
```

```
    },
    {
      "created_at": "2024-06-15T19:02:11Z",
      "entry_id": 84,
      "field1": "0",
      "field2": "46.7"
    },
    {
      "created_at": "2024-06-25T19:20:40Z",
      "entry_id": 131,
      "field1": "1",
      "field2": "44.3"
    }
  ]
}
```


The project uses a PIR sensor connected to a Raspberry Pi to detect motion and provide CPU temperature data. This project is designed to be both challenging and relevant, giving students a real-world application to work on.

The project is divided into weekly sprints, each lasting one week. During each sprint, students work on user stories that outline specific features to be added to the project. These user stories are based on the weekly theory lessons and may evolve based on feedback from a hypothetical customer. The first, introductory programming course uses hypothetical customer, but at a later courses' projects are done for real-life customers.

At the end of the course, students present their completed projects to their peers. This presentation serves as both a learning experience and an opportunity to showcase their work. The project is graded verbally by the teacher and peer students, providing qualitative feedback.

Version Control: Students use GitHub for version control, documenting their progress and ensuring that their code is well-managed and collaborative.

4.3 Project Features

The application includes a console-based menu for user interaction. Students familiar with UI programming can choose to create a graphical interface, but the primary focus is on console-based interaction.

Information is displayed in an ASCII format, ensuring that students learn to present data in a clear and readable manner.

The application includes features for creating, reading, updating, and deleting (CRUD) user information. This involves implementing password encryption, hashing, and salting to ensure data security. The data is stored in a JSON file to manage user information.

Students learn to utilize fundamental data structures and programming constructs. They also develop skills in creating flowcharts, following best practices and style guidelines to ensure their code is not only functional but also maintainable and readable. The project covers essential programming concepts such as recursive functions, basic object-oriented programming (OOP), data structures, and file handling. Additionally, students are introduced to the basics of parallel programming.

The application uses Matplotlib to create visualizations of the data. This helps students learn how to represent data graphically, making it easier to analyse and interpret.

Throughout the course, students are introduced to useful modules and libraries, which can enhance their projects. This exposure helps them understand the range of tools available in Python. Students learn to create their own custom modules, promoting modularity and reusability in their code.

The project includes comprehensive error handling and debugging practices. Students learn to anticipate and manage errors, ensuring that their applications are robust and reliable.

Students learn to break down user stories into small, manageable steps. This approach helps them tackle complex tasks incrementally, making the development process more manageable and less overwhelming.

Creating code with peer students is encouraged. This collaborative approach not only enhances learning but also helps students develop teamwork and communication skills, which are essential in real-world programming environments. The students are grouped in teams at the beginning of the course, and they make a small, one day long, scrum learning project in the team. The

programming tasks and project are returned individually, but discussion and peer guidance is allowed.

4.4 Documentation and Reporting

Students write sprint backlogs for each week, detailing the tasks to be completed and the progress made. This helps them stay organized and focused.

Students document their project in the README file on GitHub. This documentation includes an overview of the project, instructions for setting it up, and explanations of the key features and design decisions. The project code is also stored in GitHub with teacher collaboration rights to monitor the progress.

This user story-based learning project aims to provide a comprehensive and engaging learning experience. By combining theoretical instruction with practical application, it equips students with the skills and knowledge needed to succeed in their future careers and contribute to technological advancements.

5 IMPLEMENTATION

The implementation includes a comprehensive outline for a programming course, detailing the steps from initial course planning to final assessment, including weekly lesson plans, project design, and student evaluations.

5.1 Course Planning and Preparation

Clearly outline the learning objectives for the course, focusing on both theoretical knowledge and practical skills.

Create detailed lesson plans for each week, covering the necessary programming concepts and theories.

Develop five tasks for each week that align with the weekly theory lessons. Ensure these tasks are varied and progressively challenging.

5.2 Setting Up CodeGrade

Set up the CodeGrade platform to automatically grade the weekly tasks. Ensure it is configured to provide detailed feedback on the code submissions.

Develop grading standards for the tasks to ensure consistent evaluation.

5.3 Project Design and Planning

Outline the overall project, including the main features and functionalities of the Python application.

Create user stories for each week, detailing the specific features to be implemented. Ensure these stories are aligned with the weekly theory lessons. These user stories are up for change during the course to teach the students reacting to changes.

Organize the project into weekly sprints, each focusing on a different user story. Each sprint should end in a working project with the features applied so far.

5.4 Course Kick-off

Conduct an introductory session to explain the course structure, objectives, and expectations. Introduce the concept of user story-based learning and the course-long project.

Ensure all students have GitHub accounts and understand the basics of version control. Provide a tutorial for using Git with the project.

5.5 Weekly Execution

Deliver weekly theory lessons as planned.

Open the weekly tasks and set soft deadlines for the students. Use CodeGrade to grade and provide feedback.

At the beginning of each week, conduct a sprint planning session to discuss the user story for the week and outline the tasks to be completed.

Encourage students to work together on their projects, fostering a collaborative learning environment.

5.6 Project Development

Guide students to break down user stories into small, manageable steps. Ensure they understand the importance of incremental development.

Offer regular support sessions to help students with any challenges they face during the project development.

Regularly check the progress of each student's project through their GitHub repositories and provide feedback.

5.7 Mid-Course Review

Conduct mid-course reviews to assess the progress of the projects. Provide feedback and make any necessary adjustments to the course plan.

Organize peer review sessions where students can present their progress and receive feedback from their peers.

5.8 Final Project Presentation

Guide students in preparing their final project presentations. Ensure they cover all aspects of their project, including the development process, challenges faced, and solutions implemented.

Organize a presentation session where each student presents their project to the class. Encourage peer feedback and discussion.

5.9 Final Assessment

Grade the final projects verbally, providing detailed feedback on the strengths and areas for improvement.

Determine the final pass/fail grade for each student based on their overall performance throughout the course.

5.10 Course Wrap-Up

Conduct a reflective session where students can share their experiences and feedback on the course. Use the feedback to make improvements to the course for future iterations.

6 EVALUATION

The evaluation of the construction was conducted through anonymous and voluntary questionnaires for both students and ICT professionals. The ICT professionals' questionnaire (APPENDIX 1) was introduced via a LinkedIn post, while the students' questionnaire (APPENDIX 2) was presented in the classroom at the end of the course.

The students' questionnaire was conducted only once, meaning only those present in class on the day of the questionnaire participated. Both questionnaires were administered using Microsoft Forms. In total, four professionals and 26 out of the 53 enrolled students responded. Additionally, 45 students completed the course on time.

The professionals' questionnaire included three content questions, which were conducted in Finnish. At the time of the questionnaire, one respondent had a master's degree in technology, over 20 years of experience in the field, and was currently in a managerial position. The second respondent had a doctorate in technology, worked as a teacher or educator, and had 6 to 20 years of experience. The third respondent was a programmer with a master's degree in technology and over 20 years of experience. The final respondent was a programmer and cloud architect with university studies in computer science and more than 20 years of experience.

1. Question for the professionals: What challenges and opportunities do you see in user story-based teaching from the perspective of the partner company or customer? What should be considered in teaching from the company's perspective?

   According to a professional, integrating educational tools with those used by client companies can present a challenge. Designing workflows to align with client practices is essential. Project work, which favours milestone thinking over weekly tracking, is crucial for students transitioning to the workforce. He was pleased, that this approach would not only provide a new recruitment channel for clients but also familiarizes young people with company practices and required skills before potential recruitment.

Another professional noted that streamlining the initial phase of projects can be enhanced by leveraging companies that specialize in initial phase mapping, such as NAICC, which offers “strike teams” and utilizes LLM tools to assist companies effectively.

One professional highlighted that mentorship poses a challenge, as those responsible for mentoring students often have critical tasks, making time allocation difficult. As he mentions, it is essential for students to have a solid theoretical foundation before entering the workforce, as practical work takes precedence, and a lack of theoretical knowledge can lead to issues later. Work-based learning has its limits, he/she thinks, and the less students know when starting a project, the more it burdens other team members.

One of the professionals emphasized that client companies must recognize that students may require substantial support initially to learn company practices and methods, just like new employees do. He recognizes this model allows companies to identify suitable employees more effectively than traditional recruitment, as students’ knowledge, skills, and suitability have already been assessed. Teaching agile methods, such as Scrum, and basic version control practices, including the use of branches and pull requests, is beneficial before starting client projects, he/she thinks.

2. Question for the professionals: How would you improve user story-based teaching from a professional’s perspective.

According to the first professional, in the modern work environment, presentation skills and the ability to clearly express essential information are crucial. Engineering students must develop these skills through project work, enabling them to analytically summarize and present their results during milestone reviews. They should also be capable of evaluating schedule compliance and identifying necessary corrective actions for any deviations from the original plans.

Concrete role-based exercises have proven to be highly effective in teaching these skills, noted the second professional. The third professional emphasized the importance of understanding what constitutes an appropriate and sufficiently good solution for each problem, which should be as easy as possible to implement and test. However, young people often either aim for perfection or are careless, resulting in inadequate solutions,

he/she says. Achieving the right level of proficiency is a challenging task that may require decades of practice. Addressing this fundamental issue is complex and does not have a straightforward solution.

The fourth professional highlighted the importance of breaking down user stories into tasks collaboratively with the team and individually for effective communication. Additionally, practicing workload estimation with the team, using methods such as planning poker, is beneficial for accurate project planning and execution.

3. Question for the professionals: What types of projects you believe would best develop the skills students need in the workplace.

   One professional emphasized that effective project-based learning should be milestone-driven. Using tools like MS Project to schedule these milestones is essential, and the criteria for passing or rejecting milestones must be clear, unambiguous, and goal oriented.

   Another professional suggested that projects should be real-life scenarios that allow for the possibility of failure. Incorporating an Agile Coach who is external to both the case company and the university can enhance the team's self-determination.

   A solid theoretical foundation is crucial before applying knowledge in work projects, according to one professional. Traditional homework might be more suitable for building this foundation, as immediate application without a strong theoretical base can lead to gaps in knowledge.

   Engaging students in projects that use modern front-end and back-end (API) technologies, along with cloud infrastructure, is beneficial, explained another professional. The front end could use a JavaScript framework like React or Vue, the back end could be in Node.js or C#, and the database could be any PaaS database such as MySQL, Azure SQL, or PostgreSQL.

   Students should learn to use branches and pull requests, the professional noted. Understanding basic network concepts and how to secure resources is important. For

example, access to a SQL service should be restricted to the backend web app or function app, and the backend should only be accessible from the company's internal IP address. While not primarily the responsibility of software developers, understanding these basics is beneficial.

Finally, the professional explained that students should learn to break down user stories into tasks and estimate the workload of these tasks. They should be able to report their progress, identify any problems, articulate whether they need help, and estimate the remaining workload of tasks. This is crucial for tracking overall project progress.

The students' questionnaire was conducted in English for an international first-year group in Industrial Information Technology.

The first three questions were to check the knowledge of basic terms of user story programming.

1. Question: What is the significance of a Product Owner in programming projects? Fourteen individual points were found from the answers. Most of the answers describe the product owner well. Some have mixed the product owner for example with the scrum master or the developer team members:
    - The product owner directs the project, making decisions on behalf of the client and ensuring the project moves in the correct direction.
    - They define the product vision, prioritize tasks, and ensure the development team works on the most valuable features.
    - The product owner is responsible for the design and look of the product, while the development team executes this design.
    - They are often the most knowledgeable person on the team, providing solutions when team members encounter problems.
    - The product owner's decisions influence the project at every step, making them central to the project's success.
    - They provide the initial idea and feedback for the project, defining what is needed and what should be changed.
    - The product owner ensures the team delivers maximum value by aligning the team backlog with customer needs.
    - They own the product being developed, making their role highly significant.

- The product owner ensures the team creates the best product for users by communicating with both the team and stakeholders.
- They create and manage the product backlog, explaining its importance to the team.
- The product owner translates customer needs for the team and communicates with both the customer and the development team.
- They act as a bridge between business stakeholders and the development team, ensuring clear communication and alignment.
- The product owner knows all client requirements and tasks, ensuring the output fits the client's desires.
- They guide the team on what to focus on, determining project features and goals to ensure efficient delivery.

2. Question: What is the significance of User Stories in programming? The answers can be categorized into nine individual topics. Most answers were quite on point:
    - User Stories break down the project into manageable chunks, making it easier to achieve specific goals. This compartmentalization helps in organizing the work into smaller, achievable tasks, which can be tackled incrementally. This approach ensures that the development process is more structured and less overwhelming.
    - User Stories are crucial as they define features from the user's perspective, ensuring the development team understands the end user's needs and goals. They specify the features and options needed in the project, making them essential for building an app or project that meets user expectations. By focusing on user needs, User Stories help in creating a solution that is user-centric and aligned with the desired outcomes.
    - User Stories facilitate clear communication between developers, stakeholders, and users. They capture the product owner's needs and provide context for why the program is being created. This communication ensures that everyone involved in the project is on the same page, promoting collaboration and reducing misunderstandings.
    - In Agile methodologies, particularly frameworks like Scrum, User Stories are foundational. They help in capturing and communicating the needs of end-users, ensuring the development process is user-focused, collaborative, and aligned with business goals. User Stories guide the development team on what to focus

on during each Scrum cycle, helping in the creation of a product backlog and ensuring efficient project management.

- User Stories are used to add new functionalities or modify existing ones. They describe the necessary features for the project, guiding the development team in building the required functions. This ensures that the project evolves in line with user needs and business requirements.
- User Stories can serve as a learning tool for writing code, creating backlogs, and understanding potential problems. They provide insights into the challenges that might be faced during the development process and help in preparing for and addressing these issues effectively.
- User Stories are often highlighted as the most significant aspect of programming projects. They are described as the most important piece of the puzzle, helping the development team understand and create a product backlog. Their significance lies in their ability to guide the project towards successful completion by focusing on user needs and business goals.
- User Stories contain just enough information for developers to estimate the effort needed to implement them. This clarity in requirements helps in planning and resource allocation, ensuring that the development process is efficient and well-organized.
- User Stories capture glimpses of customer ideas and needs, enabling developers to demonstrate their abilities to collaborators. They help in gathering and incorporating feedback, ensuring that the final product meets customer expectations and delivers value.

3. Question: What is the significance of a Product Backlog in programming? The answers can be categorized in nine topics. Overall, the answers were on point:
    - The Product Backlog is essential for organizing and prioritizing all tasks, features, and improvements needed for a project. It ensures that the development team focuses on the most valuable work by continuously refining and updating the backlog based on feedback and changing priorities. This helps keep the project on track and aligned with user needs and business goals.
    - In Agile programming, particularly in frameworks like Scrum, the Product Backlog serves as a prioritized list of all work to be done on a product. It is a cornerstone of agile project management, providing a single source of truth for all tasks. This

ensures that the development process remains focused, collaborative, and aligned with the overall mission and long-term product targets.

- The Product Backlog acts as a central repository for all project requirements, providing transparency and visibility into the project's progress. It helps in communicating what needs to be done, what has been done, and what is currently being worked on. This shared understanding among the team and stakeholders ensures that everyone is on the same page.
- The Product Backlog helps in dividing large tasks into smaller, manageable sections, making the development process smoother and less hectic. It provides a roadmap for the development team, guiding them on what to do next and helping in the distribution of tasks. This organization improves work efficiency and teamwork.
- The Product Backlog is flexible and adaptable, allowing for changes based on new information, feedback, or shifting priorities. This adaptability is crucial in Agile methodologies, where the ability to respond to change is a key principle.
- The Product Backlog helps with estimations of time and work needed for each task. This enables better planning and resource allocation, ensuring that the development process is efficient and well-organized. It also helps in risk management by identifying potential issues early on.
- The Product Backlog often includes rephrased user stories, which quantify user needs and turn them into concrete work items. This ensures that the development team understands the requirements from the user's perspective and can deliver a product that meets those needs.
- The Product Backlog can also serve as a historical record of all modifications and comments, allowing the team to track changes and understand the evolution of the project. This documentation is valuable for future reference and continuous improvement.
- The Product Backlog provides a visualization of all items needed to fulfil the long-term product target, ensuring a shared understanding among the full Scrum team and stakeholders. It helps in communicating progress and aligning expectations with customers.

Questions four and five were about motivation of the student in programming class.

4. Question: What did you find motivating in the course? When did the work feel enjoyable? What practices from the project did you find useful in future exercises, projects, or internships? The answers were categorized into following topics:
    - Motivation
    - Enjoying their work
    - Finding Useful Practices for the Future
    - Overcoming Challenges
    - Learning Bigger Concepts

    Motivation
    - Many students found motivation in applying programming concepts to solve real-world problems, such as building a motion detector project or seeing practical applications of programming concepts.
    - The opportunity to learn new things, especially in programming languages like Python, and discovering new concepts were significant motivators.
    - Competing with peers and working in groups provided motivation, as it fostered a sense of camaraderie and learning from others.
    - Weekly tasks, in-class exercises, and the structured approach of the course helped maintain focus and motivation.

    Enjoying their work
    - Students enjoyed the process of solving problems, especially when they could independently find solutions and see their code work as intended.
    - Working in groups was enjoyable for many, as it allowed them to make new friends, learn from peers, and feel a sense of community.
    - Practical, hands-on tasks and projects were more enjoyable than theoretical lessons, as they provided a sense of accomplishment and real-world relevance.
    - Some students mentioned feeling a state of flow while coding, which made the work enjoyable and engaging.

    Finding Useful Practices for Future
    - Breaking down complex tasks into smaller, manageable steps and organizing work effectively were practices found useful for future projects.
    - Using version control systems and testing code regularly were highlighted as valuable practices for maintaining high-quality work.

- Learning to communicate effectively and collaborate with team members was seen as beneficial for future exercises and internships.
- Practices like working in sprints and using Scrum methods were found useful for understanding how to manage projects in a professional setting.
- The ability to learn independently, adapt to new challenges, and continuously improve skills were emphasized as important takeaways.

Overcoming challenges

- Some students found the increasing difficulty of the course challenging but acknowledged that overcoming these challenges was part of the learning process.
- For those new to programming, the course was particularly challenging, but they appreciated the structured learning and support provided.

Learning Bigger Concepts

- Learning Python and its applications in various fields like machine learning, AI, and data analysis was motivating and seen as highly useful.
- Understanding how to manage projects, especially using Scrum methods, was a key takeaway for many students.
- Developing logical thinking and problem-solving skills through coding exercises was seen as beneficial for future endeavours.

5. Question: What challenges did you face in the project? Were they related to, for example, endurance, motivation, or technical skills? How did you overcome the challenges? The answers were categorized into following topics:
   - Health and Endurance
   - Motivation and Procrastination
   - Technical Skills and Understanding
   - Time Management and Workflow
   - Conceptual Understanding
   - Project Specific Challenges
   - Language and Communication
   - Collaboration and Support

   Health and Endurance

   - Some students faced health challenges, such as being sick for extended periods, which affected their ability to keep up with the course.

- Fatigue and the need for sustained concentration were common challenges. Students mentioned feeling overwhelmed by the workload and the need to take breaks to manage their endurance.

Motivation and Procrastination

- Maintaining motivation was a challenge for some, especially when facing difficult tasks or when progress was slow. Procrastination also affected their ability to stay on track.
- Strategies to overcome these challenges included breaking tasks into smaller steps, celebrating small wins, and seeking support from friends or classmates.

Technical Skills and Understanding

- Many students struggled with technical skills, particularly when learning new programming concepts or debugging code. This was especially challenging for beginners.
- Overcoming these challenges involved self-learning, using online resources like tutorials and documentation, and seeking help from AI tools or classmates. Continuous practice and hands-on experience helped improve their technical skills over time.

Time Management and Workload

- Balancing the project with other coursework and personal responsibilities was a significant challenge. Students found it difficult to allocate sufficient time for coding amidst a heavy workload.
- Effective time management strategies included taking breaks, prioritizing tasks, and adjusting study plans to better accommodate coding practice.

Conceptual Understanding

- Some students faced difficulties in understanding programming concepts, such as loops, object-oriented programming (OOP), and modules. This hindered their progress in the project.
- To overcome these challenges, students relied on explanations from online resources, AI tools, and peer support. Reading more code and practicing problem-solving helped solidify their understanding.

Project Specific Challenges

- The complexity of the project, especially when transitioning from learning basics to applying them in a project, was daunting for some students.

- Gradually learning and applying new concepts, along with consistent practice, helped students manage the complexity and build confidence in their abilities.

Language and Communication

- For some students, language barriers made it difficult to understand project requirements and instructions.
- Overcoming language challenges involved self-learning, deciphering information, and seeking clarification from available resources.

Collaboration and Support

- Students often sought help from classmates, friends, and online communities to overcome challenges. Collaboration and peer support was crucial in navigating difficult tasks.
- Working with others provided different perspectives and solutions, enhancing their learning experience and problem-solving skills.

6. Question was a multiple-choice question about the students' motivation to work as a programmer in the future:

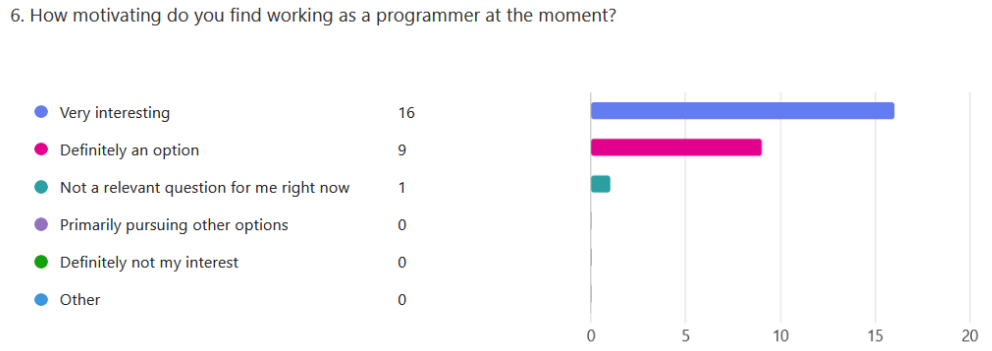

FIGURE 3. Graph of Microsoft Forms results, question 6, Questionnaire for the students

7. Question asked the students if they feel like they got a realistic experience working as a programmer:

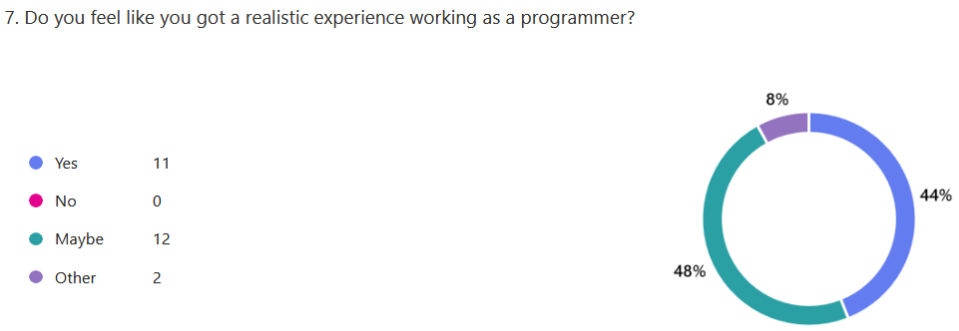

FIGURE 4. Graph of Microsoft Forms results, question 7, Questionnaire for the students

Two students answered verbally:

1. *"More like a little taste? Appetizer of the work? not a solid grasp yet"*
2. *"I haven't worked as a programmer yet ..,"*

8. Question had multiple sub-questions:

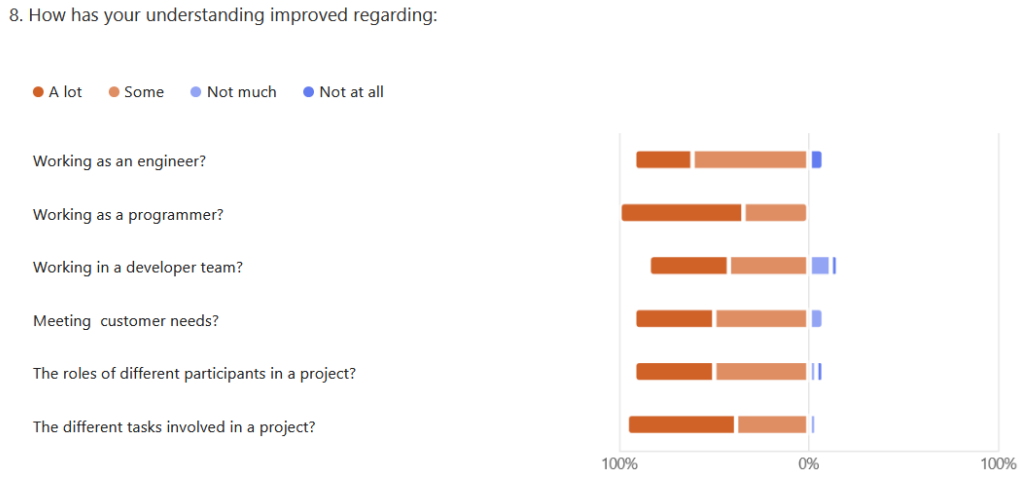

FIGURE 5. Graph of Microsoft Forms results, question 8, Questionnaire for the students

	A lot	Some	Not much	Not at all
Working as an engineer?	30.8%	61.5%	0%	7.7%
Working as a programmer?	65.4%	34.6%	0%	0%
Working in a developer team?	42.3%	42.3%	11.5%	3.8%
Meeting customer needs?	42.3%	50%	7.7%	0%
The roles of different participants in a project?	42.3%	50%	3.8%	3.8%
The different tasks involved in a project	57.7%	38.5%	3.8%	0%

FIGURE 6. Results in figures, question 8, Questionnaire for the students

9. Question was about the students' professional identity before the course:
   How would you describe your professional identity before starting this course?
   What were your initial thoughts and feelings about your role in IIT engineering?
   How confident did you feel about your skills and knowledge in IIT engineering?

   The answers were categorized into following topics:

   - Professional Identity Before Starting the Course
   - Initial Thoughts and Feelings About the Role in IIT Engineering
   - Confidence in Skills and Knowledge
   - Realizations and Growth
   - Specific Experiences and Reflections

   Professional Identity Before Starting the Course

   - Many students started the course with curiosity and motivation, eager to learn about IIT engineering and explore the field.
   - Several students had minimal or no prior knowledge of IIT engineering, feeling like "empty paper" or having only surface-level understanding.
   - Some students had a general interest in technology and computers, which drove them to pursue the course despite limited specific knowledge.

   Initial Thoughts and Feelings About the Role in IIT Engineering

   - Initial feelings were often a mix of excitement and optimism about learning new skills and diving into the field of IIT engineering.
   - Many students felt uncertain or insecure about their role, unsure of what to expect and how they would fit into the field.

- There was an awareness that the course would be challenging, requiring hard work, focus, and the ability to manage time effectively.

Confidence in Skills and Knowledge

- Most students had low confidence in their skills and knowledge at the beginning, feeling unprepared or lacking experience in programming and IIT engineering.
- Confidence generally improved over time with consistent practice, hands-on experience, and learning from the course. Students noted feeling more confident as they gained more knowledge and skills.
- Many students acknowledged that learning is a continuous process, and while they felt more confident than before, they also recognized the need for ongoing practice and improvement.

Realizations and Growth

- Students realized that IIT engineering is much more than just coding, encompassing a wide range of technologies and applications.
- The course helped students understand their potential roles in IIT engineering, from backend programming to collaborative teamwork and problem-solving.
- Facing and overcoming challenges in the course helped students build resilience and adaptability, essential traits for their future careers.

Specific Experiences and Reflections

- Some students highlighted positive experiences, such as feeling proud and curious about solving real-world problems using technology.
- Others mentioned struggles with keeping up, feeling like they were constantly playing catch-up, but also emphasized the importance of perseverance and continuous learning.

10. Question was about the development of the students' professional identity during the course:

    How has your professional identity evolved after completing this course?

    In what ways have your thoughts and feelings about your role in IIT engineering changed?

    How has your confidence in your skills and knowledge improved?

    The answers were categorized into following topics:

    - Evolution of Professional Identity
    - Changes in Thoughts and Feelings About the Role in IIT Engineering

- Improvements in Confidence in Skills and Knowledge
- Specific Experiences and Reflections
- Areas of Continued Development

Evolution of Professional Identity

- Many students now have a broader understanding of IIT engineering, recognizing it as a multifaceted field involving various technologies and roles. They feel more skilled and ready to take on challenges.
- Some students acknowledge that their professional identity is still evolving, and they are committed to continuous learning and improvement.

Changes in Thoughts and Feelings About the Role in IIT Engineering

- Students generally feel more confident and excited about their role in IIT engineering. They now see themselves as capable of contributing to real-world projects and solving complex problems.
- There is a greater appreciation for the complexity and breadth of IIT engineering, with students understanding that it involves more than just coding.
- Despite increased confidence, many students recognize that there is still much to learn and are motivated to continue improving their skills.

Improvements in Confidence in Skills and Knowledge

- Students report significant improvements in their programming abilities and technical knowledge. They feel more comfortable tackling complex problems and using various tools and languages.
- While confidence has improved, many students emphasize the importance of ongoing practice and learning to further enhance their skills.
- Some students still feel like beginners and acknowledge the need for more experience and study to build their confidence fully.

Specific Experiences and Reflections

- Working on projects has been particularly impactful, helping students apply their knowledge and see the real-world relevance of their skills.
- Students have faced challenges, such as understanding programming concepts and managing time, but have persevered and grown through these experiences.
- Many students have ambitious goals for their future careers in IIT engineering and are eager to continue learning and developing their skills.

Areas of Continued Development

- Students recognize the need to further develop their technical skills, particularly in areas like front-end development and advanced programming concepts.
- Gaining more professional experience, such as internships or projects, is seen as crucial for building confidence and expertise.
- Some students express concerns about finding jobs or internships, highlighting the need for support in transitioning from education to the professional world.

11. Question was about the evaluation of the students' own skill level before the course:

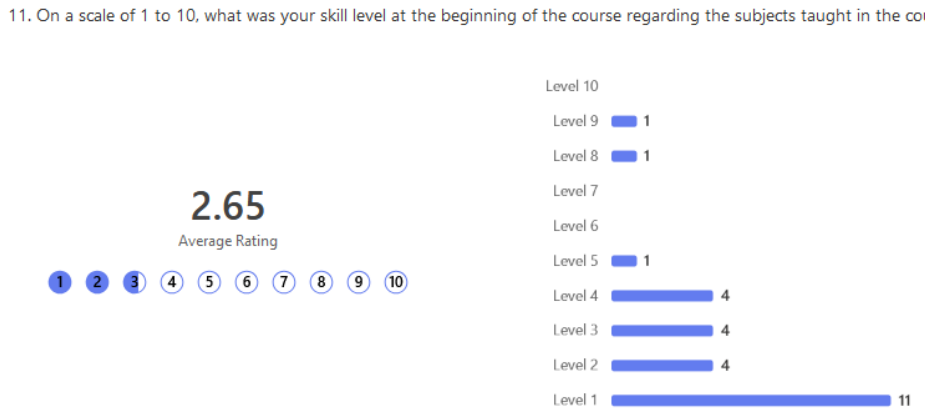

FIGURE 7. Graph of Microsoft Forms results, question 11, Questionnaire for the students

12. Question was about the evaluation of the students' own skill level after the course:

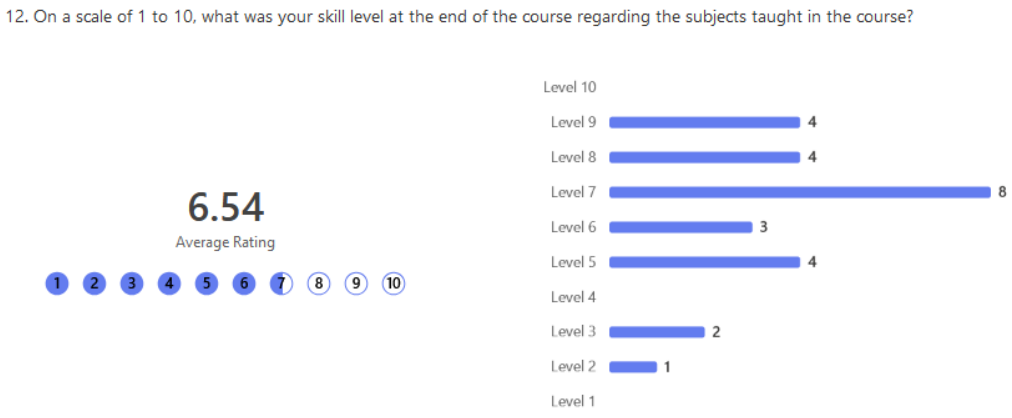

FIGURE 8. Graph of Microsoft Forms results, question 12, Questionnaire for the students

7 NEXT STEPS AND FUTURE STRATEGIES

At LAB University of Applied Sciences, as with any university of applied sciences in Finland, collaboration with industry partners is a fundamental part of our mission and identity. But why would a company collaborate with LAB IIT students? At an ICT networking event organized by the city of Lahti's International Work Lahti project, a group of employers was asked how long it takes for an experienced engineer to start making a profit for the employer. The group estimated a year.

When the same question was posed to Copilot, the answer ranged from six months to a year. The group was also asked how long it would take for a freshly graduated engineer to become profitable. They estimated one and a half to two years. Copilot's estimate varied from a few months to two years, depending on the job and the individual graduate.

The aim of user story-based education is not only to motivate students to study programming through real-life projects in collaboration with actual organizations and companies but also to provide employers with graduates who are as ready for the workforce as possible. Additionally, the goal is to immerse foreign students in Finnish work life while they are studying, reducing the training burden on employers.

Therefore, the approach is shifting from project-based learning to user story-based learning at LAB University of Applied Sciences, industrial information technology group. This method is iterative, with continuous improvement through feedback and practical experience. The specifics or duration of a project may not be known from the start but develop as the project progresses. User story-based learning offers more direct mentorship and guidance, aiding skill development and confidence building.

User story-based education is built on three pillars:

1. Demand-driven course content: The aim is to minimize the additional training companies need to provide to new hires by integrating real-life projects into university studies.

2. Hands-on learning labs and environments: These spaces encourage bold experimentation with new ideas and approaches. Organizations can collaborate with students and faculty to test solutions, create demos, and develop new features and technologies.
3. Recruitment pipelines: Organizations can create effective project and recruitment pipelines in collaboration with LAB, ensuring a steady flow of well-prepared, skilled graduates for the right jobs.

This style of teaching demands a lot from the educational institution. But it also provides flexibility and more profound integration of the institution and the students in the local professional environment.

Based on the evaluation and the interest in collaboration from the organizations, a new construction has been developed. This construction still needs further evaluation in the future.

Students from different academic year groups, as well as Finnish-speaking and English-speaking students, will work together on projects. This includes thesis writers and students from various courses, as well as students from Salpaus vocational school, LUT University, and LAB University of Applied Sciences. Although this requires organization, it will improve student's collaboration skills. The students should receive clear instructions and are expected to work at their own level. Although they are introduced to quite complex structures from the beginning, the most complex features are not included in the project's minimum requirements.

Projects will vary in length and have flexible start times. They may include nice-to-have features, demos, and R&D projects, with no responsibility for results. The students do not have an accountability for results.

Projects can be conducted remotely at LAB or on-site with the collaborating company or organization.

Possible collaboration forms include RDI project collaboration, study projects, training, thesis work, or any other needs the company may have at the time.

The goal is to complete projects that reduce the time it takes for an engineer to start generating profit for the hiring company.

For the following semester collaborative projects using user stories have been organized. Introduction to Industrial ICT engineering course is followed by compulsory Introduction to Data Pipeline course (previously Introduction to IoT Pipeline) and complementary Project in Company Co-Operation course, which can be done over the following years. Both courses are worth 15 credits. The first training should also be accomplished before the second academic year.

The projects are not fixed in length or content. The team doing the project might change in the process, as well as the goal, type, and purpose.

Collaboration project example 1, Resort cabins:

- Study project.
- Cabin reservation system integrated to an existing public website, integrated with other reservation applications.
- Customer extranet to reserve amenities like the beach sauna, fire hut, laundry room, gym, etc.
- Company intranet provides access to analyzed sensor data, customer information, and reservation management.
- Sensor application demos, which showcase applications for measuring wood storage levels, detecting water leakages, and other innovative solutions developed by students.

Collaboration project example 2, Building solution and construction technology provider:

- Thesis and training
- Collaborative project with local Vocational School
- The company offers designer tools available for free download online.
- Goal of the project is to combine all available and upcoming Precast Wall design tools to the same user interface.
- Tool should follow the design system and utilize the chosen technologies by the company.
- Concept with well documented plan boosted with prototypes.
- Technology concept and architecture plan for the proejct.
- Integration plan (API)

- Execution plan and phases to move forward.
- Execution

Collaboration project example 3, Career, and well-being coaching service provider.

- Study project.
- B2B scheduling application for career coaching.
- Integrated with other scheduling applications.
- Monthly coaching reports.

Collaboration project example 4, Heritage Museum

- Study project.
- Collaborative project with the local University
- Design and implement a mystery game for the museum, loadable from Google play and Apple store.
- Digital coloring book.
- Digital museum for learning history of the place in a fun way. Inspirations:
    - Digital interactive walls
    - Holograms and pepper ghosts
    - AR and VR
    - Interactive kiosks
    - Interactive touch screens with educational games
- 3D models of the museum artifacts.
- Cross platform, immersive experience in mix reality telling the story of the museum through time.

Collaboration project example 5, Junior University

- Study project.
- An event for high school students.
- Estimated 700 high school students come to the Lahti campuses for the whole day to participate in workshops of their choice, solving the challenges presented in these workshops.
- In the fictional city of Poukama, there has been a storm, and the focus is on how to cope with it and address the damage caused by the storm in a multidisciplinary manner.
- Analyze data: Human Mobility During Natural Disasters

- Create sensor applications from material found from your own home.

Collaboration project example 6, Software start-up company

- Study project.
- Helping hands on starting a company and getting first projects started.

Collaboration project example 7, TOIVE project

- RDI project.
- Mentoring network for promoting women and non-binary individuals in IT, gaming, and e-sports.
- Company and organization representatives as mentors
- Platform in Discord
- Towards equal working life -course starting in January
- In English
- Participants from Lahti area

Collaboration project example 8, Supporting students' own companies or freelance work.

- Student's own company ideas. E.g. Thesis on creating an AI aid for UX designing for a freelance work.

8 CONCLUSION

This research has demonstrated benefits of user story-based teaching for programming students at LAB University of Applied Sciences. The primary objective was to enhance programming education to better serve engineering students through the implementation of a research-based development model. The preplanned construction, tested with first-year Finnish and international engineering student group, has shown promising results and potential for application in advanced engineering courses.

The integration of conceptual models from previous studies, combined with practical testing and critical evaluation, has led to the refinement of the constructed models for the upcoming courses. The constructive research process, aimed at solving real-life challenges, has proven effective in creating and implementing new constructions in relevant environments. Feedback from ICT professionals and students, gathered through questionnaires, has been used in this process.

Key findings from the study indicate that user story-based learning enhances students' professional identity, motivation, and programming skills at their own level. This approach has also attracted interest from companies and organizations, highlighting the potential for collaborations.

The transition from project-based learning to user story-based learning, involving real-life customers, ICT companies, and various organizations, aims to enhance students' educational experiences. This method focuses on iterative development driven by user stories, emphasizing understanding and meeting user needs through continuous feedback and refinement. It aligns with user-centered design and Agile methodologies, preparing students for real-world tech industry challenges. The goal is to apply this approach across all programming courses, at every level, so all students can benefit.

The transition from project-based learning to user story-based learning, involving real-life customers, ICT companies, and various organizations, aims to enhance students' educational experiences. This method focuses on iterative development driven by user stories, emphasizing understanding and meeting user needs through continuous feedback and refinement. It aligns

with user-centered design and Agile methodologies, preparing students for real-world tech industry challenges.

It is clear by the results of the project, the first semester first-year students are not yet ready to work with real-life customers, so using a hypothetical customer and user story will continue to be the plan for these introductory courses. At the same time, students receive guidance on future study projects, training, internships, and job. The employment services of the city of Lahti were actively involved with the students in this course. This was well received addition and hopefully we will be able to continue this collaboration. Real-life customer involvement is for later, advanced programming and ICT engineering courses. The goal is to apply this approach across all programming courses, at every level, so all students can benefit.

In conclusion, the research underscores the value of user story-based learning in preparing students for the demands of the modern workplace. By fostering practical skills, professional growth, and industry connections, this educational model offers a comprehensive framework for developing well-rounded, job-ready graduates. While the results are promising, further research with real customers is necessary to validate these findings. Additionally, this teaching method requires significant organization and coordination with actual customers and collaborative organizations, making it a challenging, but rewarding, approach for educators.

REFERENCES

Cohn Mike 2004, User stories Applied: For Agile Software Development, [e-book]. Boston: Addison-Wesley. O'Reilly Online Learning: Academic/Public Library.

Eskola, Jari & Suoranta Juha 1998. Johdatus laadulliseen tutkimukseen, [e-book]. Tampere: Vastapaino. Ellibs Library.

Jyrkkä, Kari & Niemi Eino 2020. Ohjelmointitaito opitaan projektissa. Search date 7.7.2024. https://www.theseus.fi/bitstream/handle/10024/333392/ePooki%207_2020.pdf?sequence=2&isAllowed=y

Kinnunen Päivi 2009. Challenges of teaching and studying programming at a university of technology – viewpoints of students, teachers and the university. Doctoral Dissertation. Search date 6.7.2024. https://aaltodoc.aalto.fi/server/api/core/bitstreams/98196f2a-ed2d-4f3d-91c9-4eb6eb85eb02/content

Kulangara, Kiran Joy 2023. Designing and Building a Platform for Teaching Introductory Programming Supported by Large Language Models. Search date 9.7.2024 https://aaltodoc.aalto.fi/server/api/core/bitstreams/2eae5618-109d-43b9-9c74-38119cf3e593/content

Körkkö, Sebastian 2016 Ohjelmoinnin oppimisen haasteet ja ratkaisut. Search date 8.7.2024 https://oulurepo.oulu.fi/bitstream/handle/10024/7162/nbnfioulu-201605282035.pdf?sequence=1&isAllowed=y

Lehtinen, Teemu 2024 Questions About Learners' Code: Extending Automated Assessment Towards Program Comprehension. Search date 30.7.2024 https://aaltodoc.aalto.fi/server/api/core/bitstreams/e3c3b48e-9e1f-4fbf-a5a5-8045328171a0/content

Lukka, Kari 2006. Konstruktiivinen tutkimusote: luonne, prosessi ja arviointi. Teoksessa Soveltava yhteiskuntatiede ja filosofia (toim. Kristiina Rolin, Marja-Liisa Kakkuri-Knuutila & Elina Henttonen). Helsinki: Gaudeamus, 111-133.

Lukkarinen, Aleksi 2022. Teaching Event-driven Programming for Novice Programmers: Challenges and Advantages. Search date 29.7.2024 https://aaltodoc.aalto.fi/server/api/core/bitstreams/f0238ab9-0e63-46d5-bfa4-ae38e3b864be/content

Patton Jeff & Economy Peter 2014, User story mapping: discover the whole story, build the right product, [e-book]. California: O'Reilly. O'Reilly Online Learning: Academic/Public library.

Sippola, Julia Tytti Karoliina 2020. Katsaus ohjelmoinnin opetustyyleihin ja motivointiin. Search date: 8.7.2024 https://jyx.jyu.fi/bitstream/handle/123456789/68969/URN%3aNBN%3afi%3ajyu-202005133176.pdf?sequence=1&isAllowed=y

Toikko, Timo & Rantanen, Teemu 2009. Tutkimuksellinen kehittämistoiminta: Näkökulmia kehittämisprosessiin, osallistamiseen ja tiedontuotantoon. Search date 1.7.2024 https://trepo.tuni.fi/bitstream/handle/10024/100802/Toikko_Rantanen_Tutkimuksellinen_kehittamistoiminta.pdf?sequence=1&isAllowed=y

Vihavainen, Arto, Paksula Matti & Luukkainen Matti. Extreme Apprenticeship Method in Teaching Programming for Beginners. Search date 20.6.2024 https://www.researchgate.net/publication/228796958_Extreme_apprenticeship_method_in_teaching_programming_for_beginners. 93-98.

APPENDICES

Appendix 1 Questionnaire for the professionals
Appendix 2 Questionnaire for the students

Palaute ohjelmoinnin opetuksen suunnitelmasta

Projektiopiskelun ja kisällioppimisen menetelmiä korkeakoulujen ohjelmoinninopetuksessa on paljon tutkittu. Menetelmillä on pyritty korjaamaan ohjelmoinnin opetukseen liittyviä haasteita, kuten opiskelijoiden motivaatiota ja ammatillisen identiteetin kehittymistä. Projektiopiskelu ja kisällioppiminen on käytännön opetuksessa todettu tehokkaaksi.

Tämä tutkimus perustuu näihin aikaisempiin tutkimuksiin ja menetelmiin. Tutkimuksen tavoitteena on selvittää, voidaanko tutkittuja menetelmiä tehostaa painottamalla käyttäjätarinaa (User Story) ohjelmointiprojekteissa. Tavoitteena on opettaa opiskelijoita tuntemaan asiakkaansa ja ohjelmoimaan asiakaslähtöisesti.

Käyttäjätarinalähtöistä ohjelmoinnin opetusta harjoitellaan ensimmäisestä, ohjelmoinnin perusteet, kurssista lähtien.

1. Opiskelijoita valmennetaan ensimmäisestä kurssista lähtien asiakaslähtöisesti, oikeilla ympäristöillä ja työskentelemään oikeissa projekteissa. Projekteissa on aina yritys tai organisaatio asiakkaana. Opiskelija oppii alusta alkaen toimimaan yhdessä asiakkaan kanssa. Hän saa tuoteomistajalta käyttäjätarinan, rakentaa sen perusteella tuotteen kehitysjonon yhdessä kehitystiimin kanssa, suunnittelee sprintit ja ohjelmoi toimivan tuotteen. Opiskelijaa opastetaan toimimaan sprinttikatselmuksissa, työskentelemään asiantuntijana ja julkaisemaan tuote asiakkaan tarpeiden mukaisesti.

   Toimintamalli vaatii opiskelijalta alussa enemmän aikaa käytäntöjen ja ympäristöjen opiskeluun. Opettajan tehtävänä on ensin toimia erityisen vahvasti opiskelijan apuna ja selkänojana ja vähitellen antaa opiskelijalle tilaa hänen kehittyessään oman alansa ammattilaiseksi. Opettajalta vaaditaan paljon yhteistyökumppanien etsimisessä ja oppimisprosessin kannalta oikeanlaisten projektien kartoittamisessa.

   Asiakkaana voi toimia alan yritykset, eri alojen ohjelmointi-/ ICT-osastot tai monien eri alojen asiakkaat. Opiskelijat eivät saa projekteista rahallista palkkaa, sen sijaan he saavat harjoitusta ja arvokkaita kontakteja työelämään. Ohjelmointialan yrityksissä toimitaan aina yrityksen projektihallintamenetelmien mukaisesti.

   Mitä haasteita / mahdollisuuksia näet tällaisessa työskentelytavassa yhteistyöyrityksen / asiakkaan näkökulmasta?

   Mitä opetuksessa pitäisi ottaa yrityksen näkökulmasta huomioon?

2. Käyttäjätarinalähtöisessä opetuksessa harjoitellaan tiimissä toimimisen periaatteita, sosiaalisia taitoja ja tiimityötaitoja. Tarkoituksena on kehittää

opiskelijoiden ammatti-identiteettiä, motivaatiota ja ammatillista osaamista.

Miten kehittäisit käyttäjätarinapohjaista opetusta ammattilaisen näkökulmasta?

3. Kurssit harjoittavat ja testaavat, opettajan ohjaamana, opiskelijan sinnikkyyttä, järjestelmällisyyttä, paineensietokykyä, ongelmanratkaisutaitoja, tiedonhakutaitoja, mukautumiskykyä muuttuvissa ja vaihtelevissa tilanteissa sekä ohjelmoinnillista ajattelua. Tavoitteena on myös antaa aito kuva työskentelystä asiakasprojekteissa turvallisesti ilman tulosvastuuta. Projekteja tehdään tiimeissä koululla. Tarvittaessa esimerkiksi katselmoinnit voidaan tehdä yrityksen omissa tiloissa. Koulu tarjoaa yleisimmät ympäristöt ohjelmoinnin kehitysprojekteihin.

   Millaiset projektit mielestäsi kehittäisivät parhaiten opiskelijan työelämässä tarvittavia ominaisuuksia parhaiten?

4. Toimitko

   - Esimiestehtävissä
   - Järjestelmäarkkitehtina
   - Konsulttina
   - Ohjelmointitehtävissä
   - Opettajana / kouluttajana
   - Pilvipalveluiden asiantuntijana
   - Teknisenä myyjänä
   - Tietoturva-asiantuntija
   - Tuoteomistajana
   - Yrittäjänä
   - Muu, mikä

5. Mikä on oma koulutuksesi?

   - Tekniikan ammattikoulututkinto
   - Tekniikan ammattikorkeakoulututkinto
   - Tekniikan ylempi korkeakoulututkinto (maisteri, DI)
   - Tekniikan lisensiaatti / tohtori
   - Muu, mikä

6. Kuinka kauan olet toiminut ICT alan tehtävissä

   - 0-5 vuotta
   - 6-20 vuotta
   - Yli 20 vuotta
   - Muu, mikä

Introduction to programming course, evaluation survey

The course assignments practiced and tested your perseverance, stress tolerance, working under pressure, teamwork skills, problem-solving skills, independent information retrieval skills, collegiality, asking for and giving help, adaptability in changing and varying situations, computational thinking and working in real life customer projects. Well done, you did great!

1. What is the significance of a Product Owner in programming projects?

2. What is the significance of User Stories in programming?

3. What is the significance of a Product Backlog in programming?

4. What did you find motivating in the course? When did the work feel enjoyable? What practices from the project did you find useful in future exercises, projects, or internships?

5. What challenges did you face in the project? Were they related to, for example, endurance, motivation, or technical skills? How did you overcome the challenges?

6. How motivating do you find working as a programmer at the moment?

    - Very interesting
    - Definitely an option
    - Not a relevant question for me right now
    - Primarily pursuing other options
    - Definitely not my interest
    - Other

7. Do you feel like you got a realistic experience working as a programmer?

    - Yes
    - No
    - Maybe
    - Other

8. How has your understanding improved regarding:

	A lot	Some	Not much	Not at all
Working as an engineer?	○	○	○	○
Working as a programmer?	○	○	○	○
Working in a developer team?	○	○	○	○
Meeting customer needs?	○	○	○	○
The roles of different participants in a project?	○	○	○	○

The different tasks involved in a project?	○	○	○	○

9. How would you describe your professional identity before starting this course?

   What were your initial thoughts and feelings about your role in IIT engineering?
   How confident did you feel about your skills and knowledge in IIT engineering?

10. How has your professional identity evolved after completing this course?

    In what ways have your thoughts and feelings about your role in IIT engineering changed?
    How has your confidence in your skills and knowledge improved?

11. On a scale of 1 to 10, what was your skill level at the beginning of the course regarding the subjects taught in the course?

    1 2 3 4 5 6 7 8 9 10
    Not at all good Extremely good

12. On a scale of 1 to 10, what was your skill level at the end of the course regarding the subjects taught in the course?

    1 2 3 4 5 6 7 8 9 10
    Not at all good Extremely good