MASTER’S THESIS

Thesis submitted in fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Software Engineering

Course as Code

By: Santiago Preusche
Student Number: 2220299002

Supervisor 1: Dipl.-Ing. Thomas Mandl
Supervisor 2: Nikola Bogosavljevic

Vienna, 2023-11-30

Declaration of Authenticity

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (see Urheberrechtsgesetz/ Austrian copyright law as amended as well as the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I hereby declare that I completed the present work independently and according to the rules currently applicable at the UAS Technikum Wien and that any ideas, whether written by others or by myself, have been fully sourced and referenced. I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.”

DocuSigned by:

Santiago Preusche

C252B211AC8D42B...

Wien, November 30, 2023

Place, Date

Digital Signature

Kurzfassung

Dieses Projekt wurde mit dem Ziel entwickelt, Effizienz-, Benutzerfreundlichkeits- und Versionsprobleme für das Moodle Learning Management System bei der Erstellung von Kursen zu lösen. Ein textbasiertes Tool wurde implementiert, um diese Probleme zu beheben und den Prozess der Kurserstellung zu beschleunigen sowie die Wartbarkeit der Kurse zu verbessern. Dies wurde durch die Eingabe von Dateitypen hauptsächlich im key-value Format erreicht, die jede Einstellung jedes Kurselements beschreiben. Die Ausgabe des Tools ist eine Datei, die als Kurs-Backup von der Moodle-Website wiederhergestellt werden kann.

Schlagwörter: Moodle, textbasiert, Kurs, Versionskontrolle, Wartbarkeit, Benutzerfreundlichkeit

Abstract

This project has been developed with the objective of solving efficiency, usability and versioning issues for the Moodle learning management system when creating courses. A text-based tool has been implemented to solve these issues making the process of creating a course be faster and increase maintainability of the courses. This has been done through the input of mostly key-value file types that will describe each of the settings of each course element and the output of the tool results in a file that can be restored as a course backup from the moodle site.

Keywords: Moodle, text-based, course, version control, maintainability, usability

Acknowledgements

I would like to thank some people that made this project possible and successful:

- To my family that has always been supporting me throughout the whole process.
- To my friends.
- To my supervisor that helped me improve this project to become what it is today.
- And to all of the people that helped make this project possible throughout this year.

Table of Contents

1. Introduction **1**
2. Methods **5**
2.1. Requirements 5
2.2. Main Approach 6
2.3. Reverse Engineering 7
2.4. Development Phase 9
2.4.1. The Tool 9
3. Results **11**
3.1. Requirements 11
3.2. Reverse Engineering Results 13
3.3. Architecture 17
3.3.1. The Input 19
3.4. Code 22
3.5. Testing 23
3.6. Feedback and Validation 25
4. Discussion **26**
4.1. Main Findings 26
4.2. Potentials 26
4.3. Limitations 27
4.4. Impact 28
4.5. Extra Findings/Concerns 28

1. Introduction

Thousands of people learn from online lectures and courses everyday from all over the world [1]. It is easy and really accessible. When we refer to online education, we do not only refer to sites offering courses about every possible topic for solo learners. We also refer to whole schools and universities publishing their material, activities, lectures and exams on their own sites for their students to download and use in order for them to learn.

Going deeper into this last fact. Nowadays, course sites for universities try to recreate the university environment inside their platforms.This means that these platforms are created to offer students the same experience they would have at the university but from their own homeplaces, providing the main learning material that they will go through on each of the lectures, channels to communicate with the lecturers, ways of communicating with other students via email, phone numbers or chats, access to books or other sources of knowledge, etc. However, as there are students wanting to learn, there are also teachers looking forward to teaching. So these platforms offer a vast range of ways to create courses and add all kinds of topics and activities in different forms, such as files, videos, quizzes, forums, etc…

Since its launching in 2002, Moodle [2] has been evolving throughout the years to become a platform used by many universities and companies to provide courses to hundreds of millions of users [3]. It is an open source education platform developed in PHP “designed to provide educators, administrators and learners with a single robust, secure and integrated system to create personalized learning environments” [4]. Considering that so many organizations use this platform, one would think that the usability, design and accessibility of it is extremely good. Let’s remember that we are talking about thousands of daily users for this platform as it is being used mostly by universities.

The Moodle platform does allow students to learn from courses and also it serves as a good option for lecturers to create their courses as it offers a lot of useful features. Although there are recent studies that show that Moodle has a high usability score [5][6], there are some crucial factors that do not seem to be considered in the latest studies and that are affecting the course creators. These factors will be described in the next paragraphs from this section. Also, some of the following problems that will be mentioned may have a background related to the area of software development or similar. However, this does not mean that non developer lecturers do not face these problems too.

Starting with the fact that editors cannot undo their actions creates a huge problem at the time of updating a course. The need of an undo action has already been studied by the expert in Human-Computer Interaction Ben Shneiderman and has later been shown to be important by other renowned usability experts [7]. Moodle has no way to undo any action, not even the most disrupting ones such as whole sections or course deletions. Once it is deleted, it is gone forever and only a user with admin permissions may sometimes be able to get that information back, but in most of the cases, the lecturer must start over from scratch. This means that the lecturer or course creator must remember what activities there were inside that section, and what information each of them contained in case it deletes it by mistake. This suggests that there is a huge penalisation for mistakes being made by users, in other words, the platform fails to be error-tolerant and although error tolerance is not a first priority in a piece of software like Moodle, it is not good for the users to work under the stress of knowing that any mistake will let them to restart their current element they are editing. As Ben Shneiderman states in his book "Designing the User Interface: Strategies for Effective Human-Computer Interaction", the easy reversal of actions "relieves anxiety, since the user knows that errors can be undone, thus encouraging exploration of unfamiliar options" [8]. This means that it is not reasonable to think that for a simple mistake, a harsh penalization must come. It's worth adding that the message that appears when deleting any element doesn't mention that the action cannot be undone, so the system also fails to inform the user about the impact of their actions.

Another important lack which led to the development of this project is the fact that there is no possibility for versioning. Lecturers can not keep track of the changes they make from one course version to another. They have to trust they will remember when, where and why they make the changes they make to their courses throughout the semesters. Remember once again that there is no way to undo changes, so if something is suddenly wrong, the lecturer cannot know how long that error has been there, who made it and why it could have happened to be there. Version control is not only applicable to software development projects, it can be applied to any type of project as it serves as a history of all the changes that are made to the product throughout the development of it and keeps record of them in order for the developers (any type of developer) to easily roll back to a working version of the product. "It also creates a single source of truth" as every change and every action is recorded so there is no place for lies or shady explanations [9].

Trying to relate version control with a course, Moodle does offer the possibility to download the whole created course and see each of the topics, activities and information in it. The problem is that a backup file needs to be created in order to access this information, but as we will discuss later, this consists of hundreds of XML files which are not user friendly. The information is there and it can be accessed, otherwise this project could not have been done. But thinking about uploading these files into a git repository in order to keep some kind of version control would not be possible at all as there are hundreds of values that the user would have to keep track of, and hundreds of others that are only accessed by the system and will change from one backup to another. So there has to be another solution where instead of uploading hundreds of complex XML files, users can just upload the course they see as they see it on the platform. Meaning, correctly and easily separated by each created section, each activity with its own content, etc. So changes are clear and easy to understand.

As for the rest of the site, icons do not clearly describe their actions and most importantly, it takes way too many clicks to add significant value to the course. 19 is the max amount of possible clicks without making a mistake that can be done when creating just a section with a page activity. And 5 is the minimum amount if no settings are edited (the extra amount of settings that result from adding external plugins was not taken into account). Considering the case where all settings are edited, when trying to create a new page that has all the same settings like the other one, Moodle does provide some solution which is to select the option to copy and then change the values that need an update decreasing the amount to 6. More of this will be detailed at the *Results* section after comparing it with the developed tool (*see section 3.5*).

There are some existing authoring tools that provide a solution to some of these problems. Tools like Canvas [10] and iSpring Suite [11], among others, provide a very detailed editor for courses. With easy to use features, familiar icons and behaviors and possibilities to integrate with moodle and other learning platforms, they seem to be a great alternative to the moodle editor for some basic use of the site. This is, just adding information, resources (files, images and videos) and creating quizzes which depend on the default types of each of these tools. So in the end they are greatly limited by what they offer instead of by what moodle offers. They also seem to fail in versioning because again there is no easy way to know what is being changed in each version. Another interesting tool currently available is the JetBrains Academy [12]. This tool lets lecturers create mainly programming courses that can be posted in the jetbrains marketplace. These courses are created by multiple file types such as yaml, .java, .txt and more depending on what the lecturer wants to teach. This solves most of the problems that led to the need of developing a new tool, but creates a new one which is that the jetbrains tool only works for its own marketplace. It is not made to integrate with other sites like moodle, so although there are going to be many similarities with this tool, it does not serve as a product that can replace the tool that will be developed in this paper.

A tool that is missing is a simple text-based tool able to create courses for Moodle. This is what will be developed throughout this project (or at least a prototype of this). The objective is not to replace the Moodle course editor, but to be a good alternative for it. This project is the consequence of the errors that have been negatively affecting a relevant amount of lecturers when using the named platform. In the following chapters we will discuss if there is a solution to all the previously mentioned problems. To achieve this, we will explore the following questions:

- How can this be done in a clever and efficient way?
- How will this tool be beneficial for the creation and update of courses?

- Will the tool be accessible for all lecturers or only for those within the software development area who have encountered the previously mentioned problems?
- How will this tool affect the creation of new courses?
- Is it possible to find a solution to these problems without losing functionality?

2. Methods

2.1. Requirements

The first and most important task for this project is that the output from this software must be allowed to be imported from within the Moodle platform. This serves as the base of the entire project because if this is not achievable, then the actual project cannot be implemented. It does not state any requirement to the final product yet, just the nature of the final product itself. At this point, the product could be anything from separate and yet importable activities to a full course as the ones created on the site. With this in mind and the ultimate goal of being able to replicate the course creation that the Moodle editor already provides, new requirements were needed.

The requirements for this project were added during two separate stages of development. The first one being before starting the project and the second one during the implementation of the software itself. From the previously mentioned needs and problems with the Moodle editor, the main use case is the creation of courses that is the main requirement. Other use cases that were identified include: updating a course, creating activities and adding files, writing activities in text such as .txt or .md. However, this involves thinking about new questions that involve requirements for these created courses such as:

- How will these courses be created? Through an online editor or is it a local app?
- Will the user be able to update courses?
- Will there be a user interface like in Moodle? What should the user expect?
- What features should it support? Why?

These were the main questions that we asked before starting to implement the desired tool. They were important because they described the first requirements for the structure and usage of the project. The project architecture was based on the answers to these questions as well. Interviews were carried out with a moodle key user who provided valuable information about what would be the best for moodle course creators and what features they need the most for a correct creation of a course.

As it was mentioned before, the analysis of requirements occurred during two stages. During the second stage, which was the implementation stage, other questions arise. These involved topics based on the possible uses for the tool, expansions, efficiency and simplicity for course creation. Some of them were:

- Which are the best ways for users to create each of the features and why?
- Which file types are better for each case? How will the user use them?
- How can this tool be extended? Is this possible?

As it can be seen, these questions do not lead to requirements about architecture and main idea of the tool but to different ways in which the tool will be used and how users will interact with it. The actual requirements for the tool will be analyzed at the results (3) section later on this paper.

2.2. Main Approach

The main concept of this project was to create a tool capable of turning some user input into a real course that can be uploaded into moodle and used to teach students at FHTW. Separating this concept into three main sections we have a reading section, where the user input is read and transformed into an object that represents the created element. This object will be stored inside a structure (second section) and finally it will be rewritten as an output file that forms part of the course (third section).

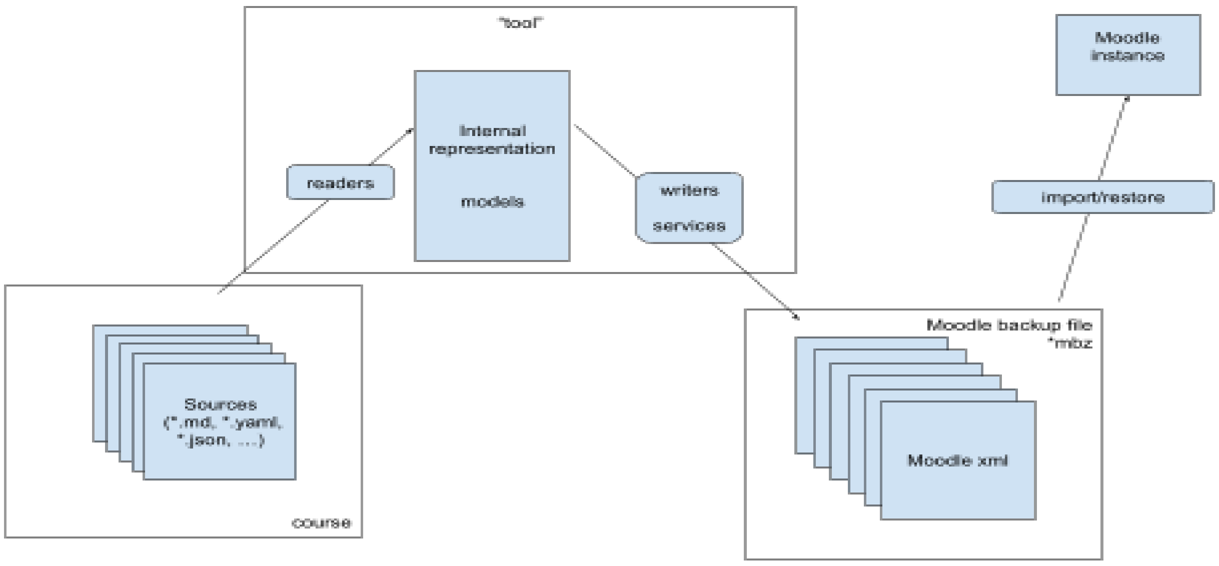


Figure 1: Diagram illustrating the flow of the program by showing how the input sources go through the tool and transform into a Moodle backup file which is imported into Moodle.

Going deeper in each section, the reading section takes care of the user input after defining it. It consists mainly of parsers that analyze the provided data and transform them into valid information for the course. For example, if the input for creating a page (which is HTML) is markdown text, the parser will transform that data from md to html. Moreover, it is really important to define the input before getting into the implementation of the reading section. It is necessary to analyze whether each moodle element will require its own input and if there will be extra inputs needed for the correct structuring of the course.

For saving the transformed data somewhere, it is necessary to have some order. This could consist of a list of sections where each section has a list of the activities that belong to that section. These activities could be dictionaries where the key is the name of the activity and the value is a list of all the activities of that type that the section has. Other approaches may include trees or other structures for saving and accessing data easily. Benefits and downsides should be analyzed for the chosen method of saving this data as it will be the most accessed part by all the reading and writing methods.

As for the writing section, once all the input data has been transformed and saved in some structure, there should be a way to write it back in files that moodle can understand. Here it will be important to find out which files does moodle understand, how are these structured, which formats does it allow and so on. Otherwise, Moodle may not be able to understand what the final output represents and the whole project will not be possible to implement.

2.3. Reverse Engineering

Reverse engineering is the act of dismantling an object to see how it works [13]. It is done primarily to analyze and gain knowledge about the way something works but often is used to duplicate or enhance the object. In this case the main objective was to understand how a file that Moodle could read is structured, how each of the elements it has represents an element from the course and how all of them interact with the moodle backup restore process.

Before going into the way that this method was applied, it is important to understand the reason why this was necessary. Although there is a lot of documentation on almost every moodle feature including the backup [14], there is no information about the files of a backup itself [15]. Each XML tag in each of the files has a meaning and the only reasonable way to get more knowledge about this was by observing how each of the files affected the restoring of the course. It was necessary to see what the output should look like before even starting the development of the tool.

There were many challenges that had to be overcome in order to achieve the first and most important objective of this project was to find a way to import a course or at least some data into the Moodle platform. As it was just mentioned, there is a feature in moodle that allows the user to restore a backup from another course into the new one that is being created at the moment. This is really important as it will be the base of the whole project. The way it works is simple: the user must go to any moodle course and create a backup. This will download a .mbz file that contains all the course information. So the user will open another course, select “restore” and upload the .mbz file that got before and it is done. However, there are some details on these instructions that are worthy to be explained in more detail.

The user can create a backup from a course from site X and restore it into a course from either site X or another site as long as they are both moodle sites and compatible versions. Some data like internal link references, user data and information related to the original site, if the backup is not being done on that same site will be lost. This is really interesting as it would lead to the possibility of other sites being able to use the tool and not just the FH Technikum Wien moodle site for which this project is being developed.

The other important part here is the .mbz file that is created during the backup process. This file is just a zip file that contains all of the files that describe the entire course. Everything is there, activities, sections, users, grades, files, etc. These course elements are represented by many extense XML files where each of the tags represent an attribute or configuration related to that particular element.

This backup restoring process is the most important part because it is the way in which we will be importing custom courses into moodle. If any course can be imported, then there should not be any problem in creating our own .mbz file with a course. This is what leads to the main method used to develop this tool: Reverse Engineering.

The steps taken to perform this method were purely based on the already mentioned backup process. A backup was made from a course and restored into the Moodle site to check that the backup system worked properly. As it did, it was now clear that there was a possible way to import external courses into Moodle. So the next step was to open the .mbz file and understand its structure and each of the files. This meant that each field, id, name, directory, etc had to be analyzed and taken into account for the correct implementation of the project. Some small steps that were performed at this point included moving files or editing the data inside them to see if Moodle accepted or rejected those changes. And finally, after understanding what each of the elements stood for, it was time to develop the tool capable of replicating each of the files with the desired information.

It is also important to mention that Moodle being an open source project also helped a lot through the implementation of this method. Although the whole code was not analyzed, the code related to the backup and other features such as activities was taken into account during the process [16]. This had mainly the objective of getting to know what were the possible values each of the tags from each xml file could have, and also in some occasions it helped understand why some errors were happening since the Moodle error description is not good nor helpful and descriptive as it should be according to the Nielsen Norman Group [17]. What helped the most to test functionalities and reverse engineer the backup feature was the use of a local Moodle instance [18]. The version used was 4.1 which is the same version that the FH Technikum Wien site started to use on July 31, 2023. However, there were visual differences as the main format used at FHTW was not available for local use. But the features and main functionality (which was the main focus of this project) were the same.

2.4. Development Phase

Following the reverse engineering way of thinking, now that the expected output has been shown, it is necessary to describe the proposed tool itself that aims to solve the problems mentioned earlier in this paper. For this part it is important to separate the development into two main aspects. On one hand the proposed solution will be described along with its architecture, design patterns, language chosen, methods and other technical details. On the other hand the course structure (input) will be analyzed and detailed in depth showing each of its components.

2.4.1. The Tool

Once the output is known, the way to reproduce it was the next objective to achieve. Starting with the programming language chosen for the development of this software, Python 3 [19] was chosen. The reason behind this choice is due to its extensive amount of libraries, the big community of developers that gives support and versatility of the language making it a fit choice for this project. Also, since Moodle is an open source project, it makes sense to use a free and powerful open source language such as Python.

The main reason for choosing an object oriented paradigm is the way this project was thought. As it was previously mentioned in the main approach *(see section 2.2)* the structure of the project would consist of three main sections (reading, saving the data and writing). To achieve this, it was necessary to have some objects capable of reading and writing each of the elements that were needed to create the course. So following Alan Kay's [20] main definition of OOP [21] where objects are defined as elements that encapsulate specific behaviors or data it was evident that this was the programming paradigm that this project needed to follow in order to achieve its objective.

As for the code and libraries, all needed was mainly to find an efficient way to read different types of files such as markdowns, yaml, json and others and another efficient way to write hundreds of xml files. Python's standard library offers some useful modules to achieve this. The usage of these libraries as well as external ones will be detailed in the next table:

Library / Package	Version	Purpose
os [22]	3.10 (same as Python)	Retrieve file information (name, type, path) as well as create new files and directories
ElementTree [23]	1.3.0	Create XML files and set the tags
BeautifulSoup [24]	4.12.2	Parse HTML documents
Markdown [25]	3.4.4	Parse markdown documents
Importlib [26]	3.10 (same as Python)	Loading code from other modules. Helpful for retrieving available classes for the course creation.
pyYaml [27]	6.0.1	Parse the first YAML documents and produce the corresponding Python object.

Table 1: List of libraries used including their version and purpose.

3. Results

During the following section we will be presenting the results from the previously described methods. These results will include the requirements that were asked for the project, the main findings from using reverse engineering as the main approach and the architecture that this method led us to as well as the results of the tests performed with target users and the evaluation of the requirements by a key user.

3.1. Requirements

As for the requirements for this project, the questions stated on section 2.1 were answered and led to the creation of the requirements shown on the following table:

No.	Requirement
1	The tool must create a file importable/restorable by moodle
2	The course must have at least the basic features: I. Name and description II. Sections III. Activities. This includes pages, files, labels, quizzes, forums and assignments. IV. A simple grading system (no complex formulas involved)
3	The course must allow at least the most frequently used activity types that make a course be worth creating.
4	Users must be able to create their courses locally
5	The course has to be conformed by files easy to read and edit by any user
6	The tool should be easy to use and allow the fast creation or modification of a course
7	The tool should give place to extensions via plugins or process file types that are not initially considered during the main development.

Table 2: List of requirements found for this project

These were the initial requirements that led to the development of this project. However, these have their own internal requirements that were described at the moment of developing each of them. These more specific requirements will be approached in much more detail during the explanation of the implementation of each of the sections of this tool. What is more important now is to fully understand what each of these mean so then there are no doubts and questions about them.

Starting with the first two, they are referring to the structure of a Moodle course. It is important that both an original course and one created by the user locally look the same, this is, separated in sections filled with activities, where each of these last ones are familiar to a regular Moodle user. Here we are not talking about creating something new but to replicate the editor behavior. It is important that for now, a basic set of actions that already can be done in Moodle (related just to the creation of courses), can be also done with this tool at least at a smaller and much simpler scale.

This final reasoning is what gives place to the third requirement, as it is necessary for the software to be able to generate a course that has the main features that almost every course on every platform presents [28]. This should be done by adding different types of activities such as purely informative activities where the lecturer just explains a topic, possibility to add files and documents for students to download and learn from, assignments or quizzes for students to be tested on their knowledge and understanding of each of the topics. That is how every course, not only Moodle courses, usually goes. There is an information section and then there is a section for the student to be tested on the topics given so far. The chosen activities for the project to support by default are: assignments, labels, pages, forums, quizzes and files (images, pdf, etc). The reasons behind this specific selection of activities comes from the interviews with the key user where it was stated that these are some of the most used features along every course by lecturers.

Going into the fourth requirement, it will be important to define how it will be done. Will users have access to some interface like a desktop app? Will they just create the corresponding files on their own and the system will detect them? These are the two main questions that have emerged from this statement and that will be discussed later with their own advantages and disadvantages of going for the implementation of one or the other.

For the fifth requirement, it is important to understand what “easy” means in this context. It implies some familiarity with the file types that will be involved in the creation of the courses. As previously mentioned, the main group of affected users that were looking forward to a solution for these problems are lecturers that teach computer/software development related topics. So this familiarity suggests file types that they have seen and used before such as markdowns (.md), YAML (.yaml), HTML, json, etc. Although the expected main users are these lecturers, they are not the only ones that we want to focus on, that is why when we say “easy to read and edit by any user” we will try to find the file types that are the simplest to learn and create even by users that face them for the very first time and do not have any kind of development experience.

This is related to the next requirement which states that the tool must be “easy to use” and allow “fast modification”. This requirement is different from the previous one because it is not referring to the input files but to the way in which these files are created and modified. This requirement asks for simplicity. It should be intuitive and simple to create a course. Regarding the speed for modification, the tool should allow users to modify their courses in an efficient way, for example by directly accessing the resource that they want to change.

The last requirement found refers mostly to the future of this software in case it works. It suggests the possibility to add plugins or make it easily extensible in order to make the project be closer to a great alternative to the Moodle editor. It also may suggest the possibility of not only applying these concepts to Moodle, but also to other platforms. This requirement, however, is too broad and not so specific, so the main focus will be set on making it work for Moodle and be extended only within Moodle resources and limits.

3.2. Reverse Engineering Results

After performing all the steps mentioned in the previous section, every file and structure from the .mbz file was understood in terms of functionality and reason for it being where it is.

As mentioned before, a .mbz file is just a zip file that contains all the files that represent a moodle course. These files are organized in directories each representing a category such as “course”, “section”, “activity” or ”file”.

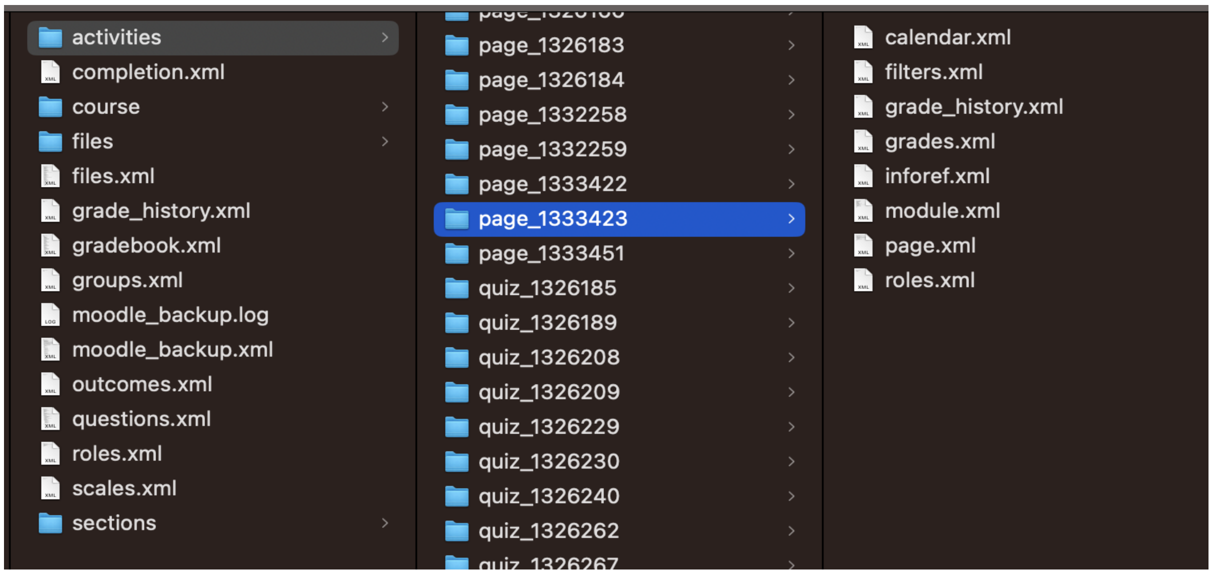
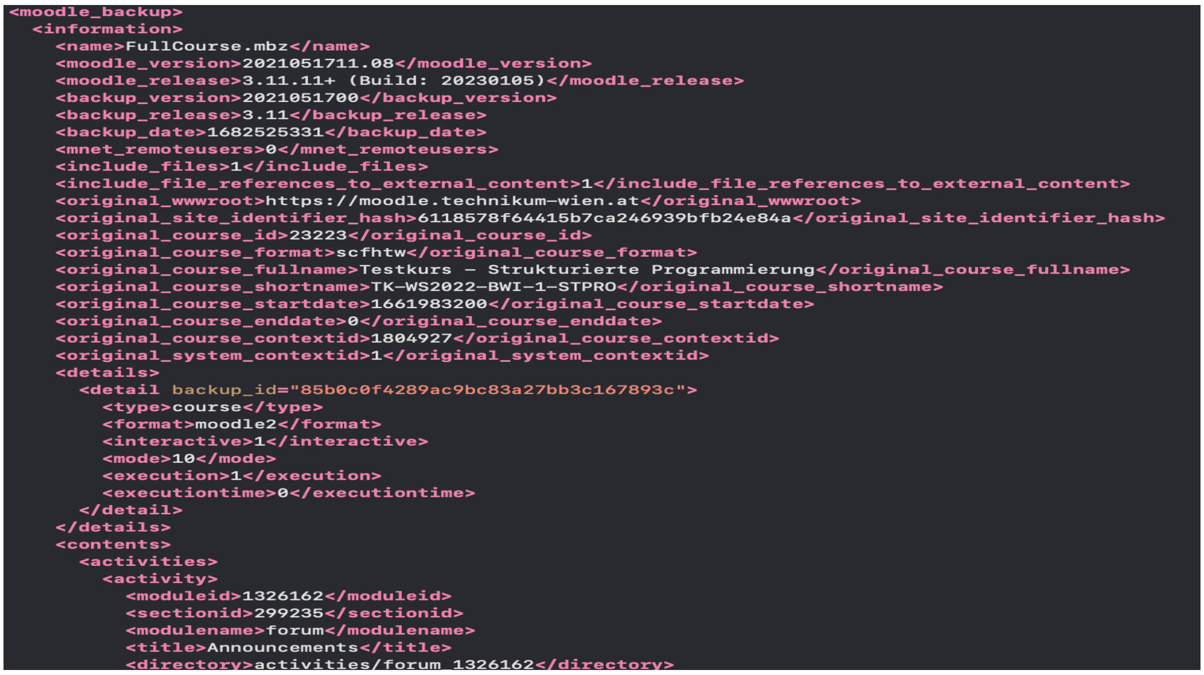


Figure 2: General visualization of the xml files being organized in different directories. On the left we see the top level of the archive file with course level information such as roles.xml or the selected folder containing all activities. In the middle panel we see folders of 'activities, with a 'page' activity selected. The right panel shows all activity level information in xml files such as grades.xml which contains grade information of this page activity.

As it can be seen in the image there are a lot of xml files contained in many different directories. In this case, the "Activities" directory is selected, showing a long list of directories that are named after the activity type they represent, followed by a number that will serve as the unique id that represents that element within all of the backup files.

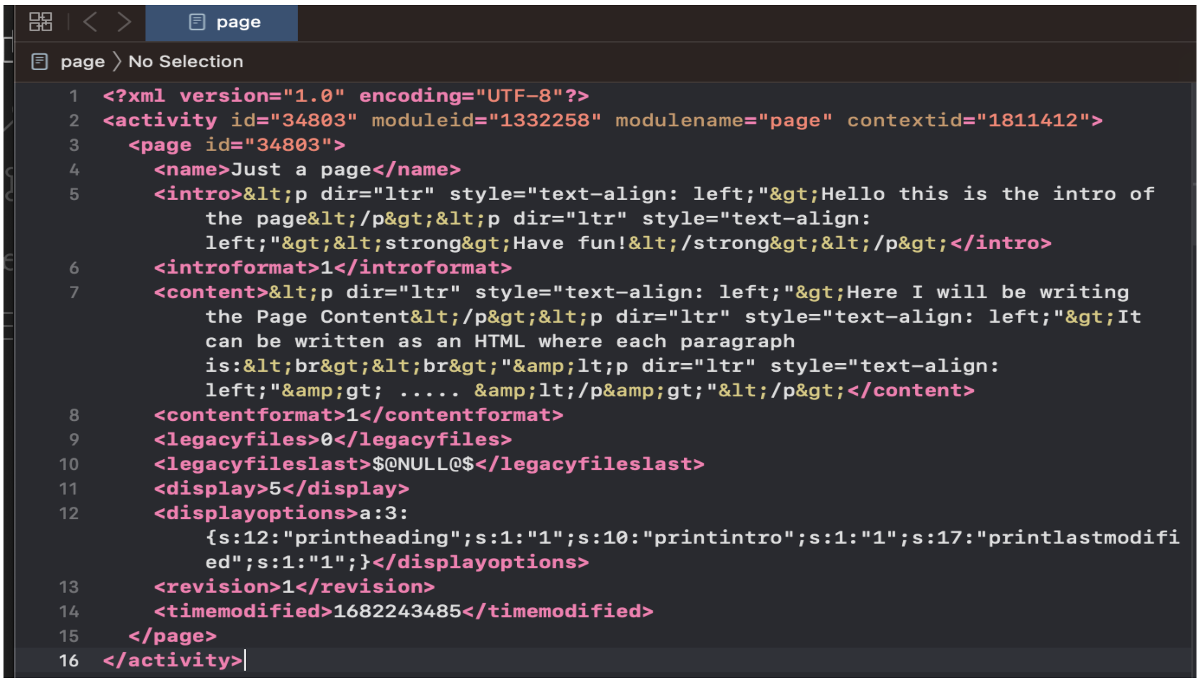
From the first observations made to each of these files we can easily get an idea of what most of these files represent or contain, "questions.xml" contains all the questions from the course, "files.xml" contains the files of the course and so on. Their names are self explanatory. However, there are some files that are worth explaining in more detail. The first one is "moodle_backup.xml". This file describes the whole course. It is an extensive xml that contains all the configuration and elements from the course. Each of the activities and sections are referenced here and it is the source of all the backup course information. Without this file, the backup will not be restored. It is in a way similar to course.xml (inside "course") directory but it does not contain course specific details but mostly site and backup specific ones.



```
<moodle_backup>
  <information>
    <name>FullCourse.mbz</name>
    <moodle_version>2021051711.08</moodle_version>
    <moodle_release>3.11.11+ (Build: 20230105)</moodle_release>
    <backup_version>2021051700</backup_version>
    <backup_release>3.11</backup_release>
    <backup_date>1682525331</backup_date>
    <mnet_remoteusers>0</mnet_remoteusers>
    <include_files>1</include_files>
    <include_file_references_to_external_content>1</include_file_references_to_external_content>
    <original_wwwroot>https://moodle.technikum-wien.at</original_wwwroot>
    <original_site_identifier_hash>6118578f64415b7ca246939bfb24e84a</original_site_identifier_hash>
    <original_course_id>23223</original_course_id>
    <original_course_format>scfhtw</original_course_format>
    <original_course_fullname>Testkurs - Strukturierte Programmierung</original_course_fullname>
    <original_course_shortname>TK-WS2022-BWI-1-STPRO</original_course_shortname>
    <original_course_startdate>1661983200</original_course_startdate>
    <original_course_enddate>0</original_course_enddate>
    <original_course_contextid>1804927</original_course_contextid>
    <original_system_contextid>1</original_system_contextid>
    <details>
      <detail backup_id="85b0c0f4289ac9bc83a27bb3c167893c">
        <type>course</type>
        <format>moodle2</format>
        <interactive>1</interactive>
        <mode>10</mode>
        <execution>1</execution>
        <executiontime>0</executiontime>
      </detail>
    </details>
    <contents>
      <activities>
        <activity>
          <moduleid>1326162</moduleid>
          <sectionid>299235</sectionid>
          <modulename>forum</modulename>
          <title>Announcements</title>
          <directory>activities/forum_1326162</directory>
```

Figure 3: Partial visualization of moodle_backup.xml

Another important file is [moduleName].xml (where "moduleName" stands for the name of the activity which is being described). Each activity has its own file describing all the properties of that type. This is the most important file for each of the activities that need to be created as it has all the information that it will need. The other files within the directory are also important and the output will not be useful if they are missing, but they do not always apply to every activity, in other words, they may or may not be used in every activity created.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<activity id="34803" moduleid="1332258" modulename="page" contextid="1811412">
  <page id="34803">
    <name>Just a page</name>
    <intro>&lt;p dir="ltr" style="text-align: left;"&gt;Hello this is the intro of
        the page&lt;/p&gt;&lt;p dir="ltr" style="text-align:
        left;"&gt;&lt;strong&gt;Have fun!&lt;/strong&gt;&lt;/p&gt;</intro>
    <introformat>1</introformat>
    <content>&lt;p dir="ltr" style="text-align: left;"&gt;Here I will be writing
        the Page Content&lt;/p&gt;&lt;p dir="ltr" style="text-align: left;"&gt;It
        can be written as an HTML where each paragraph
        is:&lt;br&gt;&lt;br&gt;"&amp;lt;p dir="ltr" style="text-align:
        left;"&amp;gt; ..... &amp;lt;/p&amp;gt;"&lt;/p&gt;</content>
    <contentformat>1</contentformat>
    <legacyfiles>0</legacyfiles>
    <legacyfileslast>$@NULL@$</legacyfileslast>
    <display>5</display>
    <displayoptions>a:3:
        {s:12:"printheading";s:1:"1";s:10:"printintro";s:1:"1";s:17:"printlastmodifi
        ed";s:1:"1";}</displayoptions>
    <revision>1</revision>
    <timemodified>1682243485</timemodified>
  </page>
</activity>
```

Figure 4: Visualization of page.xml

As it can be seen in these images and every other file, these are extense xml files with hundreds of tags. Describing each of the tags would be lengthy and would not add significant value to the analysis. Most of them are self explanatory as their name represents some attribute that we as moodle users can quickly identify, and some others are not easy to identify because they describe some moodle features that the common user does not need to understand as they refer to settings that the regular user cannot access even from the official Moodle platform. What is in fact important to analyze is that every tag is necessary in order to make the course be restored properly. Otherwise the course will not work or it will but with strange behaviors such as letting the user execute some admin actions (*see section 4.5*). Analyzing the tag also includes getting to know what format or value each of these tags have. For example dates are saved as seconds instead of milliseconds as expected. This led to unusual behaviors during development where the course elements were said to be created in the year 57650 for example.

It is worth mentioning that during the process of understanding each of the components that conform these files, many errors were found. In most cases the Moodle error message would show a stacktrace or a website with information about the error so it was easy to understand the problem. However, in other cases, there was no stack trace and just a link to an empty website. So there were two options, either brute force different options until the source of the error was found, or try to understand it by going over the code (as it is open source and of public access).

3.3. Architecture

The architecture for this project consists of two main elements: models and services. Models represent the structure of each of the data objects that will be used along the implementation. These are mainly the object itself with all the attributes. But models do not have any business logic, in other words, they have no methods, just the attributes of each of the elements that will conform the course. On the other side, the services are responsible for creating all the different outputs that form the course. Services access the data and describe all of the business logic. For every model there is a service that will handle the models' data in order to generate the different outputs.

This structure follows some design patterns and principles frequently used in software development. Although it does not follow any pattern in a strict way, it is based on many different concepts from them. Models and services are the main classes in this architecture and they only focus their behavior in a single task. This is, models structure each of the elements data and services are responsible for reading or writing the data from the models they are related to. There are no extra responsibilities for these classes. This is a clear example of the usage of the Single Responsibility Principle (SRP). This principle states that "A module should be responsible to one, and only one, actor." [29]

Another principle that this architecture follows is the design principle of Separation of Concerns (SoC). As Edsger W. Dijkstra explains in his article *"On the role of scientific thought"* [30], SoC is achieved by separating application into units, with minimal overlapping between the functions of the individual units [31]. In this case, the units would be the different models and services. Going deeper into this, the concerns would be the two main phases (reading and writing) plus the saving of the data in memory. This separation helps the software be more modular and maintainable as well as extensible. Furthermore, this is one of the reasons why it was possible to add the feature of adding custom services and models into the code. This separation made it easier for users to create their own classes and add them directly to the source files without having to know about the whole logic of the project nor having to change any of the source code. This feature will be furtherly explained once the actual architecture is presented.

This last feature tries to follow the Open-Closed Principle (OCP). Although this principle is mainly aimed at classes and modules being extended instead of modified, it can be adapted to the addition of more services and models into the project without having to modify the behaviors of the source code. This approach also gives place for the strategy design pattern as it states that new functionalities can be added without having to modify the already existent implementation [32].

Going back to the implementation itself, there are some models which represent important elements within this piece of software, and these are:

- Activity: This is the main model for every activity that is (or will be) implemented in the project. Any type of activity must extend from this main class as it states the general attributes that each of them share among.
- Section: The model that describes the sections. It is important as it holds a list of all the activities and files from that section
- Course: The model that describes the course itself. It contains the main configuration defaults for every generated course.

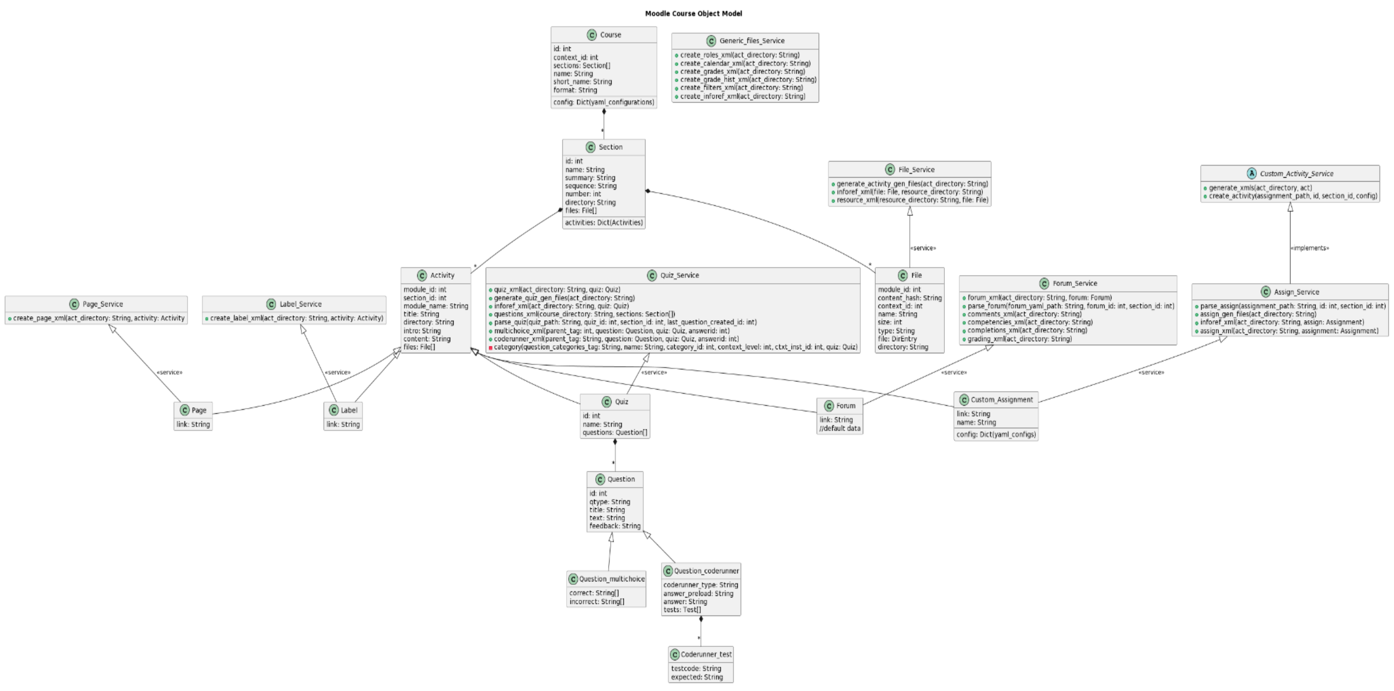
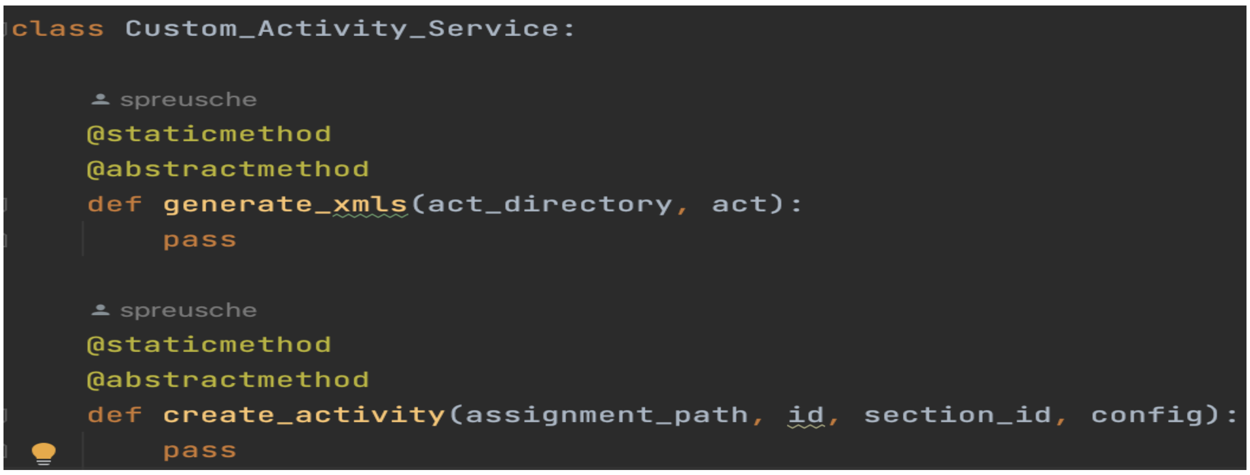

Moodle Course Object Model
Course
id: int
context_id: int
sections: Section[]
name: String
short_name: String
format: String
config: Dict(yaml_configurations)
Generic_files_Service
create_roles_xml(act_directory: String)
create_calendar_xml(act_directory: String)
create_grades_xml(act_directory: String)
create_grade_hist_xml(act_directory: String)
create_filters_xml(act_directory: String)
create_inforef_xml(act_directory: String)
Section
id: int
name: String
summary: String
sequence: String
number: int
directory: String
files: File[]
activities: Dict(Activities)
File_Service
generate_activity_gen_files(act_directory: String)
inforef_xml(file: File, resource_directory: String)
resource_xml(resource_directory: String, file: File)
Custom_Activity_Service
generate_xmls(act_directory, act)
create_activity(assignment_path, id, section_id, config)
«implements»
«service»
Page_Service
create_page_xml(act_directory: String, activity: Activity
Label_Service
create_label_xml(act_directory: String, activity: Activity)
Activity
module_id: int
section_id: int
module_name: String
title: String
directory: String
intro: String
content: String
files: File[]
Quiz_Service
quiz_xml(act_directory: String, quiz: Quiz)
generate_quiz_gen_files(act_directory: String)
inforef_xml(act_directory: String, quiz: Quiz)
questions_xml(course_directory: String, sections: Section[])
parse_quiz(quiz_path: String, quiz_id: int, section_id: int, last_question_created_id: int)
multichoice_xml(parent_tag: int, question: Question, quiz: Quiz, answerid: int)
coderunner_xml(parent_tag: String, question: Question, quiz: Quiz, answerid: int)
category(question_categories_tag: String, name: String, category_id: int, context_level: int, ctxt_inst_id: int, quiz: Quiz)
File
module_id: int
content_hash: String
context_id: int
name: String
size: int
type: String
file: DirEntry
directory: String
Forum_Service
forum_xml(act_directory: String, forum: Forum)
parse_forum(forum_yaml_path: String, forum_id: int, section_id: int)
comments_xml(act_directory: String)
competencies_xml(act_directory: String)
completions_xml(act_directory: String)
grading_xml(act_directory: String)
Assign_Service
parse_assign(assignment_path: String, id: int, section_id: int)
assign_gen_files(act_directory: String)
inforef_xml(act_directory: String, assign: Assignment)
assign_xml(act_directory: String, assignment: Assignment)
Page
link: String
Label
link: String
Quiz
id: int
name: String
questions: Question[]
Forum
link: String
//default data
Custom_Assignment
link: String
name: String
config: Dict(yaml_configs)
Question
id: int
qtype: String
title: String
text: String
feedback: String
Question_multichoice
correct: String[]
incorrect: String[]
Question_coderunner
coderunner_type: String
answer_preload: String
answer: String
tests: Test[]
Coderunner_test
testcode: String
expected: String

Figure 5: Class diagram for the architecture (see appendix A.1 for full resolution image)

The reason behind having each model related to just one service makes it much easier to parameterize the code making it easier to not only understand but also, as it was previously mentioned, to extend if needed. Going further into this last concept, although the tool currently accepts the described elements (see figure 5) it offers the possibility to be easily extended. This is possible due to the fact that it is correctly separated into two main elements. So in order to add support for a new activity, it is as simple as adding a model and service class to the project. These must follow a few rules, but all in all extension is quick and simple to achieve. This is mainly thanks to the code being able to identify each provided service and assign it to each of the activities that are being created. Once again, in order to create these classes, it will be necessary for the developer to only know how the needed files have to be created instead of having to understand the complete code.

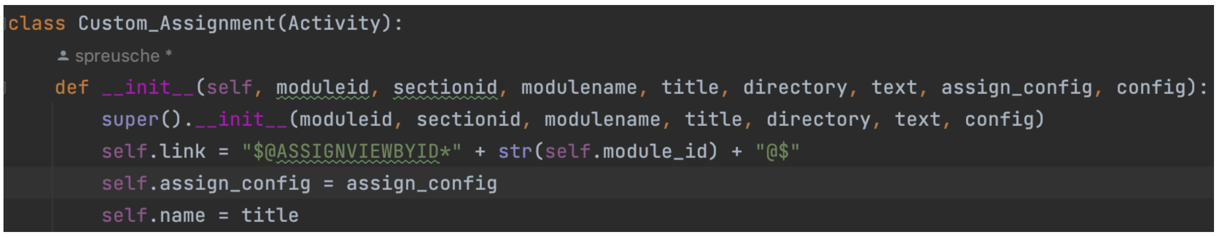


```
class Custom_Activity_Service:

    spreusche
    @staticmethod
    @abstractmethod
    def generate_xmls(act_directory, act):
        pass

    spreusche
    @staticmethod
    @abstractmethod
    def create_activity(assignment_path, id, section_id, config):
        pass
```

Figure 6: Custom activity service class that new added services must implement in order to add support to new activities which are not default from the tool.



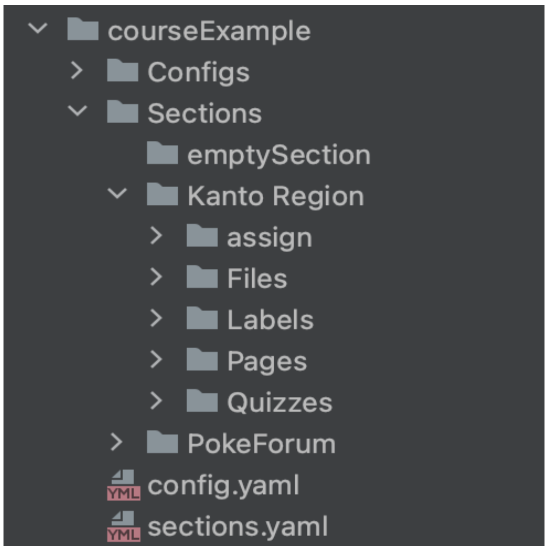
```
class Custom_Assignment(Activity):
    spreusche *
    def __init__(self, moduleid, sectionid, modulename, title, directory, text, assign_config, config):
        super().__init__(moduleid, sectionid, modulename, title, directory, text, config)
        self.link = "$@ASSIGNVIEWBYID*" + str(self.module_id) + "@$"
        self.assign_config = assign_config
        self.name = title
```

Figure 7: Example of a custom activity model class. The only condition is that it should extend from Activity class.

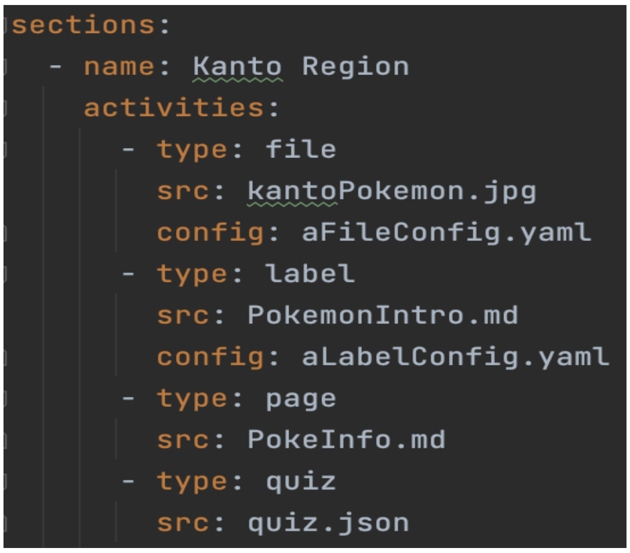
3.3.1. The Input

The input for the tool consists of different file types written by the user and organized in different directories representing each of the elements of a Moodle course. Having in mind the approach of using key-value file types that the tool of Jetbrains Academy used for its course configuration files, this project followed a similar path. In the same way as them, YAML files were used for every configuration and even some activities. The reason behind this is because these types of files are extremely easy to set and write, and also they are simple and easy to read, meaning that any human being even without technical knowledge on the subject can understand what each of the values mean [34]. Moreover, YAML files as well as the rest of the files that are going to be described next, can be easily added to version control, meaning that changes can be tracked and audited when needed. This is of high importance to the project as it is one of the main reasons why the tool was developed.

The most important configuration input files are *config.yaml* and *sections.yaml*. These two files describe the main behavior of the course. Not only the main settings, but also the order in which sections will be presented and each of the activities that these will have.



```
courseExample
  Configs
  Sections
    emptySection
    Kanto Region
      assign
      Files
      Labels
      Pages
      Quizzes
    PokeForum
  config.yaml
  sections.yaml
```



```yaml
sections:
  - name: Kanto Region
    activities:
      - type: file
        src: kantoPokemon.jpg
        config: aFileConfig.yaml
      - type: label
        src: PokemonIntro.md
        config: aLabelConfig.yaml
      - type: page
        src: PokeInfo.md
      - type: quiz
        src: quiz.json
```

Figures 8 and 9 respectively showing mentioned input files and a part of the sections.yaml file.

Continuing with YAML files, not only pure configurations were implemented to be YAMLs but also whole activities such as forums or assignments. The reason behind this is because although they do not represent a configuration as such, they end up being one on their own. These types of activities mainly consist of settings rather than content. Taking a forum for example, all the user has to decide is the title for the forum, the rest are settings that can be skipped to get a default forum or selected to create a custom one. But the main idea behind a forum relies on setting some configurations. The same happens with an assignment and may also happen with new activities in the future.

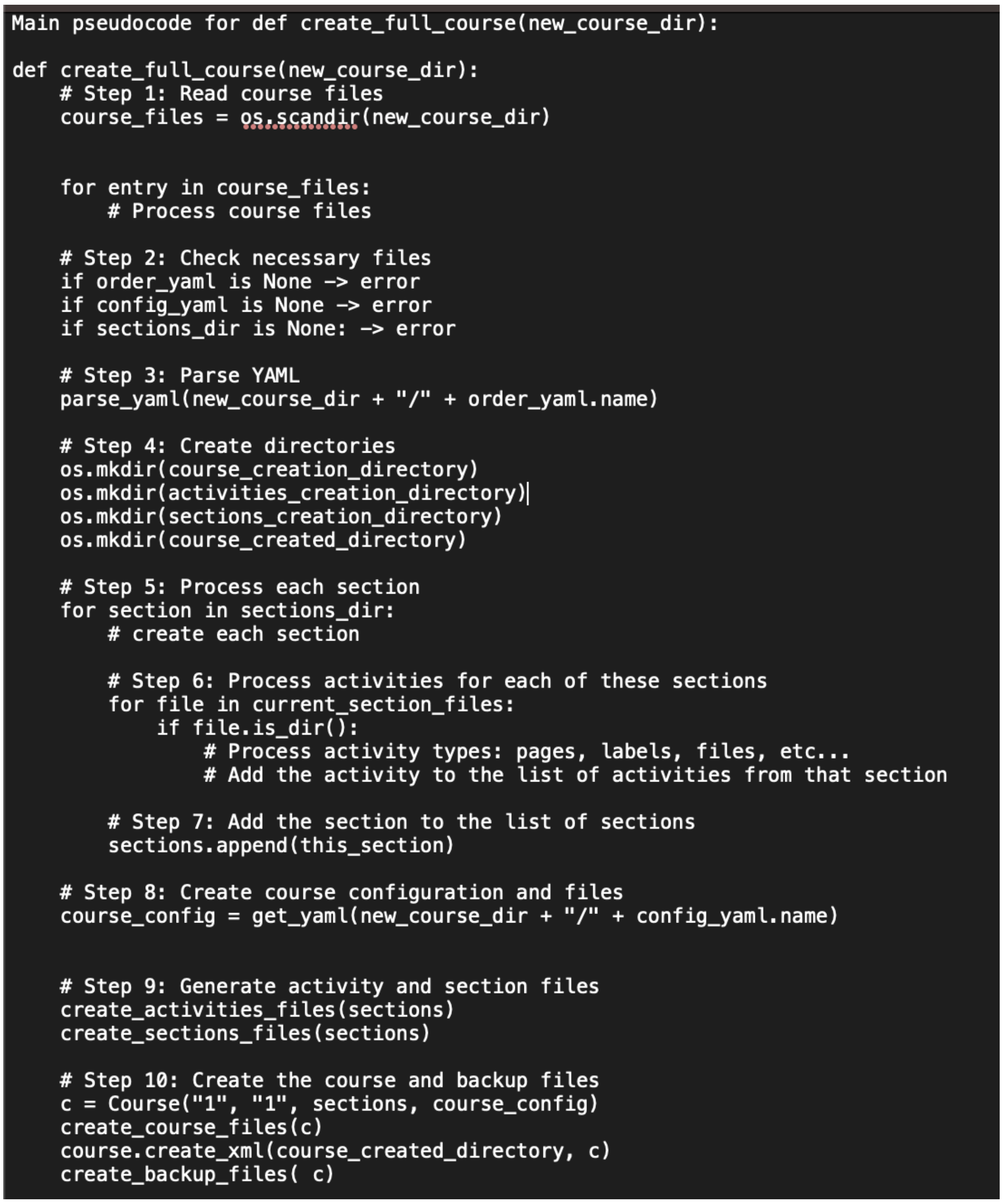
For labels and pages, after discussing it with the target user it was decided that markdown was the best option for these files. All text is interpreted by Moodle as HTML which can result in very difficult to write for inexperienced users. So instead of forcing users to learn HTML in order to write their own labels and pages, it was decided that markdown should replace it. Learning markdown is not only easy, but also easier for humans to read and understand [33].

Regarding the quiz activity, it was decided that JSON was the most suitable file type to structure it because of not only its readability but also because it is visually more clear to see all the elements that form part of the arrays. In this case, there can be many questions and having a YAML, although it is considered to be even easier to read than JSON, it can also lead to confusion when there are a lot of items with so much information (title, question, answers, feedback, etc). Another possibility for quizzes was the GIFT format. This format is currently used by Moodle for the creation of questions. However, it was not taken as a tool default because JSON is easier to learn as it can be more descriptive, and also is known on a wider scale making it easier to find any type of documentation or solution to problems that can appear while writing the file.

Finally, as it was shown in *figure 8,* all these files are organized in different directories making it easier for the user to quickly identify all the activities and configurations that are available at the created course.

3.4. Code

The pseudocode for the implementation of the described tool will be shown next. This will provide a better understanding on how each of the sections, input and other elements presented on the architecture section interact with each other in order to create the final output course.



```
Main pseudocode for def create_full_course(new_course_dir):

def create_full_course(new_course_dir):
    # Step 1: Read course files
    course_files = os.scandir(new_course_dir)


    for entry in course_files:
        # Process course files

    # Step 2: Check necessary files
    if order_yaml is None -> error
    if config_yaml is None -> error
    if sections_dir is None: -> error

    # Step 3: Parse YAML
    parse_yaml(new_course_dir + "/" + order_yaml.name)

    # Step 4: Create directories
    os.mkdir(course_creation_directory)
    os.mkdir(activities_creation_directory)
    os.mkdir(sections_creation_directory)
    os.mkdir(course_created_directory)

    # Step 5: Process each section
    for section in sections_dir:
        # create each section

        # Step 6: Process activities for each of these sections
        for file in current_section_files:
            if file.is_dir():
                # Process activity types: pages, labels, files, etc...
                # Add the activity to the list of activities from that section

        # Step 7: Add the section to the list of sections
        sections.append(this_section)

    # Step 8: Create course configuration and files
    course_config = get_yaml(new_course_dir + "/" + config_yaml.name)


    # Step 9: Generate activity and section files
    create_activities_files(sections)
    create_sections_files(sections)

    # Step 10: Create the course and backup files
    c = Course("1", "1", sections, course_config)
    create_course_files(c)
    course.create_xml(course_created_directory, c)
    create_backup_files( c)
```

Figure 10: Pseudocode for the main function of the project

As it can be seen, there are two main phases, one being the reading phase and the other being the writing one. First all the input from sections.yaml is read and labeled by the type they represent whether is an activity, section or file. After this, each of the directories that conform the course are scanned and all the different files from each of the sections is parsed, saving all the information inside the models they represent. These model objects are saved in different lists depending on what they are, for example, labels, pages, quizzes and forums are saved in their corresponding section activities dictionary. While sections, once they have all the activities and files saved, are loaded into the course sections list. These lists act as a repository for all the information. It is the structure where all data will be saved and also retrieved by the services that need it. Once the course has been created, the writing phase starts. This implies that every file has been parsed and all the information is inside the course object. During this phase, all the output files are created organizing them in all the different directories that a backup file has as it was shown at section 3.2

It is important to maintain a separation between both phases because not all the information comes from the same file. There are values such as ID's or references to other elements that need can only be added once all the files have been parsed. It would be a mistake and be very inefficient to try to write the new files as they are read because then those files will need to be accessed to add the missing information. So that would result in multiple accesses per file which are not necessary at all.

3.5. Testing

Two main tests were carried out in order to analyze both the functionality and efficiency of the developed tool. Getting into the functionality tests, it passes almost every one of them. This means:

- The tool can create a .mbz course file importable by Moodle.
- The created course can have a name and description.
- The created course can contain sections.
- The supported activities are:
  - Pages
  - Labels
  - Assignments
  - Quizzes
  - Files (resource)
  - Forums
- The tool allows the user to add extra activities that are not the default (currently the assignments feature is an example of how it works).

Regarding the grading functionality, although it is implemented it was shown that it does not work as expected. The results of testing this feature show that a gradebook can be implemented and will work correctly in the course but from the moment that course gets created, it will not let the user delete the created grade categories. This also means that the course cannot be replaced by a new version of it using the ‘restore’ feature as it will throw an error. However, if no grading system is provided, Moodle will use the default and take into consideration all the created activities that contain a grading attribute.

In order to test efficiency and see if using the tool means that courses can be created faster, the number of clicks or steps was analyzed. The following chart compares the amount of clicks needed to perform certain actions at the Moodle platform against performing the same actions in the tool. Before showing the results, there are a few considerations to take into account:

- Only settings that are also supported by the tool have been considered, otherwise it would not be a fair test.
- Since the tool does not allow to only create a single activity and post it in the online course, only the creation of the activity was considered (choosing the settings).
- For the tool side, the creation of each directory and settings file is included as if it were being created for the first time.

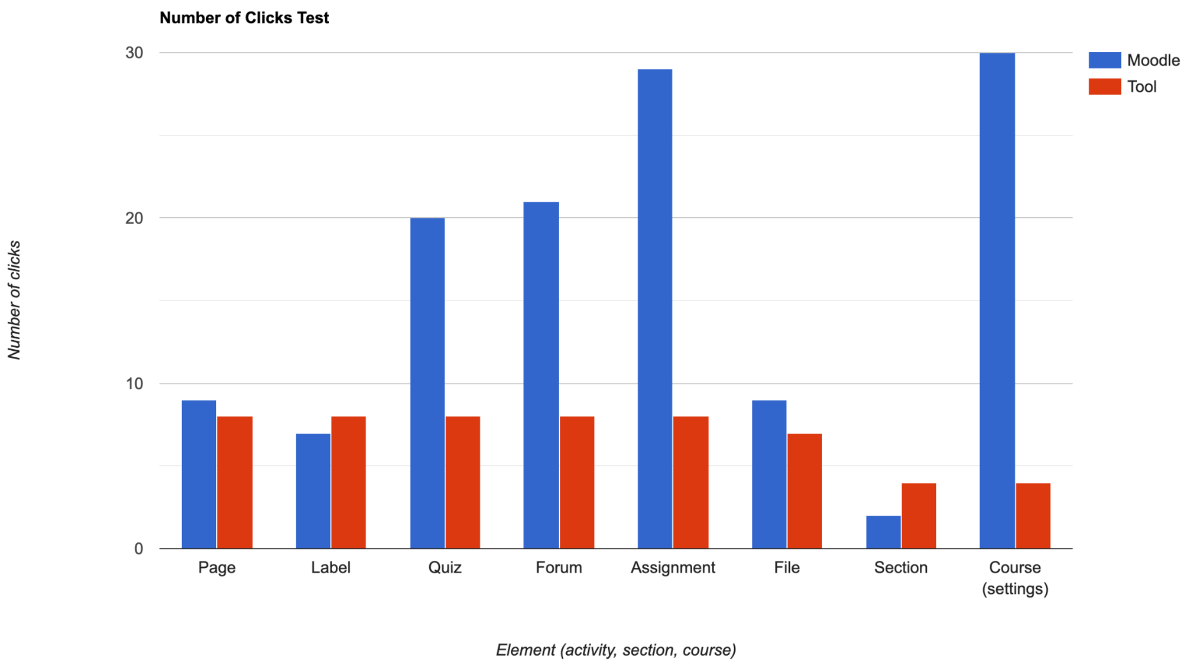


Figure 11: Bar graph comparing the amount of clicks needed to create activities, sections and courses on both the moodle platform and the developed tool.

As it can be seen, the amount of steps necessary to create a complete course on the Moodle platform are far more than the ones needed using the tool. These results already show how much more efficient it would be to use the tool in terms of clicks and speed to create a course.

3.6. Feedback and Validation

Validation for the requirements previously described (*see Table 2)* has been carried out by a key user for this project. After testing the developed tool he provided feedback that was useful both for the development of this project as well as for the future iterations that can be done to improve the functionality and quality of the tool.

Regarding the functional requirements, they do not need any specific validation from the users as it can be clearly shown that they were fulfilled (*see section 3.5 for more details*). However, starting from the fifth requirement described at the table it is not that easy to show if the requirement is fulfilled or not. In relation to the course being conformed by files easy to read and edit by any user it was determined that it is partially achieved since it is easy to read and edit but not by any kind of user. The key user suggests that the files used for the creation of the courses are simple to handle by the target users but for the rest of possible users it is not that easy since they are not familiar with any of them (it is not expected for them to be). Very similar comments were made for the sixth requirement as again it is "easy to use" for the users that are familiar with the used file types. There is a trade off between the complexity and the achieved speed of modification where efficiency (as previously tested by the number of clicks) comes at the price of adding some more complexity related to the way files are created and used along the course.

The key user's feedback for the project suggests that the project is useful and serves its purpose but it requires some knowledge of the settings. Using some provided templates helps solve this as they show all the possible options for each of the settings. An IDE plugin suggesting each of the possible settings could help solve this template-depending issue. Regarding error messages when using the tool, there used to be default Python error messages but with his feedback it was possible to catch them and print actual useful messages for the users. Another possible extension idea would be to add validation for this project in the IDE as there is no way for the user to know if the tool will execute correctly until it is executed.

4. Discussion

4.1. Main Findings

Going back to the research questions stated at section 1, it can be said that there is a clever way to solve the mentioned problems. The approach of creating a course through files that can easily be edited by users and then uploading them into Moodle through the backup feature is possible and effective. Also, the choice of the different file types based mostly on readability, maintainability and easiness to write creates the possibility of having more users for the tool as the curve of learning how to use it only consists of learning how to write a few types of file. This approach also solves the lack of version control problem as now a course can be easily added to a site like github and keep control through there.

Regarding the loss of functionality, if we base it purely on the creation of courses then there does not seem to be any loss (this is considering the potentials that will be discussed in the next section). However, if we go into course maintainability and edition in real time then, as it will be discussed later on, there are some features which are lost. Anyways, this last problem could be a good starting point for an extension of this tool, so the possibility to recover from that loss is there.

4.2. Potentials

As it was previously mentioned, the main objective of this project is not to replace the Moodle course editor but to create an alternative or extension to it where the previously stated problems are solved. However, it in fact could potentially replace or happen to be the best alternative for the groups of users that are interested and mainly benefit from the features this tool provides.

The possibility of adding new extensions for new activities and files opens the possibility of keeping up with future Moodle activity and course versions easily as only the service that handles the desired activity will have to be updated according to the needs at each moment.

Currently this tool, as we will explain later on, is not able to do it, but the idea and process studied in this paper can potentially be replicated for other platforms. As long as the data is accessible, the possibility of applying this process of thinking, reverse engineering and execution is possible. The same way users can create new service classes for custom elements within Moodle could be done for other platforms. However, this is only possible as long as the internal data on how each course is handled in each platform is accessible. This last point is currently a weakness of this project that will be discussed later on.

Other potentials include the possibility of easily sharing the courses or parts of it since now they are easy to read files that any user (considering the knowledge limitations previously studied) can create. Also it is possible to create a course in a collaboration environment as each individual can easily upload each activity to git for example and work together to create a course.

4.3. Limitations

Although this can potentially be the local version for the Moodle course editor, there are some limitations and weaknesses that affect the correct execution of the project. Some of these are the result of the design of this project and have a more straightforward solution, but others cannot be resolved due to the nature of the project itself.

One weakness that the current version of the tool presents is that only FH Technikum Wien course types can be created. The reason behind this is that this tool was originally created with the intention of just creating courses within the FHTW context. However, throughout the implementation and addition of new requirements this idea evolved into the possibility of extending this project to other contexts as well. As it can be seen, this limitation comes purely from design and it can be improved or even solved by refactoring the code. There is no other external element that could affect the improvement of this weakness.

The biggest limitation found in this project is the inability to edit the course in real time once it is live on the Moodle servers. As long as there is no user input, then the lecturer can freely restore new versions of the course until the final version is ready. However, once there are students added to the course and these start to interact with it, the lecturer can no longer create a new version for the course without deleting all the data that has been introduced into it. So if the lecturer finds out that something needs to be changed or updated, a new generated course by the tool must not be restored to the course, otherwise Moodle will understand that a new course is being created and will wipe out all existing data. For real time editing the user must use the editor provided by the platform.

This does not seem to be good for the project as it can be seen as a threat that goes against the main objective of it: creating courses. The problem is that there is no straightforward solution for this as the source of it comes from the Moodle platform itself and the way it handles the course structure and all the information inside it. The easiest way to understand this is by referring to the ids of the different elements that conform a course. When using the proposed tool, each element (activity, section, resource, etc) will have an id assigned. This value is arbitrary as it does not rely on other sources that could collide with them. It is just a way to represent each element along the code, and this is what Moodle also understands. The backup service will first check that all the ids correspond to each of the provided elements and that are used and created in the correct way. If there is something wrong it will throw an error. However, once the backup is approved by the backup service, Moodle will assign its own ids to each of the elements, so that is why when a backup from the newly created course is made, we can see that the id's do not match the ones created by the tool.

4.4. Impact

In terms of impact, users will be highly benefited. As it was explained all over this paper, with this tool users will be able to create courses faster and more efficiently. The studied approach increases course maintainability making it easier for course creators to keep track of their courses and update them every new semester or learning period.

As it was stated several times throughout this project, this tool is an alternative so there are no real threats regarding loss of jobs or usage of the official Moodle features. The idea is always to make the user have a better experience and in the best case it would make Moodle developers realize about the changes they have to implement in order to make their platform be even more efficient when creating courses.

4.5. Extra Findings/Concerns

During the implementation of this project, some interesting findings were made that although they are not part of the scope of the project, they are worth mentioning since they are mostly related to the security behind the technique used to create the courses. It was found that it is not that safe to have this kind of backup restore at moodle, as users could be able to modify the way the system behaves. For example, the user is able to create System questions that will be available for every lecturer to use in their courses. But a lecturer is not able to do that from the Moodle site. Only an admin has permissions to do that. So having an admin permission as a lecturer generates a big concern about what else can be done. For the safety of the university site more settings were not explored, and tried to just follow the rules that lecturers should follow. However, due to some programming errors, more safety issues were found.

The biggest concern comes from the fact that these are supposed to be backup files but seem to have more impact than just creating a course. A backup is meant to have only a course configuration. If within this configuration, there are admin only related settings then, should a lecturer be able to import a course from a backup? Does that mean that a lecturer also has some admin permissions? Let's remember what this paper is about. Creating courses and disguising them as backup files for them to be imported correctly.

A lecturer's main role is to create courses following the settings offered by Moodle. However, when using the backup option, there are a lot more settings that can be modified and cannot be changed from the site. Let's take the example of questions. When creating a quiz, the lecturer is allowed to add questions. These added questions are only visible inside the course they were created in. So, if another lecturer in another course wants to use the same question, it will not be possible. The lecturer has to create it again because the question only lives inside each course. This behavior is the default for the courses and that cannot be changed by a lecturer. Unless they create their quizzes from a backup course. In this project, the default context for quizzes was left at the default value (questions cannot leave a course). But that was just a design decision based on the wanted behavior by the site admin. It could have been changed and made the context level be the whole system, which means, every created question in a course would now be available and visible to all lecturers from the whole organization, in this case, the whole university.

There are also some other issues such as ID's. Continuing with the question's example. Now let's say that some value to the creator_id tag within the question XML is added. When the backup is restored, two things may happen. If the ID exists within the organization, then the question is automatically given to the lecturer with that ID. Which means that even if that lecturer does not belong to that course, it will now because it owns the question being used there. Also, creators and lecturers are usually the ones allowed to modify questions, which means that this external lecturer is now also able to edit that question. The other option is that this ID does not correspond to any lecturer within the site, then it will be assigned to an inexistent user. There is a third option as well that ends up being the same as the second, which is with an empty value, meaning that the question does not belong to anyone.

These findings are interesting to mention because they showcase the real power of a backup. It seems like the backup system assumes that the course is well generated and imports everything without contemplating the possibilities it can have. And given the ease to modify a .mbz file to make Moodle believe it is an actual course should be a concern for the developers behind the Moodle platform.

Bibliography

[1] Oxford Learning College (2022). *Online Education & E-Learning Statistics UK* [Online]. Available: https://www.oxfordcollege.ac/news/online-education-statistics/

[2] Moodle official website [Online]. Available: https://moodle.org/

[3] Moodle, *Statistics [Online]*, Available: https://stats.moodle.org/

[4] Moodle, *About Moodle [Online], Available:* https://docs.moodle.org/403/en/About_Moodle

[5] Arora, M., Bhardwaj, I., Sonia (2022). Evaluating Usability in Learning Management System Using Moodle. In: Goar, V., Kuri, M., Kumar, R., Senjyu, T. (eds) Advances in Information Communication Technology and Computing. Lecture Notes in Networks and Systems, vol 392. Springer, Singapore. https://doi.org/10.1007/978-981-19-0619-0_46

[6] Berenyi L and Laszlo G. Lecturers' Evaluation of Moodle at the University of Public Service. Proceedings of the Central and Eastern European eDem and eGov Days 2023. (178-184). https://doi.org/10.1145/3603304.3603331

[7] CHI '94: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - April 1994 - Pages 152–158. https://doi.org/10.1145/191666.191729

[8] "Designing the User Interface: Strategies for Effective Human-Computer Interaction" - Ben Scheiderman

[9] GitLab, *What is version control? [Online], Available:* https://about.gitlab.com/topics/version-control/

[10] Canvas Network, *Canvas Network About Us [Online],* Available: https://www.instructure.com/canvas-network-about-us

[11] Ispring Solutions, *Fast eLearning authoring toolkit [*Online], Available: https://www.ispringsolutions.com/ispring-suite

[12] Jetbrains, *Getting started - JetBrains Academy [Online]* Available: https://plugins.jetbrains.com/plugin/10081-jetbrains-academy/docs/jetbrains-academy-plugin.html

[13] TechTarget , *Reverse Engineering [Online],* Available: https://www.techtarget.com/searchsoftwarequality/definition/reverse-engineering

[14] Moodle Documentation [Online], Available: https://docs.moodle.org/400/en/Main_page

[15] Moodle, *Backup 2.0 general architecture [Online],* Available: https://docs.moodle.org/dev/Backup_2.0_general_architecture

[16] Moodle source code (November 2023) [Online], Available: https://github.com/moodle/moodle

[17] Tim Neusesser and Evan Sunwall (2023), *Error-Message Guidelines [Online],* Available: https://www.nngroup.com/articles/error-message-guidelines/

[18] Moodle (2023), *Installation Package for macOS [Online],* Available: https://docs.moodle.org/403/en/Installation_Package_for_macOS

[19] Python (2023), *Python 3.10.13 documentation [Online],* Available: https://docs.python.org/3.10/

[20] Wikipedia (2023), *Alan Kay [Online],* Available: https://en.wikipedia.org/wiki/Alan_Kay

[21] Alan Kay (2003), *Dr. Alan Kay on the Meaning of "Object-Oriented Programming" [Online],* Available: https://www.purl.org/stefan_ram/pub/doc_kay_oop_en

[22] Python (2023), *os — Miscellaneous operating system interfaces [Online],* Available: https://docs.python.org/3/library/os.html

[23] Python (2023), *xml.etree.ElementTree — The ElementTree XML API, [Online],* Available: https://docs.python.org/3/library/xml.etree.elementtree.html

[24] Leonard Richardson (2023), *Beautiful Soup [Online],* Available: https://www.crummy.com/software/BeautifulSoup/

[25] Python Markdown library, [Online] Available: https://python-markdown.github.io/reference/

[26] Python (2023), *importlib — The implementation of import, [Online]*, Available: https://docs.python.org/3/library/importlib.html

[27] pyYaml org, *PyYaml Documentation [Online],* Available*:* https://pyyaml.org/wiki/PyYAMLDocumentation

[28] Daria Bastanzhyieva (2023), *How to Structure Your Online Course? [Online],* Available: https://raccoongang.com/blog/how-structure-your-online-course/

[29] Clean architecture : a craftsman's guide to software structure and design - Robert C. Martin (2017)

[30] Edsger W. Dijkstra, *Selected Writings on Computing: A Personal Perspective, Springer-Verlag, 1982. ISBN 0–387–90652–5. [Online]*, Available: https://www.cs.utexas.edu/users/EWD/transcriptions/EWD04xx/EWD447.html

[31] SAP, *Separation of Concerns [Online],* Available: https://help.sap.com/doc/abapdocu_753_index_htm/7.53/en-US/abenseperation_concerns_guidl.htm#:~:text=Separation%20of%20concerns%20is%20a,and%20arrangement%20in%20software%20layers.

[32] Tutorialspoint, *Design Patterns - Strategy Pattern, [Online],* Available: https://www.tutorialspoint.com/design_pattern/strategy_pattern.htm

[33] Google (2023), *Markdown versus HTML, [Online],* Available: https://developers.google.com/style/markdown#:~:text=Markdown%20is%20easier%20to%20write,difficult%20or%20impossible%20in%20Markdown.

[34] RedHat (2023), *What is YAML used for?, [Online],* Available: https://www.redhat.com/en/topics/automation/what-is-yaml#:~:text=What%20is%20YAML%20used%20for,and%20is%20more%20user%2Dfriendly.

List of Figures

Figure 1: Diagram illustrating the flow of the program by showing how the input sources go through the tool and transform into a Moodle backup file which is imported into Moodle. 6

Figure 2: General visualization of the xml files being organized in different directories. On the left we see the top level of the archive file with course level information such as roles.xml or the selected folder containing all activities. In the middle panel we see folders of ‘activities, with a 'page' activity selected. The right panel shows all activity level information in xml files such as grades.xml which contains grade information of this page activity. 14

Figure 3: Partial visualization of moodle_backup.xml 15

Figure 4: Visualization of page.xml 16

Figure 5: Class diagram for the architecture 18

Figure 6: Custom activity service class that new added services must implement in order to add support to new activities which are not default from the tool. 19

Figure 7: Example of a custom activity model class. The only condition is that it should extend from Activity class. 19

Figures 8 and 9 respectively showing mentioned input files and a part of the sections.yaml file. 20

Figure 10: Pseudocode for the main function of the project 22

Figure 11: Bar graph comparing the amount of clicks needed to create activities, sections and courses on both the moodle platform and the developed tool. 24

List of Tables

Table 1: List of libraries used including their version and purpose. 10

Table 2: List of requirements found for this project. 11

List of Abbreviations

FHTW	Fachoschule Technikum Wien
OOP	Object Oriented Programming
SoC	Separation of Concerns
LMS	Learning Management System

Appendix A

[1] *Full class diagram image for figure 5 available at:*
https://www.plantuml.com/plantuml/png/bLVDRXit4BxpAOYSfCH1q1m58qYGDhsqQ0FtLXG
BYZkIsF1dJSwwLjI-UydbSidbMh9yiPR3RxuPpW-7_kYGMUoL90VH0WNLuP4gK5kmZg3fA
1gwDOX6qLPOu2YC9WG5Ig2_6jFwdy-cjmxeRzk__RONjY09ST-4xfXbYdAZEgD1unqUFTg
RUb_ijh1ZlSGlnk-Bxq1_U4y8uP8vD_AUY6WtL6Wat6FW4PkqTj4Rjw5tyUlFlm9c9_OR-gFW
k3eo9PiewIqRi6kYcGAIWLReFN47Ox6PYNR6AeRJyh_apMXZSgSYwfLYzfZNSD-3vXd0lFQ
3G05kTEvJ4WIV-X3lI1kZR0gkdP11wul_yIUS71e9ZYUYJDjBgEEIrcbxvcz8O5ujRNezQz8gH
bxZSnxzmhOWJqGA_MqPlz_P7ixj1QBw6CbeSs3ka5Ig6gYJu0ieYl7OmSpR47JxaqPxh0zR-
l6r5z_F9DYdqutLzdNyB8-UP5dxlI2XZdSuGdOFeDqoNeUb86rKBr7mWn7SXuaR6mwGuDu
BePUo9GatBTXUQx0dah-RcPjCkt-yhBCW3MihSU6-Gn-7F_nFQUbp9Ws8qu3pbZ8zF7R04
J9nrkqT6jLySaxijGADLQcSw-lSq7L96TkhYkJTk_6seIr3vh497CqsTs0VXX2_wPXrqB11l6AJK
qt7SAh56pgc-ePM7RScRrXat8DkXcPPCOxDel8yKEYTiR1h7fMiCSbCvDhG79niuA9QWizL
PrrNdDdx2fOXD6oyM5xeVMopG7f3BJZJMmvFAgNDGUiyyePcndMLsCBjNLXVYiC0s4IzTK
hj85n5tL1ELMRZneNySgD2DDrvmn7K0OBc0fu7IWWcBkFsbhN-wm8cn-dda4x1OmABU74
B8SdMoCjE9I1d4dJBx3NOu1-ugv7DGJYy0XtEW62l8dwsdzRLl5g6HWRnfOfA3RAPPl1wFk
UMd5sGleXoFfKAnlkml40LzZU3qhfewY2zlg6lvf_6ug31XnVRKqfFlDMcNWo4g_2dQi91LFT
W5aZci4dscnZdjcZHF3HZvBsnS8KZsuy4IJTFzF9mDpHElKoOXsRDbrn-AULRpX3sFb2hV8
ue4U3cp7dAXLNIoz583oC93o2J3Br0Q8SPbfqPlfPj7Lh6aSwdS1e0o-4GJbvLpr2ykG5J9TTZ
CupBCek3opiH-v6GP9FU_llsRUpClXOQgn8jNWtL_d3dLx9mTaB2N_hQBr-zVZNbYgIFX5oy
dCuXYbeZv0czPAf-8n52udyv2P3QWuo_apmTejuOnYCXiwkiY27TqDjR5yKVFf3Py2nYLC7c
xw4piQvqodil9wm2bLTPJaVjf90LKv3CCQGkrHc-yZ-eYZ1XXv7jjJ-2Rdib_mS0

[2] CourseAsCode source code: https://git-inf.technikum-wien.at/courseascode/testcourse
Commit: 8fade112071decbc8893f05b9acd34565aff117c