MOODLE IMPLEMENTATION OF ACTIVITY FOR ON-LINE COLLABORATIVE ONTOLOGY BUILDING

Mile Jovanov Marjan Gushev Darko Martinovikj Stefan Spasovski

Ss. Cyril and Methodius University,
Faculty of Computer Science and Engineering
Skopje, R. Macedonia

ABSTRACT

Moodle is an open source learning content management system, and it is nowadays the most used system in the educational process of the Universities, worldwide. Together with the standard Moodle core there are a variety of Moodle plugins like activity, resource, question etc. Activity plugins are used by the teacher to engage the students in learning the subject by giving them assignments.
In this paper we present the necessary procedure that needs to be followed in the process of development of new plugin for Moodle, the specifics of our new proposed activity implemented as moodle plugin, and the process of its installation. At the end, we present the benefits of the use of the implemented activity, by comparison to the other already existing activities in the Moodle installation.

I. INTRODUCTION

Moodle [1] is a distance learning management system, which is commonly used for educational purposes. Along with the standard Moodle core there are a variety of Moodle plugins. Some of them are installed together with the standard installation and some of them are optional and the administrator can install them later, by choice. Because the Moodle project is an open-source program many third-party software plugins are using the Moodle core and extending the features that Moodle offers, for their own needs [2]. The supported types of plugins are: activity, resource, question etc.
Activity plugins are used by the teacher to engage the students in learning the subject by giving them assignments. The assignments can be graded depending of the type of the activity. Also, the start or the end date of the activity assignments can be specified so the students must respect the time frame given for the activity completion. The installation and the management of these activities are made by the administrator. Each plugin requires minimal operating version of the standard Moodle installation, which depends on the core functions used by the plugin.
Further in this paper we will present the necessary procedure that needs to be followed in the process of development of new plugin (Section II), the specifics of the new proposed activity implemented as Moodle plugin (Section III), and the process of installation and benefits of the use of proposed plugin in Section IV.

II. CREATING AN ACTIVITY PLUGIN FOR MOODLE

Every activity module in Moodle must follow some basic rules for file locations and mandatory file names [3]. A plugin can be a standard plugin, included in the Moodle installation, or an add-on plugin, which the administrator can add later. To make a plugin easily accessible for the Moodle community through the official Moodle site it has to pass the plugin validation procedure, as described on the official Moodle page about plugin validation [4].

A. Mandatory files for activity plugins

The core files which must be included in every plugin directory are the following [3]:
db/install.xml: This is an XML structured file which describes all the tables that the plugin will create and use. This file is located in the subdirectory named "db" in the plugins directory. It must have an element of type XMLDB. Child of this entity is the TABLES element, which can have more than one child of type TABLE. The table itself can have more elements as children. Examples of the usual children elements are the KEYS, FIELDS and INDEXES. Each of them consists only of elements of their singular form (ex. KEY in KEYS).
The TABLES element is used as a container of all TABLE elements. The TABLE elements define which tables will be created during the installation in the Moodle's database, regardless whether is MySQL or Postgres database. There are naming conventions for names of the tables so any inconsistent and duplicate names of tables are avoided. The name of each table should begin with the name of the plugin followed by underscore concatenated with the most suitable table name. The name is defined as the content of the NAME attribute of each TABLE element. The other three attributes are COMMENT, NEXT and PREVIOUS. The COMMENT attribute is self-explanatory. The NEXT attribute indicates which table should be added next to the database after the current table. This next statement should match the next tables NAME attribute in the install.xml. Therefore, only in the last table this attribute is omitted. Symmetrically there is

©2013 Faculty of Computer Science and Engineering

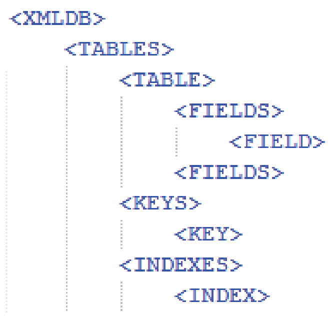
also PREVIOUS attribute which is omitted in the first table and matches the previous table NAME attribute.

The FIELDS, INDEXES and KEYS elements are only containers for the corresponding elements in singular form. The FIELD element can have the following attributes: NAME, TYPE, LENGTH, NOTNULL, UNSIGNED, SEQUENCE, PREVIOUS and NEXT. The NAME, PREVIOUS and NEXT have the same functionality as mentioned before for the TABLE elements. The KEY element has an FIELD attribute, which must be a match to a NAME of FIELD element. Additionally, if the TYPE is "primary", then the match must be from the same table, and if not, REFFIELDS and REFTABLE must not be omitted. REFFIELDS and REFTABLE correspond to a referenced table and a field in it.

The indexes must have a NAME, a field that is being indexed in FIELDS, and UNIQUE attribute that can have "true" or "false" value.

The hierarchy of the previous mentioned elements for database description is shown in Fig. 1. The files containing the code which will be executed when a user clicks an action are described in the following paragraphs.

view.php: The view.php file, as indicated by the extension, is a php code that is executed first when someone clicks on an instance of the activity module. Since the header and footer are always present, and the presence of the side blocks is configurable, the view.php is actually used to generate the viewable surface for the activity itself. Because in the vast majority of the activities, every role has a need of different view and different content, the division should be defined in this file.



```
<XMLDB>
    <TABLES>
        <TABLE>
            <FIELDS>
                <FIELD>
            <FIELDS>
        <KEYS>
            <KEY>
        <INDEXES>
            <INDEX>
```

Figure 1: Structure of the install.xml file

mod_form.php: Similarly as view.php, mod_form.php is a view generating php file that uses different trigger. The mod_form is used in order to create an instance of the plugin, showing configuration input settings for the instance that is about to be created. The edit option of these settings is visible when a user with sufficient privileges edits the instance.

lib.php: The code of all defined functions for for the module is located in this file.The module specific method names begin with the name of the module in order to avoid inter-modular conflicts. If and when the lib.php becomes too bloated then locallib.php can be used for code splitting.

version.php: Version php is the file that keeps track of the module state. It has the fields version, requires, cron, component, maturity, release, and dependencies. The value of version and requires is one integer. The first 4 digits of the integer represent the year of the module last release date, the next two the mouth, the two after that the day and the last two digits represent the ordinal number of the released date. The first corresponds with the module last release date and the dependencies corresponds to minimal Moodle version that the plugin needs in order to run.

index.php: The index.php code is executed with a trigger, just like view and mod_form. Index.php is executed when list of all instances of the desired module is requested.

db/upgrade.php: The upgrade.php is used alongside version php, in order to keep track of the modules state. It is automatically executed when upgrading from an older version.

db/access.php: the access.php is the file that is used to handle the capabilities of different roles on different files. The capability options are:

- CAP_INHERIT
- CAP_ALLOW
- CAP_PREVENT
- CAP_PROHIBIT

The cap_inherit field is dependant of the context. If none is stated then the default is CAP_INHERIT. CAP_PROHIBIT means that the prevention cannot be overridden.

B. *Installation of an activity plugin*

The installation of a new plugin is done by the administrator by invoking the installation procedure. When the procedure is started, the database tables specified in the install.xml are processed and added in the Moodle database. After that the new activity plugin is added in the list of activities and is ready for use. If the installation fails, an error message is displayed to the administrator.

Every plugin has a version number, so when a new version of the plugin is released, the plugin can be upgraded by comparing the existing version number. All the activity modules reside in the /mod directory in the standard Moodle installation. So the installation procedure is initiated by placing the activity plugin files in the mod folder.

C. *Multi-language support*

The Moodle core comes with multi-language support, currently for 86 languages. All of the proprietary software, like plugins, can be translated in one or more languages. When a new plugin is created all of the displayed strings should be stored in a separate file, each identified by a unique key-string. Adding support to another language is easy, all that has to be done is to translate all of the strings in the appropriate language.

III. COB Tool: Implementation of the Activity Plugin

As presented in [5] and [6], we have created a Moodle activity plugin, called Ontology, a tool for building ontology on a given domain. The module was gradually built, adding new features and improvements with every new version. The development of the plugin took year and a half and the plugin is actively maintained.

Our plugin consists of 69 php files and they are categorized in the following way:

- **Help files**: views for explanation of the used terms
- **Insert files**: views for adding new insertions in the Ontology
- **Edit files**: views for editing an existing insertions in the Ontology
- **Evaluation files**: teacher views for evaluation of the insertions in the Ontology
- **Adding Insertions files**: teacher views for adding the accepted insertions in the Ontology
- **Mandatory files**: view.php, index.php, lib.php, version.php, access.php, update.php

The plugin extends the database with 13 additional tables. The entity-relationship model is depicted on Fig. 3 for describing the additional database tables that are used for storing of our module data. 11 tables are used for insertion storing, one for storing the student ranks, and one for every sub-activity in the ontology, because one-domain specific ontology can be built through several sub-activities.

In Fig. 2, a student's view of making a new insertion is shown.

When a stable version of the tool was built, which supported the specified functionalities, the testing and quality assurance phase was executed in parallel with a group of 15 students. Following the specification of our model, the students were divided into 2 groups and participated in a 4 (5) sub-activities, building an ontology on the domain of Object-Oriented Programming or Compilers. One of their tasks was, using this tool and making insertions, to find and report issues and faults. At the beginning not all of them had previous knowledge in the domain of the ontology building, so a lecture was held explaining the crucial terms. Thanks to this group an interesting improvements were made.

The layout of the views was slightly changed so the number of required clicks for browsing and making new insertions was reduced to a minimum. Additionally, a help views were added containing the explanation of the crucial terms used for Ontology Creation.

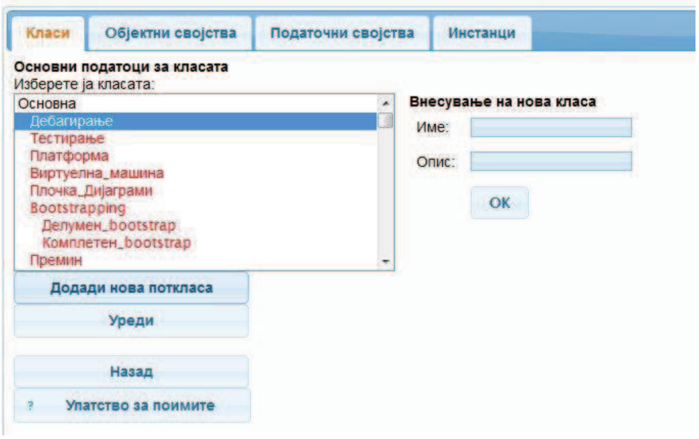


Figure 2: Student's view of making a new insertion

The main question and issue that was considered during the testing was the writing of the ontology restrictions. Some of the students were making mistakes while writing the restrictions, and other even avoided writing them. Because of that we developed a predictive push-down automaton so users only by clicking and choosing will write the restrictions. We chose the Manchester Owl Syntax [7] as the syntax for writing restrictions, because of the popularity and usage in other ontology building tools. So the users can write the restrictions with expressions using predictive push-down automaton.

In the realization we have additionally used technologies like Ajax, for better user experience. As far as installation is concerned, this tool needs to be installed on server with an existing Moodle system.

IV. Installation and Usage of the COB Tool

The installation of the tool is the same as installing any other module for Moodle: the administrator should copy the contents of this module in the appropriate mod directory, where all other modules are located and then by following the standard procedure would install the module. We have tested this plugin on different versions on Moodle, starting from 1.9 to 2.1. This plugin is installed on the official Moodle site in our university and is actively used since 2011 in the course of Compilers.

In Table 1, according to the Tool Guide by Joyce Seitzinger [8], several characteristics are chosen, and each activity module included in the standard Moodle installation is classified as good, bad or neither for that characteristic. All the marks in the table are as in the original table, with one row added - the marks for our ontology building tool, for which we tried to give marks as objective as possible and the reasons are outlined in the following text.

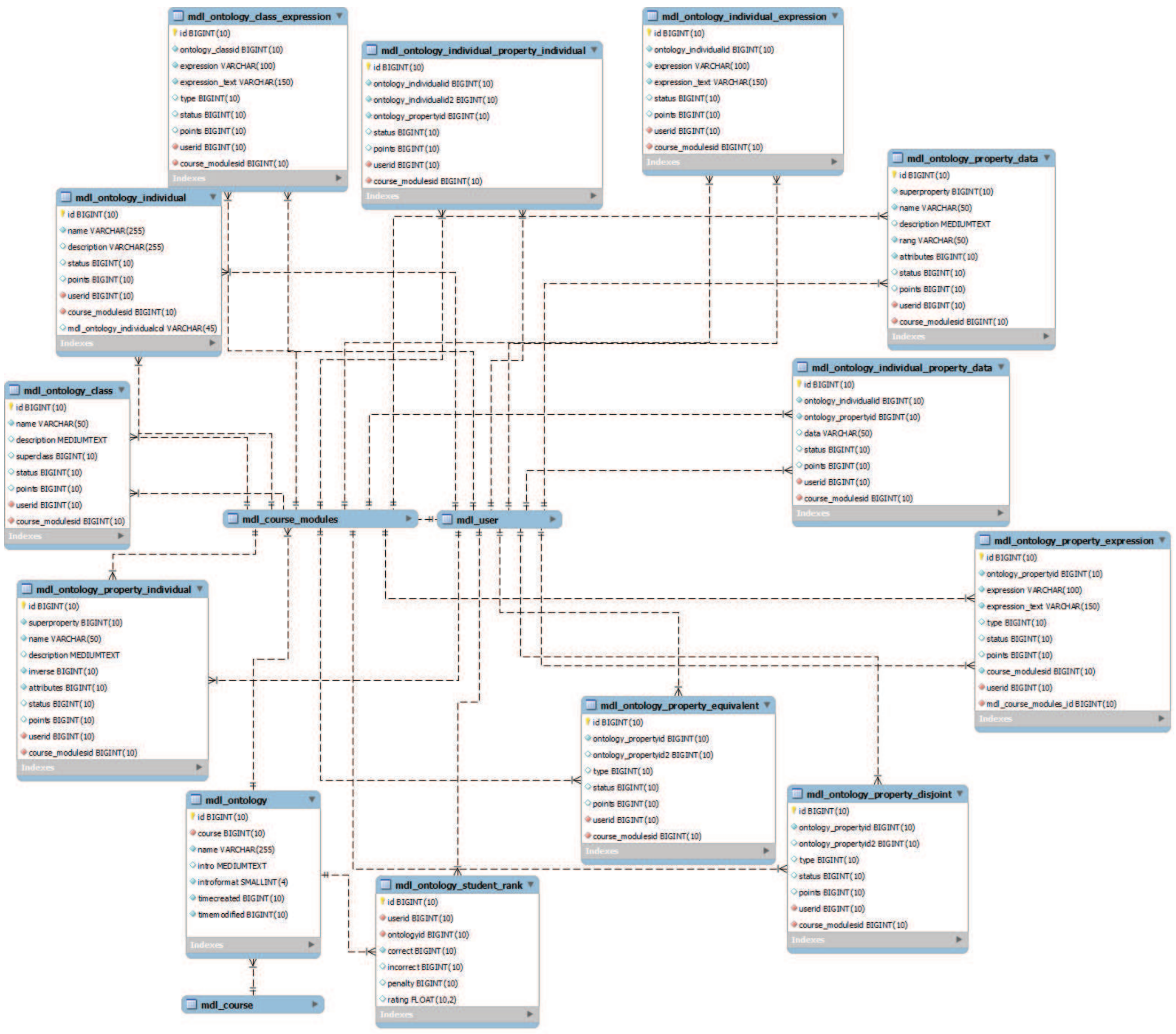


Figure 3: E-R Model of the ontology plugin

As far as ease of usage is concerned we have graded our tool as neutral, because it is robust and immense tool that invoked the need for a basic training of the students. The tool is intuitive, with vast choices of functionalities.

Because ontology creation is composed of several sub-activities, the accepted insertions from the previous week can give much information to the students about what they have missed or what did they do wrong and based on that continue making new insertions. This gives them the ability to learn from the previous sub-activities and a chance to keep up with the others. This is the reason why we marked the ***information transfer*** characteristic as good.

For every sub-activity the teacher evaluates the students' insertions. The knowledge and quantity of student's insertions generates the grade for that student. Regardless of the grades, a separate measure is also generated - the students rating mostly based on the quality of the student input [9]. Both measures assess the student knowledge and improvement. The results presented in [5] confirm the validity of the grade, as a measure highly correlated to the final grade of the student for the course in question. Because of that we gave a good mark for the ***Assess learning*** characteristic.

For information transfer, we have implemented a feedback option for every activity; hence the students can always know if they made a mistake so they will not do it again. This comprises mark between good and neutral on the ***Communication and Interaction*** characteristic.

For the ***Co-creating of the content***, our tool receives good mark, having in mind the essence of the process and the purpose of the tool. Everything valuable inserted by some student during the assignments, will be part of the final knowledge database (in the same or similar form). The tool even allows inserting partial knowledge into the ontology, useful when a student makes partially good insertion.

The last characteristic (***Bloom's***) evaluates the ability of the tool to assess the different levels of the student knowledge according to Blooms taxonomy [10] (Remember, Understand, Apply, Analyze, Evaluate and Create). Taking into consideration previously given grades 'good' for the activities like Glossary and Database, we can award our activity with 'good' also, as it is wider activity compared to the mentioned ones.

Table 1: Summarized version of the Moodle Guide by Joyce Seitzinger

	Ease of use	Information Transfer	Assess learning	Communication & interaction	Co-create content	Bloom's
Add Resource (Upload)	✓	✓	☐	✗	☐	✗
Add Resource (link)	✓	✓	✗	☐	☐	☐
News Forum	✓	✓	✗	☐	☐	✗
Discussion Forum	✓	☐	☐	✓	✓	✓
Wiki	☐	✓	☐	☐	✓	✓
Glossary	✓	✓	☐	☐	☐	✓
Quiz	✗	✗	✓	✗	✗	☐
Lesson	☐	✓	✓	✗	✗	☐
Assignment	✓	✗	✓	✗	☐	☐
Database	✗	☐	☐	☐	✓	✓
Ontology	☐	✓	✓	✓	✓	✓

V. Conclusion

In chapter II we described the main concepts for creating a new plugin for Moodle. In chapter III we gave a detailed description of our tool which can be used for reproduction and advice for development of similar tool. We have showed that the installation is simple and the tool can be used in Moodle courses around the world. The results in Table 1 further confirm that conclusion.

Acknowledgement

The research presented in this paper is partly supported by the Faculty of Computer Science and Engineering in Skopje.

References

[1] (2013) Moodle. [Online]. Available: http://moodle.com/

[2] Cankaya, S., Izmirli, S., "Activity Module Development for Moodle: A Sample Activity Module, EduGame", Proceedings of the International Conference on Computational and Information Sciences 2009

[3] (2013) Activity Modules in Moodle [Online]. Available: http://docs.moodle.org/dev/Activity_modules

[4] (2013) Process of plugin validation in Moodle [Online]. Available: http://docs.moodle.org/dev/Plugin_validation

[5] M. Jovanov, M. Gusev, and D. Martinovikj, "A new model of on-line collaborative activity for building ontology in e-learning," in Proc. of the IEEE Region 8 Conference EUROCON 2013, 2013, pp. 25–31

[6] M. Jovanov, M. Gusev, and D. Martinovikj, "COB: Online educational tool for collaborative ontology building" in Proc. of the 36th IEEE Mipro convention: Conference Computers in Education, MIPRO 2013/CE), 2013, pp. 934–939.

[7] (2013) Manchester owl 2.0 syntax. [Online]. Available: http://www.w3.org/TR/owl2-manchester-syntax/

[8] (2013) Moodle Tool Guide by Joyce Seitzinger [Online]. Available: http://www.cats-pyjamas.net/wp-content/uploads/2010/05/MoodleToolGuideforTeachers_May2010_JS.pdf

[9] M. Jovanov and M. Gusev, "Grading and rating of students in new on-line collaborative activity," in Proc. of 4th International Conference of Education, Research and Innovations (ICERI2011), 2011, pp. 1976–1981

[10] L. W. Anderson and L. A. Sosniak, Bloom's Taxonomy. National Society for the Study of, 1994