Slovak University of Technology
Faculty of Informatics and Information Technologies

FIIT-XXXXX-116308

Bc. Matej Ševčík

E-Learning SW modeling using a Class Diagram

Master's Thesis

Supervisor: Ing. Vladimír Mlynarovič, PhD.

May 2026

Slovak University of Technology
Faculty of Informatics and Information Technologies

FIIT-XXXXX-116308

Bc. Matej Ševčík

E-Learning SW modeling using a Class Diagram

Master's Thesis

Study programme:	Intelligent Software Systems
Study field:	Informatics
Training workplace:	Institute of Informatics, Information Systems and Software Engineering
Supervisor:	Ing. Vladimír Mlynarovič, PhD.

May 2026

I declare in my honor that I have prepared this work independently, on the basis of consultations and using the mentioned literature.

In Bratislava, May 2026 Bc. Matej Ševčík

Annotation

Slovak University of Technology Bratislava
Faculty of Informatics and Information Technologies
Degree Course: Intelligent Software Systems

Author: Bc. Matej Ševčík
Diploma Thesis: E-Learning SW modeling using a Class Diagram
Supervisor: Ing. Vladimír Mlynarovič, PhD.
May 2026

E-learning is a perspective field that offers many opportunities to support education. It has the potential to increase the effectiveness of learning, reduce the burden on educators, make education more accessible to the general public, and introduce innovative practices that traditional educational approaches do not allow. Despite this, e-learning is used relatively rarely in Slovak schools, even at faculties of informatics. In this project, we focused on the possibilities of applying e-learning to teaching class diagrams, with the potential for expansion to other UML diagrams. We analyzed various techniques used in e-learning and their application in teaching software modeling. The main goal of this project is to create a tool for teaching class diagrams. We applied practices that, according to our analysis, achieved the best results, and supplemented them with our own ideas and adapted them to the needs of the platform. We focused on testing the knowledge of students who used our tool. The main contribution of the project is that our tool, along with the analysis and tests, has created a foundation for further research and improvement of the application of e-learning in the field of software modeling.

Keywords: e-learning, class diagram, software modeling

Anotácia

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Študijný program: Intelligent Software Systems

Autor: Bc. Matej Ševčík
Diplomová práca: E-Learning modelovania SW pomocou diagramu tried
Vedúci diplomového projektu: Ing. Vladimír Mlynarovič, PhD.
May 2026

E-learning je rýchlo sa rozvíjajúca oblasť poskytujúca veľa možností podpory vzdelávania. Má potenciál zvýšiť efektivitu vzdelávania, znížiť záťaž na pedagógov, sprístupniť vzdelávanie širšej verejnosti alebo zaviesť inovatívne praktiky, ktoré tradičné vzdelávacie prístupy neumožňujú. Napriek tomu je na slovenských školách e-learning využívaný pomerne zriedkavo, a to aj na fakulte informatiky. V projekte sme sa zamerali na možnosti aplikovania e-learningu pri vyučovaní diagramu tried s možnosťou rozšírenia na ďalšie UML diagramy. Vykonali sme analýzu rôznych prístupov používaných v e-learningu a ich využitia pri vyučovaní modelovania softvéru. Hlavným cieľom tohto projektu je vytvorenie nástroja na výučbu diagramu tried. Aplikovali sme pri tom praktiky, ktoré podľa analýzy v predchádzajúcich experimentoch dosiahli najlepšie výsledky, pričom sme ich doplnili o vlastné nápady a prispôsobili potrebám nášho nástroja. Dôraz sme kládli na testovanie vedomostí študentov, ktorí náš nástroj využívali. Prínos projektu spočíva najmä v tom, že náš nástroj, analýza a testy pripravili priestor na ďalšie rozširovanie a vylepšovanie možností e-learningu v oblasti modelovania softvéru.

Kľúčové slová: e-learning, diagram tried, modelovanie softvéru

SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce [1]

Študent:

Meno, priezvisko, tituly: Matej Ševčík, Bc.
Študijný program: Inteligentné softvérové systémy
Kontakt: xsevcikm3@stuba.sk

Výskumník:

Meno, priezvisko, tituly: Vladimír Mlynarovič, Ing. PhD.

Projekt:

Názov:	E-Learning modelovania SW pomocou diagramu tried
Názov v angličtine:	E-Learning SW modeling using a class diagram
Miesto vypracovania:	Ústav informatiky, informačných systémov a softvérového inžinierstva, FIIT STU
Oblasť problematiky:	e-learning, diagram tried, modelovanie softvéru

Text návrhu zadania[2]

Diagram tried je najpoužívanejším UML diagramom. Slúži na grafické zobrazenie štruktúry systému pomocou tried, ich atribútov, metód a vzťahov medzi nimi. Tento diagram je významným nástrojom pri analýze, návrhu a dokumentácii softvérových systémov. E-learning je rýchlo sa rozvíjajúca oblasť poskytujúca veľa možností podpory vzdelávania. Má potenciál zvýšiť efektivitu vzdelávania, znížiť záťaž na pedagógov alebo sprístupniť vzdelávanie širšej verejnosti. V štúdiách [1][2] študenti, ktorí sa učili modelovať diagram tried pomocou navrhovaného e-learning modulu, dosiahli v cvičeniach a neskôr aj v testoch lepšie výsledky než skupina, ktorá strávila rovnaký čas štúdiom diagramu tried z iných zdrojov. To demonštruje potenciál e-learningu zvýšiť kvalitu výučby v tejto oblasti. Výskumným problémom je teda optimalizácia výučby modelovania softvéru s využitím e-learningových prístupov, pričom hlavnou otázkou je, ktoré metódy sú najefektívnejšie a ako ich vhodne kombinovať.

Analyzujte existujúce e-learningové prístupy vo vzdelávaní softvérového inžinierstva so zameraním na výučbu UML diagramov. Navrhnite vzdelávací nástroj, ktorý implementuje poznatky z tejto analýzy a sprístupnite ho v rámci prototypu novej e-learningovej platformy, prípadne ako modul do už existujúcej platformy. Otestujte efektivitu nástroja na študentoch bakalárskeho štúdia na FIIT STU pomocou testov. Porovnajte výsledky študentov, ktorí využili na prípravu nami navrhnutý nástroj a tých, ktorí sa učili z iných zdrojov. Prínos projektu bude spočívať najmä v tom, že náš nástroj, analýza a testy pripravia priestor na ďalšie rozširovanie a vylepšovanie možností e-learningu v oblasti modelovania softvéru.

---

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Acknowledgements

I would like to thank Ing. Vladimír Mlynarovič, PhD. for supervising this work.

Contents

Chapter 1

Introduction

Unified Modeling Language (UML) is a widely used modeling language. It is used for describing systems in an abstract way and from various perspectives, which makes the ability to understand it and use it a valuable asset for any professional in the field of information technologies.

Most developers claim to possess knowledge of UML. However, in many cases, their knowledge of the topic is rather limited or even inaccurate[1]. This observation highlights the need for improving the approach to teaching of software modeling and UML in educational institutions.

Traditional classes often do not provide enough time and space for an actual teacher to give the necessary amount of attention to each of the students. The answer to this problem might lie in electronic learning, or e-learning. It is a practice in which learning material is delivered through digital technologies.

E-learning can offer students interactive materials with automated immediate personalized feedback, structured and easily accessible learning materials or study and exercise recommendations, while simultaneously allowing teachers to spend more time consulting students instead of explaining the basic concepts to them.

E-learning also provides possibilities for increasing students' engagement in the study materials by introducing gamification elements or a sense of competition to the courses.

The class diagram is the most prominent diagram in UML. It is also common for students to have trouble using it properly. For these reasons, we decided to aim this study at designing an e-learning course that would act as complementary material to a software

[1] Donald Bell, "An Introduction to the Unified Modeling Language," *IBM Developer*, https://developer.ibm.com/articles/an-introduction-to-uml/, accessed February 23, 2025.

modeling university subject.

Our course will implement a number of modern e-learning approaches to improve information retention, engagement, and the overall learning experience of the students. It will be tested on a sample of students at Faculty of Informatics and Information Technologies STU in Bratislava (FIIT STU). The results will then be analyzed to determine the effectiveness and further potential of such tools in the process of education of software modeling.

Chapter 2

The Class Diagram

2.1 Overview of UML

Introduction

The UML is a standardized visual modeling language. Its first version was published in 1997 and later updated to UML 2.0 in 2005 as an attempt to unify various approaches to modeling software systems
footnoteVisual Paradigm, "What is Unified Modeling Language (UML)?," *Visual Paradigm*, https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/, accessed February 23, 2025.. The latest version of UML is UML 2.5.1 and it was published in December 2017[1].

The purpose of the language is to specify, document, and visualize parts of software systems. That is accomplished by designing standardized diagrams, each intended to describe a different aspect of the system, usually from various stakeholders' perspectives. Those may include the architect, programmer, project manager, customer, and others.

UML models contain models of business processes, interactions and relationships between individual components or the structure and functions of components. UML documents all stages of the development of a system, not only the initial part. It is mainly aimed at, but not limited to, object-oriented systems.

A model is a representation of something - in this case, a software system. A model captures the system from a certain viewpoint and omits its aspects that are not relevant to the viewpoint [1].

[1]Conrad Bock et al., "Unified Modeling Language," *Object Management Group (OMG)*, https://www.omg.org/spec/UML/, accessed February 23, 2025.

UML diagrams

UML defines a large number of elements that build up diagrams. There are in total 13 different diagrams defined. Those diagrams represent various aspects and viewpoints of the system. In some sources, 14 diagrams are mentioned. The profile diagram is usually not considered to be a separate diagram type however.

Some diagrams show the structure of the system. Those are called structural diagrams. They do not display the behavior of the modeled elements [2]. They describe the system using objects, attributes, operations and relationships.

The other type of diagrams are behavioral diagrams. They describe how the objects interact and change during runtime. They use sequence, activity, collaboration, and state to represent these changes[2].

The structural UML diagrams are:

- object diagram
- package diagram
- class diagram
- component diagram
- composition structure diagram
- deployment diagram

The behavioral diagrams are:

- state machine diagram
- use case diagram
- activity diagram
- interaction overview diagram
- sequence diagram
- communication diagram
- timing diagram

[2]Visual Paradigm, "What is Unified Modeling Language (UML)? (Behavior vs Structural Diagram)," *Visual Paradigm*, https://www.visual-paradigm.com/guide/uml-unified-modeling-language/behavior-vs-structural-diagram/, accessed April 13, 2025.

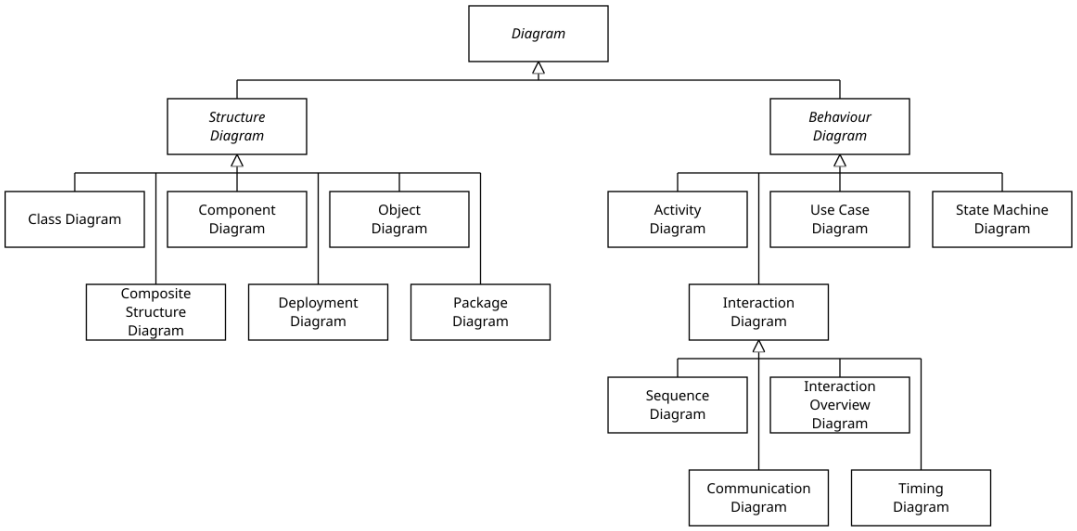

Figure 2.1: Hierarchy of UML diagrams

Source: Dave A. Ryan, "UML diagram," *Wikimedia Commons*, `https://en.m.wikipedia.org/wiki/File:Uml_diagram.svg`, Accessed: May 23, 2025.

2.2 UML modeling tools[3]

There are many tools for creating UML diagrams available. They differ in complexity, support of features, price or intended use. We selected some of the most popular ones for closer inspection and compared them.

StarUML[4]

StarUML is a fast, lightweight UML tool geared toward developers. It supports modern UML standards and allows extension via plugins. StarUML is widely appreciated for its clean interface and Markdown-compatible documentation features.

Enterprise Architect[5]

Enterprise Architect by Sparx Systems is a powerful, professional-grade modeling tool. It supports UML, SysML, BPMN, ArchiMate, and more. EA is often used in large-scale software development and system engineering projects.

Diagrams.net[6]

[3] This section was made with the use of Generative artificial intelligence (Gen AI)

[4] "StarUML," *StarUML Official Website*, https://staruml.io/, accessed May 24, 2025.

[5] "Enterprise Architect," *Sparx Systems*, https://sparxsystems.com/products/ea/, accessed May 24, 2025.

[6] "Diagrams.net (Draw.io)," *Diagrams.net*, https://www.diagrams.net/, accessed May 24, 2025.

Diagrams.net, also known as Draw.io, is a free, browser-based diagramming tool. While it's not strictly tailored for UML, it offers sufficient shapes and templates to support simple to moderately complex UML diagrams. It is therefore mainly used for making quick sketches or by students.

Visual Paradigm[7]

Visual Paradigm is a comprehensive UML and software design tool. It supports all UML diagram types and extends into system modeling, business process modeling, and database design. Visual Paradigm is favored in both academia and industry for its clarity, template variety, and export options. Visual Paradigm also provides a simpler online diagram editing tool that we used for designing diagrams in our study.

Lucidchart[8]

Lucidchart is an online diagramming application with strong collaboration features. It supports UML and other diagram types like flowcharts, ERDs, and wireframes. It is widely used in business and education settings.

Overall comparison

Tool	StarUML	Enterprise Architect	Diagrams.net
UML Support	Full	Full	Partial (manual)
Platform	Win/macOS/Linux	Windows Desktop	Web/Desktop
License	Paid (Low-cost)	Paid (Proprietary)	Free (Open Source)
Collaboration	No	Yes	Limited
Code Engineering	Yes	Yes	No
Templates	Moderate	Rich	Limited
Best Use Case	Developers	Enterprise Systems	Quick sketches

Table 2.1: Comparison of UML tools: StarUML, Enterprise Architect, Diagrams.net

Tool	Visual Paradigm	Lucidchart
UML Support	Full	Moderate
Platform	Web/Desktop	Web
License	Free + Paid tiers	Free + Paid tiers
Collaboration	Yes	Real-time
Code Engineering	Yes	No
Templates	Rich	Rich
Best Use Case	Academic / Professional	Collaborative Teams

Table 2.2: Comparison of UML tools: Visual Paradigm and Lucidchart

[7] "Visual Paradigm," *Visual Paradigm*, https://www.visual-paradigm.com/, accessed May 24, 2025.
[8] "Lucidchart," *Lucidchart*, https://www.lucidchart.com/, accessed May 24, 2025.

2.3 Class Diagram

The class diagram is the most prominent UML diagram. It describes the elements of the system as well as their relationships, creating the basis for numerous other diagrams and modeling techniques [3]. These aspects are static, meaning that they do not change with time. Despite that, it is used in multiple phases of system development, applying various levels of abstraction. It is even possible to generate code from the class diagram. It is commonly used as a part of documentation [2], as it provides an easy to understand overview of the system.

2.3.1 Perspecitves of the Class Diagram[3]

Class diagrams are one of the most fundamental diagram types in the Unified Modeling Language (UML) and serve various purposes throughout the software development lifecycle. They apply various levels of abstraction and each level of abstraction uses class diagrams slightly differently, but they all help bridge the gap between stakeholder understanding and technical implementation.

Domain Model At the highest level of abstraction, class diagrams are used to model concepts from the problem domain. These diagrams represent real-world entities and their relationships, often using language familiar to stakeholders. The focus here is on understanding and communicating the essential structure of the domain.

Logical Model As the design progresses, the domain model is refined into a logical model. This level introduces more technical detail, such as class responsibilities, access modifiers, and associations with navigability or multiplicity. It provides a blueprint for the software's internal design while remaining independent of specific implementation technologies.

Physical Model Eventually, the logical model is translated into a physical model, where class diagrams may reflect actual code structures, database schemas, or deployment components. At this stage, technical constraints and implementation details (such as data types or platform-specific features) are included.

Class diagrams also share conceptual similarities with Entity-Relationship (ER) diagrams, which are commonly used in database design. In fact, UML class diagrams can serve as a richer alternative to ER diagrams, especially when modeling object-oriented systems:

- Entities in ER diagrams correspond to classes in UML
- Attributes are represented similarly
- Relationships in ER diagrams map to associations, aggregations, or compositions in class diagrams, often with multiplicity annotations

- UML also supports advanced features such as inheritance and method specification, which are typically absent in ER diagrams

2.3.2 Components of Class Diagrams

A class diagram consists of multiple components. In broad terms, those components are classes, their attributes and operations, and the relationships between different classes.

Classes and objects are closely related. A class is a blueprint for creating objects - it describes what attributes and operations an object is going to have and how it is going to interact with other entities.

An object is an instance of a class. For example, a class Table is an abstract idea that is used for creating real tables. A table in the local library is an object - a specific instance of a table that follows the specifications described in the class Table.

Attributes

Attributes describe the properties of a class. A class defines the list of attributes along with their data types and visibility. The class may also define an attribute's multiplicity - the number of values that are to be assigned to it. The default amount is 1. In the case of an object, the attributes are assigned specific values.

Operations

Operations are activities that instances of a class can perform. They include things like performing calculations or altering a value. A class describes the visibility , input parameters, return type, and the name of the operation.

Data types

Data types define what kind of information can be stored in a variable (or returned by an operation). Examples are integer, character, string, or even an instance of another class.

Visibility

Visibility sets which entities are allowed to view and interact with an attribute or an operation. There are four types of visibility - public, protected, default and private.

Modifiers[3]

Modifiers in UML class diagrams provide additional information about the behavior and constraints of classes, attributes, and operations. Common modifiers include abstract, which indicates that a class or method cannot be instantiated or directly invoked and must be extended or overridden; final, which marks a class as not extendable or

a method as not overridable; and static, which signifies that an attribute or operation belongs to the class itself rather than to any instance of it.

UML also supports the interface modifier to represent interfaces—types that declare operations without implementing them, which classes can then realize. These modifiers help clarify the intended usage, inheritance, and design principles of class components in object-oriented modeling.

Example of a class[3]

The following figure displays an example of a class from a class diagram. It models a simplified bank account that has an identifier, balance and interest rate, allowing a potential owner to deposit or withdraw money and view the current balance of the account. It demonstrates the key features of a class, such as attributes, operations, types, operation parameters or access modifiers.

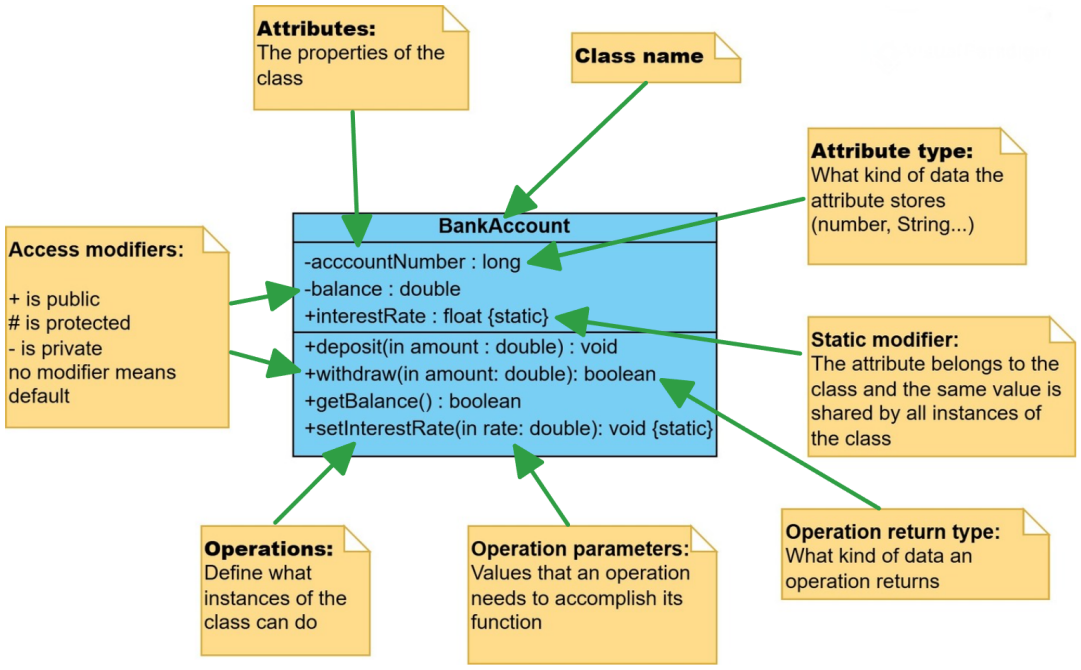

Figure 2.2: An example of a class

Made with: "Visual Paradigm," *Visual Paradigm Online*, https://online.visual-paradigm.com/, accessed May 23, 2025.

2.4 Relationships[3]

The class diagram defines several types of relationships between different classes, each representing a distinct kind of interaction or connection within a system.

Association

Association represents a general connection between two or more classes. It indicates that objects of one class are connected to objects of another. This is the most basic relationship and can be bidirectional or unidirectional.

Aggregation

Aggregation is a specialized form of association that represents a whole–part relationship between the aggregate (whole) and its parts. Unlike composition, aggregation implies a weaker relationship where the parts can exist independently of the whole.

Composition

Composition is a stronger form of aggregation. It also represents a whole–part relationship, but with the key difference that the part cannot exist independently of the whole. If the whole is destroyed, its parts are typically destroyed as well.

Inheritance

Inheritance (also known as generalization) represents an ”is-a” relationship between a more general superclass and a more specific subclass. The subclass inherits the attributes and behaviors of the superclass, allowing for code reuse and polymorphism.

Dependency

Dependency indicates a relationship where one class depends on another because it uses it temporarily, such as by calling its methods or using its data. It is typically a weaker relationship and often represents a ”uses-a” connection.

Realization

Realization is a relationship between an interface and a class that implements that interface. It shows that the class provides concrete implementations for the methods defined in the interface.

Cardinality

Cardinality tells how many instances of a class are to be associated with an instance of another class. The most common cardinalities are one-to-one, one-to-many or many-to-many.

2.5 Example of a Class Diagram[3]

The following figure displays an example class diagram. It represents a simplified model of an eshop, where customers place orders and admins manage product catalogs. Orders contain multiple items and can be paid using various payment methods.

It demonstrates many of the essential concepts of a class diagram, mainly inheritance, aggregation, composition, association, multiplicity, realization, and classes with their attributes and operations.

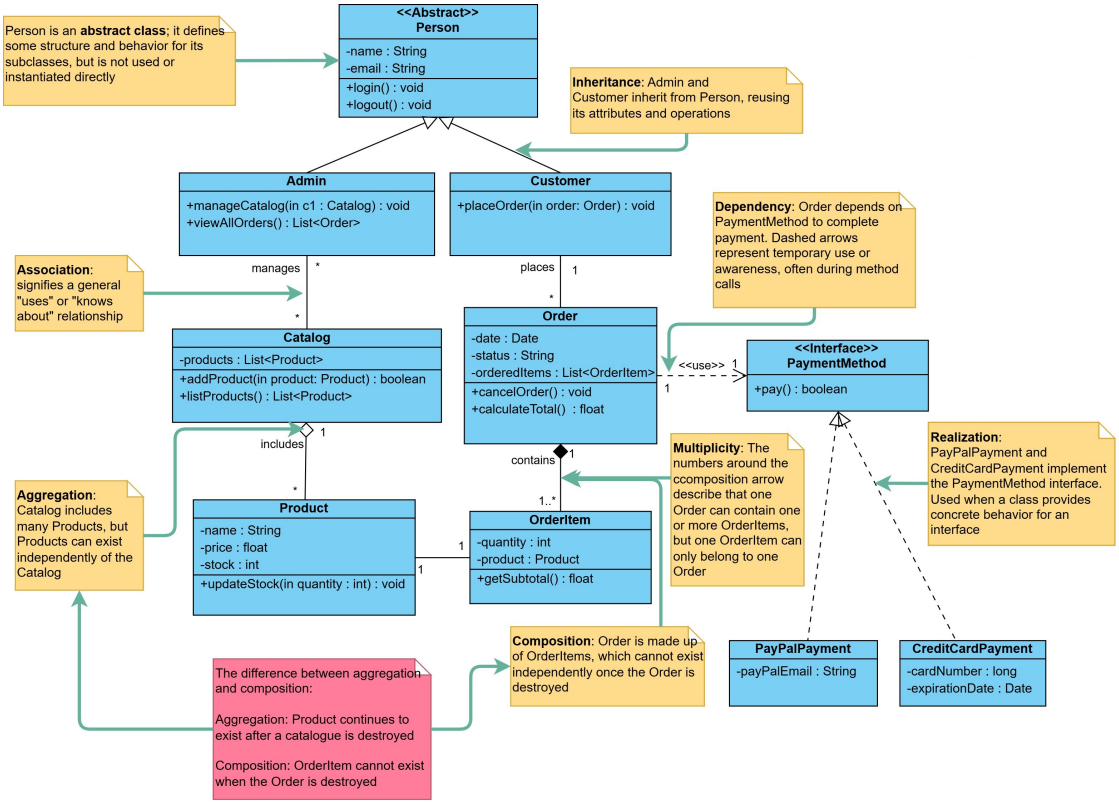

Figure 2.3: An example of a class diagram.

Made with: Visual Paradigm, *Visual Paradigm Online*, `https://online.visual-paradigm.com/`, accessed May 23, 2025.

Chapter 3

E-learning and the UML Class Diagram

3.1 Introduction to E-learning[1]

E-learning, or electronic learning, is a modern approach to acquiring knowledge and skills through digital technologies. It is a rapidly growing field that offers a wide range of possibilities for making education more effective, accessible, and flexible. With the help of the internet, multimedia tools, and interactive systems, learners can study anytime and from anywhere, overcoming many of the limitations of traditional classroom education.

E-learning is used in various forms—ranging from simple online courses and video lectures to complex platforms that offer feedback, progress tracking, and personalized learning experiences. It often complements face-to-face teaching in a blended learning model, which combines the strengths of both in-person and digital instruction.

Among the key benefits of e-learning are improved accessibility, self-paced learning, and the potential to tailor educational content to individual needs. However, it also faces challenges such as reduced social interaction, learner isolation, and varying levels of digital literacy. Despite these issues, the field continues to evolve, with current trends focusing on practices like mobile learning, gamification, microlearning, and adaptive learning technologies.

As digital tools become more widespread and learning needs become increasingly diverse, e-learning is likely to play an even more significant role in the future of education.

[1]This section was made with the use of Gen AI

3.2 Comparison of E-learning and traditional learning

Traditional learning refers to the model where a teacher or a tutor presents information and exercises to a group of people in a classroom. This model is typical of educational institutions such as universities. This approach has been utilized for a very long time and it is the standard way of teaching.

E-learning offers a number of improvements over the traditional model. Unlike traditional learning, online courses can be taken from anywhere and at any time, removing much of the geographic and time constraints for the learners. It is also significantly cheaper, since it removes the need for a teacher or a learning space. With interactive exercises and materials, e-learning can increase student engagement and interest in the topic.

On the other hand, it also presents a set of challenges that make traditional learning still the preferable choice in most cases. The most obvious drawback is that the self-directed nature of e-learning requires a level of discipline from the learners which many lack. Other problems include a lack of social interactions with other students, a lack of personalized feedback from a live instructor and the fact that some people do not have access to the technology necessary to utilize e-learning materials. For some people, digital literacy may also be a problem stopping them from using e-learning resources.

In a study performed by Ramle et al.[4] a UML course was introduced. Students that took the course were tasked by filling out a questionnaire about their experience. The reception was mostly positive, with students making claims such as *"makes me understand more"* or *"makes you get better at UML."*

Jurgelaitis et al.[5] found that the introduction of gamified mechanics to a university UML course increased both students' motivation and their average scores. This confirms the general expectations from other works that have not yet collected the necessary data for evaluation.

Lastly, Soler et al.[6] proposed a tool for automated correction of class diagram exercises. The researchers conducted an experiment where students were divided into 2 equally sized groups: one solved a set of exercises using the tool, and the other solved them manually. The number of students who solved all of the exercises correctly was higher in the group that used the tool, and this group also had better average score in a subsequent exam.

These studies clearly demonstrate that tools can be developed to increase not only the students' knowledge, but also their engagement and overall satisfaction with their educational process.

3.3 Techniques in E-learning[1]

There are many techniques and practices that can be used to enhance the learning process when it comes to e-learning materials. Some of them are aimed at increasing the learner's motivation and interest (gamification), some directly help improve and better organize the study materials (microlearning, adaptive learning) and some of them are for convenience (self-directed learning). All of them have their use and when applied correctly, they can have a positive impact on the overall learning experience of the learners.

Gamification

Gamification involves integrating game-like elements into the learning process—such as points, levels, badges, leaderboards, and challenges—to increase engagement and motivation. By leveraging learners' natural desire for competition and achievement, gamification can make education more interactive and enjoyable, especially in areas that may otherwise seem abstract or technical.

Microlearning

Microlearning is based on delivering content in small, easily digestible chunks. Each module typically focuses on a single concept or skill and can be completed in a short amount of time. This approach is well-suited to modern learners with limited attention spans or tight schedules, and it allows for quick revisions and focused repetition.

Self-directed Learning

Self-directed learning empowers students to take control of their learning paths by setting their own goals, choosing resources, and pacing their progress. While this method requires a high level of motivation and discipline, it fosters independence and lifelong learning skills. E-learning platforms often support SDL through customizable paths and open access to resources.

Blended Learning

Blended learning combines traditional face-to-face instruction with digital learning methods. This hybrid model takes advantage of the strengths of both approaches—personal interaction and immediate feedback from in-person classes, along with the flexibility and scalability of online learning.

Adaptive Learning

Adaptive learning systems use algorithms and data analytics to tailor content and assessments to the learner's individual performance and needs. By identifying strengths,

weaknesses, and learning patterns, these systems provide a personalized experience that can improve outcomes and maintain engagement.

Video-based Learning

Using videos as the primary mode of instruction is a popular and effective technique in e-learning. Videos can explain complex concepts visually, combine audio and text for better retention, and allow students to pause, replay, or speed up content based on their preferences.

Social Learning

Social learning is based on the idea that people learn effectively through observation, imitation, and interaction with others. In e-learning environments, this is facilitated through social features such as discussion forums, comment sections, peer reviews, live group sessions, and integration with social media. These tools allow learners to share knowledge, ask questions, provide feedback, and learn from each other's experiences. Social learning fosters a sense of community and can increase motivation by creating a more collaborative and connected learning environment.

3.4 Applying E-learning to UML Class Diagrams

Since the experience of studying computer science contains a large amount of self-study in any case, there are many online resources and courses in the field. Some of them utilize the techniques mentioned in the previous section. We investigated how they were used in more detail to determine whether such practices could be relevant for our intended course.

3.4.1 Gamification

Gamification is the most commonly mentioned practice when it comes to research of software modeling e-learning, and there are numerous attempts at applying it in the context of teaching the UML class diagram. The goal is to increase the students' interest and motivation in the given tasks by exploiting their natural drives.

In *Actionable Gamification: Beyond Points, Badges, and Leaderboards* [7], Yu-kai Chou introduces the Octalysis Framework, a comprehensive model for analyzing and designing engaging experiences through gamification. The framework is built around eight core drives that motivate human behavior, such as meaning, accomplishment, empowerment, ownership, social influence, scarcity, unpredictability, and avoidance. Unlike traditional gamification approaches that focus mainly on superficial rewards, Octalysis emphasizes the psychological factors behind user motivation, enabling the creation

of more meaningful and effective game-like experiences in both digital and non-digital contexts.[1]

The practice of gamification involves several techniques. Cagnazzo et al.[8] mark a student's accomplishment by levels - the student starts at level one and is awarded experience points for completing exercises. The amount of experience points gained is affected by the difficulty of the given exercise as well as the number of mistakes made by the student, providing additional motivation to reduce the mistakes for the student. After gaining enough experience points, the student's level increases. The study proposes even the introduction of quests to further engage the learners.

The same study also implements a customizable avatar that represents the student in the application. They can alter the avatar's appearance by changing its clothes, hairstyle and other visual features. Completing exercises unlocks new styling options, again increasing the student's motivation. Lastly, the tool immediately displays feedback to the student by highlighting mistakes in various colors and by changing the student's avatar's facial expression based on the student's performance.

In another work, Jurgelaitis et al.[5] implement, in addition to previously mentioned levels and experience points, badges that are awarded for accomplishing specific goals and coins that can be exchanged for certain items. The application displays immediate feedback after completing tasks and visualizes the student's progress. Leaderboards with students ranked by their experience points are also included, invoking the students' natural competitiveness.

There are other options to incorporate the concepts of gamification into the teaching of modeling. Besides some of the already described techniques, Bucchiarone et al.[9] connect modeling with the hangman game - each mistake draws another part of the hangman, until either the exercise is completed or the hangman picture is finished (which means the player has lost).

3.4.2 Microlearning

Microlearning is the practice of breaking the study material down into small, bite-sized pieces so as to avoid overwhelming the learners. Ramle et al.[4] mention that the Learn section of their application divides the study materials into smaller sections. The general reception of the application was positive and students claimed that it helped them understand the concepts better. That suggests that applying microlearning to the topic of UML may improve the students' learning experience and possibly their resulting knowledge as well.

However, it should be noted that, even though microlearning improves motivation and knowledge retention, it is more suitable for early stages of learning when basic concepts

are explained. It might not be the ideal approach when tackling more complex concepts or when constructing the bigger picture out of the smaller lessons[2]. Therefore, microlearning should not be applied to every concept, but rather to the earlier lessons only.

3.4.3 Social learning

Ramollari et al.[10] designed a system in which students were encouraged to discuss and share their solutions to modeling exercises. Besides discussions and comments, the students could attach their solutions to other students' work as hints, alternatives or suggestions for improvement. The efforts were evaluated positively.

3.5 Learning Management Systems (LMS)[1]

LMS have become an increasingly popular choice for educational institutions and individual educators seeking flexibility, transparency, and cost-effective solutions. These platforms offer a wide range of features to support online learning, from course management and content delivery to assessment and communication tools.

In this section, we briefly examine several prominent open-source LMS platforms, evaluating their general characteristics, strengths, and limitations. Each system offers a unique set of features. We will perform a comparison based on our assumed requirements that will be described later in section.

Moodle[3]

Moodle is a widely-used and feature-rich open-source learning management system that supports a broad range of teaching styles and learning needs. It offers powerful tools for content delivery, assessment, tracking learner progress, and supporting collaborative learning through forums and wikis. Thanks to its extensive plugin ecosystem, Moodle supports gamification, microlearning, adaptive learning, and integration with external tools. While highly customizable, its interface and complex configuration may feel overwhelming to users seeking simplicity.

[2]Christopher Pappas. "Microlearning in Online Training: 5 Advantages and 3 Disadvantages." *eLearning Industry*, March 15, 2016. Available at: `https://elearningindustry.com/microlearning-in-online-training-5-advantages-and-3-disadvantages`. Accessed May 15, 2025.

[3]"Moodle," *Moodle Project*, https://moodle.org/, accessed May 25, 2025.

Open edX[4]

Open edX is a scalable and modular LMS originally developed by Harvard and MIT. It is especially strong in supporting self-paced and adaptive learning through structured content, video lectures, quizzes, and analytics tools. Its powerful authoring tool, Studio, allows course creators to design interactive and engaging learning experiences. While Open edX is ideal for research-driven or large-scale deployments, it has a steeper learning curve and requires more technical resources to deploy and maintain.

Chamilo[5]

Chamilo is an open-source LMS focused on ease of use and fast deployment, making it suitable for institutions prioritizing simplicity over advanced customization. It includes built-in tools for course management, assessments, certification, and collaboration. Chamilo enables tracking student progress and provides basic support for gamification and interactive learning. Its clean interface and low technical demands make it a solid choice for smaller teams or pilot e-learning projects, though it offers fewer integrations than larger platforms.

ILIAS[6]

ILIAS is a mature, flexible LMS that supports complex learning scenarios, including competency-based education, learning paths, and detailed tracking of user progress. Originally developed for academic institutions and public organizations, it emphasizes structure, reusability of content, and rich assessment tools. ILIAS also includes collaborative tools such as blogs, forums, and group workspaces. While powerful, its interface can appear outdated and may require configuration to meet modern usability expectations.

Canvas LMS[7]

Canvas LMS offers a modern, user-friendly experience with a focus on intuitive design and ease of use. Developed by Instructure, its open-source version includes core features such as course creation, grading, analytics, and communication tools. Canvas supports REST APIs, making it easy to extend or integrate with external tools like custom diagram editors or assessment modules. Though some advanced analytics and AI features are exclusive to the commercial version, the open-source release remains highly capable and suitable for most academic settings.

[4] "Open edX," *Open edX Project*, https://openedx.org/, accessed May 25, 2025.
[5] "Chamilo," *Chamilo Association*, https://chamilo.org/en/, accessed May 25, 2025.
[6] "ILIAS," *ILIAS Open Source e-Learning*, https://www.ilias.de/en/, accessed May 25, 2025.
[7] "Canvas LMS," *Instructure*, https://www.instructure.com/canvas, accessed May 25, 2025.

Overall comparison

Feature	Moodle	Open edX	Canvas LMS	ILIAS	Chamilo
Simplicity (UI/UX)	Moderate	Moderate	High	Low	High
Progress Monitoring	Yes	Yes	Yes	Yes	Yes
Customization	Very High	High	Moderate	Moderate	High
Ease of Content Extension	High	High	Moderate	Moderate	High
Gamification	Yes (plugins)	Limited	Yes (plugins)	Yes	Yes
Microlearning	Yes	Yes	Moderate	Yes	Yes
Adaptive Learning	Yes (plugins)	Yes	Limited	Limited	Moderate
Community and Plugin Availability	Very Large	Large	Moderate	Moderate	Growing
Feedback Channels	Yes	Yes	Yes	Yes	Yes
Ideal Use Case	Schools, Universities	MOOCs, Corporations	Schools, Universities	Corporate, Government	Schools, NGOs

Table 3.1: Comparison of selected open-source Learning Management Systems

3.6 Common issues in Learning UML

There are several common challenges students face when learning UML. Some of them are not encountered by students exclusively but also by graduates and practitioners.

Fresh computer science graduates are expected to possess at least a basic understanding of the use of UML in software development, but that is not often the case [11]. In addition to that, even more experienced practitioners or programmers commonly claim to know UML, but later prove to only have a very limited or even wrong understanding of the concepts[8].

This gap arises because UML is widely recognized as complex and difficult for novices to grasp. Studies have consistently shown that students struggle with selecting the appropriate level of abstraction, applying correct syntax and semantics, and translating mental models into UML diagrams. Moreover, the same recurring mistakes continue

[8]Donald Bell, "An Introduction to the Unified Modeling Language," *IBM Developer*, https://developer.ibm.com/articles/an-introduction-to-uml/, accessed February 23, 2025.

to be observed over the years, indicating persistent conceptual misunderstandings and difficulties in mastering the modeling process itself [12][1]

There are multiple research papers describing the process of identifying errors made in class diagrams. The studies [13], [14], [12] made catalogues of such mistakes, allowing easier identification of errors and potentially using them as educational materials. Chren et al.[14] revealed that the most common mistakes were related to associations, lack of domain understanding, applying inappropriate levels of detail, placing operations into wrong classes or incorrect use of inheritance.

Reuter et al.[12] provided a comprehensive taxonomy of problems encountered by students, including confusion between class diagrams and other UML diagram types, difficulties in determining appropriate attributes, operations, or classes, uncertainty in modeling associations and their properties, and problems distinguishing between relationships such as inheritance, realization, aggregation, and composition[1].

These studies highlight the need to improve the way UML is taught in most universities, as the students' knowledge often appears insufficient. Areas for improvement include providing more practical exercises, immediate feedback, increasing students' engagement, or connecting modeling with coding.

Chapter 4

Analysis summary

4.1 Related work

There are multiple studies that tried to create e-learning modules or tools aimed at enhancing the teaching of the UML class diagram. They applied various approaches and ideas to the task.

Automated diagram assessment tools

Much of the other researchers' efforts were aimed at creating modeling tools that would enable students to draw their own class diagrams from scratch and then automatically evaluate their submissions. The goal is to provide fast and personalized feedback to the students and reduce the teachers' workload.

In these studies, the requirements for those tools were fairly similar; the drawing tools should be simple and intuitive and should be able to export the created diagrams in a format compatible with the evaluation algorithm (XMI in most cases). Then the diagram would compare the student's solution with a premade example solution and list errors or recommendations for improvement for the student.

One of the main issues with automated class diagram assessment is the fact that a modeling task is very likely going to have multiple correct solutions. One way of dealing with this issue is to store a number of possible solutions for each given problem and use the solution that is most similar to the student's current submission [6]. However, it is also possible to only use one sample solution. Doing so presents the risk of providing misleading feedback, missing errors or marking correct options as errors. In practice, these issues do not seem to be significant [15]. A less common approach to diagram quality assessment was using machine learning which achieved promising results, though only provided grading of the diagrams without specific feedback [16].

Modi et al.[17] described a tool for drawing and automatically assessing submitted diagrams. Unlike most other tools of the kind, which only support one type of diagram, the proposed tool supports a number of various UML diagrams, including the class diagram. The study also includes a brief comparison of similar tools and concludes that such tools can decrease the time and effort needed from teachers when evaluating students' diagrams.

Schramm et al.[15] also introduced an automated tool for correcting students' diagrams by comparing them to an example solution. The results of the study did not suggest that using an automated system for correcting diagrams yields better results than relying on feedback from a teacher. However, they demonstrate that such a system can save the teachers' time and resources without compromising the students' learning process.

E-learning applied to the UML Class Diagram

There were other studies aimed at teaching the Unified Modeling Language class diagram using other techniques than an automated tool for correction of the diagrams. They created platforms with various forms of quizzes and exercises and some of them applied e-learning practices such as gamification or microlearning.

In a study performed by Ramle et al.[4], a simple e-learning application for teaching several types of diagrams was designed and tested on a small sample of students. The section dedicated to explaining the concepts contains small-sized bits of information as to avoid overwhelming the students (an example of the use of microlearning). The application was generally well received by the students and, according to a survey filled by the participants, also helped them understand UML better.

Cagnazzo et al.[8] proposes using gamification mechanics to enhance the modeling classes. It introduces a web-based tool that allows the students to draw diagrams and have them automatically evaluated for mistakes. The gamified practices include levels, where a student's accomplishment is determined by a numeric value. The level can be increased by solving exercises. Another mechanics implemented in the tool are offering the students the option to customize their avatars (a picture that represents them in the application) and visually enhanced feedback that highlights their mistakes.

There are more proposed enhancement for the teaching of UML. Almadi et al.[18] propose creating a recommender system to suggest the most relevant modeling exercises to students. Lethbridge et al.[19] describe how using tools that convert diagrams to code and vice versa in classrooms helps students understand the abstract concepts of modeling. Shmallo et al.[13] arranged an activity during which students were to find mistakes in each other's diagrams. Such an activity significantly improved both their modeling skills and motivation.

Study	Notable Features	Key Findings
Ramle et al. [4]	A course on UML, microlearning, self-directed learning	Well structured, interactive exercises and self-paced approach in the course were well received by students and improved their understanding of UML
Jurgelaitis et al. [5]	A gamified online course used in a university software modeling class	Gamification increased both students' motivation and their average scores
Soler et al. [6]	An automated diagram assessment tool used in a classroom	The group who used the tool scored higher in an exam than the group who used any other resource of choice
Cagnazzo et al. [8]	An automated diagram assessment tool, learning platform with gamified elements	Instant feedback on modeling exercises and gamified elements improved both student engagement and modeling skills
Bucchiarone et al. [9]	Online learning tool applying gamified mechanics	Using a gamified learning tool improved motivation and results of the students. They also found using the tool enjoyable
Ramollari et al. [10]	Collaborative learning environment – students discussed, evaluated and helped each other in modeling exercises	The extensive collaboration helped the students comprehend the taught materials, found it to be an effective learning method and the feedback on the system was largely positive
Shmallo et al. [13]	Identification of mistakes in diagrams as a collaborative exercise (not an e-learning tool, however)	Searching for mistakes in each other's diagrams and discussing them both engaged the students and improved their modeling skills
Schramm et al. [15]	An automated diagram assessment tool	The proposed automated diagram assessment tool saved teachers' time and energy without reducing the quality of provided education
Almadi et al. [18]	Recommender system to propose exercises to students	Personalized exercise recommendations help students better understand UML and promote self-directed learning
Lethbridge et al. [19]	Modeling tool that converts diagrams to code and vice versa	Students understand UML better when they see code corresponding to the diagrams

Table 4.1: Summary of notable e-learning tools and methods for teaching UML and modeling

4.2 Summary

UML is a language used for modeling systems from various perspectives. The class diagram is the most prominent of the UML diagrams. It is used in various stages of system development as well as for documentation purposes.

For these reasons UML is a common part of the curriculum at universities related to computer science. Despite its inclusion, it is often observed that students as well as graduates and even professionals struggle with UML. To solve this problem, it is a worthwhile effort to improve the education of modeling in universities.

Attempts have been made to improve the situation with the use of e-learning - the use of digital technologies to deliver educational content. E-learning can provide numerous benefits such as increased student engagement, instant personalized feedback or saving the time and attention of teachers. It is also possible to use e-learning as an addition to traditional classrooms rather than their replacement - such practice is called blended learning.

There are numerous commonly used e-learning practices with various intended benefits. The ones most suitable for educational content related to software modeling are gamification, microlearning, blended learning and social learning.

We analyzed several studies that implemented these techniques in UML learning tools. The most popular technique was gamification, where the tools support functions resembling games. Examples are giving experience points or some sort of currency for completing exercises and allowing students to create customizable avatars to represent them in the learning environment.

Most of the studies were aimed at creating tools to enhance the in-person classes at universities, not replace them. Some of them presented courses with study materials, practical examples and tests; some introduced new collaborative learning activities, and some studies proposed tools for automated diagram assessment.

The results of these studies were predominantly positive, showing improvements both in students' scores and their engagement with the course topics. Another benefit of the described tools is that they free up the educators' time and energy that they would otherwise spend reexplaining concepts and correcting exercises.

These findings clearly document the potential of introducing e-learning tools to enhance the teaching of UML in university courses related to software modeling.

Based on the analysis, we decided to create an online course aimed at the UML class diagram. It will implement some of the e-learning techniques, most notably gamification and microlearning, and will test it out on students of a software modeling subject at FIIT

STU. Our hypothesis is that taking such a course will improve the students' knowledge and understanding of the UML class diagram. Should we prove the hypothesis correct, further research can be performed to extend the scope of the course to include other types of diagrams and software modeling materials.

Chapter 5

Objectives and methodology

5.1 Objectives

The goal of this study is to create an online course for teaching the UML class diagram. The target audience is computer science students taking a software modeling course at FIIT STU.

It is a pilot project to determine whether it is viable to teach software modeling with the use of e-learning. We want this study to be a starting point for further educational content at FIIT STU should this project prove successful. In such a case, other researchers could extend the course by adding additional features and content and, most importantly, including other types of diagrams.

We present the following research questions based on our initial analysis and objectives:

1. What are the most popular and most successful e-learning practices?
2. Which of these e-learning practices are applicable to a software modeling course?
3. How can students' skills in software modeling be effectively evaluated?
4. Is it worthwhile to increase the use of e-learning at FIIT STU in courses related to software modeling?

5.2 Methodology

The course will be implemented in an established e-learning platform. It will contain theoretical materials, interactive exercises, self-assessment components, as well as e-learning practices based on our analysis.

The course will be taken by second-year students of computer science who will be pre-

sented with a test and a questionnaire to assess the impacts of the course. The test will be made up of theoretical questions and practical exercises, while the questionnaire will be aimed at the overall reception by the students. The most important questions will target the students' interest in such materials and whether they think it is an improvement over the standard way of teaching software modeling.

The students will be divided into two groups: one that used our materials and one that was instructed to search for materials on their own. Both groups will be given the same tests to determine their knowledge after the preparation time.

Chapter 6

Design

6.1 Solution description

When designing our intended class diagram course, we will take into account a number of requirements. Firstly, we have the platform or LMS on which to implement the course, then there are e-learning techniques to use, specific types of exercises and other educational content we want to apply, and lastly we need to propose an effective and objective evaluation method to determine the potential of the project for further research.

E-learning platform

We want to create the course on a preexisting LMS. Doing so will significantly reduce the development time, allowing us to focus on the educational content.

The platform should satisfy the following conditions:

- easy to use and learn
- record and store students' scores and progress for result comparison and material recommendations
- option to propose changes and improvements, report mistakes and bugs
- easy to include new features and content
- free to use or open source
- support for e-learning techniques described in the next section
- support for interactive exercises

E-learning techniques

Based on our analysis, we concluded that the most suitable e-learning techniques are microlearning, gamification, adaptive learning and social learning. Our course will also

apply self-directed learning, since a course can be taken at any time and at any place, and blended learning, as it is meant to be a complementary tool to normal classes rather than a stand-alone learning material.

Under microlearning we understand breaking down content in the sections dedicated to explaining the concepts into small pieces.

Gamification offers the largest amount of possible features in an online course out of all the e-learning techniques. We plan to implement a system for awarding experience points for completing tasks, where reaching certain milestones will trigger some changes in the aesthetics of the course. They will be used for tracking and visualizing progress in the course and exercises will be ranked by difficulty - the more difficult exercises will award more points. Another gamification element we want to include is visually enhanced feedback.

Adaptive learning should allow the course to propose content to the student based on their mistakes in exercises.

When it comes to social learning, extensive collaborations on modeling tasks are not allowed by the time available to test out the proposed course. For this reason, we will restrict the practice to merely displaying a number of metrics to the users, such as the average score on a given exercise, average completion time or the overall progress of other students in the course. Doing so will apply some social pressure on the students to perform better and potentially trigger their competitive tendencies. This practice is sometimes mentioned as a part of gamification.

E-learning techniques that are to be implemented:

- microlearning
- gamification
- adaptive learning
- social learning
- self-paced learning
- blended learning

Content

This section is aimed at describing what specific types of content the course should contain. We will divide it into three sections - one theoretical section dedicated to explaining concepts, one for practical exercises, and one for tests for skill assessment. There will also be a final test available for the purposes of documenting the student knowledge after taking the course.

A desirable addition to the final test is an introductory test, which can be a part of adaptive learining; students will be recommended exercises and activities based on their

answers in the introductory assessment.

The theoretical section will contain a set of chapters each dedicated to a single concept or component of the class diagram. The illustrative examples should be interactive - hovering over an element will highlight the corresponding text, and the models should also be modifiable so that the students can explore the concepts in a more personal way.

Since students understand UML better when they see corresponding code [19], there should also be a snippet of code available for each diagram demonstration.

Examples of complete class diagrams with precise descriptions of concepts and modeling decisions will be included as well.

The exercise section will be referred to after the completion of each theoretical chapter. The student will be given material recommendations based on the mistakes they made in the exercises. The available exercises will be of the following types:

- assign the type of relationship based on the description
- find what is missing in a diagram with drag-and-drop fill-in option
- find mistakes in a diagram
- compare a diagram with its corresponding code. Find places where they do not align
- match class diagram components with their descriptions and pictures
- alter a diagram according to provided conditions
- theoretical questions

The section with tests will consist of a number of exercises selected from across all of the course's topics. The results will be recorded and, again, the student will receive material and study topic recommendations based on the mistakes they made.

Evaluation

The first part of evaluation will consist of a questionnaire with questions about subjective opinions and impressions of the students. The most important questions will concern whether the students thought the course was helpful, whether it improved their understanding of the class diagram and, most importantly, if they thought the course was an improvement over merely using materials from lectures or reading articles.

The second, objective part of evaluation will be a test containing theoretical questions and practical exercises about the class diagram. The test will be taken by both students who completed the course and those who did not, but still received an introduction to the class diagram in a university class.

6.2 Requirements

From the solution description, we can extract lists of functional and nonfunctional requirements.

Functional requirements

- Users can propose changes, improvements and report bugs easily
- Automated exercise/test correction with and instant personalized feedback
- Support for:
  - Gamification- points, milestones, visual feedback, difficulty indicators, badges...
  - Microlearning
  - Adaptive learning - content recommendations based on a student's results
  - Social learning - progress tracking, displaying other students' performance (anonymized)
- Support for various interactive exercises and learning materials
  - Interactive diagrams (hover to highlight, modifiable diagrams)
  - UML code snippets linked to diagrams
  - Full examples with explanation of modeling decisions
  - Drag and drop exercises
  - Matching exercises
  - Find and correct mistakes in a diagram
  - Modify diagrams based on provided conditions
  - Theoretical questions
- Students need to provide anonymous feedback on the content
- Tests of students' skills
  - Introductory assessment
  - Short tests within the course
  - Final assessment

Nonfunctional requirements (e-learning platform)

- Easy to learn and use
- Open source or free to use
- Capable of storing and retrieving student data
- Scalable and maintainable for future expansion
- Easily extensible (adding new features/content)
- Restricted access - allow access only to students and teachers at FIIT STU
- Content can be accessed anytime, anywhere

6.3 Research questions

What are the most popular and most successful e-learning practices?

We performed research and based on our findings, the most prominent e-learning practices are gamification, microlearning, self-directed learning, blended learning, adaptive learning, video-based learning, scenario-based learning and social learning (see Section 3.3).

Which of these e-learning practices are applicable to a software modeling course?

Based on other researchers' endeavors and our own schemes, the most promising e-learning practices for our software modeling course are gamification, microlearning, adaptive learning and social learning, along with blended learning and self-directed learning.

How can students' skills in software modeling be effectively evaluated?

A test with exercises similar to those described in (see proposed exercises 6.1) will be our main source of evaluation. On top of that, the students will be asked to create class diagrams as part of the subject this course is complementary to, so we can study the relationship between points awarded for their submitted diagrams and the points they received in the test. The experiment also needs students who would not take the course but will take the test anyway so that we can tell if taking the course improved their grades on the modeling task.

This approach was chosen since we are lacking a tool for automated diagram correction assessment while not having enough capacity to check all the diagrams manually. Designing a fully functional tool of this sort is beyond the scope of this project.

Is it worthwhile to increase the use of e-learning at FIIT STU in courses related to software modeling?

Outside the test and modeling exercise in which we will be able to assess the objective contribution of our course to the students' understanding of the class diagram, we will also conduct a feedback questionnaire to find the students' perception of the project; even if they achieve improvement in their software modeling skills by taking the course, future courses would not be successful unless they also take interest in taking them.

6.4 Workflows[1]

[1]This section was made with the use of Gen AI

Class diagram of the system

This diagram describes the entire system as we intend it to function. It shows all the major entities that should be found there - users, course and its content, tools for tracking progress and modules for additional features such as adaptive learning and gamification.

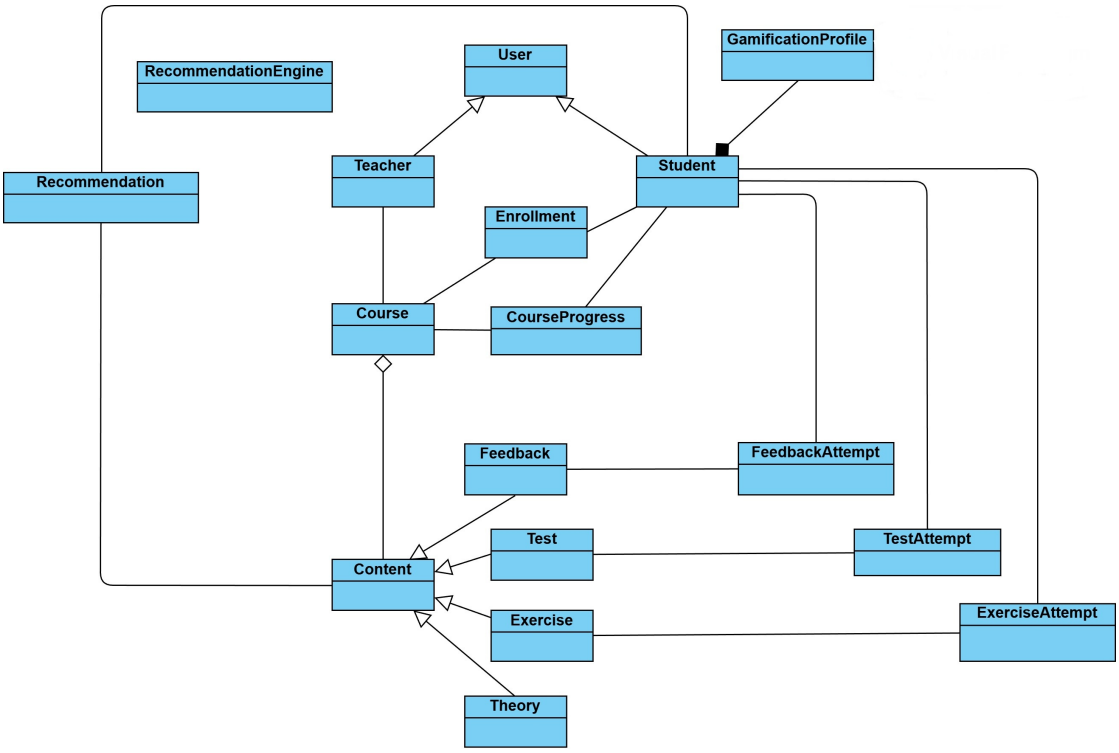

Figure 6.1: Class diagram of the designed system.

Made with: Visual Paradigm, *Visual Paradigm Online*, `https://online.visual-paradigm.com/`, accessed Dec 9, 2025.

Activity diagram - student

The activity diagram describes a student's interaction with the system. The student will log in to the system and view the course. They will complete the theoretical section including the micro-exercises in between modules and receiving feedback on their responses. Then they will complete the practical section containing more exercises. Exercises will be recommended based on how successful their responses were across various topics.

Lastly, the student will enter the evaluation section, where they will complete a test (which will not give them instant feedback this time), and fill in a feedback questionnaire.

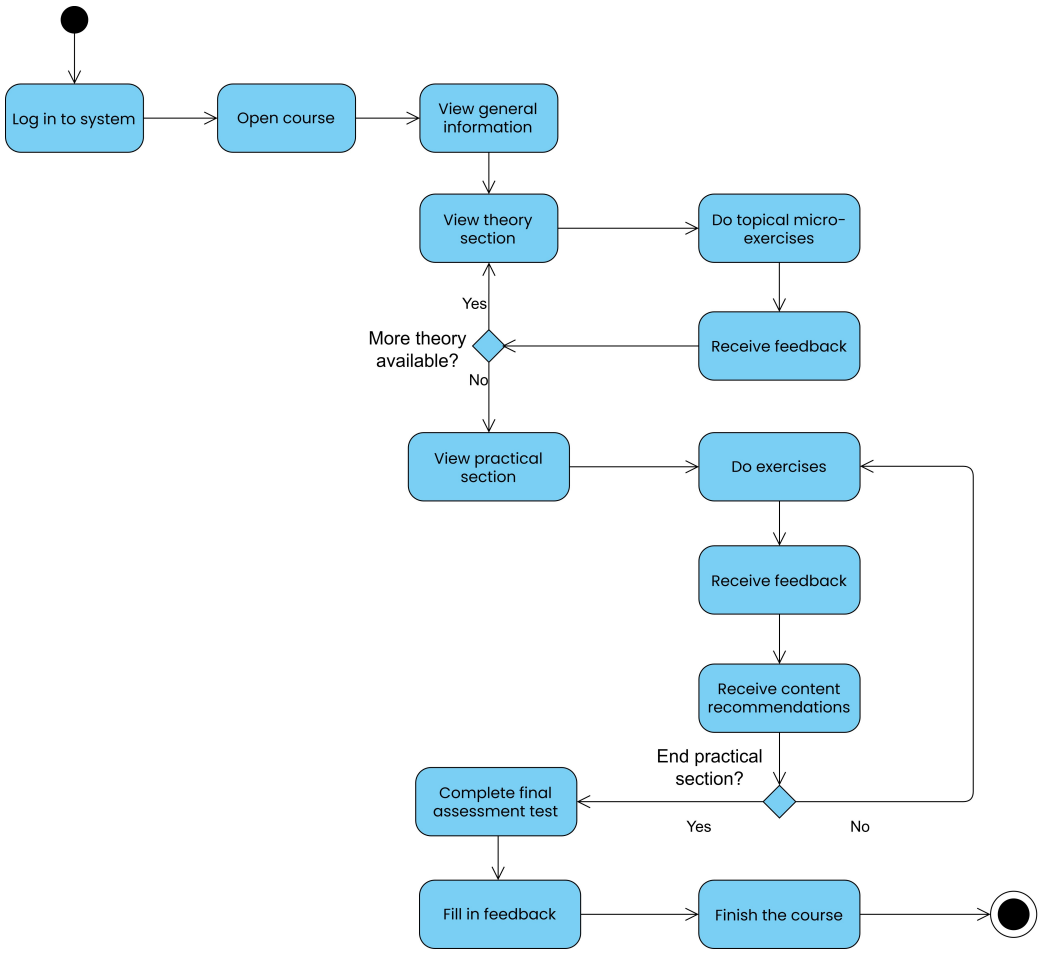

Figure 6.2: Activity diagram from a student's perspective.

Made with: Visual Paradigm, *Visual Paradigm Online*, `https://online.visual-paradigm.com/`, accessed Dec 9, 2025.

Sequence diagram - whole system

The sequence diagram describes the interaction between all the major entities in the system - teacher, system, student. The teacher needs to authenticate themselves and create a course along with its contents. Then the student can enroll in the course and complete it in the flow already described in the activity diagram (see figure 6.2). Lastly, the teacher can review the student's results and make their conclusions.

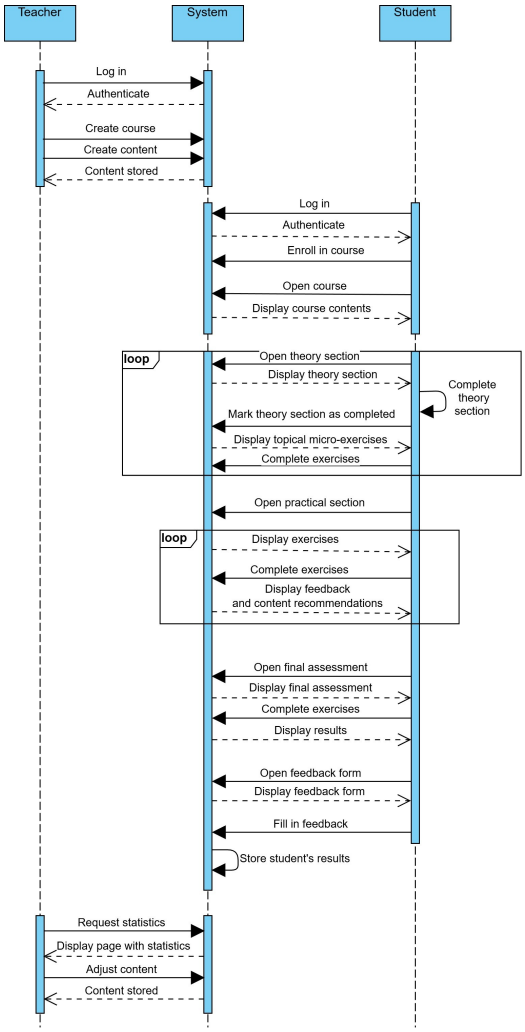

Figure 6.3: Sequence diagram of the whole system's workflow.

Made with: Visual Paradigm, *Visual Paradigm Online*, `https://online.visual-paradigm.com/`, accessed Dec 9, 2025.

Chapter 7

Implementation

We decided to implement our course in Moodle LMS. Moodle satisfies the vast majority of our requirements - both functional and nonfunctional.

It is open source, easy to use, and it is possible to deploy on local servers, allowing us to control the amount of computational resources dedicated to it for future scaling. We can restrict access to a set of users defined by us. It stores information about students' progress, generating statistical reports about quizes automatically. Content can be added easily, with loads of plugins developed and maintained by community available that introduce features not present in basic Moodle itself.

Moodle supports most of the proposed exercise types out of the box - matching exercises, code snippets, drag and drop, multiple choice, matching exercises and more. The biggest hindrance is the absence of a UML editor. However, we found no other LMS that would provide this functionality without the need for a workaround, so that is not a decisive issue. For the first version, we decided to leave out exercises with direct diagram manipulation. Exercises are corrected automatically, feedback can be provided for each answer in an exercise. Tests and feedback forms are also implemented and ready to use in Moodle.

7.1 Nonfunctional requirements - Moodle[1]

Easy to learn and use
Moodle has extensive documentation and multiple tutorials for educators available. The interface for creating content is intuitive and easy to navigate.

[1]This section was made with the use of Gen AI

It allows generating login credentials for students, meaning the students will receive their credentials via email, log in to the platform and can start completing the course right away. That is desirable, as it allows them to focus solely on learning without the need to set anything up.

Open source or free to use

Moodle is open source, meaning it is free and fully customizable.

Capable of storing and retrieving student data

Moodle stores data about students' quiz and question attempts. The data structure is designed to support adaptive learning practices and detailed statistics. It automatically generates reports on each quiz.

Scalable and maintainable for future expansion

Moodle is designed to host multiple large courses, possibly even functioning as a website for whole educational institutions. We deployed our Moodle site on a private server that belongs to FIIT STU. The allocated resources were selected based on the expected number of users, but additional servers can be dedicated if more users are present.

Easily extensible content

Adding new content or adjusting old one is easy with Moodle, as well as adding additional courses in case we want to create materials for other types of diagrams. Moreover, Moodle supports plugins developed by community that implement various features like new types of exercises or gamification practices.

Restricted access

It is possible to turn off the option to self-enroll into the site. Then we can generate accounts for a list of users from a .csv file, which allows full control over who can access our courses without the need to configure external authentication services such as LDAP or SSO. Should we decide to make our courses public for all internet users to access, self-enrollment can be turned back on.

Content accessibility

As long as the Moodle server is running, students can access the materials.

7.2 Functional requirements

7.2.1 E-learning techniques

Microlearning is easy to implement - we broke the theoretical section into small chapters, where each chapter contains only 1 concept. There are also exercises in between theory to make the content more engaging.

Blended learning in our course is implicit. The course is to be used as a part of an actual university class.

Self-paced learning - students will be allowed to take the course whenever it suits them, and will not have a time limit on finishing it after they start.

7.2.2 Theoretical content

We placed the chapter with explanations of concepts of the class diagram at the start of the course. It is implemented as a number of Moodle's book resource[2] instances.

The **book resource** contains a number of pages with static content. It is structured like an actual book with a table of contents, chapters and subchapters. This way the student is aware and in control of the flow of their progress. It can be exported and printed to a .pdf file for future reference by the students as well.

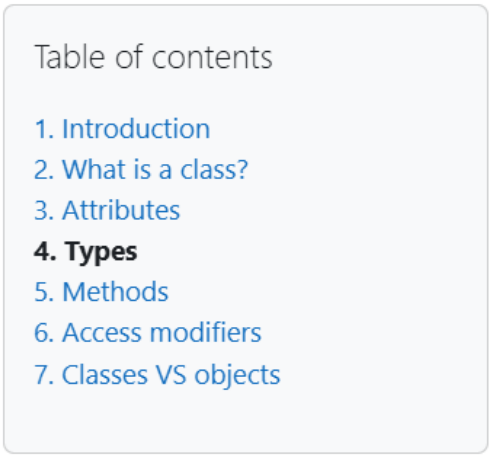

Figure 7.1: Clear separation of topics for microlearning

[2]Moodle, "Book resource", `https://docs.moodle.org/501/en/Book_resource`, accessed December 13, 2025.

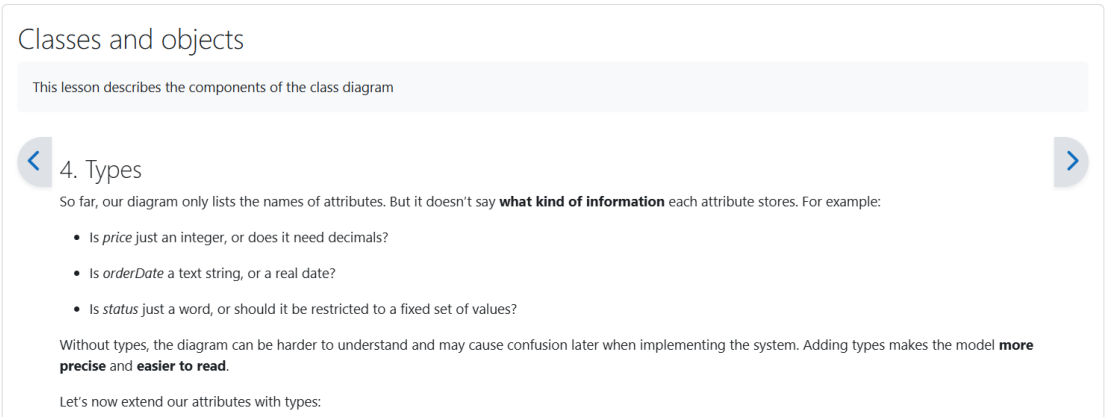

Figure 7.2: Layout of individual section in the theoretical section

There were other options, mainly Moodle's Lesson activity[3] which provides more space for adaptive learning. However, it cannot be referred to in later stages of the course, which is what made us choose the book instead. Appropriate theoretical sections are linked in feedback to exercises in case a student makes a mistake.

Short topical exercises are placed in between the explanations to promote engagement and memory retention in students.

7.2.3 Exercises

Moodle allows us to implement the majority of the exercise types described in earlier chapters (see Section 6.1). The biggest issue is that Moodle does not support a diagram drawing tool by itself. This will be discussed in later sections of this work. Another, more prominent, problem is automatically grading such diagrams, which is what led us to avoiding the implementation in this stage of the project.

Basic moodle questions still allow for a number of possible exercise types that do not present issues. Most notably, those would be generic theoretical questions, questions for comparing code and diagrams, and selecting options for improvements in a given diagram from a list.

[3]Moodle, "Lesson activity", `https://docs.moodle.org/501/en/Lesson_activity`, accessed December 13, 2025.

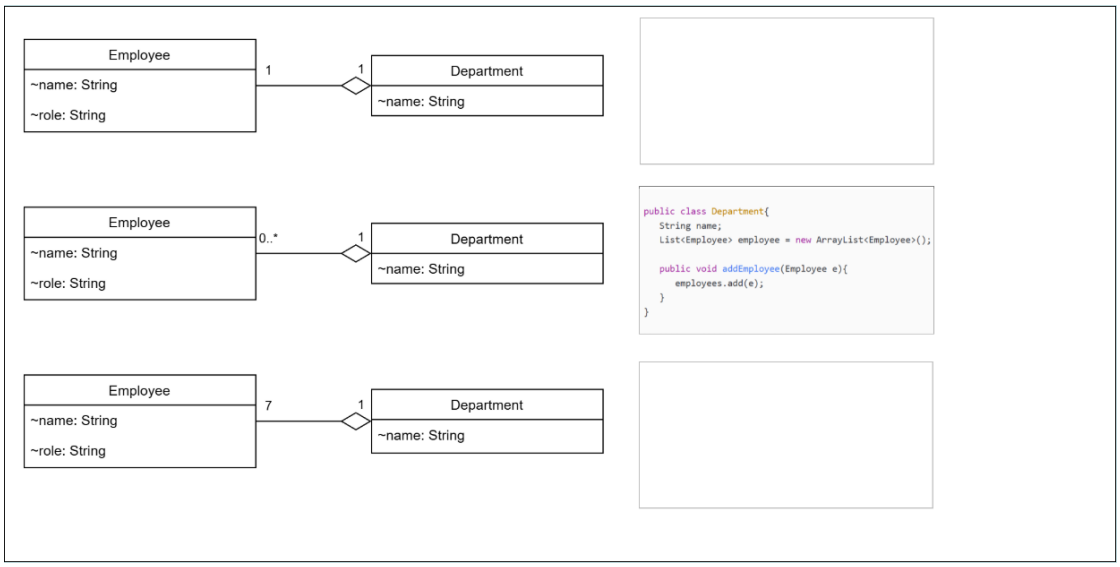

Figure 7.3: Exercise for matching code with corresponding diagrams

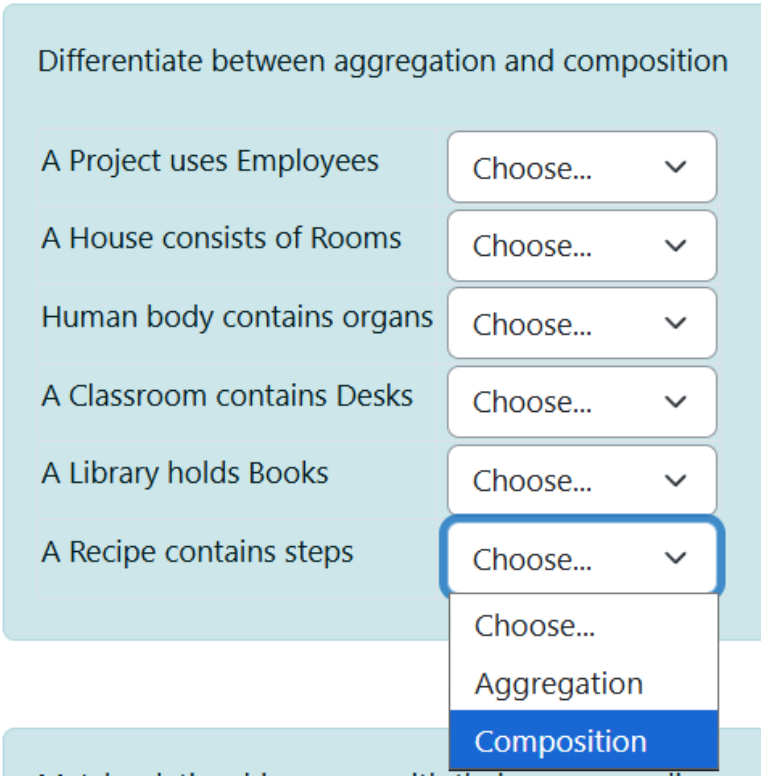

Figure 7.4: A generic theoretical question about the class diagram.

7.2.4 Testing students' knowledge, feedback

Moodle's quiz activity[4] is a selection of exercises from a defined set. It is possible to have static quizes that contain the same exercises every time, or select a number of exercises at random from a set.

We used the quiz activity both for the practice section and the final assessment. In the quiz settings, the practice sections show instant feedback for each response in each exercise after the student clicks a check button found under each exercise. They are allowed to retry each exercise multiple times, getting hints in between attempts as well.

In the final assessment section, the immediate feedback is not present, and each question only allows one attempt. On top of that, students have to tell how certain they are of their responses. If they declare high confidence but select a wrong answer, they will get a penalty for it. If they express low confidence but select a correct response, they will be awarded with fewer points than maximal amount.

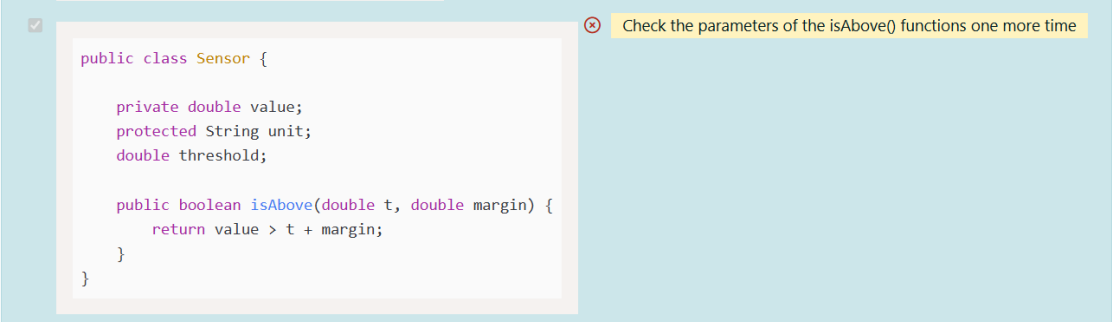

Figure 7.5: An incorrect answer displays can display feedback or a hint to the student.

[4]Moodle, "Lesson activity", `https://docs.moodle.org/501/en/Quiz_activity`, accessed December 13, 2025.

7.2.5 Statistical reports

Moodle provides statistics regarding individual quizes, exercises and the overall success of individual students. Reports can be downloaded in multiple formats including .csv. Statistics can be general, showing only the basics such as overall scores, or more nuanced and advanced statistics like effective weight or discrimination index mainly useful for adaptive learning practices.

		First name / Last name	Email address	Status	Started	Completed	Duration	Grade/10.00	Q. 1 /3.33	Q. 2 /3.33	Q. 3 /3.33
☐	AU	**Admin User** Review attempt	xsevcikm3@stuba.sk	Finished	13 December 2025 2:32 PM	13 December 2025 8:39 PM	6 hours 6 mins	**4.17**	0.83	-	3.33
☐		**Admin User** Review attempt		Finished	13 December 2025 2:26 PM	13 December 2025 2:26 PM	25 secs	**0.00**	0.00	-	0.00
☐		**Admin User** Review attempt		Finished	13 December 2025 2:28 PM	13 December 2025 2:32 PM	4 mins 11 secs	**6.67**	3.33	3.33	0.00
☐		**Admin User** Review attempt		In progress	13 December 2025 8:39 PM	-	-	-	0.83	-	-

Figure 7.6: Basic statistical report for a quiz.

Quiz name	Classes and objects - quiz
Course name	Introduction to the UML Class Diagram
Number of complete graded first attempts	1
Total number of complete graded attempts	3
Average grade of first attempts	0.00%
Average grade of all attempts	36.11%
Average grade of last attempts	41.67%
Average grade of highest graded attempts	66.67%
Median grade (for all attempts)	41.67%
Standard deviation (for all attempts)	33.68%
Score distribution skewness (for all attempts)	-0.7221
Coefficient of internal consistency (for all attempts)	12.24%
Error ratio (for all attempts)	93.68%
Standard error (for all attempts)	31.55%

Figure 7.7: Advanced statistical report for a quiz.

Question statistics

Attempts	3
Facility index	33.33%
Standard deviation	57.74%
Random guess score	
Intended weight	33.33%
Effective weight	17.41%
Discrimination index	-39.74%
Discriminative efficiency	-40.00%

Figure 7.8: Advanced statistical report for a single exercise in a quiz.

7.2.6 Feedback questionnaire

Collecting feedback from participants is one of the most important parts of the whole project. Moodle's feedback activity[5] lets students answer questions about their perception of the course. The responses are anonymous.

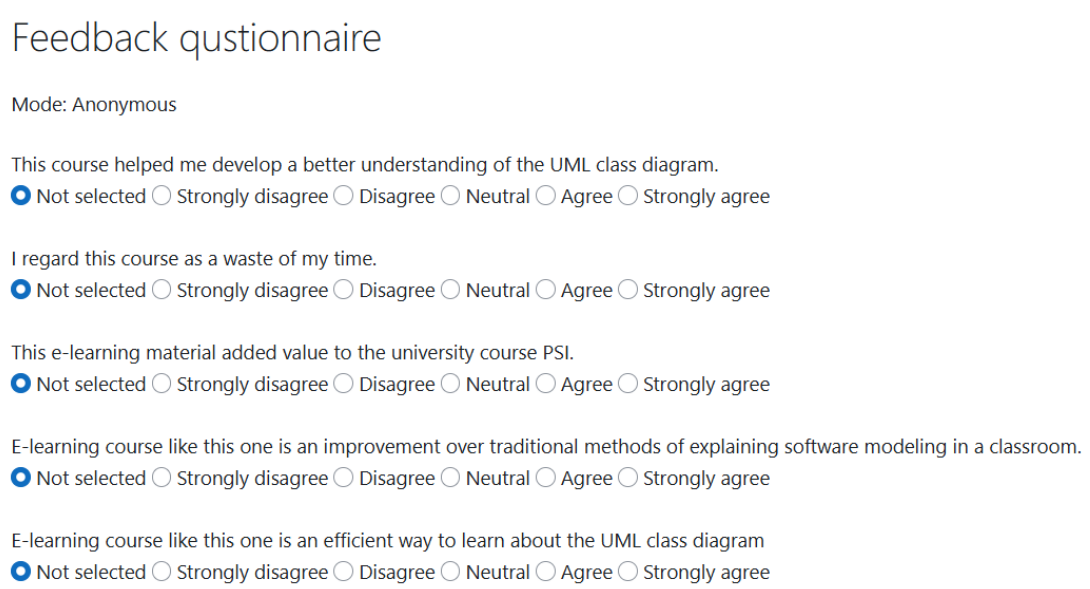

Figure 7.9: A sample from feedback questionnaire for students.

[5]Moodle, "Feedback activity", `https://docs.moodle.org/501/en/Feedback_activity`, accessed December 13, 2025.

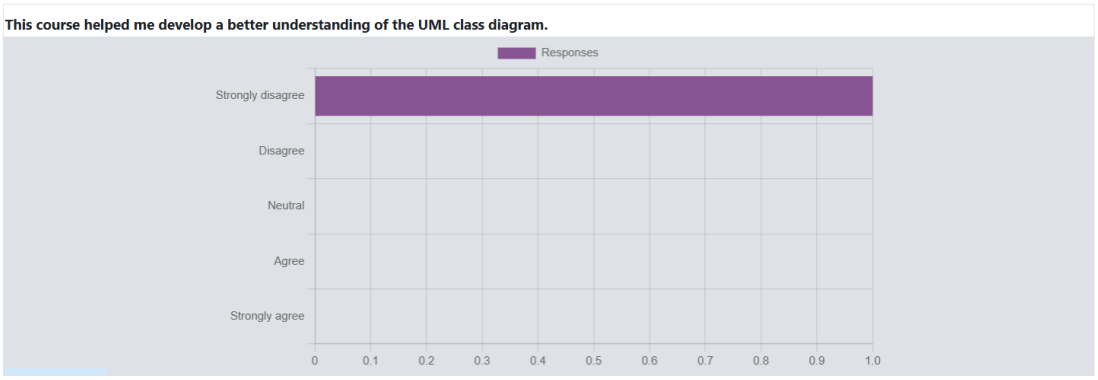

Figure 7.10: Viewing the results of the feedback questionnaire.

7.3 Research questions

Some of our research questions were addressed in the design chapter. This chapter concerns with acquiring answers to these questions:

How can students' skills in software modeling be effectively evaluated?

Previous sections (see Sections 7.2.4 and 7.2.5) describe how students can be tested and how their results will be processed. The final assessment quiz consists of various exercises both theoretical and more practical, while also requiring the students to tell how sure they are of their answers. The test will be taken by a group of students who did not take the course as well as a group who did take our course.

Is it worthwhile to increase the use of e-learning at FIIT STU in courses related to software modeling?

This question is addressed by the feedback questionnaire (see Section 7.2.6). Participants of the experiment will express their sentiments regarding this project. Results of the final assessment quiz will be taken into account as well - if students who took the course score significantly higher than those who did not take it, it would be an indication that making e-learning materials on the topic of software modeling is promising.

List of Abbreviations

FIIT STU Faculty of Informatics and Information Technologies STU in Bratislava

Gen AI Generative artificial intelligence

LMS Learning Management Systems

UML Unified Modeling Language

List of Figures

List of Tables

References

[1] Lvar Jacobson and James Rumbaugh Grady Booch. "The unified modeling language reference manual". In: (2021). URL: http://debracollege.dspaces.org/bitstream/123456789/404/1/UML%20Reference%20Manual%20by%20James%20Rambaugh.pdf.

[2] Martina Seidl et al. *UML@ classroom*. Springer, 2015. DOI: 10.1007/978-3-319-12742-2.

[3] Bernhard Rumpe. *Modeling with UML*. Vol. 98. Springer, 2016. DOI: 10.1007/978-3-319-33933-7.

[4] Rosni Ramle et al. "UML Diagram E-Learning Application: A Pilot Study". In: *Multidisciplinary Applied Research and Innovation* 2.1 (2021), pp. 418–425. DOI: 10.30880/mari.2021.02.01.044.

[5] Mantas Jurgelaitis et al. "Implementing gamification in a university-level UML modeling course: A case study". In: *Computer Applications in Engineering Education* 27.2 (2019), pp. 332–343. DOI: 10.1002/cae.22077.

[6] Josep Soler et al. "A formative assessment tool for conceptual database design using UML class diagram". In: *International Journal of Emerging Technologies in Learning (iJET)* 5.3 (2010), pp. 27–33. DOI: 10.3991/ijet.v5i3.1402.

[7] Yu-kai Chou. *Actionable gamification: Beyond points, badges, and leaderboards*. Packt Publishing Ltd, 2019. URL: https://books.google.sk/books?hl=sk&lr=&id=9ZfBDwAAQBAJ&oi=fnd&pg=PP5&dq=Actionable+Gamification&ots=xvjYK0ZPBI&sig=IEJ6v3IosuKephU8Z5C1JjHriXs&redir_esc=y#v=onepage&q=Actionable%20Gamification&f=false.

[8] Christian Cagnazzo et al. "Umlegend: A gamified learning tool for conceptual modeling with uml class diagrams". In: *Proceedings of the 2nd International Workshop on Gamification in Software Development, Verification, and Validation*. 2023, pp. 2–5. DOI: 10.1145/3617553.3617883.

[9] Antonio Bucchiarone et al. "Gamifying model-based engineering: The PapyGame tool". In: *Science of Computer Programming* 230 (2023), p. 102974. ISSN: 0167-6423.

DOI: https://doi.org/10.1016/j.scico.2023.102974. URL: https://www.sciencedirect.com/science/article/pii/S0167642323000564.

[10] Sonja Trapp et al. "Collaborative learning of UML and SysML". In: (2011). DOI: 10.3991/ijep.v1i2.1663.

[11] Rupinder Kaur, Muhammad Zubair Malik, and Maninder Singh. "Improving Delivery of UML Class Diagrams Concepts in Computer Science Education Through Collaborative Learning". In: *2023 IEEE Frontiers in Education Conference (FIE)*. IEEE. 2023, pp. 1–5.

[12] Rebecca Reuter et al. "Insights in students' problems during UML modeling". In: *2020 IEEE Global engineering education conference (EDUCON)*. IEEE. 2020, pp. 592–600. DOI: 10.1109/EDUCON45650.2020.9125110.

[13] Ronit Shmallo and Tammar Shrot. "Constructive use of errors in teaching the UML class diagram in an IS engineering course". In: *Journal of Information Systems Education* 31.4 (2020), pp. 282–293. URL: http://jise.org/Volume31/n4/JISEv31n4p282.html.

[14] Stanislav Chren et al. *Mistakes in UML Diagrams: Analysis of Student Projects in a Software Engineering Course. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET). 100–109*. 2019. DOI: 10.1109/ICSE-SEET.2019.00019.

[15] Joachim Schramm et al. "Teaching UML Skills to Novice Programmers Using a Sample Solution Based Intelligent Tutoring System." In: *FLAIRS*. 2012. URL: https://cdn.aaai.org/ocs/4388/4388-21578-1-PB.pdf.

[16] Gustav Bergström et al. "Evaluating the layout quality of UML class diagrams using machine learning". In: *Journal of Systems and Software* 192 (2022), p. 111413. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2022.111413. URL: https://www.sciencedirect.com/science/article/pii/S016412122200125X.

[17] Salisu Modi, Hanan Taher, and Hoger Mahmud. "A Tool to Automate Student UML diagram Evaluation". In: *Academic Journal of Nawroz University* 10 (Sept. 2021), pp. 189–198. DOI: 10.25007/ajnu.v10n2a1035.

[18] Sara H. S. Almadi, Norhayati Mohd Ali, and Novia Admodisastro. "Class Diagram Recommender System (CDRS): An Educational Tool for UML Class Diagram". In: *Proceedings of the University Carnival on e-Learning (IUCEL)*. ID No. UPM 019. 2018, pp. 434–437.

[19] Timothy Lethbridge et al. "Teaching UML using umple: Applying model-oriented programming in the classroom". In: May 2011, pp. 421–428. DOI: 10.1109/CSEET.2011.5876118.

Appendix A

Work Schedule

A.1 Work Schedule for DP1

week 1	planning and outlining the work, researching the topic
week 2-3	writing about UML and the class diagram
week 4-5	research and writing about e-learning
week 6-8	researching the applications of e-learning to the class diagram
week 9-10	writing about e-learning's applications to the class diagram
week 11-12	proposing our approach to the rest of the project

Statement on the execution of the plan: The purpose of DP1 was to research the topic of the project. At the beginning of the semester, exploration of UML and e-learning was carried out. We described the basic concepts of the class diagram. The subsequent topics were drawn out, however. We got delayed at research about e-learning and its applications on UML, and had to compensate for the lost time during the exam period.

A.2 Work Schedule for DP2

week 1	research about various LMS
week 2-3	familiarizing with Moodle LMS
week 4-6	implementing the theoretical sections of the course
week 7-11	implementing the concepts of the exercises in the course
week 12	writing the design and implementation chapters

Statement on the execution of the plan: The goal for DP2 was to implement the course. The intentions were not completely fulfilled, as we still have a number of tasks left, namely creating more exercises, implementing gamification and social learning. We did create a draft of the course as well as drafts for all the exercise types we want to use.

A.3 Work Schedule for DP3

week 1-2	implementing gamification and social learning practices
week 3-4	finishing creating additional exercises
week 5	deployment of the course on a private FIIT STU server
week 6-10	performing experiments on students and evaluating them
week 10-12	finish writing the text for the thesis