Diese Arbeit wurde vorgelegt am Lehr- und Forschungsgebiet Informatik 9
The present work was submitted to Learning Technologies Research Group

Analyse von Aktivitätslogging-Ansätze und Konzept für Datenkorrelation mit Lernstilen
Analysis of Activity Logging Approaches and Concept for Data Correlations with Learning Styles

Masterarbeit
Master-Thesis

von / presented by
Judel, Sven
335203

Prof. Dr. Ulrik Schroeder
Prof. Dr. Matthias Jarke

Aachen, September 21, 2018
# Contents

**List of Figures** iii  
**List of Tables** v  
**Listings** vii  
**List of Abbreviations** ix  

1. **Introduction**  

2. **Learning Analytics** 7  
   2.1. Definitions 7  
   2.2. Related Fields 9  
   2.3. Learning Analytics Process 9  
   2.4. A Reference Model for Learning Analytics 10  

3. **Moodle** 13  
   3.1. Moodle Core and Plugins 13  
   3.2. Moodle Events 14  

4. **Learning Style Models** 17  
   4.1. Popular Learning Style Models 17  
   4.2. Critiques 18  
   4.3. Felder-Silverman Learning Style Model 20  

5. **Plugin Research** 25  
   5.1. Initial Plugin List 25  
   5.2. Related Plugins 27  
   5.3. Additional Plugins 30  

6. **Plugin APIs** 33  
   6.1. Getting Events 33  
   6.2. Getting Grades 35  
   6.3. Getting Further Informations 39  
   6.4. Final Plugin List 42  

7. **Visualization Plugin Research** 45  
   7.1. Collecting Plugins 45  
   7.2. Plugin Comparison 46  
   7.3. Downloads 47
# Contents

8. Implementation ........................................ 49
   8.1. Architecture ........................................ 49
   8.2. Importing and Storing Learning Style Data .................. 54
   8.3. Analytic View Frontend .......................... 55
   8.4. Backend ............................................ 62

9. Evaluation ............................................. 67
   9.1. Study Preparations ............................... 67
   9.2. Study Conduction ................................. 74
   9.3. Results ............................................ 75

10. Conclusion and Future Work .......................... 79

Appendix .................................................. 82
   A. Bibliography ......................................... 83
   B. Frontend Mockup Example .......................... 87
   C. Handout Felder-Silverman Learning Style Model .......... 89
   D. Handout Index of Learning Styles Results ............. 91
   E. User Study Results .................................. 93
   F. Digital Appendix .................................... 95
List of Figures

2.1. Learning Analytics Process (Deduced from [3][12]) ........................................ 10
2.2. LA Reference Model (Deduced from [3]) .......................................................... 11
4.1. Kolb’s Four-Stage Cycle (Deduced from [40]) .................................................... 18
5.1. Moodle Core Plugin Grouping ................................................................. 28
7.1. Example Data Visualized with Straight Lines .................................................... 47
7.2. Example Data Visualized with Smooth Lines .................................................... 47
8.1. Controller Class Diagram ................................................................. 52
8.2. Formular to Upload a JSON File ............................................................... 54
8.3. Database Table Storing the Learning Style Data .............................................. 54
8.4. Dropdown Element to Select the Visualization Type (on the left) and the
     Value Format (on the right) ................................................................. 56
8.5. Multiselect Element to Select Events to Display .............................................. 57
8.6. Control Element to Select the Time Period ..................................................... 58
8.7. Learning Style Selection with Sample Selection .............................................. 58
8.8. All Four Control Elements for Selecting Learning Styles .................................. 59
8.9. Influence of Users Matching more than One Selected Learning Style ................. 59
8.10. Final First Version of the Frontend ......................................................... 61
8.11. Student View Frontend with Example Visualization of Course Visits ................. 62
9.1. Example of ILS Results from NC State ......................................................... 71
9.2. Course Page of the Course Used for the Study ................................................. 72
List of Tables

4.1. Algebraic Signs as Learning Style Indicators ........................................... 22
4.2. FSLSM Characteristic Overview ............................................................. 24

5.1. Suitable Plugins from the Initial List ...................................................... 26
5.2. Suitable Plugins from the Core Plugin Groups ......................................... 29
5.3. Suitable Third-Party Plugins .................................................................... 31

6.1. Final List of the Plugins Used in the LSA-Plugin ....................................... 43
7.1. Example Data of PDF Views in a Moodle Course ....................................... 46

E.1. Comments of User Study Participants ..................................................... 93
E.2. Wishes and Suggested Improvements of User Study Participants ............... 94
Listings

3.1. Event Namespace ....................................................... 14
3.2. Event Class Header .................................................... 14
3.3. Declaring and Triggering an Event ............................... 15

6.1. First Version of the Developed API ............................... 34
6.2. Basic Version of the API with Filtering .......................... 35
6.3. Function to Get the Total Grades ................................. 37
6.4. Function to Get the Assignment Grade of a Single User .... 38
6.5. Function to Get the Workshop Grades of a Single User .... 39
6.6. Function to Check if a Block Plugin is Installed .............. 39
6.7. Getting the Dedication Time of a Single User .................. 40
6.8. Getting the Dedication Time of All Users ....................... 41
6.9. Getting the Use stats of a Single User .......................... 42

8.1. Load Template Function of Template Builder Class ........... 50
8.2. Handle Access Function of the Overview Controller .......... 51
8.3. Render Function of the Basic Controller Class ................. 52
8.4. Extract templates/chart_export_buttons.php .................. 52
8.5. Chart Export Buttons Render Function of the Overview Controller Class .................. 52
8.6. Overview Controller within the view.php ........................ 53
8.7. Function to Store Uploaded Learning Style Data ............... 55
8.8. Parameter Values for the Visualization Settings ............... 56
8.9. List of Possible Plotable Events for the Assign Plugin ....... 57
8.10. Functions to Prepare and Download displayed Data (Extract of js/chart_export.js) .................. 60
8.11. Plot Function of the Chart Builder Class ...................... 62
8.12. Extract of get_y_value of Chart Builder Class ............... 65
8.13. Extract of get_y_value of Chart Builder Class to Get Quiz Grades .................. 65
8.14. Function to Calculate Relative Values of Chart Builder Class .................. 66
List of Abbreviations

AA .................. Academic Analytics
API .................. Application Programming Interface
CBE ................. Computer-Based Education
CSS .................. Cascading Style Sheets
CSV .................. Comma-Separated Values
EDM .................. Educational Data Mining
FSLSM .............. Felder-Silverman Learning Style Model
GNU .................. GNU is not Unix
HTML ............... Hypertext Markup Language
ID ................... Identifier
ILS .................. Index of Learning Styles
IP ................... Internet Protocol
ISO .................. International Organization for Standardization
JSON ................ JavaScript Object Notation
L2P .................. Lehr- und Lernplattform der RWTH Aachen
LA .................. Learning Analytics
LAK .................. Learning Analytics and Knowledge
LMS .................. Learning Management System
LS .................. Learning Styles
LSA-Plugin .......... Learning Styles Analytics Plugin
LSI .................. Learning Style Inventory
LSQ .................. Learning Style Questionnaire
LTS .................. Long Term Support Release
MBTI ............... Myers-Briggs Type Indicator
MIME .............. Multipurpose Internet Mail Extensions
MOOC ............... Massive Open Online Course
Moodle ............ Modular Object Oriented Development Learning Environment
MVC ............... Model-View-Controller
PC .................. Personal Computer
List of Abbreviations

PDF .................... Portable Document Format
PHP .................... PHP: Hypertext Preprocessor
PLE .................... Personalized Learning Environment
RWTH .................. Rheinisch Westfälische Technische Universität
SoLAR .................. Society for Learning Analytics Research
SQL .................... Structured Query Language
STEM .................. Science, Technology, Engineering and Mathematics
TEL .................... Technology Enhanced Learning
URL .................... Uniform Resource Locator
xAPI .................... Experience API
XML .................... Extensible Markup Language
Abstract

Learning is an individual process where learners prefer different ways to receive, adapt and perceive information, called learning styles. It is assumed, that using learning material which matches these preferred ways, can increase the learning efficiency. To adapt to the needs of the different learning styles it has to be determined where their difficulties are grounded. Looking at performance data like achieved grades in quizzes or assignments is the first step. If performance flaws are detected for some learning styles, the reasons need to be determined. Indicators like the used material (related to a quiz or assignment) or the communication of problems with other learners in the course can be analyzed. The results can, therefore, be used to improve the course by updating or creating material, changing structures or providing other help to support each learning style. Therefore, in this thesis, a Moodle plugin was developed which aims at linking learners learning style data and their activity and performance data within a course. These linked data are presented to the user in different visualizations. This allows the described investigation with results presented in a more human-readable way than just reading big tables containing many log data and numbers. To evaluate the plugin, a usability study was done aiming at detecting usability flaws and determine improvements.
Zusammenfassung

Chapter 1 Introduction

Learning is an individual process in which learners receive, adapt and perceive information in different ways. These ways are referred to as learning styles [16]. Not only the way new things are learned is effected by these styles but also the behavior in Learning Management Systems (LMS) like a Moodle course. Moodle is an open-source LMS designed for educators, administrators and learners to create Personalized Learning Environments (PLE). These PLE can be stand-alone courses or support face-to-face classes in school or university. It is assumed, that if learning is done in the preferred way of a student, the learning can be more efficient [23]. To provide a better learning experience to each student in eLearning systems, it needs to be determined how these different learning styles behave in such an environment. This knowledge can be used to resolve e.g. which learning styles are poorly supported, leading to updating or creating new materials for the course.

Moodle logs many user actions and interactions and also performance data like achieved grades and present them in different ways. Visualizing them with techniques of the field of Information Visualization makes the data more easy to understand, analyze and communicate in contrast to big tables with texts and numbers [24]. But these effects only apply, if the viewer is familiar with the displayed data [14]. In this context, it includes knowledge of learning styles - especially the Felder-Silverman Learning Style Model - and knowledge about the logging in Moodle with its events.

The aim of this thesis was to answer the question of how learning styles can be linked with activity and performance data gathered from a Moodle course. To achieve this goal, several other questions needed to be answered first:

- What and how does Moodle log?
- Which interfaces are available to get activity and performance data from?
- How can the data be extracted from the interface (automatically, manually download, not at all)?

Additionally, it is asked, how the linked data can be provided.

In this context, a Learning Styles Analytics Plugin (LSA-Plugin) for Moodle was developed. This plugin collects, processes and visualizes data of students actions in and interactions with a Moodle course based on the students learning styles. Therefore existing plugins for data collection and visualizations were considered and analyzed to find the most suitable ones to use. This research was split up into three steps:

1. Search for suitable plugins that provide action or performance data

[https://moodle.org](https://moodle.org)
2. Analyzing how the data of the suitable plugins can be extracted and chose the plugins that should be used within the LSA-Plugin

3. Search for suitable visualization plugins and select the most suitable one

Different requirements were set up for the plugins to filter the great number of available plugins and select the most suitable ones for the LSA-Plugin. The learning style data are provided from an external source and the LSA-Plugin needs to offer the possibility to upload a file containing the learning style data sets for the enrolled students in a course.

This paper is structured as follows: Chapter 2 describes the concept of Learning Analytics, discusses definitions of this term and looks at related fields. The cycle of Learning Analytics is presented together with a reference model for Learning Analytics, the LSA-Plugin is mapped to. Chapter 3 shortly introduce Moodle and its event-based logging system and Chapter 4 describes some learning style models, looks at critiques about the concept of learning styles and finally describes the Felder-Silverman Learning Style Model which is used within the LSA-Plugin. Further, Chapter 5, 6 and 7 describe the three steps of research for the LSA-Plugin. At first, a list of promising plugins from different sources is generated, analyzed and filtered and in the next step, the options of extracting data from the leftover plugins investigated and a final list of plugins generated which will be used to get activity and performance data from. In the third step, visualization plugins are searched and compared and the most suitable one selected.

Chapter 8 shows the architecture of the LSA-Plugin, the modular way it is constructed in and how the different tasks like uploading learning style data or generating a chart are done. Chapter 9 describes the usability study used to evaluate the implemented LSA-Plugin. It states the study construction including the pilot study, the actual study conduction and the results. Furthermore, suggested improvements of the study participants are discussed and for each detected usability flaw a solution recommended.

Finally, Chapter 10 summarizes this thesis and lists possible improvements for the LSA-Plugin.
Chapter 2 Learning Analytics

Teaching impacts need to be monitored and adjusted to the demands of the learner to be truly effective [12]. Therefore, teachers need to reflect on their teaching and the learning process of their students. These reflections should be used to constantly analyze and update teaching methods and learning materials. If students got difficulties, teachers need to adjust their material or provide additional support. [11, 12]

Traditional approaches for monitoring and reflection involve student evaluations, analyzing grades and frustration rates, and instructor perceptions gathered at the end of a course. As a consequence, the evaluation suffers from a limited quantity of data, delivered only by busy students and instructors who are willing to share their opinion. Additionally, collected data can be reported very late. [13]

Only measuring the learning outcome to proof learning is also doubtful. Furthermore, only a fraction of students’ behavior is actually perceivable. This applies especially to technology enhanced learning (TEL), where students are not always nearby and not easy to observe [12]. But with this increasing number of educational resources moving online, a great amount of data surrounding interactions with these resources becomes available. Analyzing this data automatically and using the findings and other observable activities of the students to improve teaching and learning emerged from the research field of Learning Analytics (LA) [13].

2.1. Definitions

Many definitions for Learning Analytics are available today. A very popular definition which is also used by Wikipedia is the one given by the first International Conference on Learning Analytics and Knowledge (LAK) in 2011 from the Society for Learning Analytics Research (SoLAR):

“Learning analytics is the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimising learning and the environments in which it occurs.” [27]

This is a very general definition covering a broad range of learning and learning environments. It also can include offline data collected by hand or self-reported data of the learner. The goal of understanding and optimizing learning and environments also does not limit to online learning. Ferguson pointed out, that this definition is coupled with two assumptions: Learning Analytics makes use of pre-existing, machine-readable data and that LA techniques are suited to handle big data which would not be practicable to deal with manually [20].

---

1 https://en.wikipedia.org/wiki/Learning_analytics
2 https://solaresearch.org/
2.1. Definitions

But definitions can change over time e.g. the definition in the Horizon Report 2012 is different from the one given in the Horizon Report 2016:

“Learning analytics refers to the interpretation of a wide range of data produced by and gathered on behalf of students in order to assess academic progress, predict future performance, and spot potential issues. [...] The goal of learning analytics is to enable teachers and schools to tailor educational opportunities to each student’s level of need and ability in close-to-real time.” [32]

“Learning analytics is an educational application of web analytics aimed at learner profiling, a process of gathering and analyzing details of individual student interactions in online learning activities. The goal is to build better pedagogies, empower active learning, target at-risk student populations, and assess factors affecting completion and student success” [33]

While the definition of 2012 is more general and not specific about where the data originates from, the definition of 2016 focuses more on the usage of LA in combination with modern web technologies. Here, online learning activities include the actions performed in a learning management system (LMS) like Moodle. The goals formulated in the two definitions are very similar. Providing teaching to each student’s level of need is a part of building better pedagogics and a possible consequence is the empowering of active learning of all students and therefore at-risk students. One unique aspect is the demand of results being delivered in close-to-real time.

Another definition is given by Tanya Elias in 2011:

“Learning analytics is an emerging field in which sophisticated analytic tools are used to improve learning and education. It draws from, and is closely tied to, a series of other fields of study including business intelligence, web analytics, academic analytics, educational data mining, and action analytics.” [13]

It aims at highlighting the connections of learning analytics to other fields of study. Here, in contrast to the definition of the Horizon Report of 2016, web analytics is again just a part of LA.

A more specified definition is given by Dyckhoff et. al. who seek to unite learning analytics with action research. Action research is a methodology that enables teachers to investigate and evaluate their work to improve teaching practice and assure quality. In contrast to LA which is triggered by observations made based on already collected data, action research is driven by a research question arising during teaching. Action research often uses qualitative methods to generate a holistic picture, while LA is mostly based on quantitative methods [3].

“In the context of our research, we understand learning analytics as the development and exploration of methods and tools for visual analysis and pattern recognition in educational data to permit institutions, teachers, and students to iteratively reflect on learning processes and, thus, call for the optimization of learning designs on the one hand and aid the improvement of learning on the other.” [11]

Besides the differences in some details, all definitions include converting educational data into useful actions to foster learning [3,12]. This is also what the LSA-Plugin aims at. Taking the already collected activity data of the users, merging them with the learning styles of the users and enabling the user to visualize this data in an appropriate way and draw own conclusions.
2.2. Related Fields

As mentioned above, several areas of research converge with LA. Action research was already discussed when talking about the definition of LA by Dyckhoff et. al. [11]. Another area is academic analytics (AA) which describes the application of business intelligence tools and practices in higher education [3, 13]. Business intelligence is described as “a broad category of applications and technologies for gathering, storing, analyzing and providing access to data to help enterprise users make better business decisions” [25]. Examples in AA literature refer mostly to detecting students at risk who might drop out of a course or abandon their studies. The focus is on enrollment management and the prediction of student success with statistical software. It is oriented towards educational institutions but LA includes more stakeholders. LA also include more analytics methods [3].

Another popular research area is educational data mining (EDM). It is concerned with understanding students and the settings in which they learn by developing methods to explore data from an educational context. Data mining methods are applied with the objective to resolve educational research issues. The focus of the educational domain is given in both, EDM and LA, but LA uses more methods than only data mining techniques to support the users in analyzing the learning process [3].

Recommender Systems aggregate data about user’s behavior or preferences to draw conclusions for recommending items he or she might be interested in. This is done with information retrieval and machine learning algorithms like decisions trees or artificial neural networks. These methods are also used in LA but have to be adapted from the domain of commercial recommendations to the educational domain [3]. The LA objective of adapting material and personalize learning is a shared goal within the research area of personalized adaptive learning. There are two possibilities to adapt and personalize. The first is adaptivity and is defined as the ability to modify course materials by varying parameters and rules. The second one is adaptability, the possibility for learners to personalize the course materials on their own. Most literature on personalized adaptive learning focuses on adaptive intelligent educational systems where the presentation of the course material is adapted to each learner based on the information about the learner and the current context. This is done by telling learners what to do next by automatically matching teaching material to the individual needs. But with the concept of personal learning environments, the focus moves towards adaptability such that the learner is able to self-direct, self-organize and self-control his or her learning. Therefore the used system should help the learner decide what to do next by recommending different learning entities [3].

2.3. Learning Analytics Process

Learning Analytics and its related fields have something more in common: Doing it only once is not enough. Especially LA is not about by looking at an online course and how well the different students are doing once and changing material in the currently most needed or desired way. Each change in the course needs to be evaluated. LA is an iterative cycle containing three major steps, as illustrated in Figure 2.1.

As stated above, educational data is the foundation of LA. The first step in LA is to collect data from all educational environments and systems. But these data may be too large or
2.4. A Reference Model for Learning Analytics

To classify LA solutions, Chatti et. al. [3] developed a reference model (see Figure 2.2).

This model is based on four dimensions: **What**, **Who**, **Why** and **How**.

**What** asks what kind of data is gathered, managed and used for the analysis. Based on the fact, that LA is a data-driven approach, it is interesting, where the data comes from. Centralized educational systems like LMSs accumulate large log data of student’s activities and interaction data like reading slides, doing quizzes or watching lecture recordings. If the data comes from distributed learning environments, additional challenges arise identifying users and sessions and merging data from different, heterogeneous sources with different formats. Furthermore, the other challenges of data collection and pre-processing described above arise also for centralized learning environments [3].

**Who** asks who is targetted by the analysis. The answer to this question has a big impact on the development of a LA tool. Stakeholders of such tools include students, teachers, educational institutions, researchers and system designers. They got different perspectives, goals and expectations of the tool and the LA exercise they perform with it. Students will probably be interested in improving their grades while teachers might be interested in improving their teaching by adapting their material. Therefore, LA should provide goal-oriented feedback for each stakeholder. Furthermore, non-data-mining-experts contain irrelevant attributes and need to be merged. Furthermore for the merging of data from different sources, user and sessions need to be identified. Data pre-processing is done to clean and reduce the data and transform it into a suitable format [3].

In the next step, based on the pre-processed data and the analysis objectives, different LA techniques can be applied to explore the data. This is done in order to discover (hidden) patterns. But analyzing and visualizing this mined information is not enough. The primary aim of the whole analysis process is to take actions, including monitoring, analysis, prediction, adaption and personalization [3].

The third step of the LA Process is to improve the previously performed analytics exercise. In the post-processing, new data from additional sources can be added, data sets can be redefined and new indicators identified. With these new elements, the cycle can be done again [3].
2.4. A Reference Model for Learning Analytics

![Diagram of Learning Analytics Reference Model]

**Figure 2.2.: LA Reference Model (Deduced from [3])**

should optimally have the tool embedded into their eLearning toolset. A wider user acceptance serves the intended objective of improving teaching and learning. But involving all stakeholders and fitting their needs is a complex task. Some interests might be contradictory, e.g., teachers feeling controlled if administrators look through courses in order to find some best practice examples. Additionally, privacy concerns need to be considered. Data misuse needs to be prevented, boundaries of analytics defined and the identity of the users protected [3].

**Why** asks for the reason, the system analyses the collected data. There are different objectives of LA and some are presented below.

- **Monitoring and analysis** aim at tracking student activities and generate reports in order to support decision-making. It should support pattern detection such that teachers can make decisions on how to design future learning activities.

- **Prediction and intervention** develop a model that attempts to predict knowledge and future performance of learners based on their current activities and accomplishments. This can be used to intervene if a student tends to have difficulties or even fail the course or is at risk to drop out.

- **Assessment and feedback** have the objective to support the (self-)assessment of improved efficiency and effectiveness of the learning process. Intelligent feedback provides interesting information, based on the learner’s context.

Other LA objectives are **Adaption, Personalization and recommendation** and **Reflection**. They all need a tailored set of performance indicators and metrics. These need to go beyond achieved points or grades to support different ways of learning like self-organized learning or informal learning [3].
How asks for the methods, the system uses to perform the analysis. The methods aim at detecting interesting patterns in the data sets. Most existing LMSs already contain reporting tools that provide basic statistics of students’ activities and interactions with the system. These include the total number of visits, the distribution of visits over time or percentage of material read. Some other methods are:

- **Information Visualization** is about representing the obtained LA results in a user-friendly visual form in contrast to the raw reports and tables of data provided by statistics. Different techniques like charts, scatterplots or maps can be used to present the data in a clear and understandable format. The most difficult part is to select the best visualization to achieve the given analysis objective.

- **Classification** groups data in classes. It is also called supervised learning because the data objects are labeled with pre-defined classes. The function or model distinguishing the data classes is used to predict the class of unlabeled data objects.

- **Clustering**, in contrast to classification, does not know the classes of the data objects in advance. Instead, the available data objects are grouped or clustered, so that objects within a cluster are similar to each other and dissimilar to objects in other clusters.

There are plenty more techniques and methods out there and the main challenge is to select the most suitable one to design and develop a usable and useful tool which helps the users to achieve their analytics objectives. The main questions are how available raw data can be converted into actionable knowledge and it can become a trustworthy empirical base for decision-making [3].

The LSA-Plugin can be mapped on these four dimensions in the following way: At first, the what asks for the data and environments. The environment of the LSA-Plugin is a Moodle course and the data are, on the one hand, the activity data from the users and, on the other hand, the learning style data of the users. Targetted by the analysis (the who) are students, teachers and researchers. Students, their actions and learning style data are analyzed and students are able to compare their actions and performance to the average of the other students with the same learning styles. Teachers can review the usage of their provided activities, learning materials and the interaction between the students in forums or chats. These usages are grouped by the learning styles and the teacher can determine, if and where adaption is needed, to support some students in a better way, increase their success and predict dropouts. Researchers can use the LSA-Plugin to investigate the characteristics of the learning styles. They can ask if strong active learners are more active in forums or chats than strong reflective learners or maybe detect other correlations of learning styles and behavior that was never thought about. But this kind of research clearly needs more than only the data from one course. This also answers the question of why the plugin analyzes the collected data. The question of how the analysis is performed is answered by a combination of classification and information visualization. The classes are the eight different learning styles which are subcategorized in strong, moderate and mild. Visualized are the activities of the grouped user in a way the users can choose by themselves, to transform the raw data into knowledge.
Chapter 3  Moodle

In this thesis the online learning platform Moodle (Modular Object Oriented Development Learning Environment) is used to get activity data from users. Moodle is an open-source Learning Management System (LMS) which is designed for educators, administrators and learners to create Personalised Learning Environments. It is built by the Moodle project which is (financially) supported by a network of over 80 Moodle Partner service companies worldwide. Many universities, schools and companies use Moodle for education and training. There are more than 100,000 registered Moodle sites in more than 200 countries containing over 15 million courses with more than 130 million users [53, 58]. Moodle has been designed to be compatible, flexible and easy to modify. To achieve these goals, Moodle has been written in PHP which runs on any computer with a minimum of effort and was published under the GNU General Public License making it an open-source software. By using common technologies like libraries and abstraction, it achieves a high modularity. Moodle can also be linked to other systems like mailing servers or student directories [10, 42].

Here at RWTH Aachen University, some features of Moodle are used in the L2P [2]. The L2P (Lehr- und Lernplattform der RWTH Aachen (engl. Teaching and Learning Platform of RWTH Aachen University)) is the current LMS for RWTH Aachen University. But it is planned to switch from L2P to a Moodle-based system, called RWTHmoodle [3]. Because of all these factors (open-source, modularity, runnable on all systems with minimum effort and being widely used), Moodle is the most suitable environment to do this thesis with.

3.1. Moodle Core and Plugins

The Moodle core consists of courses, activities and users. For each course multiple enrolment methods (enrolling themselves or being enrolled by a teacher) and different roles (e.g. guest, student, teacher) with different rights and empowerments within a course are provided. E.g. a teacher can upload materials like lecture slides to the course so students can view and download them or view students submissions for assignments and grade them [61].

Moodle can be enhanced easily by including new plugins without adjusting the core. There are already a lot of plugins in the Moodle core like assignments, chats, forums or resources enabling basic teaching and training. But different courses need different plugins based on its purpose and the number of participants. In a programming course, it might be needed to compile and run code which is submitted by students automatically. Additionally, plagiarism needs to be detected and therefore, new plugins are needed [42, 67].
3.2. Moodle Events

Moodle does the needed activity logging itself by collecting large amounts of log data about students’ activities. It is recorded whatever activities are involved, like looking at provided materials (e.g. lecture slides), taking quizzes, submitting an assignment or communicating with other students in a chat or a forum. Additionally, the results of some actions like quiz grades are logged and can be analyzed at any time [49, 63].

Each log in Moodle is a single event in the system. These events are used to describe that something has happened. They are atomic pieces of information and primarily the result of user action (e.g. the user with ID 1 visits the course with ID 17). Other sources of events are the cron process or actions done via the command line by an administrator [46, 49].

Each plugin defines the events it wants to report for itself. The Moodle core provides an abstract base class each plugin can extend for any of its own events. This includes all necessary properties like the eventname, the performed action, the target on which the action is taken, the ID of the course the event happened in and a timestamp. With this approach, events become active objects that are able to provide callback functions (such as the event name and a description). All events are stored in a separate file, usually located in .\classes\event\ in the root directory of the plugin. Also a naming convention for the events is given by `<component>\event\some_object>_verb`. The component is the combination of the plugin type and the plugin name. Additional limitations on the naming are given by a defined list of verbs that can be used. In the code this is applied by defining the namespace `<component>\event` for each new class and naming the classes `some_object>_verb` [46].

For instance, the forum plugin contains an event for creating a post. The PHP file of this class is stored in .\mod\forum\classes\event\post_created.php. The namespace is defined as shown in Listing 3.1 and the class as an extension of the abstract class defined in the Moodle core as shown in Listing 3.2. An event is triggered by the plugin it belongs to and the plugin also decides when an event is triggered as shown in Listing 3.3 taken from the file ./mod/forum/post.php. Some of the parameters for the event are the object ID of the post, the ID of the discussion the post is created in and the ID of the forum containing the discussion and the post.

```php
namespace mod_forum\event;

Listing 3.1: Event Namespace
```

```php
class post_created extends \core\event\base {

Listing 3.2: Event Class Header
```
3.2. Moodle Events

```php
$event = \mod_forum\event\post_created::create($params);
$event->add_record_snapshot('forum_posts', $fromform);
$event->add_record_snapshot('forum_discussions', $discussion);
$event->trigger();
```

Listing 3.3: Declaring and Triggering an Event

The goal is to use these atomic events as indicators to correlate the learning style data with. To achieve this, step 1 is aimed at finding plugins which work with events and logs and filter the best one for later use in the LSA-Plugin.
Chapter 4 Learning Style Models

Learning is an individual process in which learners receive, adapt and perceive information in different ways - each learner has different learning styles [16]. Based on this premise it is assumed, that if learning is done in the preferred style, it will be more effective [23]. There are many learning style models out there, all classifying learners according to where they fit on a scale describing how they work with information. They aim to help to understand the learners better and design own teaching models. Lecturers and teacher can adjust their teaching to support all the different kinds of learners in the best possible way. In engineering education, three learning style models have become very popular because of their high usage as subjects of studies in this and other fields [41]: the Myers-Briggs Type Indicator [37], Kolb’s Theory of Learning Styles [34] and the Felder-Silverman Learning Style Model [18]. Although it was originally developed to assess the various learning styles of engineering students, the Felder-Silverman Learning Style Model (FSLSM) is now used with students across majors and in general science education [15, 41].

4.1. Popular Learning Style Models

The Myers-Briggs Type Indicator (MBTI) is a paper and pencil instrument developed by Isabel Briggs Myers and her mother Katharine Briggs over a twenty year period [37, 39]. This instrument aims at gathering the psychological types defined by Carl Jung in 1921. He saw patterns in human behavior and described them in four categories:

- Are people oriented outwardly or inwardly?
- On what kind of information do people rely?
- How is the judgment of people based?
- How do people interact with the world?

Each of these four categories is scored with the MBTI. Based on the results, a person is assigned to one of 16 (four categories with two faculties each to combine: $2^4$) personality types. The MBTI was designed for use in education, counseling/therapy, career guidance and workplace team-building but gained dominance within the assessment practices of career guidance and personnel selection. Psychologists use it to gather information on possible career paths and job placements. The MBTI is based on an explicit theory of personality and has typically been found to be reliable at the level of the four categories. But there are questions about the reliability and validity of the 16 personality types and also evidence of limited correspondence between the MBTI and other global measures of personality.
4.2. Critiques

The psychologist David Kolb proposed that individual learning styles emerge due to genetics, life experiences and the demands of the current environment [4, 34, 40]. It was developed in the 1970’s and is one of the first learning style models based on an explicit theory. Furthermore, it is said, that it is one of the most influential ones [1, 7]. Kolb regarded learning as a four-stage cycle, which is displayed in Figure 4.1.

Concrete experiments serve as a foundation for observations on which an individual reflects and begins to build a general theory of what happened and what this could mean. Based on these theories and hypothesis, abstract concepts and generalizations are formed and implications of them tested later in new situations. These implications lead back to the first step by making new experiences which can be handled the same way as the previous ones. Kolb introduced four learning styles build on this cycle and states that people naturally prefer a certain learning style. Each learning style describes the preferring of two activities within the cycle. In most cases, people are also good skilled in the activities their learning style refers to. Diverging people like to feel and watch (having concrete experiences and doing reflective observations), assimilating people like to think and watch (doing reflective observations and abstract conceptualization), converging people like to think and do (doing abstract conceptualization and active experiments) and accommodating people like to feel and do (doing active experiments and having concrete experiences).

4.2. Critiques

It is important to note, that the concept of learning styles is not universally accepted. The main critique is that learning style models lack a sound theoretical basis and that the instruments used to measure or determine learning styles have not been appropriately validated [16]. This stretches to a point where is asked if learning styles even exist or whether they are useful when practically applied [11]. It is said that “the literature on learning styles is theoretically incoherent and conceptually confused” [6]. Furthermore, there is clear psychological evidence that presenting learning material and information in a way, matching the preferred learning style of a learner does not increase learning [23]. Coffield gave evidence from empirical studies showing that matching learning styles to tasks decrease educational attainment [7, 8].

The Learning Style Inventory (LSI) was devised by Kolb to test his theory of experiential learning. This instrument was improved over thirty years but found to have low face validity with managers [11]. Furthermore, Geller suggested in 1979, that the LSI fails to distinguish between learning styles of individuals or even large groups [11, 22]. The test-retest reliability of the LSI also has been found lacking [60]. To assess these individual differences in learning preferences, Honey and Mumford invented the Learning Style Questionnaire (LSQ) in 1982. This questionnaire probes general behavioral tendencies instead of asking people directly how they learn [11, 30]. Measuring observable behavior instead of the psychological basis was suggested to be more valid by a study of Allison
and Hayes in 1988 where the LSI and LSQ were compared \[1, 2, 60\]. Additionally, critique in Kolb’s model was also given because it fails to differentiate between primary and secondary learning processes and between learning activities and typologies of learning \[23\].

Another critique lays in the decontextualization of the tests, trying to determine the learning style of a student. But context shapes how an individual behaves. They respond to various challenges throughout a day that require the usage of a broad range of learning styles \[6\]. Questions with general terms like traditional/common way or by visiting a website force the student to make a response based on a limited range of information. The student does not know what problem he or she is facing in which context. A problem in education might be solved differently to problems occurring at home or within friendships. By not knowing the context, learners tend to believe they should solve the problem on their own but in reality, problems can be addressed by a larger collective \[7, 8, 23\]. The context of the learner itself is also not considered. Teachers should consider age, educational level and motivation as well as learning styles because these might change across a range of situations as well as throughout time \[11\]. Additionally, individuals may not be able to assign a category to their behavior and give answers that are socially desirable \[23\].

Referring to the decontextualization of learning styles, Laurillard describes learning styles as context-dependent, rather than applying to individual learners. If individual differences exist, they exist at the level of how the student perceives and interprets his or her learning environment \[36, 60\].

Further critique is on commercialization. Many different learning style models are out there and these are accompanied by different tests to determine which style is best suited for a student \[11\]. Coffield stated in 2012 that the existence of 70 learning styles demonstrates the disorganized nature of this research field. The absence of an agreed model or agreed vocabulary creates further confusion \[6\]. Many of these tests are sold to schools and higher education facilities at varying prices. From tests of $5 per student to training programs for educators of $1225 per trainee \[11\].

But with all these critiques, there is no disputing of the fact that learners have learning preferences. It is the effectiveness of learning styles that is under scrutiny. While individuals have these definite preferences, learning through the preferred style may not result in an increase in learning effectiveness \[11\]. Teachers should avoid trying to categorize or confine learners to one learning style \[60\].

To cope with the critiques, Felder and Spurlin \([19]\) pointed out that several qualifying statements have to be made to clarify the intended uses of the instrument (here referring to the Index of Learning Style (see below)) and guard against possible misuses. At first, it needs to be made clear that learning style dimensions are continua and not either/or categories. These preferences are not a reliable indicator of learning strengths or weaknesses and can be affected by a student’s educational experiences. Profiles, based on learning styles, suggest behavioral tendencies rather than being infallible predictors of behavior. Identifying learning styles is not aimed at labeling individual learners and modifying instructions to fit the labels. Students need skills associated with both learning styles of each dimension and therefore teachers should find a balanced teaching style in which all learners are sometimes thought in a manner that matches their preferences and sometimes in the opposite manner. \([19]\)
4.3. Felder-Silverman Learning Style Model

In this thesis, learning styles refer to the Felder-Silverman Learning Style Model, which is one of the most widely used models [15]. It was originally designed in combination with teaching styles to improve engineering education. Today it is one of the most recognized models for identifying learning styles in eLearning environments among STEM (science, technology, engineering and mathematics) students and often used in technology-enhanced learning. [18,26]. The authors did not create new dimensions but based their model on Kolb’s learning styles and Myers-Briggs Type Indicators.

This model became popular very quickly and was used by other researchers and therefore often discussed and worked on [15]. The authors first introduced five dimensions of learning: Perception, Input, Organization, Processing and Understanding. The Organization dimension, classifying learners as Inductive or Deductive, was omitted a few years after the publication of the paper. Felder came to the belief that at least below the graduate school level, the best method of teaching is induction (called problem-based learning, inquiry learning, or a variation on these terms) in contrast to the traditional college teaching method (deduction) which starts with fundamentals and proceeding to applications. He assumed that more students would prefer induction. However, this assumption was refuted by later sampling. In reality, students prefer deductive presentation by getting told exactly what they need for the exam - not one word more or less. Felder did not want teachers to be able to determine that their students prefer deductive presentations. He feared that they would use this as an excuse to rely on the traditional but less effective lecture paradigm in their courses. This led to the omission of this dimension from the model reducing it to four remaining dimensions.

Moreover, also the input dimension was edited. The first version with the two learning styles Visual, which includes pictures, diagrams, chart, plots, animations, etc. and Auditive, which includes the spoken word and other sounds, did not really cover the written word. While it is perceived visually, it is not processed like a picture. Consequently categorizing it as visual information would be wrong. The brain generally converts written words into their spoken equivalent and process them the same way spoken words are. Therefore the learning style pair was changed to Visual and Verbal were spoken and written words are both included in the verbal learning style.

Now, with these two changes applied, the FSLSM consists of four dimensions with two learning styles each: Processing: Active/Reflective, Perception: Sensing/Intuitive, Input: Visual/Verbal and Understanding: Sequential/Global [17,18,26]. Each learning style describes one preference for working with information or learning itself. In contrast to other learning style models, the categorisation is about tendencies of learners, indicating that even learners with a high preference for certain behavior can act sometimes differently.

- The first dimension states how a learner prefers to process information. Active learners tend to experiment with the learning material. It involves doing something with it in the external world - applying it, trying things out, testing it in some way or explaining it to others. Furthermore, active learners tend to be more communicative and prefer to work and learn in groups where the material and information can be discussed. They do not learn much in situations where they are forced to be passive (like in most classical lectures). Reflective learners, on the other hand, prefer to think about and reflect on the learning material. They prefer to do this alone or in a very small group, e.g. with one good friend. If they can not think about the given material and information they do not learn much (also not given in most lectures). But with active learners in class and the possibility to discuss, argue
and brainstorm, the learning experience of reflective learners is also increased in comparison to classes or lectures where every student is passive. In practice (e.g. engineering) both types of these learners are needed. Reflective learners are the theoreticians, creating mathematical models, defining the problems and propose possible solutions. The active learners evaluate these ideas and design and carry out experiments to find a solution that works. [18 26]

- The **Perception** dimension covers in which way learners like to take information. **Sensing** learners observe and gather data through the senses. They like to learn facts and concrete learning material which they can remember well. Furthermore, they prefer to solve problems with standard approaches and also tend to be more patient with details. While not liking surprises or complications, they tend to be more realistic and practical relating learned material to the real world. They operate carefully which can make the sensing learners look slow. In contrast, **intuitive** learners perceive by the way of the unconscious by doing speculation, imagination, hunches or similar. They like innovation and complications, principles, theories and the underlying meanings while disliking repetition and getting bored by details. Being able to grasp new concepts makes them look fast but in some situations also careless. An important difference between these two learners is the comfort with symbols. Words are symbols and translating them into what they represent comes naturally to intuitive learners, but is a struggle for sensing ones. Because of this struggle, sensing learners might need to read texts several times to understand them, putting them at a disadvantage in timed tests where they might run out of time. On the other side, intuitive learners impatience with details may lead to a too quick answer to a question and careless mistakes. [18 26]

- The previously mentioned **Input** dimension with the two learning styles **Visual** and **Verbal** (at first visual and auditive) is about the preferred way of having information presented. Studies have shown that most people learn in one way and tend to ignore information or learning material presented in another way. Visual learners remember better what they see (pictures, diagrams, etc.), while verbal learners remember textual representations (written or spoken) better. It was shown that most people of college age and older are visual learners which is not very compatible with most classical college teaching, consisting of tellings and lecturing (verbal) by a teacher or presenting textual information on slides or handouts. [18 26]

- The fourth and last dimension (**Understanding**) characterizes learners accordingly to their understanding. **Sequential** learning describes the most formal way of education. It means presenting learning material and information in a logically ordered progression based on a schedule. After one block of material has been covered, the students are tested and if they pass, they move on to the next stage or else they redo the block. Sequential learners prefer small incremental steps and have a linear learning progress, mastering the material and information more or less as it is presented. They solve problems in the same manner by using a linear reasoning process, following logical stepwise paths. **Global** learners, on the other side, can not learn in this way. They use a holistic thinking process, learn in large leaps and need to “get” the whole picture at once. They tend to absorb material and information almost randomly and after learning enough, they start “getting it”. By not giving them all the material and information at once they may be lost for the time a block is presented in the traditional small steps. They will not be able to show just a basic understanding or solving the simplest problems. But after they put all the pieces
together, solve the puzzle, they may understand the material and information well enough to leave sequential learners behind. In problem-solving, they also make intuitive leaps, but might not be able to explain how exactly they solve the problem. What characterizes someone as a sequential or global learner is what happens before the whole picture is understood. Sequential learners can also struggle with not fully understanding the material, but they can nevertheless do something with it (e.g. homework). They are able to use the pieces they have absorbed and logically connected to solve assignments or pass tests. Another difference is found after the global learners got the big picture they still may struggle with some details of the subjects. Sequential learners, on the other hand, may know a lot about a specific aspect but have problems relating it to different aspects of the same subject. [18, 26]

Based on this four dimensions and the two contained learning styles, \(2^4 = 16\) combinations are possible and theoretically, that many kinds of learners could exist. But in practice, everybody uses both faculties of each dimension and the learning styles just show the preferred ones of a learner. To determine the preferences of a learner, Felder and Silverman created a questionnaire (called the *Index of Learning Styles (ILS)* [17]) containing 44 questions with respective two answer options for each, e.g.:

I understand something better after I
(a) try it out.
(b) think it through.

The aim is to compute a value expressing how much a learning style is preferred [17][26]. For each of the four dimensions, the questionnaire contains 11 questions. Both answer options indicate a preference for one of the two learning styles of the related dimension (e.g. the question given above relates to the processing dimension). Only one answer for each question can be chosen and every question has to be answered. If both answer options seem to apply, the more frequently applying should be chosen.

After the questionnaire is finished, the answer is evaluated. If answer option (a) was chosen, the value of the dimension is increased by one and if answer option (b) was chosen, decreased by 1. The result for each dimension is a value between \(+11\) (choosing (a) only) and \(-11\) (choosing (b) only) with steps of \(\pm 2\). The algebraic sign \((+/−)\) indicates the learning style as displayed in [Table 4.1](#).

<table>
<thead>
<tr>
<th>Processing</th>
<th>Perception</th>
<th>Input</th>
<th>Understanding</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Active</td>
<td>Visual</td>
<td>Sequential</td>
</tr>
<tr>
<td>−</td>
<td>Reflective</td>
<td>Intuitive</td>
<td>Verbal</td>
</tr>
</tbody>
</table>

**Table 4.1:** Algebraic Signs as Learning Style Indicators

This value also characterize the preference for a learning style in each dimension in three categories: A value of one or three denotes a *mild* preference, five and seven a *moderate* preference and nine and eleven a *strong* preference. While having a mild preference indicates a good balance in a dimension, a moderate or strong preference indicates that a learner learns more easily in an environment that supports the preferred learning style and have difficulties if not (the higher the value the more these effects apply).
This questionnaire is often used and a well-investigated instrument. Many studies dealt with the analysis of response data to the ILS regarding the distribution of preferences for each dimension and with verifying its reliability and validity [19, 26]. These studies seem to state that the ILS is reliable, valid, and suitable but also issues arose like possibly existing dependencies between some learning styles. To gather more information about the characteristics of learners based on their learning styles, an in-depth analysis of the FSLSM was done by Graf et. al. [26]. With the given descriptions of the learning styles, the related questions in the ILS were grouped according to the similarity of semantics. These groups were analyzed with the goal to detect the most representative groups for each learning style. As a result, it was stated that for active learners trying something out is more important than discussing learning material and information or explaining it to others. On the other side, for reflective learners, the social behavior (working on their own or with a good friend) is more relevant than thinking about the material and information. For sensitive learners, it is more important to get concrete learning materials while being careful with details is not that relevant for this group. Intuitive learners, on the other hand, prefer the abstract material, like new approaches and are also characterized by being not careful with details. For visual learners, only one semantic group exists which states that they like pictures and other visualizations. This group is also highly representative. Verbal learners prefer written words the most and are also strongly characterized by struggling with visual styles. Spoken words represent also a relevant aspect but represent the lowest of each of the three semantic group for this learning style. For both learning styles in the understanding dimension, all semantic groups show high relevance. Sequential learners are detail-oriented, progress sequentially and go from parts to the whole. In contrast, the global learners try to get the overall picture by having a non-sequential progress and work with relations and connections. These results can be used as indicators for teachers and lecturers to improve their teaching for each dimension and its learning style. Table 4.2 summarizes the characteristics of the eight learning style to give an overview of how learners can be supported. For instance, active learners need to try new things out and therefore to support them, the teaching should give them the opportunity to do so. Reflective learners, on the other side, should be able to think about the material and information and to work on their own.
### 4.3. Felder-Silverman Learning Style Model

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Learning Style</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understanding</td>
<td>Active</td>
<td>• Like to try things</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Like to work in groups</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Like to explain their view and interpretation and discuss them with others</td>
</tr>
<tr>
<td>Perception</td>
<td>Reflective</td>
<td>• Prefer to think about the learning material</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Prefer to work alone or with one good friend</td>
</tr>
<tr>
<td>Perception</td>
<td>Sensing</td>
<td>• Observe and perceive the environment with their senses</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Like and can remember better facts and concrete material</td>
</tr>
<tr>
<td>Perception</td>
<td>Intuitive</td>
<td>• Like to fantasize and speculate about the world and try to figure out if they are right</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Like innovations and complications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Like principles, theories and the underlying meanings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Get bored by repetitions and working with details</td>
</tr>
<tr>
<td>Processing</td>
<td>Reflective</td>
<td>• Prefer to solve problems with standardize procedures</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Like principles, theories and the underlying meanings</td>
</tr>
<tr>
<td>Processing</td>
<td>Sensing</td>
<td>• Prefer to work alone or with one good friend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Observe and perceive the environment with their senses</td>
</tr>
<tr>
<td>Input</td>
<td>Visual</td>
<td>• Remember best what they have seen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Pictures, graphics, visualizations, demonstrations, ...</td>
</tr>
<tr>
<td>Input</td>
<td>Verbal</td>
<td>• Remember best spoken or written words</td>
</tr>
<tr>
<td>Understanding</td>
<td>Sequential</td>
<td>• Get bored by repetitions and working with details</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Like principles, theories and the underlying meanings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Like innovations and complications</td>
</tr>
<tr>
<td>Understanding</td>
<td>Global</td>
<td>• Prefer to work alone or with one good friend</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Observe and perceive the environment with their senses</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Like and can remember better facts and concrete material</td>
</tr>
</tbody>
</table>

Table 4.2: FSLSM Characteristic Overview
Chapter 5 Plugin Research

In order to be able to create the LSA-Plugin, the first step was to look at already existing plugins of Moodle. These core and additional plugins were searched, filtered and evaluated based on a developed list of interesting data:

- Data on quiz views, attempts and results
- Data on the forum views and updates
- Data on chat views
- Data on gradebook results
- Total time spent on course
- Number of total views
- Clicks on learning materials
- Video views

Moodle logs many user actions and interactions and also performance data like achieved grades and present them in different ways. These logs can be viewed with different plugins but many only provide limited events (e.g. views of course or views of PDFs), or display the data in big tables that are hardly human-readable. Therefore, a set of plugins has to be assembled that implements the given list.

The special challenge regarding the views of learning materials refers to missing events triggered by the Moodle module Label. It allows adding text, pictures, links, videos and other media to the course page. But just the access of the course page is logged which does not state if a user reads through the label or watches the video. Therefore, only visits of materials added with the File module can be counted.

5.1. Initial Plugin List

This research started with a list of plugins which provide learning analytics given by Moodle. For each plugin, it was given the plugin type, if it is one of the core or an additional plugin, for whom it is useful, a short description and the reported usage. This usage is based on the Report of the Plugins Usage Survey[1] which was open in 2015 [55]. 353 responses were collected before the survey was closed. Primarily targeting administrators of registered Moodle sites, the survey focussed on capturing usage of different types of plugins on Moodle sites. The objective was to determine, whether plugins were used or not [48].

The survey did not cover all the plugins on the learning analytics list and therefore only for 15 of the 31 plugins the reported usage were given. Before evaluating the plugins of this list, the attribute *Latest supported Moodle version* was added. If the plugin did not support a Moodle version higher or equal to version 3.2, it was not suitable for this thesis. This decision is based on multiple factors. The first limitation on suitable Moodle versions was the fact that the last long-term support release (LTS) at the beginning of the research was version 3.1 released on 23 May 2016 [57]. Most of the plugins on the initial plugin list support the latest Moodle version (at the time of the research, it was 3.4). Only two plugins support up to Moodle version 3.3, one Moodle version 3.2, two Moodle version 3.1 and two Moodle versions below 3.1. By looking at the plugins supporting versions 3.1 to 3.3 the one supporting up to Moodle version 3.2 appeared as a suitable plugin and therefore it was to decide if version 3.1 is also still suitable. Considering the improvements of version 3.2 in comparison to version 3.1 and the fact that none of the two plugins that support Moodle up to version 3.1 is suitable for this thesis, it was decided to use version 3.2 as the threshold for all plugins.

Based on this attribute four plugins (*Progress Bar*, *Configurable Reports*, *Engagement Analytics*, and *GISMO*) were rejected. *Progress Bar* also links to the plugin *Completion Progress* which is a faster, more efficient and easier to use plugin and the successor of *Progress Bar*, but this plugin was also rejected because it just visualizes the course completion progress which is no data that can be used in a meaningful way in this thesis.

The next step was to look at the remaining plugins, starting with the ones of the Moodle core. Each plugin entry in the Moodle Documentation was read to find out, what the plugin is about and if it is suitable. Plugins where rejected if they just visualize data (e.g. the plugin *Course overview* creates a graph showing the activity in a course - calculated by the number of views and times specific activities like quiz submissions or responses to a discussion forum were done - without providing details on what was done exactly or by which user), provide too abstract information (e.g. the plugin *Activity Completion* allows a teacher to select course activities which completion he or she wants to track, but it is not logged, when an activity was completed or how (no results of quizzes)) or do not log anything interesting (e.g. the plugins *Feedback* and *Survey* allow to create surveys and let users answer them - even if they ask for user activities, automatic data mining is more consistent and cannot be lied about).

After this filtering, three of the 31 plugins remained, listed in Table 5.1:

<table>
<thead>
<tr>
<th>Plugin Name</th>
<th>Type</th>
<th>Stand./Add.</th>
<th>Rep. usage</th>
<th>Category</th>
<th>LSMV*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logs</td>
<td>Report</td>
<td>Standard</td>
<td>71.4%</td>
<td>Course reports</td>
<td>3.4</td>
</tr>
<tr>
<td>Overview report</td>
<td>Report</td>
<td>Standard</td>
<td>N/A</td>
<td>Grades</td>
<td>3.4</td>
</tr>
<tr>
<td>Course dedication</td>
<td>Block</td>
<td>Additional</td>
<td>N/A</td>
<td>Blocks</td>
<td>3.2</td>
</tr>
</tbody>
</table>

*Latest supported Moodle version

The *Logs* plugin logs all triggered events in a Moodle course for each activity. It provides a tabular view of these data and the possibility to download them amongst other things as a CSV-file. Additionally to the name of the event and the user who triggered the event, it also logs the timestamp (in milliseconds from the Unix epoch) and the object class and object name of the event. The plugin can then be used to analyze user behavior in a more detailed way.

---

2 https://moodle.org/plugins/block_progress
3 https://moodle.org/plugins/block_configurable_reports
4 https://moodle.org/plugins/report_engagement
5 https://moodle.org/plugins/block_gismo
6 https://docs.moodle.org/34/en/Logs
the event, it shows the time and date, the affected user and other details like the IP address. For more specific results, Logs allows filtering for participants, a special day, an activity, the actions (create, view, update, delete, all changes), the source (CLI, Restore, Web, Web service, other) and the education level (participating, teaching, other). While the teaching events refer to events or actions usually performed by a teacher which affect the students’ learning experience (e.g. creating a quiz), the participating events refer to events or actions which could be related to a user’s learning experience (e.g. submitting an assignment) [50].

Overview report [7] is a plugin which shows the grade of one user for each course he or she is enrolled in. It does not show the single grades of each grades plugin within the courses. For example, if a course got only two quizzes and nothing else that can be graded, the grade shown in Overview report is the sum of the grades of the two quizzes. Disadvantages are that the plugin interface displays the grade of just one user at a time and there is no possibility to download a whole list [54].

The plugin Course dedication [8] estimates the dedication time to a Moodle course by the participants. This is done by using the Session and Session duration concepts applied to Moodle log entries. Every time a user accesses to a page in Moodle, a log entry is stored. Each series of two or more consecutive clicks in which the elapsed time between every pair of successive clicks does not overcome a fixed maximum time is called a Session. The Session duration is the time between the first and the last click of the series. In the plugin, the limit between clicks can be fixed between 1 and 150 minutes. It allows viewing the dedicated time spend by each participant for the whole time the course is active or a specific period within this time and to download the resulting table as a CSV-file [5].

The maximum time for a timeout has to be chosen carefully. E.g. if the time is set to 30 minutes, the user can be inactive or even logged out and come back after 25 minutes and the plugin will add the time to the whole dedication.

Additionally, two plugins were marked which did not provide log data but were interesting in other aspects: Ad-hoc database queries and Logstore xAPI.

Ad-hoc database queries [9] allows administrators to set up arbitrary SQL select queries. These queries act as ad-hoc reports and can be executed by any user with the appropriate permissions. The results of a query are displayed as an HTML table and also downloadable as a CSV-file. This should avoid the workload of creating a whole new admin report plugin [31].

Logstore xAPI [10] is a plugin to emit events from the Moodle logstore as xAPI statements. These statements are structured by using nouns, verbs and objects. If possible, the plugin could be used to export the analyses from the LSA-Plugin as xAPI statement to support a consistent and established format in learning analytics. This also allows the usage of the generated data by other plugins or learning analytics software [64, 65].

5.2. Related Plugins

The next round of finding and evaluating plugins was to look at related and similar plugins of the Moodle Core. Therefore, the 17 plugins of the Moodle core on the initial list were grouped based on their category in the Moodle Documentation which is visualized
5.2. Related Plugins

Subcategories refer to categories included in other ones like Quiz activity contained in Activities.

![Figure 5.1: Moodle Core Plugin Grouping](https://docs.moodle.org/34/en/Grader_report)

The plugins of the Activities category are all activities which user interactions were logged by the Logs plugins and also provide no additional logging or desired user information (e.g. the results of choices are not interesting by now). Only the grades of Assignments are to get but the activity plugin itself does not provide such a functionality. The grades can be viewed in the Grader report amongst the grades of quizzes, workshops or other activities. Therefore this plugin was noted to may be used later.

More interesting were the plugins of the subcategory Quiz reports. Two of these plugins (Quiz grades report and Quiz responses report) provide the results of all quizzes ever done and e.g. not only the grade of the best one of each course participant. But while Quiz grades report adds the points achieved in each question of a quiz, Quiz responses report adds the answers given by the participants. Because the points of the questions provide more basic feedback than the given answer, Quiz grades report will be used in this thesis and Quiz responses report is rejected. It is easier to see that some learning styles struggle with some questions by seeing lower points than the exact answer, leading to looking up if it was a correct one or not. In comparison to the Grades report it provides more details on the results of quizzes but not on the other activities so it can not replace it.

[11](https://docs.moodle.org/34/en/Grader_report)
5.2. Related Plugins

The additional plugins of the Blocks category were all considered as not suitable. Most of these plugins are used to simplify the usage of Moodle by adding different features. For example, enables the plugins Global search block a user to search the whole site (not only one course) for specific content. Other blocks connect a Moodle site or course to third parties like the Flickr block which displays photos based on used tags or the Youtube block which pulls videos with defined tags into the course. Blocks supporting admins or teachers like Mentees block (quick access to mentees profile pages) did not any logging and did not provide interesting information. The block Quiz result block itself suggests using the Activity result block instead which was already rejected in the first round as it has to be activated and configured first and there are other plugins already providing the information this block could provide and it would just display them in the course.

In the Tracking process category at first, the subcategory Grades was investigated. The Grader report was found earlier with the disadvantage of a missing export of the grades. Grader export provides this export feature and is therefore added to the list. Based on how easy the export of the grades will be (maybe Grader report provides an API for an automatic export) one of the two will be used (see Chapter 6). Most of the subcategories of Grades were explanations on how grades in Moodle work (e.g. Managing grades explaining terms like Grade item or Grade letters). The remaining plugins were also not suitable because they introduce a new system (e.g. Scales) or do not log or provide data (e.g. Grade import).

The other two subcategories were about Competency-based education (CBE) and Badges. While CBE has to be activated and configured and only log specific events referring to it could be gathered, it was rejected. An event, logging the earning of badges can also be seen with the Logs plugin and therefore no further look was taken at this subcategory.

At the end of the second round of the plugin research, three more suitable plugins as listed in Table 5.2, were found:

<table>
<thead>
<tr>
<th>Plugin Name</th>
<th>Type</th>
<th>Stand./Add.</th>
<th>Rep. usage</th>
<th>Category</th>
<th>LSMV*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grader report</td>
<td>Report</td>
<td>Standard</td>
<td>N/A</td>
<td>Grades</td>
<td>3.4</td>
</tr>
<tr>
<td>Quiz grades report</td>
<td>Report</td>
<td>Standard</td>
<td>N/A</td>
<td>Quiz reports</td>
<td>3.4</td>
</tr>
<tr>
<td>Grade export</td>
<td>Report</td>
<td>Standard</td>
<td>N/A</td>
<td>Grades</td>
<td>3.4</td>
</tr>
</tbody>
</table>

*Latest supported Moodle version

The Quiz grades report is a part of the quiz activity and can be viewed within the quiz administration. Therefore only the results of a single quiz can be viewed and no comparison to other quiz results are possible within this plugin. It shows the achieved grade for each attempt together with the date and time the attempt was started and completed. Additionally, it shows the achieved points for each question of the selected quiz. The tabular presentation can be downloaded in different formats and all scores are visualized in a graph. For more details on a student’s attempt, the specific attempt can be clicked and the view of the student on the selected attempt is displayed.

The Grade export plugin is part of the grade administration of a course and allows exporting the grades of the course without presenting them in a tabular or visual form. They can be exported to OpenDocument spreadsheet, plain text file, Excel spreadsheet and XML file. It can be selected which grades should be exported. Besides the total grade, every single gradable activity like quizzes or assignments and also earning badges can

---

12 https://docs.moodle.org/34/en/Quiz_grades_report
13 https://docs.moodle.org/34/en/Grade_export
be included in the exported report. Additionally, settings can determine if only new or just updated grades will be exported or if user profile fields should be included. It had to be determined if these exported data are good enough to do the desired visualization in the LSA-Plugin and make the plugins Overview report and Quiz grades report dispensable.

5.3. Additional Plugins

The third and last round consists of searching the Moodle plugins at https://moodle.org/plugins/ with some keywords. The website allows filtering Moodle plugins by type, version or using keywords. Matching plugins are presented in groups of 30 plugins at a time. The user can display up to 30 more (or fewer if less than 30 plugins remain) by clicking a button that triggers a GET-Request.

Processing packages of 30 plugins manually take a lot of time. Therefore the website was investigated to get to know how the results are received and displayed. This is done by a POST-Request to an ajax service of Moodle with some arguments in the request body, amongst other things the search query and the batch number. This batch number is increased for each time, the user wants to see more plugins, initially starting with 0. The results of these requests are up to 30 plugins or none if the batch number is too high and there are no plugins remaining. For each plugin, there are given a lot of information from which are some displayed on the website like the name, the short description, the main screenshot (if available) or the total number of downloads. Additional information where the ID of the plugin, the index in the search query or the plugin type.

A script was written in JavaScript with the usage of the jQuery library for the communication with the Moodle service to automatize the collection of plugins found by searching for different keywords. It was not filtered for purpose, plugin type or Moodle version to get as many results as possible so no maybe interesting plugin will be missed. The script took a keyword, used it as the search query and increased the batch number as long as the result of the request contains one or more plugins. From each plugin the ID, the index, the name, the short description, the plugin type and the URL was stored in an array.

The next challenge was to get the latest supported Moodle version of the plugins which was not given in the response to the requests. This information needed to be received from another Moodle website. The website https://moodle.org/plugins/pluginversion.php takes either a plugin name or an ID as a parameter to display information about previous versions of the plugin and the current versions and the Moodle versions supported. Based on the url of a plugin, the script extracted either the name (e.g. `mod_attendanceregister` from https://moodle.org/plugins/mod_attendanceregister of the plugin Attendance Register) or the ID (e.g. 447 from https://moodle.org/plugins/view.php?id=447 from the plugin Kaltura Video Package) and did a GET-Request for the whole website (e.g. https://moodle.org/plugins/pluginversions.php?plugin=mod_attendanceregister or https://moodle.org/plugins/pluginversions.php?id=447). On the received websites the first appearance of the class `moodleversions` is searched and the length of the span element calculated. After that, the text within this element is split up at each comma and the last entry of the resulting array taken as this is the latest supported Moodle version which is added to the attributes of the belonging plugin in the local array of the script.

---

14 https://jquery.com/
The last step was to export the array in a suitable way. Based on the fact that multiple runs of the script were done with different keywords, an easy to merge format was necessary and because some future work had to be done with the resulting lists (see below), each array was exported as a CSV-file. To do this, linebreaks had to be removed from the name and short description and also each semicolon as this was used as the separator in the CSV-file. Finally, the values of each plugin in the array were written as one line in the CSV-file with additionally the keyword, the plugin was found with and the resulting file downloaded automatically.

The script was run with five keywords: report (206 results), log (367 results), activity (252 results), analytic (25 results) and tracking (23 results). These keywords come from the fact that most of the considered plugins were from the plugin type Report and were about learning analytics as it was called in the initial list. How this is done is described above as activity logging and therefore these keywords were used too. An Excel macro was used to merge the five CSV-files into one Excel sheet. This generated list was sorted by the unique ID of each plugin and checked for multiple entries. Plugins which occurred on two or more lists were reduced to just one entry, merging the keywords to make the detection of duplicates possible. As a result, 201 plugins were removed from the list. The next step was to sort out each plugin which is not available for Moodle version 3.2 or higher resulting in the rejection of another 310 plugins.

The remaining 362 plugins were evaluated manually. At first, the plugin type and short descriptions were considered. Themes just add new layouts for the Moodle site and courses. Authentification, availability, enroll and plagiarism detection plugins were rejected because they add new functionalities and log (if any) only activities within themselves. If the short description did not give enough information, the noted URL was visited and the decision whether the plugin is suitable or not was made based on the plugin description and additional information and the criteria already used for evaluating the plugins on the initial list. A special case was the plugin IntelliBoard. It uses the service of IntelliBoard as a third party but this service has to be paid. The free version only supports 500 users which is a strong limitation to the analysis leading to the rejection of the plugin. Plugins which were considered suitable were installed, tested and evaluated. By this amongst other things the plugin Attendance Register was rejected. It was first considered as an alternative for the Course dedication plugin but needed more configuration and was not that easy to use and therefore removed from the list of suitable plugins. Only one plugin was considered suitable which details are shown in Table 5.3:

<table>
<thead>
<tr>
<th>Plugin Name</th>
<th>Type</th>
<th>Stand/Add.</th>
<th>Rep. usage</th>
<th>Category</th>
<th>LSMV*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Stats</td>
<td>Blocks</td>
<td>Additional</td>
<td>N/A</td>
<td>Blocks</td>
<td>3.4</td>
</tr>
</tbody>
</table>

Table 5.3.: Suitable Third-Party Plugins

*Latest supported Moodle version

Use Stats\footnote{https://moodle.org/plugins/block_use_stats} samples the log records of a specific user to estimate the time he or she has spent in the course. In contrast to the course dedication plugin, it does not rely on sessions but on the actual events triggered by the user within an adjustable threshold. The creators of the plugin based this decision on the hypothesis that all activities in Moodle underlie a constant logging track generation. They also state that the more Moodle is used as a daily content editing tool, the more accurate the report becomes \cite{21}.
5.3. Additional Plugins

With these six plugins (Logs, Overview report, Course dedication, Quiz grades report, Grade export and Use Stats), the last step of comparison could be done. Comparisons like Overview report vs. Quiz grades report vs. Grade export were done within and based on the results of the second step and partly on how the data could be exported from the plugins by the LSA-Plugin.
Chapter 6 Plugin APIs

The second step was to find out how to extract the data from the plugins. This criterion also influenced the decision on which plugins will be used in the LSA-Plugin.

6.1. Getting Events

The Logs plugin is the only plugin considered suitable to get the desired event data from. The wish for this API was to take the same parameters as the plugin provides filters in its interface (course, user, date and activity). These data should be used to extract the desired data for the events and selected time period.

At first, the lib.php of the plugin was searched for a usable function. This file contains functions for extending the navigation, verify if a given store instance is supported or if a user is allowed to view the report. No function returns logged activity data. The next step was to look at the locallib.php. It contains a function to get the log records for a specific course and user. Another function allows filtering the returned log set for a specific day. Other functions were provided to generate graphs or prepare data to be used with the Moodle API Charts API. The most promising function report_log_usercourse was described in the comment with Select all log records for a given course and user. Returned is an array of stdClasses containing two attributes where the first attribute, day, describes the days passed since the selected course started and the second one, num, how many events were logged for the selected user in the selected course. No additional information were given and also no other function returned the desired data.

Because none of the two files containing public usable functions provide the desired functionality and no other plugin was considered suitable to deliver event log data, an own API had to be designed and implemented.

6.1.1. Designing the API

To know how to get the data, it was necessary to determine where the data is stored. Therefore the function report_log_usercourse from the locallib.php of the Logs plugin was considered again. The SQL query accesses a database table which name is returned based on the instance of the reader of the used log manager. Reader and log manager are parts of the new logging system, implemented for Moodle version 2.7. The log manager contains all enabled log store subplugins and states in which order they appear. Three log stores are implemented in the Moodle core:

- Standard log
- Legacy log
- External database log
6.1. Getting Events

Standard log is the primary log store and replaces the log table of the old logging system. The legacy log itself does not conduct any log writing but works as an emulation layer allowing new reports to access historical data. It translates the existing log table into the new format and transforms new SQL queries to match the legacy table format. External database logs stores data in a predefined external database. The Moodle core does not provide an interface to read from such external databases. The next step is to look at the readers. In the \texttt{report\_log\_usercourse} function, there is an optional parameter $\logreader$ which can be used to address a specific log reader. If the parameter is empty, the first log reader is used. Some detections have to be done to make sure an existing log reader is selected, the list of log readers is not empty and based on the selected log reader, which database to access.

6.1.2. Implementing the API

With this knowledge, a first version of the API was written, which is shown in Listing 6.1, returning everything contained in the referenced database.

\begin{verbatim}
private function first_api_version(){
    global $DB;
    $logmanager = get_log_manager();
    $readers = $logmanager->get_readers();
    $reader = reset($readers);
    $logtable = $reader->get_internal_log_table_name();
    if ($reader instanceof logstore_legacy\log\store){
        $logtable = 'log';
    }
    return $DB->get_records_sql('SELECT * FROM {' . $logtable . '}');
}
\end{verbatim}

\textbf{Listing 6.1: First Version of the Developed API}

In the test Moodle setup, only the standard log was enabled. The first entry of the returned records was taken to look at the database schema. Additionally, the overview of the Moodle database schema from Zoola Analytics was read to see connections between single tables. The important fields are

- \textit{id}
  
  The ID is needed because each returned set of records by a SQL query needs a unique identifier. Because the targeted set of records is a subset of the records of all event logs and no merging with other tables is needed, the ID of the log table is sufficient.

- \textit{eventname}
  
  This field stores the eventname of the triggered event in the given naming convention described in section 3.2.

- \textit{contextinstanceid}
  
  The context instance ID refers to the course module the event was performed in, with or by. This ID is unique for the whole Moodle site which made an additional filtering for the ID of the course unnecessary.

\footnote{https://moodleschema.zoola.io/index.html}
6.2. Getting Grades

- **userid**
  This field stores the ID of the user who performed the action, triggering the logged event.

- **timecreated**
  The data and time the event was triggered are stored in this field as an unix timestamp.

These fields match the requirements on filtering like in the graphical interface of the logs plugin. In the next step of the implementation, the filtering options were added to the SQL query. The resulted code is shown in Listing 6.2. Filtering for a user ID was omitted to get a whole list of event logs and filter the users in a later step based on which learning styles should be displayed. This should prevent doing SQL queries multiple times with just one parameter changing.

```php
private function second_api_version(){
    $logmanager = get_log_manager();
    $readers = $logmanager->get_readers();
    $reader = reset($readers);
    $logtable = $reader->get_internal_log_table_name();
    if ($reader instanceof logstore_legacy\log\store){
        $logtable = 'log';
    }
    return 'SELECT * FROM {' . $logtable . '} l' .
    ' WHERE l.eventname = :eventname AND l.contextinstanceid = :contextinstanceid' .
    ' AND l.timecreated >= :timestamp_lower AND l.timecreated <= :timestamp_upper';
}
```

**Listing 6.2: Basic Version of the API with Filtering**

Another change made is not returning the result of a SQL query but the SQL query itself as a string. This decision was based on the aim of making the functions modular. Another function is needed to get the necessary parameters for the filtering and therefore a third function will be used to do the actual execution of the query with the selected parameters. The last step needs to be the adaption to the legacy log but because reasonable data needs to be generated, this would take too long to be part of the first version of the LSA-Plugin. The foundation for further adaptions is given by the if-clause which asks for the kind of log reader that is chosen from the log manager. Later, a parameter for choosing a specific log reader can be added.

6.2. Getting Grades

Three plugins were listed to get grades from:

- Overview report
- Grader report
- Quiz grade report
- Grade export
6.2. Getting Grades

For quizzes is it interesting to get as well the best grade of a user as the average grade. The average grade has to be computed by dividing the sum of the points achieved in each run of the quiz by the times of runs the user has done. If either grade report or grade export are capable of returning these grades, the quiz grade report could be omitted. Furthermore, if one of them also returns the total grade of a course or enables computing it, the overview report could be omitted too.

6.2.1. Grader report vs. Grade export

Therefore the comparison of the grade report and grade export was done at first. Grade export provides one lib.php but no locallib.php. This file only contains a class grade_report_grader which extends the class grade_report. No further functions were given and the class does not provide any static functions. Conclusively no API is given in the plugin.

Grade export also only got a lib.php. This file contains an abstract class grade_export, a class grade_export_update_buffer and a public function export_verify_.grades. The public function returns for a given course ID if there is a valid set of grades to export and similar to the function of the grader report, the class grade_export_update_buffer does not provide any static functions. The abstract class provides two abstract class which both do not return any grades. Therefore also grade export does not provide an API.

Before designing an own API for getting the grades, the two provided classes in the lib.php of the two plugins were studied. grade_export_update_buffer is designed to update exported fields and no function returns any grades. With these results, the plugin Grade export is unsuitable and omitted. The class grade_report_grader of the grader report is described in the comments as a “class providing an API for the grader report building and displaying” (line 29). In this class, grades are represented by grade items - the way they are stored in the database. The class is aimed at generating HTML code for displaying the table viewable in the graphical interface of the plugin. It is possible to instantiate the class within the LSA-Plugin and do the necessary preparation (loading users and grade items from the database) to get this code. This code could be parsed to extract the desired information about the grades but this is not desired. As stated above, the desired data should be returned by one function call. Instantiating a class, doing preparations and also needing to parse the results is too much overhead. Concludingly also the plugin Grader report was omitted.

As a result, no plugin was available to return all grades for each enrolled user in the course, the LSA-Plugin is used in. The alternative approach was to get the grades from single sources.

6.2.2. Overview report

At first, the overview report was investigated to find an API for the total grade of a course. The lib.php of this plugin provides the class grade_report_overview, also extending the grade_report class which is described as a “class providing an API for the overview report building and displaying” (line 29). Additionally, two public functions are defined where one returns the setting definitions for an object of the Moodle Forms API and the other one adds nodes to the myprofile page. The grade_report_overview class does not have any static function returning grades and an object of this class is used to return the HTML code for the table which is displayed in the graphical interface of the plugin. This is also not suitable and therefore the plugin was omitted and because no
6.2. Getting Grades

other plugin was considered suitable to return the total grade during the plugin research, an own API needed to be written.

But during the testing of the grade_report_grader class, the lib.php of the grades itself necessary used because it contains the grade_item class. During the research for the right library to include, the querylib.php of the grades were also studied. This file contains a public function grade_get_course_grades which takes a course ID and one or more user IDs and returns the total grades for these users in the given course. Therefore, this API was used in the function shown in [Listing 6.3] to return the total grades. A variable is used to store the pulled grades. If the variable is empty, the API is called and the grades extracted from the returned standard class.

```php
private function get_total_grades_of_current_course() {
    global $CFG, $COURSE;

    if (empty($this->total_grades)) {
        require_once($CFG->dirroot.'/grade/lib.php');
        require_once($CFG->dirroot.'/grade/querylib.php');

        $this->total_grades = grade_get_course_grades($COURSE->id, get_all_user_ids())->grades;
    }

    return $this->total_grades;
}
```

Listing 6.3: Function to Get the Total Grades

6.2.3. Quiz grades report

The quiz grade report as a part of the quiz plugin has its code also contained in the quiz directory (mod\quiz\report\grading). But this folder did neither contain a lib.php nor a locallib.php. These files can be found in the quiz folder itself. The lib.php file contains two functions providing the desired functionality of getting a user’s best and average grade:

- **quiz_get_best_grade**
  This function takes an object containing the module instance ID of a quiz and a user ID and returns the best grade the user has achieved in the selected quiz.

- **quiz_get_user_attempts**
  The two parameters for this functions are a module instance ID of a quiz (no need for an object) and a user ID. It returns all achieved grades and additional information like the date and time the quiz was graded. Two optional parameters allow further filtering. One filters for the status of an attempt and is set to finished per default.

Because of the planned filtering for special time periods, the API function for getting the best grade was rejected. By getting only the overall best grade, it would not be possible to view e.g. only the best grade achieved in the last week. Instead, all user attempts are filtered by the timestamp and after that, the best grade is computed from the left attempts. The implementation of these features can be seen in [Listing 8.13].
6.2. Getting Grades

6.2.4. Further Gradable Plugins

Two more gradable activities should be supported by the LSA-Plugin:

- Assignment
- Workshop

Even though Badges are also listed in the grader report, they are earned and therefore just summing up how many are earned without knowing which is not detailed enough. Therefore each badge instance is handled as an own activity with one event *Badge earned*.

Assignment Grades

The assignment plugin does not contain an own report and it was directly looked at the *lib.php* and *locallib.php* of the *assign* directory. Currently, Moodle contains as well an *assignment* folder as an *assign* folder. The assignment contains the old plugin which was replaced by the assign plugin in Moodle version 2.3 [43]. In the *lib.php* there is a function `assign_get_user_grades` which takes an instance object of the assign plugin and a user ID as an input. If the user ID is set to 0 (default value), the grades of all users who submitted a solution for the given assignment are returned. To reduce the overhead of calling the API for every single user, a variable is used to store for each assign instance the grades. If this variable does not contain any entries for the given instance ID, the API of the assign plugin is called. The assign object is taken from the database, given to the API and the returned grades stored in the responsible variable with the right key. If no entry is available for a specific user ID which indicated that the user has not submitted any solution (yet), 0 is returned. The code of the function is shown in [Listing 6.4](#).

```php
private function get_assign_grade_of_user($assignInstanceID, $userid){
    if(empty($this->assignGrades)){
        $this->assignGrades = array();
    }
    if(empty($this->assignGrades[$assignInstanceID])){
        global $CFG, $DB;
        require_once($CFG->dirroot . '/mod/assign/lib.php');
        $assign = $DB->get_record_sql('SELECT * FROM {assign} WHERE id = :id', array('id' => $assignInstanceID));
        $this->assignGrades[$assignInstanceID] = assign_get_user_grades($assign);
    }
    if(empty($this->assignGrades[$assignInstanceID][$userid])){
        return 0;
    }
    return intval($this->assignGrades[$assignInstanceID][$userid]->rawgrade);
}
```

Listing 6.4: Function to Get the Assignment Grade of a Single User
6.3. Getting Further Informations

Workshop Grades

The workshop (or peer review) plugin provides two grades. One for the submitted work and one for the assessment. The plugin contains both a `lib.php` and a `locallib.php` in its directory. Although the `lib.php` provides many public functions, none of them returned any grades or grade items. The `locallib.php` does not contain any public functions but many classes. One of these - the workshop class - contains the function `get_gradebook_grades` which returns the grade items for the submission and assessment for a single user ID. Because there is no possibility to get grades for more than one user ID at the same time, the function calling the workshop API (see [Listing 6.5]) also does it for every single user. To instantiate a workshop object, the database entry of the workshop which grades are wanted needs to be pulled and also the course module and the course object. After that, the function to get the grades can be called and the grades, stored in a standard class object, is returned.

```php
private function get_workshop_grade_of_user($workshopInstanceID, $userid) {
    global $CFG, $COURSE, $DB;

    require_once($CFG->dirroot . '/mod/workshop/locallib.php');

    $workshop = $DB->get_record_sql('SELECT * FROM {workshop} WHERE id = :id', array('id' => $workshopInstanceID));

    $cm = get_coursemodule_from_instance('workshop', $workshop->id, 0, false, MUST_EXIST);

    $workshopObject = new workshop($workshop, $cm, $COURSE);

    return $workshopObject->get_gradebook_grades($userid);
}
```

[Listing 6.5: Function to Get the Workshop Grades of a Single User]

6.3. Getting Further Informations

Two plugins were found to get time spend in a course from:

- Course dedication
- Use stats

Both are additional plugins of the block type such that at first, it needs to be checked if these blocks are instantiated in the course. This is done by the function displayed in [Listing 6.6]. The function takes the name of the plugin (dedication for course dedication and use_stats for use stats) and checks in the database for all instances of the plugin. If instances are found, it is checked if one of them is located in the course, the LSA-Plugin is used.

```php
private function check_for_block_plugin_installed($pluginname) {
    global $DB, $COURSE;
```

[Sven Judel - Activity Logging Approaches and Data Correlations with Learning Styles]
6.3. Getting Further Informations

```php
$all_plugin_instances = $DB->get_records_sql('SELECT * FROM {block_instances} WHERE blockname = :bn',
array('bn' => $pluginname));

foreach($all_plugin_instances as $instance){
    if($DB->record_exists_sql('SELECT * FROM {context} WHERE id = :id
AND instanceid = :courseid',
array('courseid' => $COURSE->id,
'courseid' => $instance->parentcontextid))
        ){
        return true;
    }
}

return false;
```

Listing 6.6: Function to Check if a Block Plugin is Installed

This function can be applied to check for any block, only the name of the plugin needs to be known. After that, a closer look was taken onto the plugins to determine if they provide APIs.

6.3.1. Course dedication

The course dedication plugin provides two classes in the file `dedication_lib.php`. Because no public functions were given, these two classes were studied. The class `block_dedication_manager` is described as a class generating dedication reports. It has no static functions but a public function `get_students_dedication` which takes an array of user objects which only need to contain the user ID and returns the dedication time for each user. To instantiate a dedication manager, a course object, a minimum and maximum timestamp and a timeout limit need to be given to the constructor. The timeout limit needs to be given in seconds and the lowest possible timeout is one minute (60 seconds).

Because the dedication time of all users can be returned, two functions were implemented to get the dedication time of all users and store them in a variable (Listing 6.8) and to return the dedication time of a single user (Listing 6.7). This should reduce the overhead because the API does not need to be called for every single user. If the dedication time of a user is required, it is checked if the variable in charge of storing the dedication time of all enrolled user is empty. If so, the dedication times were got from the API as an array with each user ID as a key and the dedication time as the value. After that or if the dedication times were already gotten, the value of the user ID in the array is returned.

```php
private function get_course_dedication_of_user($userid)
    {
        if(empty($this->dedicationtimes))
            {
                $this->dedicationtimes = $this->get_course_dedication_time();
            }
        return $this->dedicationtimes[$userid];
    }
```

Listing 6.7: Getting the Dedication Time of a Single User
6.3. Getting Further Informations

```php
private function get_course_dedication_time()
{
    global $COURSE, $CFG;

    require_once($CFG->dirroot.'/blocks/dedication/dedication_lib.php');

    $longest_allowed_timeout_in_seconds = 60; // 1 minute

    $ownDedication = new block_dedication_manager($COURSE, $this->timePeriodBeginning, $this->timePeriodEnd, $longest_allowed_timeout_in_seconds);

    $userIDs = get_all_user_ids();

    $user = array();
    foreach ($userIDs as $userID)
    {
        $u = new stdClass();
        $u->id = $userID;
        $user[] = $u;
    }

    $results = $ownDedication->get_students_dedication($user);

    $return = array();
    foreach ($results as $result)
    {
        $return[$result->user->id] = $result->dedicationtime;
    }

    return $return;
}
```

Listing 6.8: Getting the Dedication Time of All Users

To get the dedication time from the API, a dedication manager needs to be instantiated. The course object is given by the global $COURSE object and the minimum and maximum time by class variables storing the beginning and the end of the selected time period. The timeout limit is set to one minute per default but in a future version, this can be implemented in a more dynamic way. For all enrolled users which IDs are stored in the variable $userIDs a standard class needs to be created and given the ID. The array of these objects is given to the dedication managers API function and the data structure of the result is flattened and finally returned.

6.3.2. Use stats

The directory of the use stats plugin contains both a lib.php and a locallib.php. lib.php provides two public functions where one enables checking if the plugin supports a given feature and the other one sets up theme notifications. The locallib.php provides more functions but none with a clear name or commentary that the desired data are returned. After investigating the functions, two were considered suitable to return the desired data if they are combined:

- use_stats_extract_logs
  This function takes two timestamps in within the log thread for a user with a given ID for a course object is determined and returned.
6.4. Final Plugin List

- **use_stats_extract_aggregate_logs**
  The log thread, returned by the function above, can be given to this function together with a parameter called \$dimension which states how the use stats are computed. The most usual case is to set the parameter to *module* which returns the use stats values also display in the graphical interface of the plugin on the course site.

Listing 6.9 shows the implementation of the function to get the use stats of a single user. Although it is possible to hand over an array of users to the extract function, it is clearer and better readable to extract and aggregate the logs for each user on its own.

```php
private function get_use_stats_time_of_user($userid)
{
    global $COURSE, $CFG;

    require_once($CFG->dirroot.'/blocks/use_stats/locallib.php');

    $logs = use_stats_extract_logs($this->timePeriodBeginning, $this->timePeriodEnd, $userid, $COURSE->id);
    $res = use_stats_aggregate_logs($logs, 'module');

    return $res['coursetotal'][$COURSE->id]->elapsed;
}
```

Listing 6.9: Getting the Use stats of a Single User

6.4. Final Plugin List

During the investigation of how data can be extracted from the suitable plugins collected in the first step. The Logs plugin was the only plugin on the list to get log data from but does not provide an API. Therefore the plugin was omitted and an own API written which took some parts of the Logs plugin and provides the desired functionality.

While looking at the grader report and grade export plugins from which one should enable extracting grades for specific activities and the total course grade, it was discovered, that both do not provide an API. The overview report also does not define a function to get the total course grade such that these three plugins were removed from the list. But during the investigation and trials with these plugins, the lib.php and querylib.php of the grades itself were discovered. These files contain one class and one function which enable to write an own API to get the total grade for all enrolled users in a course. Because quizzes can be done multiple times, getting only one grade (maybe only the best) is not detailed enough. The quiz plugin provides some own reports and also two functions to extract the desired data. To get the grades for assignments and workshops, it was looked at the plugins and their lib.php and locallib.php files. A function to pull grades for all enrolled users is given for the assignments. Getting the two grades for a workshop is done by instantiating a workshop object and using one of its public functions.

Further information, provided by the plugins course dedication and use stats, could be extracted by creating a dedication manager to get the dedicated time of a user and calling an API function of the use stats plugin to get the estimated time spent in a course. Table 6.1 shows the final list of plugins that are used in the LSA-Plugin to get event logs, performance data and further information.
### 6.4. Final Plugin List

<table>
<thead>
<tr>
<th>Plugin Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Own logs API</td>
<td>Extracting the event logs from the database. Implemented because the Logs plugin does not provide a function to get logs from with suitable filtering.</td>
</tr>
<tr>
<td>Grades</td>
<td>Getting the total grades for the course.</td>
</tr>
<tr>
<td>Quiz grades report</td>
<td>Getting the average and best grade a user has achieved in a quiz. While the API also provides a function to extract the best grade of a user, the best grade is computed manually to enable additional filtering for the time period, the grade should be investigated in.</td>
</tr>
<tr>
<td>Assign</td>
<td>Getting the achieved grades from a specific assignment.</td>
</tr>
<tr>
<td>Workshop</td>
<td>Getting the achieved submission and assessment grade for a specific user.</td>
</tr>
<tr>
<td>Course dedication</td>
<td>Getting the dedication time to a course for the enrolled user.</td>
</tr>
<tr>
<td>Use stats</td>
<td>Getting the estimated time a user has spent in a course.</td>
</tr>
</tbody>
</table>

*Table 6.1.: Final List of the Plugins Used in the LSA-Plugin*
Chapter 7 Visualization Plugin Research

The third and last step before implementing the LSA-Plugin was to search for plugins which take data and visualize them in multiple ways. It was required to have a broad range of visualization types like bar charts or graphs. Additionally, the results should also be displayable in a table but this was no mandatory ability of the used visualization plugin. A table could be generated by hand with plain HTML for a first version of the LSA-Plugin such that during the research for the most suitable visualization plugin, this feature was not considered. Furthermore, the visualization should be downloadable as an image and the raw data as a CSV, JSON, XML file or similar.

7.1. Collecting Plugins

By googling “Moodle charts” a link to the documentation of the Charts API is the first result. This API is built into the Moodle core and provides a simple interface to generate dynamic charts [44]. Different visualization types are represented by different classes:

- Bar
- Stacked Bar
- Horizontal Bar
- Stacked Horizontal Bar
- Line
- Smooth Line
- Pie
- Doughnut

One instance of the chart is not limited to only one of these chart types but they can be mixed. Furthermore, the API provides a clickable text element below the chart which displays the visualized data in a table. Clicking the text a second hides the table again. Data are given to the API in form of objects of the core class chart_series which contains a title for the data and the data itself in an array. To label each data, the labels can be given to the chart also as an array. Additionally, customizations can be applied to the chart title, the axis and the colors of the displayed data. This satisfies all requirements towards the desired API and also provides the tabular data display.
7.2. Plugin Comparison

7.1.1. Additionally Plugins

The Charts API is the only visualization feature implemented in the Moodle core. Therefore other plugins were searched in the Moodle plugin directory. With the script also used to collect additional plugins for learning analytics (see section 5.3), further visualization plugins were searched. It was run with three keywords:

- *chart* (20 results)
- *visualize* (4 results)
- *visualise* (1 result)

Only one plugin occurred multiple times (*Grade distribution* for chart and visualize). Some of them were also included in the list of plugins which were investigated as being suitable to provide learning analytics data in section 5.3. From the collected 25 plugins, two were given a closer look. The plugin *Pie Chart* provides an interface which takes arbitrary data and visualizes them in a pie chart [29]. This functionality is also provided in the plugin *Chartist* together with visualizations in bar charts or tables [28]. Most of the other plugins like the *Analytics graph* also provide visualization but only for specific data (e.g., the grade distribution) and do not allow using them with own data sets.

7.2. Plugin Comparison

None of the collected additional plugins provide any further features than the Charts API. Especially because the API is integrated into the Moodle core, the usability of the LSA-Plugin does not suffer from requiring the user to install another plugin to get it working. Data can be visualized in many different ways including a tabular view. Therefore the Charts API is chosen to do the visualization for the LSA-Plugin.

For the first version of the LSA-Plugin, the chart types *Smooth Line* and *Doughnut* are not provided. A doughnut chart is a pie chart with a blank center and does not provide data in a different way than a pie chart does. Using smooth lines instead of straight ones is dangerous because some data can be misinterpreted [35]. Consider the example constructed in Table 7.1:

<table>
<thead>
<tr>
<th>Learning Style</th>
<th>Active</th>
<th>Reflective</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Strong</td>
<td>Moderate</td>
</tr>
<tr>
<td>PDF views</td>
<td>200</td>
<td>230</td>
</tr>
</tbody>
</table>

*Table 7.1:* Example Data of PDF Views in a Moodle Course

Visualizing these data with straight lines results in a chart similar to the one displayed in [Figure 7.1]. The small decrease in views from moderate active to mild active learners is not very visible in this chart but by looking at the chart with smooth lines in [Figure 7.2] a non-expert might get confused. The increasing value between moderate active and mild active learners might be interpreted as an increasing number of views in between these categories. Especially because in reality there is no smooth transition between the single categories and only value points are connected, such interpolation, computed by the visualizer contain these possibilities of misinterpretation. Therefore, smooth lines are not used in the LSA-Plugin.
7.3. Downloads

The Charts API uses an HTML Canvas element to display the chart. This visualization can be converted into an image object which can be provided as a download. To process the raw data, a file could be generated on the server side via PHP or on the client side with JavaScript which runs through the table and extracts the single values from the HTML code. How this is done can be seen in subsection 8.3.3.
Chapter 8 Implementation

With the final list of plugins to use for gathering activity and performance data and doing the visualization, the LSA-Plugin can be implemented. The goals were to achieve easy extendability and high maintainability by implementing the plugin in a modular way.

8.1. Architecture

The LSA-Plugin was initialized with the basic structure of any Moodle plugin containing the following directories and files:

- **db/**
  This directory contains files to (un-)install and upgrade the database tables created and used by the LSA-Plugin. Additionally, the capabilities are defined with which access rights can be controlled.

- **lang/**
  This directory contains the localized strings used in the LSA-Plugin stored in a folder named by the language code of ISO 639.

- **index.php**
  This script would be run if an instance of the LSA-Plugin is called. But because the plugin will be built on a Model-View-Controller pattern (see below), this script is never called.

- **lib.php**
  Similar to the lib.php files of the plugins that are used within the LSA-Plugin, this file contains functions it wants to provide to other plugins in Moodle. Especially the functions to add, update and delete an instance of the plugin to a course are defined here.

- **LICENSE.md**
  This markdown file contains the GNU license text.

- **mod_form.php**
  This file contains the formula of the setting page of the LSA-Plugin.

- **README.md**
  Read-me file for the git repository.

- **settings.php**
  This file contains the whole plugin setting page.

---

8.1. Architecture

- **version.php**
  Defines the version number, required Moodle version, the component name and the maturity of the LSA-Plugin.

- **view.php**
  The first view defined for the LSA-Plugin.

To achieve the desired modularity, the LSA-Plugin was conducted based on the Model-View-Controller (MVC) pattern. Here, the models are the different data that can be visualized. But creating a single model file for each of these data is unnecessary overhead. Only values itself are required and these are either the number of times one or more events were triggered, grades or time spend in the course. Furthermore, the Charts API does not differentiate between these kinds of data. Therefore, only views and controllers were implemented in a formal way and the views stored in the root directory and the controller in `classes/controller/`. Used formulas, created with the Moodle Forms API, are stored in `classes/forms/` (see section 8.2) and utility classes in `classes/utilities/`.

### 8.1.1. Views

To satisfy the aims of the plugin, three views needed to be implemented. None were combined to stick with the MVC pattern and to maintain the modularity of the plugin.

- The analytic view for the teachers and admins (see section 8.3)
- The view for the students (see subsection 8.3.5)
- The interface to upload learning style data (see section 8.2)

Because the main focus of the LSA-Plugin is the analytic view for teachers and admins, this view was conducted for the `view.php`. For the other two views, two new files (`student_view.php` and `learning_styles_import_view.php`) were created and stored in the root directory of the LSA-Plugin. Each view file first requires needed files (among others the related controller for the view) and sets up some required variables like the module instance and module context. After that, the controller can start working by checking, if the accessing user is allowed to visit this view, handle submissions and render the single template elements used in the view.

To achieve a high modularity within the plugin, each element that should be displayed in a view is written as a template file and stored in the directory `templates/`. These templates are built by a utility class `template_builder` which enables assigning values to variables used in a template. If a template should be displayed, the builder buffers its code, assigns the values and returns the output (see Listing 8.1).

```php
public function load_template(){
    $file = $this->templatePath . DIRECTORY_SEPARATOR . $this->
    templateName . '.php';
    $templateExists = file_exists($file); 
    if($templateExists){
```

2 Deduced from https://github.com/moodlepeers/moodle-mod_groupformation/blob/master/classes/util/template_builder.php
8.1. Architecture

CSS files needed to style the templates are stored in the directory `css/` and JavaScript files handling events and enabling some interactivity are stored in `js/`.

8.1.2. Controller

A controller provides the functionality of a view and renders its template elements. The first task is to check if the visiting user has the rights to access the view. Listing 8.2 shows the `handle_access` function of the `overview_controller`, responsible for the analytic view for teachers and admins. The rights are taken from the capabilities, defined in the `db/access.php`. If a student tries to access the `view.php`, he or she gets redirected to the `student_view.php`. Similar functions are used to redirect teachers and admins from the student page to the analytic view and to deny students accessing the view to upload learning style data and redirecting them back to their view.

```
public function handle_access()
{
    $context = context_module::instance($this->cmid);

    if (!has_capability('mod/lsanalytics:editsettings', $context)) {
        $redirectionTarget = new moodle_url('/mod/lsanalytics/student_view.php', array('id'=>$this->cmid));
        redirect($redirectionTarget->out());
    }
}
```

Listing 8.2: Handle Access Function of the Overview Controller

Because of these and further similarities, an abstract basic controller was implemented and two controller inheriting from it (see Figure 8.1). The student overview controller inherits from the overview controller because they mostly need the same functions and only some adjustments to hand over a user ID instead of learning styles to the chart builder had to be done.

Rendering templates is another common functionality of the controller. The render function of the basic controller is shown in Listing 8.3. The `view` variable contains a template builder object and the `templateName` variable is set in the functions of the three controller classes where a specific template should be displayed (see Listing 8.5). Here, the name is the name of the file, containing the code of the elements to display (e.g. Listing 8.4).
8.1. Architecture

Figure 8.1.: Controller Class Diagram

Listing 8.3: Render Function of the Basic Controller Class

```php
<?php
//...
if (!defined('MOODLE_INTERNAL') || die()) {

    public function render() {
        $this->view->set_templateName($this->templateName);
        return $this->view->load_template();
    }
}
```

Listing 8.4: Extract templates/chart_export_buttons.php

```php
<?php
...
if (!defined('MOODLE_INTERNAL') || die()) {

    public function render_chart_export_buttons() {
        $this->templateName = 'chart_export_buttons';

        return parent::render();
    }
}
```

Listing 8.5: Chart Export Buttons Render Function of the Overview Controller Class

The third big task of the controller is to handle the submissions for the formulas on each page. How this is done as described in section 8.2 for the upload of learning style data and in section 8.4 for the analytic and the student view.
8.1.3. Controller within the View

Listing 8.6 shows the usage of an instance of the overview controller within the view.php. After its instantiation, the controller checks for the access rights of the current user. With a function, defined in the LSA-Plugins locallib.php, JavaScript files are included and after the Moodle header code is printed, CSS files too. The next step is to check for submission of parameters to display a chart with and to get this configured chart or a dummy visualization (see section 8.4). If something happened during the creation of the chart, notifications are generated and the controller displays them. Finally, the single template elements of the page are rendered and only if the previously received chart object is null, another notification is displayed.

```php
$controller = new mod_lsanalytics_overview_controller($cm->id,
    $moduleinstance->id);

$controller->handle_access();

// Import jQuery and js file.
lsanalytics_add_jquery($PAGE, 'learning_style_selection_form_interaction.js');
lsanalytics_add_jquery($PAGE, 'visualization_settings_form_interaction.js');
lsanalytics_add_jquery($PAGE, 'chart_export.js');

echo $OUTPUT->header();

echo '<link rel="stylesheet" href="css/learning_style_selection_with_categories.css">';

echo '<link rel="stylesheet" href="css/visualization_settings.css">';

echo $OUTPUT->heading($pagetitle);

$controller->handle_submission();

$chart_to_display = $controller->plot_chart();

foreach($controller->get_notifications() as $notification){
    echo $OUTPUT->notification($notification);
}

echo $controller->render_visualization_settings();

echo '<div style="width: 80%; float: right;">';

if($chart_to_display !== null){
    echo $OUTPUT->render($chart_to_display);
}
else{
    echo $OUTPUT->notification(get_string('no_learning_style_data', 'lsanalytics'));
}

echo '</div>';

echo $controller->render_chart_export_buttons();
```
8.2. Importing and Storing Learning Style Data

Moodle provides with the *Forms API* a tool to create own formulas for Moodle pages. With this, the interface to upload a JSON file with learning style data, shown in Figure 8.2, was created. The user can select a file in a provided dialog or drag-and-drop it into the upload element and clicks on the “Save changes” button.

The uploaded file is given to the controller and it tries to insert the JSON data into the database. Figure 8.3 shows the database table which stores the learning style data (see Listing 8.7). To relate the learning style data to the right user within the right LSA-Plugin instance, an ID is added for both of them. For each dimension, a field is provided to store its value. Additionally, the timestamp of the date, the data is uploaded is stored to allow versioning of the users’ information. As stated in Chapter 4, learning styles may vary and teachers may want to compare visualizations with learning style data gather at different times for the same students.

This versioning is the reason every single entry in the uploaded JSON file needs to be checked first. If one set of data is corrupted or not in the right format, the other data would be uploaded and this one data set would have to be fixed and uploaded later.
This would result in a different timestamp for this data set. Only correct data should be stored. This is achieved by the two functions `$DB->start_delegated_transaction` which buffers database transactions and `allow_commit` which triggers the execution of all transactions but only if no error is thrown during the processing of the uploaded data.

```php
private function store_imported_learning_style_data_in_database($ls_data)
    {
        global $DB;

        $this->uploadSuccessful = true;

        $ls_data_rows = json_decode($ls_data, true);

        $currentTime = new DateTime('now');

        $allEnrolledUserIDs = get_all_user_ids();

        try {
            $transaction = $DB->start_delegated_transaction();

            foreach ($ls_data_rows as $ls_data_row) {
                if (in_array($ls_data_row['id'], $allEnrolledUserIDs)) {
                    $newEntry = new stdClass();
                    $newEntry->lsanalyticsid = $this->lsanalyticsid;
                    $newEntry->userid = $ls_data_row['id'];
                    $newEntry->processing = $ls_data_row['processing'];
                    $newEntry->perception = $ls_data_row['perception'];
                    $newEntry->input = $ls_data_row['input'];
                    $newEntry->understanding = $ls_data_row['understanding'];
                    $newEntry->timecreated = $currentTime->getTimestamp();

                    $DB->insert_record('lsanalytics_learning_styles', $newEntry);
                }
            }
            $transaction->allow_commit();
        } catch (Exception $e) {
            $this->uploadSuccessful = false;
            $transaction->rollback($e);
        }
    }
```

Listing 8.7: Function to Store Uploaded Learning Style Data

If the upload and storing is successful, the user is then redirected back to the analytic view and gets a notification about the inclusion of the new data into the database. If the upload or storing fails, a notification with an error message is printed. By canceling the upload, the user gets redirected to the analytic view without any notification.

8.3. Analytic View Frontend

The limitations of the Moodle Forms API are very strict. Only predefined and formatted elements can be used and decisions on arrangements are restricted to the vertical order
8.3. Analytic View Frontend

of single blocks. Creating arbitrary formulas is not possible and therefore the interface with form elements to configure the chart was created without the API. This led to taking care of the formula submission and reading out data correctly. Submitting data was done by giving the `<form>` element values for the `action` and `method` attributes to generate a post request to the same page. Furthermore, Moodle provides a function to read out the `$_POST` variable.

8.3.1. Settings for the Visualization

The visualization can be configured directly in two ways by choosing the visualization type and the value format. Both parameters are set with dropdown elements (see Figure 8.4). The values of each option are set up in the chart builder class as shown in Listing 8.8 together with the values for the activity data. The values for the time periods were hardcoded in the template itself because of the interaction (done with JavaScript) with the single radio buttons, if the Other option is selected. Naming the label for the control to select activity and performance data Type of activity data was a remain from the initial implementation where grades and earned badges were not added. Activity and performance data selection and time periods do not directly influence the visualization. They change plotted values which might result in a different chart.

![Figure 8.4: Dropdown Element to Select the Visualization Type (on the left) and the Value Format (on the right)](image)

```php
$the->visualizationTypes = array(
    'bar_chart',
    'stacked_bar_chart',
    'horizontal_bar_chart',
    'stacked_horizontal_bar_chart',
    'graph',
    'pie_chart'
);  
$the->valueFormats = array(
    'absolute',
    'relative_between_events',
    'relative_between_learning_styles'
);  
$the->listOfCourseActivities = $this->get_all_course_activities();

Listing 8.8: Parameter Values for the Visualization Settings
```
The selection of activity and performance data is done with a modified multiselect element (see Figure 8.5). Each activity in the course becomes an `<optgroup>` element and each plottable event an `<option>` element. Which event of which activity can be selected was setup manually and the lists stored in the constant `SUPPORTED_ACTIVITIES_AND_EVENTS` within the chart builder class. An example of how the events are listed is given in Listing 8.9 where the events for the assign plugin are shown. An event is added to the list if it can be triggered by a student. Furthermore the event `average_grade` is not an event of the assign plugin but used to trigger the calculation of the average grade for the selected assignment.

The function `get_all_course_activities` first reads out all created modules in the course. Each modules type is checked and if it occurs as a key in the constant containing all supported activities, an activity instance is created. It consists of the module name, the name of the instance, the list of supported events and a unique module name, made up from the context id of the module and its module and instance name. To visualize the earning of badges, all badges of the course are extracted from the database and prepared in the same way, the regular activities are with just the one event `badge_awarded`. The events for the course, like the total grade or the dedication time (if the plugin is used in the course) are prepared also separately to e.g. check for installed additional plugins.

![Figure 8.5.](image)

### Figure 8.5.: Multiselect Element to Select Events to Display

```php
'assign' => array(
    'average_grade',
    'feedback_viewed',
    'submission_created',
    'submission_updated',
    'submission_viewed'
)
```

### Listing 8.9: List of Possible Plotable Events for the Assign Plugin

The time periods are selected with radio buttons. Four options are provided (see Figure 8.6). `Last week` and `last month` are calculated automatically with the data, the plotting is done as the last day of the time period. `Whole semester` goes back to the date, the course was created and therefore capture all triggered events. Time is not selectable, even for the `other` option, where the user can select the first and last day of the time period manually.
8.3. Analytic View Frontend

The time is always set to 0:00:00 and the end date to the next day such that each event of the selected day is considered. To prevent misuses of the other option, checks on the reasonability of the dates are done, e.g. if the end date is earlier than the start date, the dates are switched.

8.3.2. Learning Style Selection

To select learning styles and their categories, a whole new control element, which is shown in Figure 8.7, was implemented. Its design is based on the values representing the preferences in a dimension. Each dimension is represented by one line and the learning styles and categories ordered in a way that the values calculated are display from +11 at the left end to −11 on the right end. This is taken from the visualization of the results of the Index of Learning Styles from Felder and Silverman ([17]). Green indicates that a learning style and some of its categories are selected and should be visualized, gray indicates that they are not. No indicate that the boxes should be clicked, the cursor image is changed to the pointer image if it lays over the control.

To simplify the selection, it is possible the click and hold the mouse to toggle the selection of categories. With this, a row or a column can be (de)selected quicker than clicking each category box. To deselect a whole learning style or select one with all its categories, the dark green box have to be clicked - the click and hold movement does not work with these boxes.

To show the results over all learning styles, a checkbox is given which states if the results of single categories for one learning style should be merged or not. Selecting the upload date for the learning style data is done with a dropdown element and an additional button links the teacher or admin to the upload page for learning style data (see Figure 8.8).

Currently, the selection of the learning styles returns events triggered by users which matches at least one selection. This means that if e.g. the views of PDFs should be shown...
8.3. Analytic View Frontend

Figure 8.8.: All Four Control Elements for Selecting Learning Styles

for strong active and strong visual learners and one user matches both criteria, his or her visits are added to both values. This is shown in Figure 8.9 where User 2 is not a strong visual learner and User 3 not a strong active one.

Figure 8.9.: Influence of Users Matching more than One Selected Learning Style

8.3.3. Exporting Data and Visualization

The Charts API uses an HTML5 Canvas Element to display the visualization. Its content can be converted into an image object with a JavaScript function and a download started, containing this object (see Listing 8.10). To export also the raw data, displayed in the table, a JavaScript function was written which iterated through the HTML table and prepares them in a CSV file. Doing this on the client side with JavaScript was decided
because if done on the server, the CSV file will always be created and therefore adds a small overhead to the time the page needs to load. Especially if the user tries out different visualizations for the same data, he or she only wants to download once as an image or CSV file, this unnecessarily increases the time needed to perform the desired task.

```javascript
var exportChartVis = function() {
    var img = $('canvas')[0].toDataURL('image/png');
    download(img, 'visualization.png');
};

var exportChartRaw = function() {
    var csvContent = 'data:text/csv;charset=utf-8,Learning Style';
    $('.chart-table-data').find('table').children().each(function() {
        $(this).children().each(function() {
            csvContent = csvContent + $(this).text() + ';
        });
    });
    csvContent = csvContent + '
\n';
};

var encodedUri = encodeURI(csvContent);
download(encodedUri, 'raw-data.csv');
```

Listing 8.10: Functions to Prepare and Download displayed Data (Extract of js/chart_export.js)

8.3.4. Arranging the Controls

The single control elements can be ordered within the frontend in many different ways. To achieve a good ordering, seven friends and acquaintances (all Bachelor or Master students in Computer Engineering) were given the single controls (without the learning style selection control) as images within a PowerPoint file. They were asked to order the elements in a way that feels natural and logic for them. One returned mockup is shown in Appendix B. It should be mentioned that during the time, the mockups were created, the element for selecting the time period got a different style which was changes shortly after because it was too high to fit well into the desired layout. From the seven mock-ups, common groupings and positionings were identified and used. The learning style selection control was no part of the mockup because it was conceived as a block over the nearly the whole width of the page and also because students were asked to do the mockup and this target group does not operate this element.
8.3. Analytic View Frontend

The final layout for the analytic frontend with a dummy visualization is shown in Figure 8.10. The displayed instance is generated in the course the plugin evaluation was done (see Chapter 9).

Figure 8.10: Final First Version of the Frontend

8.3.5. Student View

The student view contains a subset of the template elements used in the analytic view, as shown in Figure 8.11. Only the selection of learning styles is omitted because the learning styles of the visiting student should be used. The user selects the activity and
8.4. Backend

Based on the frontend layout, the generations of a chart was split up into single modules to simplify changes that might be necessary to fix flaws detected during the user study (see Chapter 9). These modules are called in function, shown in [Listing 8.11] of the chart_builder class to create the desired chart step by step.

```php
public function plot(){
    /*
    * Check if the necessary parameters were given
    * Using one representative from the visualization settings and
    * learning styles and the user id
    * Either a user id or learning style data should be given (not null)
    */
    if($this->selectedVisualizationType !== null or (!is_null($this)
        any_learning_styles_given() and $this->userID !== null)){
        //Not all parameters given -> return null
        return null;
    }

    //Sort the categories for the x-axis
    $this->sort_list_of_selected_ls_categories();
```
Listing 8.11: Plot Function of the Chart Builder Class

At least a visualization type and either a learning style category needs to be selected or a user ID given (analytic view or student view). If one is not given, null is returned which leads to the display of a notification within the view as seen above. Because the categories of learning styles can be selected in an arbitrary order, the entries in the arrays of each learning style need to be sorted in the right way (based on the signum of the learning style). This simplifies iterating through the list and labeling the x-axis in the right order.

In the next step, a chart instance of the selected visualization type is created. If no user ID is given initially (which is the case if the plotting is done for the analytic view), the needed user IDs based on the selected learning styles are read out from the database. The last big step is to compute the series for each selected activity and performance data and to check if they need to be visualized with relative values. If the chart is created for the student view, these data are computed just for one user which is achieved by additional filtering in the SQL query. Therefore the own API function was extended to separate these two use cases. The gathered values are stored in a variable and after that the four learning style categories of the user determined, the average values for them calculated and the user value added in front of the series. After that, all series are added to the chart.
instance, the labels for the x-axis are generated and added and finally the chart object returned such that it can be rendered in the view.

8.4.1. Modelling the FSLSM

To represent the Felder-Silverman Learning Style Model in the LSA-Plugin an abstract class `mod_lsanalytics_fslsm` was written. It contains six constants to model the different parts of the FSLSM:

- **DIMENSION**
  An array containing the names of the four dimensions of the FSLSM.

- **LEARNING_STYLES**
  An array containing the names of the eight learning styles defined in the FSLSM.

- **SIGNUM**
  An array with each learning style as a key and the signum of it as the value (e.g. `'ACTIVE' => 1`). Used in combination with the stored value of each dimension to determine which learning style is the preferred one.

- **DIMENSION_OF_LEARNING_STYLE**
  An array with each learning style as a key and the index of the containing dimension within the `DIMENSION` constant as the value (e.g. `'Visual' => 3`).

- **CATEGORIES**
  An array containing the three categories the preference for a learning style could be represented in.

- **CATEGORY_INTERVAL**
  An array with each category as a key and the minimum and maximum value identifying the category as the value (e.g. `'Strong' => array('min' => 9, 'max' => 11)).

Other classes require the file containing this class and use its definitions of the FSLSM. This increases the modularity of the plugin and reduces the risks of human error if for each usage of the FSLSM within the code, everything would be written by hand or hard-coded.

8.4.2. Getting Users for Selected Learning Styles

To know which users need to be taken events from, all user IDs matching one or more of the selected learning styles have to extracted from the database. The utility class `chart_builder` provides a private function `get_needed_user_ids` to search for uploaded learning style data, matching the selected ones. It filters the table of the uploaded learning styles for the selected timestamp and the selected range of values for the dimensions and returns the user IDs without redundancy. Redundancy would be given, if the learning style categories should be grouped such a `SELECT DISTINCT` in the SQL query is necessary.

8.4.3. Getting Activity and Performance Data

The first step is to extract the necessary event details from the selected activity and performance data. For each desired data, an object with the following attributes is created
8.4. Backend

- **eventname**
  The eventname as specified in the `SUPPORTED_ACTIVITIES_AND_EVENTS` constant.

- **label**
  The name of the event, returned by the static `get_name` function of the event class or generated with strings from the LSA-Plugin for special events or performance data like the average total grade of the course.

- **contextinstanceid**
  The context instance ID is set during the gathering of all course activities.

- **activityType**
  The type of the activity, the event belongs to.

- **moduleInstanceID**
  Only used for quiz, assign and workshop activities and needed for the interaction with the APIs to get grade data from.

For each of the returned event objects, the needed values for the parameters of the SQL query, which is gathered from the self-written API (see section 6.1) are extracted and the query executed. The returned records contain the events triggered by any enrolled user in the course. In the next step, the function iterates through the selected learning style categories and extracts the needed data for each user ID in the category. This is done with the function `get_y_value` and the default way is shown in [Listing 8.12] where `$sql_res` is the variable containing the records of the executed SQL query. The user ID of each record is checked and the counting variable `$temp` increased by one if it matches the currently investigated user ID, given to the function as a further parameter.

```php
foreach ($sql_res as $r) {
    if ($r->userid === $userid) {
        $temp = $temp + 1;
    }
}
```

**Listing 8.12: Extract of `get_y_value` of Chart Builder Class**

This default way is not usable for special data like grades. The `get_y_value` function contains a switch-case statement to determine the right technique to extract and return data, based on the eventname. [Listing 8.13] shows how the average grade for one user in one quiz is computed. The API function of the quiz plugin `quiz_get_user_attempts` returns all attempts of the user identified by the given user ID. These attempts are filtered based on their timestamps such that only the results achieved during the selected time period are left. An additional function `compute_average_grade` which takes all attempts, extracts the grade value of each one, sums them up and returns this sum.

```php
case '\mod_quiz\event\average_grade':
    $quiz_attempts = quiz_get_user_attempts($event->moduleInstanceID, $userid);
    $quiz_attempts = $this->filter_timestamps_out_of_bound($quiz_attempts, $time_params['timestamp_lower'], $time_params['timestamp_upper']);
    $temp = $temp + $this->compute_average_grade($quiz_attempts);
```

Sven Judel - Activity Logging Approaches and Data Correlations with Learning Styles 65
8.4. Backend

Listing 8.13: Extract of get_y_value of Chart Builder Class to Get Quiz Grades

The values for each user ID in each category are summed up and if categories should be group also the values of each category are summed up to represent the values of the whole learning style. An array is created containing the resulted values in the right order, matching the order of the learning style categories. Before returning the values, it is checked if the value format is set to relative (learning styles) (see Listing 8.14).

```php
if ($this->selectedValueFormat === 'relative_between_learning_styles') {
    $sum = 0;
    foreach ($numOfUserOccurrences as $value) {
        $sum = $sum + $value;
    }
    if ($sum !== 0) {
        for ($i = 0; $i < count($numOfUserOccurrences); $i = $i + 1) {
            $numOfUserOccurrences[$i] = 100 * ($numOfUserOccurrences[$i] / $sum);
        }
    }
}
```

Listing 8.14: Function to Calculate Relative Values of Chart Builder Class

Checking for the value format to be set to relative (events) is done back in the plot function and if so computed by summing up all values within one series and after that the relative values calculated. Finally, the series are added to the chart object, the x-axis labels generated and added and the chart object returned such that it can be rendered in the view.
Chapter 9 Evaluation

According to Dix et. al. [9], the evaluation of a system has three main goals:

- Assess the extent and accessibility of the system’s functionality
- Assess users’ experience of the interaction
- Identify and specific problems with the system

Rogers et. al. [62] stated that evaluation “is needed to check that users can use the product and like it”. To get these kinds of feedback, the LSA-Plugin was tested on potential target users. This technique is called usability testing and can be defined as “representative users attempting representative tasks in representative environments” [38]. This is a very broad definition and meant to be that way because usability testing can be applied in various stages during the design and development. It can be done very early with prototypes that have only been build on paper (paper-prototypes) and also later with the finished product. All these techniques share the aim to improve the interface by finding flaws in it.

The main focus of the LSA-Plugin is the analytic view of teachers and administrators. The student view is a smaller version of it, leaving out the selection of the learning styles (and its control element) and replacing them with the ones of the student, using the plugin. Therefore, only the teacher view was evaluated.

9.1. Study Preparations

The preparation of the study included determining what the study should aim at (as given above), how many participants are needed, what information were necessary to answer the studies questions and what the participants should do with the system. A study setup had to be assembled, participants recruited and the whole plan tested with a pilot study.

9.1.1. Study Group

According to different literature sources, a usability study should include 5 [66] to 7 [59]. Virzi states that five participants is the magic number and that five users will find about 80% of the usability flaws [38, 66]. Nielsen and Landauer say that seven participants are optimal for small projects and about 15 participants for medium to large projects and the highest ratio of benefit to costs is achieved if 3.2 participants do the usability testing [38, 59].
9.1. Study Preparations

The reality is, that most usability studies will never uncover all usability flaws. Even if all flaws would be uncovered, not all will be fixed. As stated above, the aim of usability testing is to find the major flaws that will cause the most problems and to fix them. In industry most usability testing is based not on what should or be done, but on how much resources are left in the development process. This includes time, money and available users that are willing to participate.

Based on the recommendation from Virzi and Nielsen and Landauer \cite{66,59}, it was aimed at getting between five and seven potential users to participate. This interval was chosen to meet the two mentioned rules of thumb. Less than five users would lead to a too limited study with not enough input to separate major flaws from flaws just annoying a few users. More than seven users would be too much because, on the one hand, the project is relatively small and, on the other hand, as the project is a part of a master thesis, the resources (especially time) are limited. Because only the teacher view of the LSA-Plugin should be evaluated, the participants of the study had to fulfill the characteristics of the related target group as good as possible:

- Experience in teaching
- Experience with learning management systems as a teacher (in best case with Moodle)
- Good knowledge of the FSLSM

Teaching experience was needed to know about how to work with students and how to impart knowledge. Experience with learning management systems was required such the participants know about the possibilities of learning management systems like setting up quizzes, uploading slides or other materials, managing assignments or providing a discussion forum. If these experience arise from using Moodle, the participant would not need to understand a new system first, before working with the LSA-Plugin. A good knowledge of the FSLSM would result in better understand what a task is about. It was also expected to be more easy to understand the learning style selection control because of the arrangement of the single dimensions and learning styles which was based on the presenting of the results of the Index of Learning Styles.

9.1.2. Recruiting Participants

The first bottleneck was the limited contact with teachers or educators at all. Some contacts remained from previous working relationships and other teachers were contacted based on very low geographical distance (e.g. working in the same building). Because not many potential and available participants had good or any knowledge of learning styles and especially the FSLSM, a summary of this learning style model was created (see Appendix C). It was used as a handout and combined with verbal explanations to teach the unknowing participants during the study before they interacted with the plugin.

One week was scheduled to conduct the usability testing of the LSA-Plugin and the desired number of sessions set to five. Six potential participants were invited to the study from which five responded. The earliest a fifth user could make it was the next Tuesday and therefore the time span of the study was expanded to get five sessions done.
9.1. Study Preparations

9.1.3. Initial Survey

In order to create a detailed picture of the participants’ profile, a short web-based survey was created. It aimed at getting a more concrete picture of the previous experience in teaching and knowledge of learning styles. The questions were the following:

1. **Age** (in years)

2. **Profession** (Bachelor student, Master student, PhD student, Postdoctoral, Professor, Other)

3. **Previous experience...**
   a) ... in teaching (in semesters)
   b) ... in using learning management systems as a teacher (in semesters)
   c) ... with Moodle (in semesters)

4. **Knowledge of Learning Styles** (Never heard of it - Heard of it - Worked with it - Good knowledge - Very good knowledge)

5. **Knowledge of the Felder-Silverman Learning Style Model** (Never heard of it - Heard of it - Worked with it - Good knowledge - Very good knowledge)

Asking for the age and profession was done to relate it to the experiences of the participant. The questions on the experiences were asked to know how much a participant had worked without using learning management systems at all and how long he or she had worked with Moodle. On the one hand, experience in Moodle aimed at understanding its single activities. Knowing that teachers and administrators can add assignments, upload PDF files or provide a forum or chat to the students so they can collaborate. On the other hand, it was interesting to determine how much the different experienced users know about the event based logging system in Moodle. Especially because the selection of the data a user wants to visualize is mostly based on the single events which were named by Moodle and therefore might be confusing at first.

The scale to rate the participants’ knowledge of learning styles and the FSLSM is based on a consultation of friends and acquaintances. They were asked to rate themselves without a given scale. The most frequently used terms were *Never heard of them*, *Heard of them* and *Good knowledge*. After that, they were asked a few questions to map these terms to a knowledge level. E.g. knowing about the existence of the four dimensions of the FSLSM and being able to name some or explain what they are about, was referred to *Heard of them* while being able to name all dimensions or explain them without or just a few mistakes was considered *Good knowledge*. The term *Worked with it* were added to fill the gap where people were once a bit more involved in the topic but forgot many things which they would remember by getting a refreshment of the knowledge. *Very good knowledge* was to separate participants who do more work with these topics like studying it or doing research in it.

9.1.4. Task List

The participants should go through the plugin in a goal-driven way. Therefore a task list was created, containing six assignments. Each referring to one of the main plugin features.

1. Upload the new Learning Style data to the plugin
9.1. Study Preparations

2. Visualize in a bar chart how often active and reflective users have visited the forum "Lecture Discussions"

3. Visualize in a horizontal bar chart how often sequential and global users (without the distinction in categories) have visited the forum "Lecture Discussions"

4. Find out how many posts were created and how many were updated in the forum "Lecture Discussions" in the last week by strong sensing learners

5. Find out if visual learners have visited the PDF slides more often than the images extracted from the lectures during the whole semester in contrast to the verbal learners

6. Visualize in a pie chart and in relative values how many strong verbal and how many strong visual learners have viewed the slides from the first lecture

On an ideal task list, each task is clear, unambiguous and does not need additional explanation [38]. In this case, task 1 needed additional explanation on where the learning style data were located such that they can be uploaded. This explanation was given verbally as a part of the instructions on how the results of the ILS were transformed in JSON format. Therefore a redundant explanation on the task list was left out.

While the first task (uploading learning style data) was a job which is not done often but important to get the plugin working, the other five tasks aimed at exploring the features of the plugin. At first, only one event should be visualized and the learning style selection control has to be used such that the user gets a feeling for it and feedback on its intuitivity can be given. After that, different visualizations types have to be used together with different combinations of learning styles - with or without the distinction in the three categories (strong, moderate, mild) - and also the interaction with the multisiselect control becomes more complex to visualize more than one event at the same time.

The only feature of the LSA-Plugin that was not investigated in this usability study was the selection of the upload date of the learning style data. Because of the limited time to prepare the study, no big scenario could be set up to create a useful task were these control and its positioning could be tested.

9.1.5. Tools and Data Collection

Another important task was to decide which tools should be used and how to collect the desired data. Because of the limited time and that the study was conducted by only one person, as much data collection as possible should be done automatically. Automated data collection and other tools should reduce the load of the moderator.

The initial survey was created with an online survey tool called SoSci Survey. It allowed to set up the drawn survey very quickly and also provides the possibility to generate serial numbers. Each number can do the survey exactly one time and therefore they were used as ids for the participant to relate the collected data of the different activities to each other without knowing who the participant was. SoSci also stores the responses of the survey and the only action performed by the moderator is visiting the survey link and entering the id such that the participant only has to click on a button to start the survey. Both can be done before the participant arrives and therefore no work has to be done by the moderator during the work with the survey.

https://www.soscisurvey.de/en/index
9.1. Study Preparations

The ILS was done by using the online version provided by the North Carolina State University. In this tool, a name is required which is set to the id of the participant. The questionnaire is split up on four sites with eleven questions each and after submitting the answers the results are presented as shown in Figure 9.1.

![Example of ILS Results from NC State](https://www.webtools.ncsu.edu/learningstyles/)

This visualization was used in combination with another handout (see Appendix D) to explain how these values are transformed into a JSON format and how they are grouped in the three categories (strong, moderate, mild). The results of the ILS were not logged and therefore the moderator only had to visit the link and enter the id which can also be done before the participant arrives.

Capturing the interaction with the LSA-Plugin was necessary to recognize patterns. This should answer questions like how is a task done, in which order does the participant the single controls to configure the plot in the desired way and how does he or she operate the selection of activity data and learning styles. To achieve such a capturing, screen-capturing was done. With the captured videos, the time needed for each task can be read out and there is no need to have someone use a stopwatch and note the times. This also provides the chance of human error which would falsify the results [38]. Additionally, notes were taken with pen and paper of interesting behavior or things the participants said. Therefore the participant needs to be encouraged to say out loud, what he or she is thinking. This technique is called **Think aloud** [38]. The participant should state why he or she does something, what he or she expects from the action currently performing. It allows capturing some keywords which can later be talked about during the feedback part. A drawback of this technique is that talking interrupts performing. The participant might need a bit longer to complete a task than in a non-testing session. But because the time performance of the participant is not the main focus of the study, this impact is tolerated. Another possible drawback will arise if the participant is not very talkative. Reminding him/her to talk reminds them also of being in a testing situation and it may also interrupt his or her thinking about the task [38].

Screen-capturing and making notes with pen and paper were also done during the feedback. It is very challenging for the moderator to ask questions, look at the participant to not look disinterested and note, what was said, but it is more natural for the participant to know that his voice or he or she as a person is neither audio nor video recorded [38].
9.1.6. Study Setup

For the testing sessions, a new Moodle course was created with a small number of dummy user accounts assigned to some learning styles. The course page is depicted in Figure 9.2. It contains of one forum with the intention to discuss topics of the lecture, the slides of the first three lectures as PDFs, the images extracted from the slides, a course chat, two assignments, one test for self-assessment and the test instance of the LSA-Plugin.

Different activities were performed with the dummy accounts to generate event logs that can be visualized during the sessions. Some were performed in a way that a correlation between the learning styles and the activities can be seen (e.g. decreasing number of accesses of the forum the less active and more reflective a user is).

For this study, it was assumed that Moodle is mostly used on desktop devices like PCs or notebooks. This assumption is grounded on the facts that Moodle courses need some space to display all elements (e.g. for navigation and administration). Furthermore, Moodle is rarely used without other materials. A teacher will upload material mostly from their computers and enter grades from a list either printed out or on a list on another monitor. If the mobile version of Moodle is used, the users are mostly aware of the lose of overview and sometimes usability. Therefore the hardware setup was arranged to simulate an office working place. A monitor, a keyboard and a mouse were placed on a desk such that the participant can work with it. The computer, running Moodle, was not visible for them. Additionally, refreshments (water and snacks) were provided and placed on the desk next to the monitor.

9.1.7. Study Procedure

After all big questions were answered, the procedure of the study was fixed. The schedule contained the basic components of the study like the order in which the survey, the lecturing about the FSLSM and the interaction with the LSA-Plugin should be done. Additionally, the framework was defined with some bullet points about where to include which information. The schedule was defined as followed:

1. (Before the participant arrives) Prepare study setup
2. Participant is welcomed and thanked for his or her time
3. Participant takes place and gets offered some refreshments
9.1. Study Preparations

4. Participant get informed what the study is about, what he or she will do and that the interaction with the plugin will be screen-captured.

5. Participant does the survey.

6. (If needed:) Participant gets lectured about the FSLSM.

7. Participant does the Index of Learning Styles.

8. Participant gets explained how the results of the ILS are transformed into JSON.

9. Participant gets explained how the values of the ILS are mapped into the three categories.

10. Participant is presented with the study course.

11. Participant is presented with the LSA-Plugin.

12. Participant performs the tasks on the task list.

13. Participant gives feedback.

14. Participant is thanked again and farewelled.

15. (After the participant has left) Summarize own notes; Add concluding notes.

No text was completely tidied up because it felt unnatural and remembering that much information and also doing the study was considered too much load on one person.

The feedback session was planned as an unstructured interview without any predefined interview guide. All questions should arise from the comments the participants make during the performing of the jobs on the task list. This should allow investigating all statements of the participants in the most possible depth and breadth. Participants are able to come up with thoughts and ideas, a researcher might not think of in advance such that in a structured interview, these topics might not be investigated. It is also more suited to understand the participants understanding of the plugin and their perspectives.

9.1.8. Pilot Study

Because of the limited amount of available target user and the short amount of time during the conduction of the thesis, a second evaluation would not be possible. Therefore each part of the study (procedure, survey, handouts, setup, task list and tools) needed to be tested to detect flaws in its design. The needed time for the whole study and for the single tasks was measured. Additionally, the recording of notes by the moderator was trained and a rough format developed to simplify the process. The pilot study was run with three master students of computer engineering to keep the willing participants from the target group for the actual study.
Some small flaws were found in the survey and the plugin itself. At first, the survey only asked for previous experience in using learning management systems and Moodle without the limitation on teaching experience. This resulted in a higher number of semesters in these experiences than experience in teaching. In succession, the two questions were updated. There were also difficulties in understanding some questions of the ILS because of unknown words to a German student answering the English questionnaire. Therefore the introduction to the ILS was extended with an explicit advise asking the moderator for unknown words. The moderator added for each asked word an entry on a list to provide a translation faster than using a translation website which needed to be accessed first. In the plugin, one label of a checkbox was not correctly placed and some responsive elements needed some adjustment. Furthermore, some bugs were found, e.g. division by zero if no user in the selected learning styles had performed any selected activity and therefore nothing could be visualized. Another bug was given in the learning style selection control where a learning style was not marked as selected if a category was selected. With the recorded videos, the action sequence that causes the bug could be read out, the bug reproduced and finally fixed.

The strategy of unstructured interviews based on the comments of the participants during the performing of the jobs on the task list was validated. All pilot study participants made comments mostly related to parts of the plugin they were using to solve a specific task. These comments and statements were taken up during the interview and by further explaining, what they mean - sometimes with demonstrating it in the plugin - they also added some wishes or suggested improvements.

The time measurements showed that the three pilot sessions lasted between 46 and 53 minutes and the processing of the task list between 9 and 12 minutes. In the invitation to the actual study, the participants were told that the study would last about one hour and for each session, including preparation and summarization, 90 minutes were scheduled.

9.2. Study Conduction

Each of the 5 sessions followed strictly the planned procedure successfully. The study setup was build up in an office at RWTH Aachen University and each participant was asked to come there. After welcoming the participant and offering him/her a seat and refreshments, the introduction was done. The next action was the participant doing the survey.

Two PhD-Students, one postdoctoral, one professor and one private educator took part in the study. The participants were between 28 and 56 years old and got between 4 an 50 semesters of teaching experience. While they got between 4 and 29 semesters of experience with using a learning management system as a teacher, only two participants got experiences with using Moodle as a teacher (1 and 2 semesters). This matched the requirements on the participant to have experience in teaching and with learning management system. The missing experience with Moodle only affected one participant who took some time to read through the navigation block and administration block before doing the first task. This was done to get a first impression of what can be done in the system. Even after overlooking the button to upload learning style data the blocks were considered again. At this point, the participant was given the hint that each needed control element is located in the area of the LSA-Plugin. Four participants stated to have heard about learning styles and one stated to have good knowledge. The knowledge about the Felder-Silverman Learning Style Model was a bit lower. Two participants had never heard about it, two did have heard about it and
9.3. Results

One participant stated to have good knowledge. This led to some difficulties with the learning style selection control where one participant struggled with finding the learning styles within the control.

After the survey was done, the four participants without good knowledge of the FSLSM were lectured about it and the one with good knowledge just given the handout. Each participant did the ILS and was lectured about the categories and how the results are transformed into JSON format. Combined with this explanation, the file containing the new learning style data is presented to the participant such the first task can be performed.

While performing the jobs on the task list the participant worked on his or her own and was only interrupted if a hint was needed. This happened a lot during the first task were participants did not find the upload button and while one starts searching the blocks, others asked for help. Interesting comments, critique and questions the participants asked themselves or the moderator were noted. Furthermore, conspicuous behavior was recorded such that the participant can be asked about it later. Also, two bugs were found and noted and both times solvable before the next session started. In the second last session, the activities created for task 4 where some activities of the last week should be visualized where done more than a week ago. By visualizing nothing, the participant got confused but calmed by the moderator and for the last session, new activities were created.

One participant was not talkative at the beginning but after a short conversation during the performing of the jobs to get him/her more comfortable, more comments were made and a good feedback session was done after the performing of the task list. Each feedback session was started with the invitation to summarize the experience with the LSA-Plugin. The comments made by the participant were taken up again to get further information about why the comment was made and address the referring element or problem. Besides understanding the problem, the participants were also invited to suggest improvements or made wishes for additional changes or features they think would increase the usability.

Finally, the participant was thanked again and farewelled.

9.3. Results

The hand-recorded notes were stored in two tables (see Appendix E). The first table contained comments made during the performing of the jobs and the second one wishes and suggested improvements. Entries with the same semantical ground were grouped and counted to determine the major statements to deduct the major flaws in the plugin.

9.3.1. Comments

Two kinds of comments were made by each participant:

- “Upload learning style data” button was expected at the top of the page
  Before the plugin can be used in a meaningful way, learning style data need to be provided. Therefore the participant expected something to provide this functionality at the top of the plugin page. The button was overlooked that often that two participants started to search for the feature in the navigation block and administration block. To fix this flaw, the button should be put as expected to the top of the page. But this solution also has to be evaluated to check if this cause other usability flaws (e.g. space lost to the button on a small screen).
9.3. Results

- The naming of the relative value options not clear enough
  The critique of the naming of the two relative value formats (relative (events) and relative (learning styles)) was caused by not understanding what the notes in the brackets describe. Task 6 was always solved via trial and error while each participant did not pay attention to the actually visualized values. This results in wrong visualizations (both learning styles were assigned 100%) and by interpreting it graphically, wrong answers were given. One participant made a wish for an information button describing the meaning of the two kinds of relative values. Such a button can be added e.g. in form of a question mark icon next to the dropdown element which displays the information text when clicked or hovered.

Another comment given by three participants criticizes the positioning of the “Create Plot” button. One participant stated that the position at the bottom disturbs the flow because based on the task list the learning styles are chosen first followed by the settings for the plot at the top and after that, the user has to go to the bottom to create the plot. Two other participants expected this button next to the actual plot. Another participant who did not state any disturbance by the positioning of the button suggested placing the same button additionally next to the plot. By this, the user would be able to trigger the creation at multiple places which might be an advantage if just a small change is done at the top of the page. To fix this flaw, the two options of repositioning the button more closely to the plot or adding an additional one need to be evaluated. A controlled experiment should be set up to test which solution is the most suitable one.

All other comments were made just by one or two participants but two of them are worth a closer look:

- The naming of dummy visualization should be clearer
  This refers to the dummy data visualized if the plugin page is initially called. The x-axis is labeled with three shortenings for months and values called Learning Style Data and Learning Style Data2 with values between 150 and 300 are plotted. These dummy data were generated during the development of the LSA-Plugin to test the Charts API and never changed after. One participant tried to map the visualization to the selected parameters for the visualization and became more confused by not finding any learning styles in it. There are two possible solution for this: The first one renames the dummy data explicit as Dummy Data and the second one would be to plot real values. But plotting real values for some learning styles requires, that learning style data are uploaded. Otherwise, just a notification about the missing learning style data would be displayed which might be confusing for a user who works with the plugin for the first time. Therefore renaming the dummy data is the more suitable way and can be combined with the solution for the next flaw.

- Colors of the learning style selection control were confusing (gray was interpreted as selected)
  Just one participant got the idea of green marking selected learning styles and categories initially. Two other participants took some time before they stated the correct hypothesis about what the colors mark and tried to plot it. To solve this confusion a solution in combination with the solution of the flaw above can be implemented. Just one learning style with two categories will be selected (e.g. Active - Strong and Active - Mild) and only these two are used on the x-axis of the dummy visualization. By this, the differences between the green and gray color could be made clearer, but this needs to be evaluated.
9.3. Results

All other comments were not mentioned often enough to be considered major flaws. Two participants were a bit annoyed by the scale of the plot. Steps of 0.2 were not suitable for a visualization of total numbers and another participant misread a 2.0 as the number twenty. On the one hand, this was caused by the low amount of events that were created for the test course. On the other hand, this should be evaluated with a larger test group and may be fixed to prevent such misreadings.

Another participant mentioned that the naming of the events is not good and clear, especially Course Module Viewed. It could be further investigated if a layer mapping the event names defined by Moodle to more intuitive names should be implemented.

9.3.2. Wishes

No wish was made by each participant. The highest number of participants making the same wish was three for two wishes:

- Grouping PDFs and other files of the same (mime-)type
  This wish arose from task 5 where visits of PDFs should be compared to the visits of the images extracted from the PDFs for visual and verbal learners. All three PDFs and images had to be selected individually and the plot was not easy to read for each participant. Some participants just went with this interference and did not clearly stated this wish, but adding such a feature might improve the usability.

- Add checkboxes to each selected activity type data
  Not all participants were skilled in using the multiselect element. Especially the selection of more than one event resulted in confusion and the usage of the Ctrl-Button was only discovered in a trial and error procedure. The first try was always just clicking on the next event the participant wanted to select which lead to the deselection of the previously selected one. One participant discovers the possibility of selecting multiple events by clicking and moving the mouse by chance. As a result, the wish for checkboxes was made. One of these three participants made the alternative wish for just clicking an event should select it without deselecting others. This wish was also made by a fourth participant. It needs to be evaluated which suggested solution is the more suitable one.

Additionally to the wish for grouping files based on their type, one participant wished for grouping also activities. This should enable investigations on a broader level where e.g. the usage of all forums is compared to the usage of all chats.

Another reasonable wish made by two participant needs to be considered. While the selected events are marked with a blue background color, the activities with selected events are “just” highlighted by the numbers in brackets in front of the activity type and instance name. The two participants wished for an additionally highlighting of the activities by colors.

All other wishes were done by single participants but some of them are worth mentioning. To decrease frustration, the plugin should detect and prevent senseless configurations for the plot. E.g. in a pie chart, only one event can be visualized and if two or more are selected, nothing is plotted. Therefore the plugin could either disable some selections based on previously made ones or show a warning if the user selects senseless parameters.

Live updates were another feature wished by one participant. It was referred to the changes of visualization type and value formats which do not afford a recalculation of the plotted values. This would simplify finding the most suitable visualization for desired data because it gets rid of the delay of reloading the page.
9.3. Results

The labeling of some controls was also criticized by one participant. Renaming *Type of Activity Data* to *Activity and Performance Data* was suggested because besides the logged events, achieved grades and badges can be visualized. Furthermore, it was suggested to limit the length of the activity selection control to prevent increasing the height of the whole page. The learning style selection control would be moved down if the activity selection control becomes higher than the plot. With e.g. 13 events for one forum activity, this can happen really quick if the user does not fold the activities after looking at the events.
In this thesis, the question of how learning styles can be linked with activity and performance data gathered from a Moodle course was answered. In this process, it was looked at which activity and performance data Moodle logs with its event-based logging system and which plugins are available to get some of these logged data from. These possible data sources were held against some requirements and the most promising ones were collected. The final criterion was the question of how these data can be extracted from the plugins. An automatic approach was desired and if the analyzed plugin did not provide one, it was checked, if other plugins could replace the selected one or if an own API could be developed within the time of this thesis.

Here, the Logs plugin stood out which displays data on each triggered event in a table with additional filtering. But the plugin does not provide an API to extract the displayed data - a manual download would be needed. An own API was developed, based on the code of the Logs plugin, and used to get desired event data from the database. Performance data has to be gathered from different plugins with their own APIs. It was looked at grades from quizzes, assignments, workshops and for the course and earned badges. Other additional plugins also provide APIs to get their data like the dedicated time for a course.

The gathered knowledge was applied by developing a Moodle Learning Styles Analytics Plugin (LSA-Plugin) which collects, processes and visualizes data of students actions in and interactions with a Moodle course based on the students learning styles. Merging the heterogeneous data from multiple sources was a big challenge and also creating a good interface. The first version of the LSA-Plugin was evaluated by a usability study with the aim of detecting usability flaws and maybe suggesting fixes. Comments and wishes were collected and reasonable improvements collected and described. But the plugin needs to be tested in a bigger setting to find more flaws and ways to improve the plugin. Some are stated below:

Besides the improvements suggested in the discussion of the usability study results in section 9.3 some more work can be done to further improve the LSA-Plugin.

One improvement can be done for the visualization of the analytic view. Currently, only the sum of activities is visualized in the analytic view while the average values are given in the student view. A checkbox should be added to the analytic view to enable the user to switch between these two options.

During the development, the chart builder class became very big. Single parts like the list of supported activities and events and visualization types should be outsourced to achieve a higher degree of modularity. Furthermore, data defined in some PHP code (e.g. FSLSM) should be transferred to JavaScript code such that it will not be redefined.
This could also improve the handling of the time periods in which values are currently hard coded such that the PHP and JavaScript scripts could access them more easily.

A new module which can be implemented in a future version of the LSA-Plugin could expand it with more standardization. With the usage of the Logstore xAPI, the entries of the logstore_standard_log can be transformed into xAPI statements. Providing an interface which is able to process these statements is also a step towards working with more homogeneous data.

Currently, the LSA-Plugin reads to the logstore every time, a chart should be plotted. In the test courses, generated during the development only a few users were enrolled and test activities were done manually. Therefore, the number of logs was pretty small. Imagining a course with more than 1000 user (considering Massive Open Online Courses (MOOCs)) which do multiple actions and interactions during one day, the number of logs will grow pretty fast. Additionally, the logs from other courses are stored in the same database resulting in the need to check each entry if it is related to the course, the plugin is used in. This also applies to the data stored by the used additional plugins like Course dedication and Use stats. The more logs are generated, the longer it will take the plugin to run through each database.

One option to cope with this challenge is to implement a cron process. A cron process is a script regularly run in the background. It can be used to run through the databases containing needed data to run the plugin and store them in a database within the LSA-Plugin instance. Each entry can be filtered for the course ID and log data furthermore based on the supported events. Such a process could be run e.g. every 15 minutes and an explaining disclaimer added to the interface. But if a user is willing to wait for the plugin to run through all entries to get the latest possible data, an option can be provided in the plugin settings to switch between these modes.

There are more interpretations of learning style selections. One use case is to visualize only the data of users matching all selected learning styles. In the example given in Figure 8.9, only one bar would be shown with only the values of User 1. This setting can be achieved e.g. with an additional checkbox but senseless configurations (selecting contradicting learning styles resulting in an empty set of user IDs) have to be prevented.

Furthermore, the control element should be improved. As found out during the evaluation, it was not very intuitive to operate. Selecting all learning styles per default and having them all colored green led to confusion. Therefore, the approach of selecting only a few learning styles, matching a dummy visualization (see subsection 9.3.1) should be implemented and evaluated. The option to click and hold a mouse button and move the cursor over the single box elements was not discovered by any participant of the study. This interaction was conducted and implemented to increase the usability of the element and should be more promoted by the layout and control design. An on-click tooltip next to the control could be the first step but will not be enough because not everybody reads tooltips, especially if the control is operatable by single clicks such that no other way is thought of.

Currently only one comparison - the average values for the selected activities - can be done in the student view. A student could additionally compare him- or herself to subgroups of the course like his or her project team. But this should be considered carefully because in too small groups it might be possible to link actions to group members based on what the student know about them.

---

1 https://docs.moodle.org/34/en/Cron
Another privacy concern relates to the accessibility of the plugin for students for whom no learning style data are given. Only students who provide their learning style data, such that the teacher can upload them to the LSA-Plugin, should be able to visit the plugin. This is also motivated by the need for learning style data because without them, no visualizations are possible. Therefore the `handle_access` function of the student overview controller should be extended.
Appendix A Bibliography


[2] Christopher W. Allinson and John Hayes. The learning styles questionnaire: An alternative to kolb’s inventroy?


A. Bibliography


A. Bibliography


Appendix B Frontend Mockup Example
Appendix C Handout Felder-Silverman Learning Style Model
Appendix D Handout Index of Learning Styles Results

Converting ILS Results into JSON

FSLSM Categories
## Appendix E  User Study Results

<table>
<thead>
<tr>
<th>Comments</th>
<th>8E32</th>
<th>AP7G</th>
<th>BSDP</th>
<th>FAW1</th>
<th>GMKV</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Upload learning style data” button was expected at the top of the page</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>5</td>
</tr>
<tr>
<td>The naming of the two relative value options not clear enough</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>5</td>
</tr>
<tr>
<td>“Create plot” button at the button distrubs the flow/was expected near</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>to the actual plot</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The naming of dummy visualization should be clearer</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Colors of the learning style selection control were confusing (gray was</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>interpreted as selected)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checkbox for grouping learning style categories not outstanding</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>If the multiselect control is folded just the number in brackets do not</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>highlight selected fields good enough</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finding single learning styles in the learning style selection control is</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>hard</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scale with 0.2 steps confusing (Task 4)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>The naming of the events confusing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Learning style selection control not easy to identify</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Floating point value for low values confusing (2.0 read as 20)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

*Table E.1.: Comments of User Study Participants*
<table>
<thead>
<tr>
<th>Wishes</th>
<th>8E32</th>
<th>AP7G</th>
<th>BSDP</th>
<th>FAW1</th>
<th>GMKV</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grouping PDFs and other files of the same (mime-)type</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Add checkboxes to each selectable activity type data</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>3</td>
</tr>
<tr>
<td>Highlight activities with selected events with colors</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Selecting multiple events by just clicking the names without the need of holding the “Ctrl”-Button</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>More flow in creating the plot (maybe with graphical support)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Information button about the relative value formats</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>System should prevent or warn if sensless configurations are done</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Learning style selection control a bit smaller</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Live update of the plot if visualization format or value format is changed</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Additional “Create plot” button next to the actual plot</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Rename “Create plot” to “Update plot”</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Rename “Select Activity and Performance Data”</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Limit the length of the activity types list</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Make it possible to group all activities based on their kind</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Reset buttons for selected activity data and selected learning styles</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>If only one event is selectable for an activity, just clicking on the label of the activity should be enough</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Table E.2.: Wishes and Suggested Improvements of User Study Participants
Appendix F  Digital Appendix
Eidesstattliche Versicherung

Judel, Sven 335203

Name, Vorname Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/Masterarbeit* mit dem Titel

Activity Logging Approaches and Data Correlations with Learning Styles

selbstständig und ohne unzulässige Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätz-lich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen, September 21, 2018

Ort, Datum Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen, September 21, 2018

Ort, Datum Unterschrift