electronics

MDPI

Article

Multi-Agent System Observer: Intelligent Support for Engaged E-Learning

Igor Vuković [1], Kristijan Kuk [2], Petar Čisar [2], Miloš Banđur [3], Đoko Banđur [3], Nenad Milić [4] and Brankica Popović [2,*]

[1] Ministry of the Interior of the Republic of Serbia, 11080 Belgrade, Serbia; igor.vukovic@mup.gov.rs
[2] Department of Informatics and Computer Sciences, University of Criminal Investigation and Police Studies, 11080 Belgrade, Serbia; kristijan.kuk@kpu.edu.rs (K.K.); petar.cisar@kpu.edu.rs (P.Č.)
[3] Faculty of Technical Sciences, University of Pristina in Kosovska Mitrovica, 38220 Kosovska Mitrovica, Serbia; milos.bandjur@pr.ac.rs (M.B.); djoko.bandjur@pr.ac.rs (Đ.B.)
[4] Department of Criminalistics, University of Criminal Investigation and Police Studies, 11080 Belgrade, Serbia; nenad.milic@kpu.edu.rs
* Correspondence: brankica.popovic@kpu.edu.rs

Abstract: Moodle is a widely deployed distance learning platform that provides numerous opportunities to enhance the learning process. Moodle's importance in maintaining the continuity of education in states of emergency and other circumstances has been particularly demonstrated in the context of the COVID-19 virus' rapid spread. However, there is a problem with personalizing the learning and monitoring of students' work. There is room for upgrading the system by applying data mining and different machine-learning methods. The multi-agent Observer system proposed in our paper supports students engaged in learning by monitoring their work and making suggestions based on the prediction of their final course success, using indicators of engagement and machine-learning algorithms. A novelty is that Observer collects data independently of the Moodle database, autonomously creates a training set, and learns from gathered data. Since the data are anonymized, researchers and lecturers can freely use them for purposes broader than that specified for Observer. The paper shows how the methodology, technologies, and techniques used in Observer provide an autonomous system of personalized assistance for students within Moodle platforms.

Keywords: educational data mining; engaged learning; intelligent tutoring systems; Moodle; multi-agent system

Citation: Vuković, I.; Kuk, K.; Čisar, P.; Banđur, M.; Banđur, Đ.; Milić, N.; Popović, B. Multi-Agent System Observer: Intelligent Support for Engaged E-Learning. *Electronics* **2021**, *10*, 1370. https://doi.org/10.3390/electronics10121370

Academic Editors: Imre J. Rudas and György Eigner

Received: 16 May 2021
Accepted: 5 June 2021
Published: 8 June 2021

1. Introduction

With the implementation of new technologies in educational systems, the goal is to increase the quality of teaching, that is, to improve the process of students' acquisition of knowledge [1]. Most educational institutions today use the internet in the learning process [2], and virtual teaching is becoming part of the program, even of institutions known for a long, campus-based tradition [3]. On the other hand, the current example of the rapid spread of the COVID-19 virus puts into focus the significant advantage of the application of distance learning, which is the continuity of the teaching process in different emergencies.

Contemporary distance learning has, above all, enabled students to access teaching content on their plan, independently deciding on the schedule and how much time they will spend learning [4]. The most common way that teaching is provided to students via the internet and other computer networks is learning management systems (LMSs) [2]. LMSs offer lecturers the opportunity to distribute information to students, produce teaching materials, prepare assignments and tests, initiate discussions, manage distance learning classes, and facilitate learning through collaboration through forums [5].

Modular Object-Oriented Dynamic Learning Environment, abbreviated as Moodle, is the most used and widespread LMS solution [4]. Although LMS Moodle has many features

that help lecturers create and manage courses available through a computer network, it does not meet students' individual needs [6]. Artificial intelligence support for lecturers and students could provide what Moodle lacks [7]. Learning is a cognitive activity that varies from student to student, and tailoring e-learning to students requires modeling their cognitive characteristics, especially learning styles, as the most explored cognitive traits [8]. Data learning techniques can be applied in distance learning systems [9], aiming to study user behavior, perform behavioral assessment and improve the system to support the user [5].

The Observer system presented in this paper is a multi-agent system that enhances the Moodle platform into intelligent tutoring systems (ITS). Such systems try to understand the process of individual learning. They are tasked with building an appropriate data structure to represent the student's cognitive characteristics as realistically as possible [1]. The Observer system is somewhat different because it focuses on a single learning style, and it is an engaged learning style. It monitors engaged learning indicators continuously, drawing students' attention to whether they are on track to complete the course successfully or need additional effort.

Engaged learning can be implemented without the use of technology, but technology facilitates engagement in a way that is difficult to achieve otherwise [10]. Research related to engaging learning is focused on the student's activities, his/her involvement, and his/her efforts in achieving academic success [3]. A specific study presented in [2], which compared 17 blended courses using Moodle LMS, showed that the number of clicks, time spent in interaction with distributed teaching material, and an overall number of visited pages significantly increased the positive effect to the final grade in the exam. Having that in mind, the idea behind the implementation of the Observer system was to provide information to students about their progress and current level of engagement by comparing previous measurements with current ones, as well as with the indicators of measurement of other students in the same course.

What sets Observer apart from most LMS learning support systems is the data source. Although Moodle keeps detailed logs regarding the use of systems to track what materials a student has accessed [5], the Observer system uses data that it collects, anonymizes, and stores on its own. In this way, a large amount of information is provided that can extend the system's purpose shown in this paper. Additionally, Observer enables the creation of a centralized system that can connect an arbitrary number of Moodle platforms of different organizations, thereby enhancing its data learning capabilities as well as third-party analytic capabilities.

The rest of the paper is organized as follows: A brief overview of related work is presented in Section 2, emphasizing the difference between them and our approach. In Section 3, we discuss engaged learning from the perspective of the basis for ITS development. Section 4 presents the proposed multiagent system Observer, while in Section 5, we describe the technologies used to develop it. Finally, in Sections 6 and 7, we summarize our findings, draw some conclusions, and state directions for future work.

2. Related Work

MIMLE is a learning system presented in [1] that uses one of the diagnostic techniques of an ITS system called model tracing. It offers students theorems and definitions through a help window to help solve a given problem. The applicable agent uses the Markov process of deciding whether or not a help window should appear to minimize disruption to the student. It is a reflex agent that makes decisions based on counting correct and incorrect answers and measuring a student's reaction time. The Observer system tries to be less disruptive to the student but with a different approach than MIMLE, choosing the particular Moodle interface item called a block.

One of the solutions implemented using the Moodle system is presented in [7]; it is a multi-agent system based on the Java Agent Development Environment (JADE). Based on the information collected from the Moodle databases, the system provides information to

students and lecturers about student activity. For example, the number of posts on each student's forum, whether there are students who have not posted yet, whether there are any disagreements and similar activities. The Observer system's most significant difference is the data source for analysis since Observer itself collects them and is not connected to the Moodle database.

Predicta presented in [4] is a Java desktop tool for tracking students based on Moodle database records. It selects tables according to user needs and prepares data for analysis in WEKA data learning software, displaying results in the dedicated module. The second component of the system predicts student success, that is, whether there is a risk of failure. Predicta uses the Moodle database, but concerning the LMS, it is the external solution itself as well as Observer. The difference is that with Observer, the filtering of data is done by lecturers or course creators who choose teaching material or Moodle pages from which information is gathered.

In [6], a nonagent-based example, such as Predicta, is presented, but it supports a specific learning style, as is the case with the Observer system. It is a solution that supports self-regulated learning across all three stages. The planning phase is supported here by choosing a predefined learning profile after which the system offers links to access the learning recommendations in the main menu of the Moodle plug-in. The learning phase provides a service that recommends learning materials (various content, exercises, and tests) that are neither too easy nor too difficult for a particular student. At this stage, self-regulation is based on the student's free choice among the learning material offered. The reflection process system supports two different tools: the progress rating tool and the visualization tool. Although always present, or to the extent determined by the course creator, the Observer system in the student learning process, using the LMS, participates only in measuring its engagement. Observer predicts the course's outcome by processing the measuring and gives suggestions while trying to stay unobtrusive.

In order to achieve as little disruption as possible, the Observer system uses a novel approach of agents interacting with students, using the Moodle block. The application of the block itself is innovative since it is used to implement the agent. The application of the block allows the data to be filtered by the creators of the course themselves, choosing positions in the learning material where the block will be placed. The Observer system improves the individual work of students, offering them the opportunity to be informed of, based on indicators of this engaged learning style, whether their activities are within the group of students with a chance for success or they need to invest additional effort. As an additional contribution, the Observer system provides an anonymized dataset for further analysis with a potential connection of an external entity via a web service or creating an analytical solution within the system itself.

3. Theoretical Background

Engaged learning theory represents a fundamental basis for the development of Observer. It is an active student's cognitive process and involves creating, reasoning, problem-solving, decision making, and evaluating what has been achieved. The theory behind this learning style is based on the idea of creating successful, collaborative teams engaged in ambitious projects whose significance goes beyond classroom frameworks [10]. Before broader acceptance of LMS, student engagement was most often measured by attendance, which is a raw measure of participation because it does not measure quality but has proven to be an essential variable in determining student success [3]. The time that students spend on a particular task, called engagement time, is of great importance for our research also.

Student engagement refers to the degree of attention, interest, and commitment when learning or in class. Students committed to improving their skills and knowledge will undoubtedly be active within the virtual learning environment, such as Moodle, so engagement can be estimated using such systems logs [11]. The Observer system measures learning engagement in a virtual environment based on the number of clicks, time spent

on the teaching material, and the number of reviews, all of which have proved to be an important indicator of success [2]. After measurement completion, the system provides suggestions to the student regarding his/her engagement by comparing it with their previous engagement and the engagement of other students.

Concerning engagement indicators, one should note the research [3] conducted on a large sample of students—91,284 who used Blackboard LMS and 1515 who used Moodle—which is considered to be statistically significant. The study found a general correlation between student engagement indicated by clicks and academic performance indicated by grades. However, there is a limitation related to the number of clicks as an indicator of engagement. Although a link to a student's final success in a particular course exists, its importance to success is not determined.

In applying the indicator, portability is essential, which is related to the question of whether there is a universal set of variables for predicting student success. Studies have shown that the portability of prediction models is not high between different courses. However, a prediction is successful when it comes to a particular course, while grouping data at the student level makes it easier to find portability between different courses [2]. For this reason, during indicators processing, Observer considers all data related to a particular student, i.e., the system user, and the data of other students who work with the same course's teaching material.

4. Observer ITS System

The Observer system is a multi-agent system that assists students in engaging in learning by using LMS Moodle. The system independently collects student activity data, anonymizes and stores it in its database, prepares to extract engagement learning indicators, creates a training set and data-learning set, and implements two different machine learning algorithms for passing on suggestions to students. It can be used with a specific Moodle platform by educational or other institutions, or it can be the focal point of multiple Moodle platforms.

Software agents can be defined as programs that act on behalf of human beings, resolving inconsistencies, finding and integrating information from different sources, filtering out irrelevant or unwanted information, and adapting them to human needs [7]. Pedagogical agents are autonomous software entities that support the learning process through interaction with students, lecturers, and other participants and in collaboration with other similar agents [12]. The Observer system uses the following: tutoring agents; an agent who collects data, stores them in the system's internal database, and informs tutoring agents about student suggestions from the system; a data preparation agent; and an agent who learns from the data and forwards the results (as shown in Figure 1).

Agents communicate through a WebSocket protocol that enables two-way communication and has already been used to develop multi-agent environments for this purpose [13]. Different from the usual purpose of communication channels between agents, in the Observer system, instead of conversation using some of the agent languages, only a limited set of data are exchanged through messages, due to the potentially high dynamics in communication.

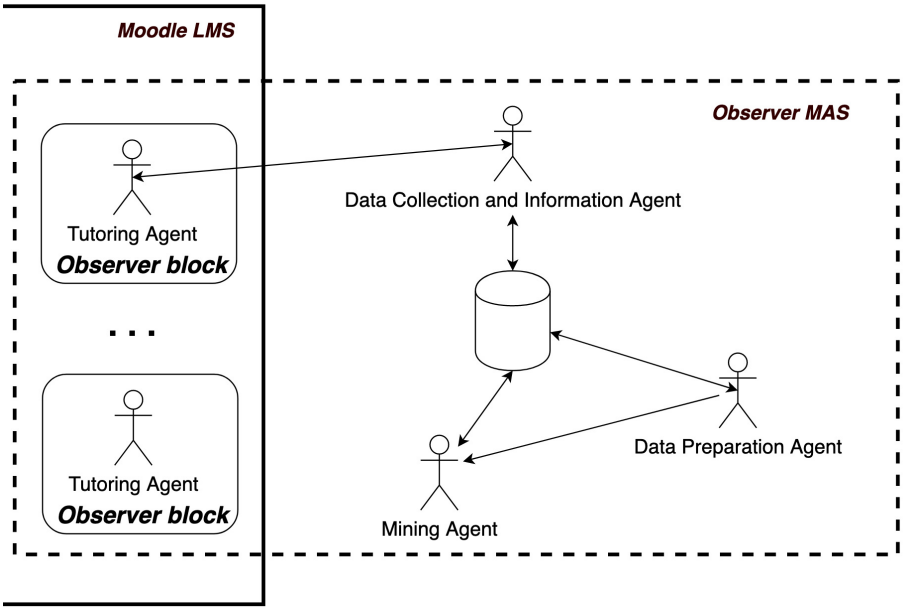

Figure 1. Model of the multi-agent Observer system.

4.1. Tutoring Agents

The Observer ITS tutoring agent is dedicated to monitor the student's work, record his/her interaction with the teaching material, and inform students how the system values their efforts. For that purpose, the LMS Moodle elements named blocks are used to implement the agent. The blocks are graphical interface elements added to the left, right, or center column of the Moodle Page. Blocks are placed by the Moodle administrator or course creator on each specific page of teaching material where they deem it necessary or binds the presentation to the block setting's context, allowing blocks to be placed on multiple pages at once. For an example of the Observer block placed on the LMS Moodle page, see Figure 2.

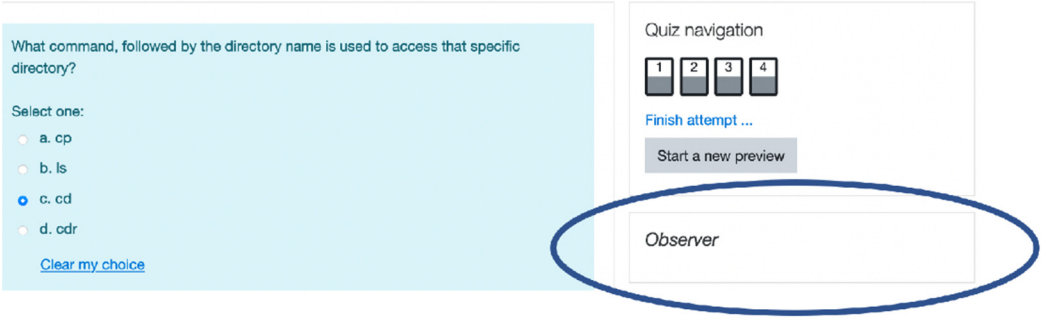

Figure 2. The layout of the Observer block added to the quiz page.

It should be noticed that some activity modules (e.g., quiz modules) do not allow blocks to be displayed by default. Administrator or course creator could solve the problem in the setup of the quiz module.

Block Observer performs three essential functions from the perspective of the system. The first function is to retrieve the username and email address from the Moodle database and generate hash values, using the MD5 algorithm to obtain the user (student) identification number. This number is linked to all subsequently collected data, which makes them anonymous. Figure 3 shows a section of the block's PHP and JavaScript program code that provides an identification number and creates a hidden component of an HTML

document whose value is student identification. Figure 4 contains the JavaScript code for the module.js file that creates and runs an agent.

```
$this->content = new stdClass;
$this->content->text = '';
$attributes = array();
$attributes['id'] = 'observer_user';
$attributes['value'] =  md5($USER->username.$USER->email);
$attributes['type'] =  'hidden';
$this->content->text .="<span id='suggestions'></span>";
$this->content->text .= HTML_WRITER::empty_tag('input', $attributes);
```

Figure 3. Observer block PHP program code.

```
M.block_observer = {
    initObserver: function () { agent=new Agent(); agent.start(); };
...
class Agent {
   start(){
      var user=document.getElementById("observer_user");
      var x = document.getElementsByClassName("page-header-headings");
      this.communication(
        {
           user: user.value.
           course_title: x[0].innerHTML.replace(/(<([^>]+)>)/ig,""),
           full_title: document.title,
           time: Math.floor(new Date().getTime()/1000);
           type: 'open'
        });
document.addEventListener('mousedown',this.addEvent);
   }
   communication(message){
      return new Promise(function(resolve,reject){
         var ws = new WebSocket('ws://localhost:8080/');
         ws.onopen= function(){
         ws.send(JSON.stringify(message));
   };
```

Figure 4. JavaScript code of the Observer block.

The space in the created block, shown in Figure 2, is used by an agent to display the Observer system suggestion. Offering suggestions to the student is another crucial role of the Observer block for the system's functioning. The third function is to implement agent code and include the agent in the Moodle environment. Opening each new Moodle page that houses the Observer block creates an agent class object.

After creation, the agent starts to collect information, including identification number, the course title, and HTML document title, as well as the time that the agent is created, and sends a message to the agent for data collection and information. In response, the tutoring agent receives a current Observer system suggestion, which then displays via the block. On the creation, the tutoring agent also starts to register students clicks on the Moodle page. After each click, the agent sends a message to the data collection and information agent in the form of a JSON object.

Algorithm 1 shows the process of data collection by a tutoring agent.

Algorithm 1 Data collection by a Tutoring Agent.

```
   Input: The text content of the clicked HTML element
   Output: Collected data in JSON format
function createTutoringAgent(t)
   type←"open"
   uID←getUserId("observer user")
   pTitle←getTitleFromPage()
   sendToDataCollect&InfoAgent(type,t,uID,pTitle)
   addClickListener()
end function
function onClick(tText,t)
   type ←"click"
   uID←getUserId("observer user")
    pageTitle←getTitleFromPage()
   sendMessageToDataColl&InfoAgent(type,t,uID,pTitle,tText)
end function
if Course page is open then
   createTutoringAgent()
end if
if Click registered then
   onClick(targetText, currentTime)
end if
```

4.2. Data Collection and Information Agent

There are three agents on the server-side of the system and they have the highest workload in processing the collected information. The first is a data collection and information agent who communicates with the tutoring agents, receives new information from them, and sends suggestions from the system. The message received from the tutoring agent is parsed, after which the JSON object is created.

Agents use objects when storing the collected data in the database. Among the data obtained from the JSON object is time. The timestamp accuracy depends on the settings on the client device itself, which the data collection and information agent could unify by downloading time from the server. The purpose of the time information is not to determine when something happened but decide how long the student stays on a particular page; therefore, accuracy is not necessary.

In addition to storing data, the agent checks in the database for system suggestions intended for the student for whom the particular tutoring agent works. The agent translates the results of machine learning from a numerical value into a human-readable message. The agent also adds information about how old the suggestion is, and both pieces of information are sent back to the tutoring agent.

4.3. Data Preparation Agent

Preparing data for mining involves extracting student engagement indicators based on the information gathered by the tutoring agent. The preparation agent is activated every 10 min and prepares, up to that point, unprocessed data. Figures 5 and 6 show the raw data from the Observer system repository and the data after processing and extracting the engaged learning indicator from the raw data.

For the data preparation process, the relevant information is a raw data record type. The type "open" suggests opening the page whereby the tutoring agent is created and counting this kind of record system, extracts the first indicator—the number of views. The "click" type marks the data collected after registering the student click and gives another indicator. For the third indicator, the agent uses recorded times. Except for engagement indicators, the agent includes the username, name of the course, and process start time in a prepared set. The information added to the indicators is later used in the data mining process and as references for gaining insight into the results, i.e., informing.

```
{
  "_key" : "491838",
  "_id" : "student_activity/491838",
  "_rev" : "_aO57_fS-_2",
  "user" : "ad23ff0b59939d565800cc4acb3f848f",
  "course_title" : "Linux - Basic Course",
  "full_title" : "Linux Basic: Lesson: mv",
  "time" : 1585347655,
  "type" : "open",
  "processed" : "yes"
},
{
  "_key" : "491835",
  "_id" : "student_activity/491835",
  "_rev" : "_aO57_fW--F",
  "user" : "ad23ff0b59939d565800cc4acb3f848f",
  "course_title" : "Linux - Basic Course",
  "full_title" : "Linux Basic: Lesson: rm",
  "time" : 1585347654,
  "type" : "click",
  "processed" : "yes"
}
```


Figure 5. Raw data obtained from a tutoring agent.


```
{
  "_key" : "491876",
  "_id" : "duration/491876",
  "_rev" : "_aO57_fW--C",
  "user" : "ad23ff0b59939d565800cc4acb3f848f",
  "title" : "Linux - Basic Course",
  "views" : 2,
  "clicks" : 2,
  "updateTime" : 1585347720,
  "time" : 29
}
```


Figure 6. Extracted indicators of engaged learning.

Upon completing the data preparation, the agent forwards a message to the data mining agent containing information about the processing's start time. The process of data preparation is shown in Algorithm 2.

Algorithm 2 Extraction of student learning indicators by the Data Prep. Agent.

Input: list ← (uID_1, t_1, $type_1$), (uID_2, t_2, $type_2$), . . . (uID_n, t_n, $type_n$)
- List of collected data filtered by student uID
Output: Indicators of engaged learning - *clicks*, *time*, *views*

clicks ← 0, *views* ← 0
for (i ← 0; i < *list.length*) **do**
  if I = 0 **then**
    t_{start} ← *list* [0][2]
    user ← *list* [*i*][0]
  end if
  if *list*[*i*][3] = "open" **then**
    views++
  end if
  if list[i][3] = "click" **then**
    clicks++
  end if
  t_{end} ← *list* [*i*][2]
end for
time ← ($t_{end} - t_{start}$)
saveIndicatorsForStudent(*user*, *clicks*, *time*, *view*)

4.4. Data Mining Agent

The work of a data mining agent starts with the message that it receives from the data processing agent. Based on the information about the start of the preparation process contained in the message, the agent recognizes new data and creates a query to the database. The result of the query is a multidimensional array with prepared data and references. Due to the disproportionate value of the indicators in the prepared set (the number of views and clicks can be significantly less than the number of seconds spent using the course material), Observer normalizes the data by the minimum and maximum values method. The agent uses the k-nearest neighbors (k-NN) algorithm to classify the data by calculating the approximate distance between different points on the input vectors and then assigns to the unmarked point the class of its k nearest neighbors. The distance $D(i,j)$ between the samples we determined by calculating the Euclidean distance.

The last activity of a data mining agent is to determine the direction of regression. The purpose of the regression direction, which is determined using the least-squares method, is to include the time-flow component as a vital factor of engaged learning, especially in the case of making too many multi-day breaks between the use of teaching materials distributed through the LMS Moodle. The system uses the implemented regression method's results to supplement the suggestion with information about the student's overall engagement. Figure 7 shows the result of the data mining agent's work, where the current suggestion is obtained with the k-NN algorithm and overall with linear regression.

Figure 7. Example of Observer block suggestion.

Algorithm 3 presents the process of forming a suggestion intended for a specific student.

Algorithm 3 Forming a suggestion for a student by Data Mining Agent.

Input: $list \leftarrow (uID_1, clicks_1, time_1, views_1), \ldots (uID_n, clicks_n, time_n, views_n)$
- obtaining learning indicators of all users in the last ten minutes
Output: A suggestion for a student
allClicks←getClicksSortByNum()
allTimes←getTimesSortByDuration()
allViews←getViewsSortByNum()
$(C_{norm}, T_{norm}, V_{norm})$ ←minMaxNormalization(*allClicks*, *allTimes*, *allViews*)
trainingSet←getKnnTrainingSet(C_{norm}, T_{norm}, V_{norm})
for (*i*← 0; *i* < *list.length*) **do**
  currentSugestion←applyKnnlAgorithm(*list*[*i*], *trainingSet*)
  prevResults←getPreviousStudentSugestions(*list*[*i*][0])
  overallSugestion←applyLinearRegression(*currentSugestion*, *prevResults*)
  createMessageForStudent(*currentSugestion*, *overallSugestion*)
end for

5. Implementation of Observer System

For developing the multi-agent Observer, we use JavaScript programming language. The principal reason for choosing JavaScript is related to the concept of monitoring students' work, which is based on their interaction with the teaching material. Since everything relevant for the Observer system is happening on the Moodle platform's client-side, JavaScript is the logical solution. JavaScript is the leading scripting language for internet browsers, a de facto standard when it comes to the client-side, and essential for the development of modern web applications [13]. The implementation of the Node.js software environment allows JavaScript to be used on the server-side as well. It is an open-source software based on Google's V8 core and event-driven architecture that enables asynchronous I/O, as well as creating extensible server applications without using threads [14]. It is efficient and allows the development of web applications that work intensively with real-time data [15]. Unlike traditional, multi-threaded software models, where the thread has to wait for the demanding I/O operation to finish, Node.js can delegate such activities as asynchronous operations, thus not degrading performance [16]. However, the Observer works inside environments with large numbers of students or connecting multiple Moodle platforms. In that case, there could be a blockage of the only thread used by Node.js (looper thread), primarily during the execution of a data preparation agent and a mining agent code.

Agents need to be run as child processes to avoid blocking due to processing large amounts of data [17]. The multi-agent approach in itself has the potential to expand the system by adding new agent centers, which, through distributed processing, ensures load balancing but also avoiding the existence of a single point of failure of the entire system, enabling the execution of computer-intensive tasks [17].

The NoSQL ArangoDB database is the core of the system in which agents store all collected data, whether raw, prepared, or data representing learning results. What distinguishes NoSQL databases is their horizontal extensibility, storing different data structures with less demanding hardware [18]. ArangoDB is a multi-model database supporting various data models, storing key-value pairs, documents, and graphs; all data can be accessed in the same ArangoDB Query Language (AQL). Compared to other similar databases that specialize in graphing, Arangodb leads in performance, but its main advantage is its multi-model architecture [19].

6. Results and Discussion

Agents in the Observer system are designed to meet a minimum set of fundamental traits that characterize software agents, such as being autonomous, capable of operating as a standalone process and performing actions without user intervention [20]. We create agents according to the reactive or executive agents' model, representing the type of agents that can only directly follow the basic program (task-oriented model of agents versus goal-oriented model). They are considered primitive, not applying reasoning for the causes or effects of their action. We use this model in the research because of its main characteristic: efficiency based on simplicity [21]. As they are executed within browsers, tutoring agents must not significantly affect resources and interfere with the work of the Moodle platform. On the other hand, agents that accept and process data are oriented toward reactivity due to the projected large amount of data.

To evaluate the system's operation, we experimented with actual university Moodle LMSs during distance learning due to COVID-19. The experiment with the Observer system was performed on two different courses, undergraduate and master studies, which were attended by a total of 36 participants.

Over three months, the system collected a total of 4913 records from tutoring agents. Table 1 shows a preview of the results of the work of the data preparation agent. The table contains the learning indicators collected by Observer after normalization as part of the preparation for the application of data mining algorithms and performed by the data mining agent.

Table 1. Learning indicators extracted and normalized.

Student ID	Visits	Clicks	Time
04a7aa6645b28f848253ec5132290ae1	0.04348	0.01117	0.07731
268046f034b2e014dabf73ba8e780cf9	0.02174	0.00559	0.00000
34c01e64351a8f1e0176d90d32b0a99a	0.39130	0.39665	0.55339
3acb0baab491ea28de32247da3e9a992	0.08696	0.05028	0.14701
5d781e010054aeeebf26d029664d6618	0.34783	0.46369	0.26440
...	...	...	...

Figure 8 shows the result of applying the final kNN classifications to the data collected at the end of the course, shown in Table 1, on which the predicted exam outcomes are performed.

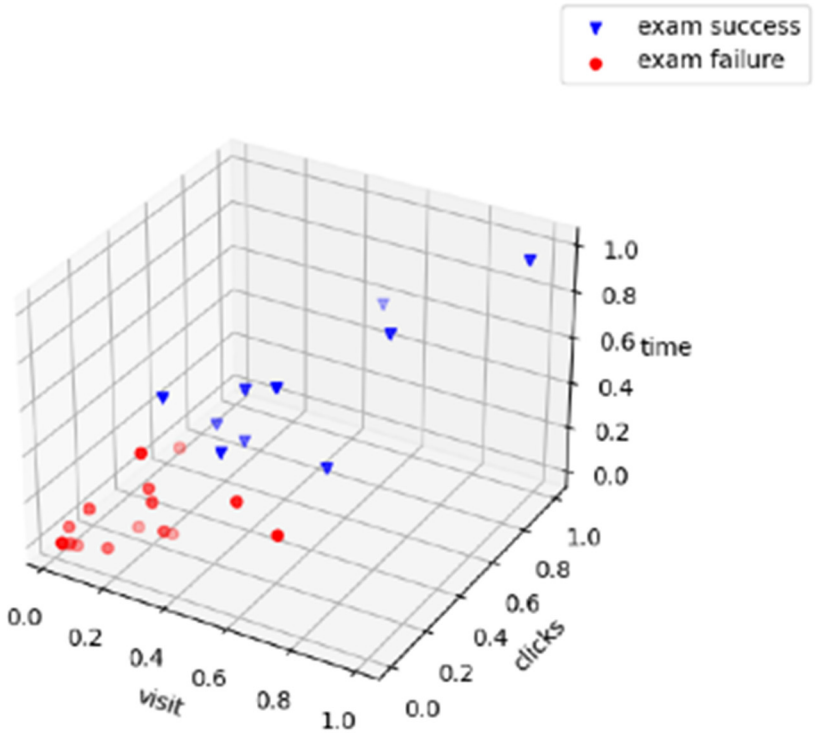

Figure 8. Result of k-NN classifications base on collected data.

After the end of the course, 31 students took the exam. Based on students' data known to the creator of the courses, we created hash values and compared the results of the system prediction with the actual results on the exam. The system achieved an accuracy of 0.742, or related to the confusion matrix, we observed eight true positives, fifteen true negatives, two false positives, and six false negatives.

At this point, we have to address some limitations we had regarding objective evaluation of proposed system effectiveness. Although it was implemented in the university Moodle LMS for distance learning due to COVID-19, we were not able to communicate directly with students in order to get their feedback regarding the Observer utilization. At this point, we do not have information about their experience with the Observer system and whether the suggestions they received from it affected, in any way, their further engagement. We could measure and evaluate only the parameters of their engagement and consequently, produced predictions of their success on exams, for which the system achieved a reasonable accuracy score (0.742). Our plans are to address this issue in the experiment we are going to perform in the next semester in autumn.

7. Conclusions

In this paper, we introduced Observer—the multi-agent system for intelligent support for engaged students. It combines two different machine learning methods that, based on the student engagement indicators, predict success at the end of the course and then make suggestions based on that prediction. The primary measures of engagement, teaching attendance and additional indicators previously found to be related to the student's final success in the course, are utilized in the Observer system. They include the number of clicks, number of views that are, access to various pages, and the time that the student spends using the Moodle Course's teaching content. Although there are limitations to the use of indicators, such as the fact that click counting cannot determine the learning process's quality, it should be noted that neither can the grade itself. It only indicates that the assessment criteria are met, which may or may not be a measure of learning effectiveness [3].

The goal of the proposed system is to support the student but not to interfere with his/her work. The element of the Moodle interface called block is selected for communication with students because it occupies a small portion of the Moodle page. Observer block contains only a brief suggestion and information on how old the suggestion is. In addition, the Observer system is devoid of redundant functionality, designed to be mostly autonomous, which is why we choose the agent architecture. Other used technologies provide a reasonable basis for upgrading the role of the Observer system in distance learning. Node.js is designed to build a real-time system, and the multi-model ArangoDB database can handle a wide variety of datasets, most notably enabling the use of graphs that allow new dimensions to analyze and learn from data.

This paper has shown that through the application of data mining, an intelligent and mostly autonomous system can be developed to support students in engaged learning, while not neglecting the critical role of lecturers and course creators. Lecturers who provide teaching material distributed through the Moodle platform by selecting the appropriate Observer block location give significance to any data collected and analyzed. The database containing anonymized data allows their use much more widely than Observer's primary purpose. Analysis can be directed toward evaluating existing courses, comparing them, and other research since free access to data without compromising student privacy is provided.

Further research will focus on considering the possibility of supporting other learning styles, especially for self-regulating students.

Author Contributions: Conceptualization, I.V. and K.K.; methodology, I.V., K.K. and B.P.; software, I.V.; validation, B.P., P.Č., M.B., Đ.B. and N.M.; formal analysis, K.K. and B.P.; investigation, I.V., M.B., Đ.B. and N.M.; resources, all authors; writing—original draft preparation, I.V., K.K. and B.P.; writing—review and editing, I.V., B.P., P.Č., M.B. and N.M.; visualization, I.V.; supervision, B.P., P.Č. and K.K.; project administration, B.P., P.Č. and M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kuk, K.; Milentijević, I.; Rančić, D.; Spalević, P. Pedagogical agent in Multimedia Interactive Modules for Learning–MIMLE. *Expert Syst. Appl.* **2012**, *39*, 8051–8058. [CrossRef]
2. Conijn, R.; Snijders, C.; Kleingeld, A.; Matzat, U. Predicting student performance from LMS data. *IEEE Trans. Learn. Technol.* **2017**, *10*, 17–29. [CrossRef]
3. Beer, C.; Clark, K.; Jones, D. Indicators of engagement. *Curric. Technol. Transform. Unkn. Future* **2010**, *2010*, 75–86.
4. Felix, I.; Ambrosio, M.A.P.; Neve, P.S.; Siqueira, J.; Brancher, J.D. Moodle Predicta: A data mining tool for student follow up. In Proceedings of the 9th International Conference on Computer Supported Education, Porto, Portugal, 21–23 April 2017; Volume 1, pp. 339–346.

5. Romero, C.; Ventura, S.; Garcıa, E. Data mining in course management systems: Moodle case study and tutorial. *Comput. Educ.* **2018**, *51*, 368–384. [CrossRef]
6. Kopeinik, S.; Nussbaumer, A.; Winter, L.C.; Albert, D.; Dimache, A.; Roche, T. Combining self-regulation and competence-based guidance to personalise the learning experience in moodle. In Proceedings of the IEEE 14th International Conference on Advanced Learning Technologies, Athens, Greece, 7–10 July 2014; pp. 62–64.
7. Scutelnicu, A.; Kinshuk, F.L.; McGreal, R.; Liu, T.; Graf, S. Integrating JADE agents into moodle. In Proceedings of the 5th International Conference on Computers in Education, Hiroshima, Japan, 10–12 January 2007; pp. 1–6.
8. Despotović-Zrakić, M.; Marković, A.; Bogdanović, Z.; Barać, D.; Krco, S. Providing adaptivity in moodle LMS courses. *Educ. Technol. Soc.* **2012**, *15*, 326–338.
9. Pradana, C.; Kusumawardani, S.S.; Permanasari, A.E. Comparison clustering performance based on moodle log mining. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *722*, 1–11. [CrossRef]
10. Kearsley, G.; Shneiderman, B. Engagement theory: A framework for technology based teaching and learning. *Educ. Technol.* **1998**, *38*, 20–23.
11. Naika, V.; Kamat, V. Predicting engagement using machine learning techniques. In Proceedings of the 26th International Conference on Computers in Education, Manila, Philppines, 26–30 November 2018; pp. 17–20.
12. Kuk, K.; Rančić, D.; Pronić-Rančič, O.; Ranđelović, D. Intelligent agents and game-based learning modules in a learning management system. *Smart Innov. Syst. Technol.* **2016**, *58*, 233–245.
13. Jensen, S.H.; Møller, A.; Thiemann, P. Type analysis for javascript. In *Proceedings of the 16th International Symposium on Static Analysis*; Springer: Los Angeles, CA, USA, 2009; pp. 238–255.
14. Wang, J.; Dou, W.; Gao, Y.; Gao, C.; Qin, F.; Kang, Y.; Jun, W. A comprehensive study on real world concurrency bugs in Node.js. In Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, Urbana, IL, USA, 30 October–3 November 2017; pp. 520–531.
15. Lei, K.; Ma, Y.; Tan, Z. Performance comparison and evaluation of web development technologies in PHP, python, and node.js. In Proceedings of the IEEE 17th International Conference on Computational Science and Engineering, Chengdu, China, 19–21 December 2014; pp. 661–668.
16. Chang, X.; Dou, W.; Gao, Y.; Wang, J.; Wei, J.; Huang, T. Detecting atomicity violations for event-driven node.js applications. In Proceedings of the IEEE/ACM 41st International Conference on Software Engineering, Montreal, QC, Canada, 25–31 May 2019; pp. 631–642.
17. Lukić, A.; Luburić, N.; Vidaković, M.; Holbl, M. Development of multi-agent framework in JavaScript. In Proceedings of the ICIST 2017 Proceedings, Druskininkai, Lithuania, 12–14 October 2017; pp. 261–265.
18. Corbellini, A.; Mateos, C.; Zunino, A.; Godoy, D.; Schiaffino, S. Persisting big-data: The NoSQL landscape. *Inf. Syst.* **2017**, *63*, 1–23. [CrossRef]
19. Fernandes, D.; Bernardino, J. Graph databases comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB. In Proceedings of the 7th International Conference on Data Science, Technology and Applications, Porto, Portugal, 26–28 July 2018; pp. 373–380.
20. Abar, S.; Theodoropoulos, G.K.; Lemarinier, P.; O'Hare, G. Agent based modelling and simulation tools: A review of the state-of-art software. *Comput. Sci. Rev.* **2017**, *24*, 13–33. [CrossRef]
21. Mostafa, S.A.; Ahmad, M.; Mustapha, A.; Mohammed, M.A. A Concise overview of software agent research, modeling, and development. *Softw. Eng.* **2017**, *5*, 8–25.