Moodle Plugins for Highly Efficient Programming Courses

Sun Zhigang, Su Xiaohong

School of Computer Science and Technology, Harbin Institute of Technology, China, {sun,sxh}@hit.edu.cn

Zhu Ning, Cheng Yanyu

Undergraduate College, Harbin Institute of Technology, China, {zhun,chyy}@hit.edu.cn

Abstract

This paper introduces three Moodle plugins, Online Judge Assignment, Moss Plagiarism and Github Assignment, which are designed for courses involving programming, such as CS1, Data Structure, Algorisms, Compilers and Operating Systems. Online Judge extends the advanced uploading of files assignment to instantly compile and run submissions against predefined test cases. According to the number and weights of passing test cases, it can grade submissions automatically. Students who do not reach the maximum grade can improve and upload their programs again and again for higher grades. It supports more than 40 kinds of programming languages. Moss is a plagiarism plugin. It compares all submissions in the same/related assignments, lists the most similar pairs and let teachers to confirm who are cheating. Its underlying engine is designed to parse source code but this plugin extends this ability to support pdf, doc, docx and odt documents. Github, as an assignment type plugin, can synchronize commits from git repositories in github.com and store them as submissions. Therefore, teachers can track student projects in details and grade them directly in Moodle. Moreover, for team-based projects, each group uses one shared repo and the plugin can statistic workload percentages of every group member. It has been proved that these plugins can significantly reduce teachers' labour and improve students' enthusiasm, make programming courses more efficient. All of them are open source and free. You are welcome to download them from http://git.io/hmdl .

Keywords

Moodle, online judge, Moss, plagiarism, Github, team-based projects

Introduction

Moodle is a great universal platform with many build-in tools to construct online courses. It is suitable for nearly all kinds of courses for any person, any institution and any discipline. But different courses usually have different requirements. In department of computer science or software engineering, most of the courses are related with programming and teachers wish Moodle can provide these features:

First, grade assignment submissions by online judge. Online judge systems can compile and execute submissions, and test them with pre-constructed data to judge correctness. This method is widely used in many programming contests, such as ACM International Collegiate Programming Contest (ACM-ICPC, 2012) and TopCoder(Inc., 2012). There are also many online judge websites (Wikipedia, 2012) provide huge problem sets. They are so attractive that many students take initiative to practice programming skill on them. Naturally, online judge is applied in courses and has been proved to be successful (Kurnia et al., 2001)(Cheang, Kurnia, Lim, & Oon, 2003).

Second, detect plagiarism behaviours in assignments. Most of our courses have more than 100 students. Some teachers even need to instruct more than 400 students in one course. It is impossible for teachers to find plagiarism manually. Turnitin (August, 2010) has been integrated with Moodle well but it is expensive and not suitable for source code. There are several tools (Hage, Rademaker, & Vugt, 2010) designed for code, and Moss (Aiken, 2012)(Schleimer, Wilkerson, & Aiken, 2003) is the most popular one.

Third, track the procedure of student projects. Some courses require students to do projects in groups within 4-16 weeks. Usually, 1-3 checkpoints are set at which students make presentations and teachers give evaluations. There are two problems are hard to solve:

1 Teachers don't know what happened between checkpoints. Is there any procrastination?
2 Teachers don't know whether the division of labour told by students is true. Is there anybody in the group who did only a little or even nothing?

Github.com makes it possible to track the whole history of projects. But github is not designed for courses management. Tutors, students, groups and grades are better to be managed in Moodle and Moodle should synchronize students' activities with github and present them to teachers in more convenient interfaces.

This paper introduces Online Judge Assignment, Moss Plagiarism and Github Assignment plugin for Moodle. We have been using them in C/C++ Programming Language (CS1) course since 2006 and in Operating Systems course since 2008. There are also many users from different countries. Feedbacks show that they can give great convenience, fairness and efficiency to both teaching and learning.

Online Judge

Moodle Online Judge contains two plugins:

1 A local plugin implements an online judge library called judgelib. It contains three modules: judge engines, judge daemon and library API.
2 An assignment type plugin provides UI and invokes judgelib to judge submissions.

Figure 1 is the architecture of Moodle Online Judge. The client (we implement only one client - online judge assignment type) submits judge tasks to judgelib through API. Each task has source code, test data, resource limits and which judge engine should be used. Tasks are written into database and marked as ONLINEJUDGE_STATUS_PENDING. The judge daemon always resides in memory and regularly check database for pending tasks. When one pending task is found, it is marked as ONLINEJUDGE_STATUS_JUDGING and sent to specified judge engines. The judge engine will take some time to judge and return the result. Then the task's status is set as the result. Finally, an onlinejudge_task_judged event is triggered so that the client can know the result. The judge daemon is tricky. It can work in parallel mode. So it is safe to concurrently run several judge daemons for higher throughout.

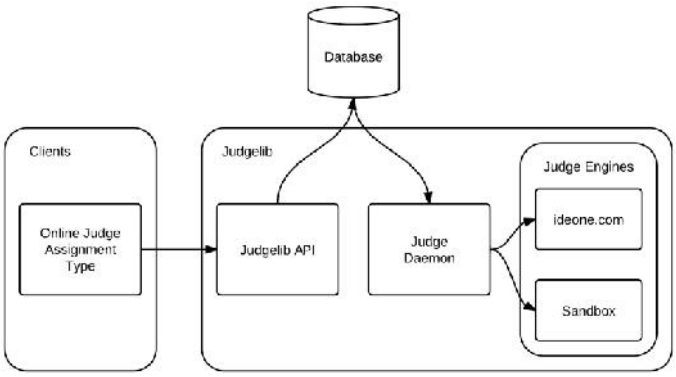


Figure 1: Online judge architecture

The assignment type is extended from Moodle's official advanced uploading of file assignment type, and inherits most of the standard features such as uploading file, due time limit, grading manually and feedbacks. In its setting page, teachers can choose programming languages, set memory and CPU time limits. Its settings block can direct teachers to rejudge all submissions or configure test cases. Each assignment may have several test cases which consist of input data, output data, grade and feedback for error. Submissions are judged separately against each case. If one case is passed, its grade is added up to the final grade. Otherwise, the feedback is shown to students as hint (see figure 2).

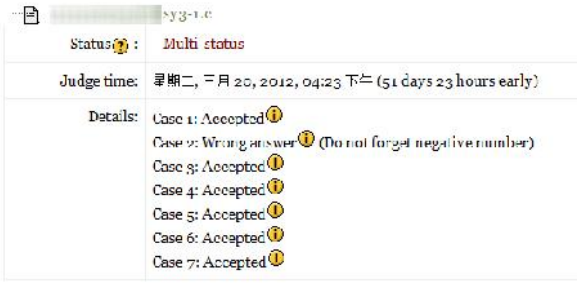


Figure 2: Online judge result of a submission

Judge engines are designed as subplugins. Therefore, it is easy to integrate more engines. Now there are two judge engines: sandbox and ideone. Sandbox uses libsandbox(Liu, 2012) as a restricted environment to execute C/C++ submissions on Linux. It can monitor the behaviour of target programs in runtime. When program's execution exceeds the specific time or memory limit, or it calls a disallowed system call, sandbox will terminate it immediately. Ideone engine supports more than 40 programming languages (e.g. Java, Perl, Python, C#) by calling the web services of ideone.com. It is safer but has longer latency than sandbox. Moreover, ideone allows only 2000 submissions/month for one free account. But they are considering providing big quota to users of Moodle Online Judge.

Moss Plagiarism

The official website of Moss (Aiken, 2012) says:

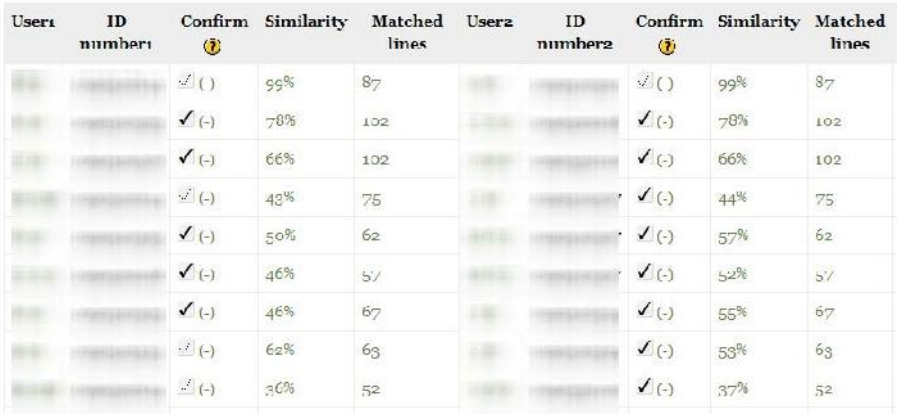
> Moss (for a Measure Of Software Similarity) is an automatic system for determining the similarity of programs. To date, the main application of Moss has been in detecting plagiarism in programming classes. Since its development in 1994, Moss has been very effective in this role.
> The algorithm behind moss is a significant improvement over other cheating detection algorithms (at least, over those known to us).

It supports dozens of program languages: C, C++, Java, C#, Python, Visual Basic, JavaScript, FORTRAN, ML, Haskell, Lisp, Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, a8086 assembly, MIPS assembly and HCL2.

Moss works as a service with a client program written in Perl. This client can (a) upload files; (b) specify in which programming language the uploaded files are; (c) specify a set of files as the skeleton code which will not be considered as plagiarism; (d) set the maximum number of times a given passage may appear before it is ignored. On succeed, it will print an URL and exit. Accessing the URL in any web browser can get the plagiarism detection result pages, which show the possible cheating codes side-by-side and highlight the similar code segments.

Moss had been used in our courses for 2 years before Moodle was applied. Therefore, the requirement of integrating them is straightforward. The first version of Moss plugin is implemented as a block. After Moodle 2 introduced the Plagiarism API("Plagiarism API," 2012), it was migrated and upgraded to Moodle 2. The new version not only inherits all the abilities of Moodle Plagiarism API and Moss, but also introduces four significant features.

The first feature is confirmation. The results of Moss are only reference. Teachers must check the results manually to confirm who is cheating. Moss plugin adds a confirmation status on each submission. Teachers can look up a sorted list of potential plagiarism (figure 3) and click to see the side-by-side comparison. If they are sure that there must be someone involving cheating, they can press the confirm button and related students will be notified through Moodle message system. They can also easily give a punish grade to confirmed students in the same page. And of course, there are unconfirm buttons if they changed their minds (e.g. students give a reasonable explanation. In figure 3, the first pair of person with 99% similarity is unconfirmed because one of them occasionally uploaded another person's file. He presented his own file and can explain every detail in it to prove it belongs to him.)



User1	ID number1	Confirm	Similarity	Matched lines	User2	ID number2	Confirm	Similarity	Matched lines
[redacted]	[redacted]	✓ (-)	99%	87	[redacted]	[redacted]	✓ (-)	99%	87
[redacted]	[redacted]	✓ (-)	78%	102	[redacted]	[redacted]	✓ (-)	78%	102
[redacted]	[redacted]	✓ (-)	66%	102	[redacted]	[redacted]	✓ (-)	66%	102
[redacted]	[redacted]	✓ (-)	43%	75	[redacted]	[redacted]	✓ (-)	44%	75
[redacted]	[redacted]	✓ (-)	50%	62	[redacted]	[redacted]	✓ (-)	57%	62
[redacted]	[redacted]	✓ (-)	46%	57	[redacted]	[redacted]	✓ (-)	52%	57
[redacted]	[redacted]	✓ (-)	46%	67	[redacted]	[redacted]	✓ (-)	55%	67
[redacted]	[redacted]	✓ (-)	62%	63	[redacted]	[redacted]	✓ (-)	58%	63
[redacted]	[redacted]	✓ (-)	36%	52	[redacted]	[redacted]	✓ (-)	37%	52

Figure 3: Moss Plagiarism plugin results page

The second feature is support of pdf, doc, docx, rtf and odt documents. Although never mentioned in homepage, Moss's client supports plain text. So Moss Plagiarism plugin parses out plain texts from common document formats and let Moss do the rest.

The third feature is history. Moss engine never stores files for long. And teachers usually reset their courses at beginning of new semesters to clean all submissions of unenrolled students. Moss plugin can keep those files and will include them in future measuring. Moreover, assignments in different courses with the same moss tag (set by teachers) can also be measured together.

The fourth feature is multi-configures. Since the advanced uploading of file assignment allows multi-files submissions, the plugin gives the ability to use different configure for different files. For example, an assignment requires three submission files. One is report file and the others are Java source code files. Teachers can set *.doc, *.pdf and *.docx to be measured as plain text, and *.java to be measured as Java source code. The result page can show the results of the two configure together or separately.

Unlike turnitin("Plagiarism Prevention Turnitin," 2012) and crot (Butakov, 2012) plugin who show result immediately after uploading files, Moss does detection only at specified time (default to the due dates of assignments) because Moss requires that all files must be submitted together. Another weakness is that Moss does not do web search. Only files submitted to Moodle are measured. But this is not a big problem because it is common that more than one student copy the same origin. They cannot hide from Moss.

Github Assignment

Github.com is not only an amazing website for developers, but also a helpful tool for teachers and students. Some supervisors require their students to host graduation projects in github. So that supervisors can watch the projects to know the latest updates and commented on commits to give advices. For supervisor, everything is easier and more trustable than before. For students, the public projects can be impressive show to their future employers. Therefore, github should also be helpful for course projects.

However, github is not designed for courses. Github assignment type plugin fills the gap between courses and github. It allows teachers to track projects without leaving Moodle. The features are:

1 Collect students' github usernames and repository URLs.
2 If the assignment is in group mode, group members share the same repository. Otherwise, everyone has his/her own repository.
3 Synchronize all commit histories of all repositories and cache the logs.
4 Statistic workload percentages of group members.
5 List all repositories by optional sort order (name, last update date).
6 Direct links to github pages those teachers & students may concern.
7 Inherits standard functions of assignment type, such as grading.

Figure 4 shows a project named taskmanager-for-linux-chengjisihan. Two group members were contributing to it.

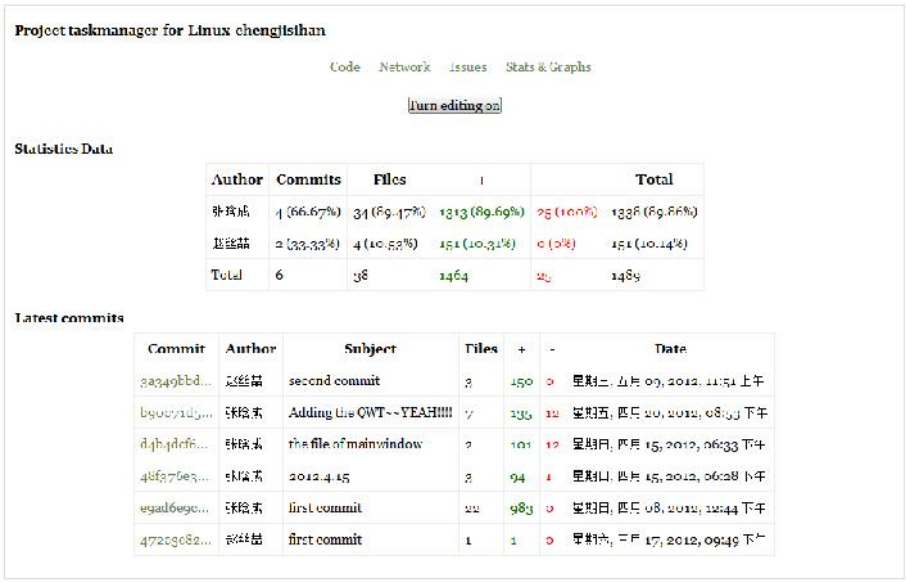


Figure 4: Project status page of github assignment type plugin

This plugin has a simple architecture. Besides the user interface, it provodes a command line synchronization script which invokes git command and Github API (Duplessis, 2012) to fetch data. Git working directories are cached in Moodle temporary directory and logs are stored in database. The CLI script can be run through crontab in any period.

Now, the plugin only supports public repository. Github has an educational plan which gives teachers free accounts to create as many private repositories as need for courses. Since perhaps some teachers or students want the code to be private, the plugin will support private repository in the next version.

Case Study

Online Judge, Moss Plagiarism and Github Assignment plugin have given significant contributions to our courses. They can be used together or separately. Some teachers said they change the way of teaching and learning in programming courses. Moreover, most students are also happy to accept them.

Online Judge

In 2006, Online Judge plugin was first applied in C Programming courses for Computer Science and Software Engineering students. Now it has been widely used in other majors and courses and also other institutions and countries. Some users even contributed Polish and Portuguese translations.

In our university, every semester, there are more than 2500 students will enrol C Programming Language courses which are instructed by no more than 10 teachers. Some teachers must instruct more than 400 students concurrently. But school only supports 1-4 teaching assistants for each teacher. In the days without Online Judge, there were less assignments and only part of them could be graded because teachers could not afford the workloads. But recent years, it is normal that a student who gets a pass have to code more than 50 programs. And all the programs are graded well. Although teachers must spend time to design assignments and test cases, the assignments can be easily shared among teachers and reused in the future. Therefore, teachers are much happier than ever. Somebody even added “infinite” online judge assignments and announced that only the student who conquers the most number of the assignments can get 100 points and so on.

Most of students also enjoy the procedure of conquering online judge assignments. They treat it as an adventure of online game. If the grades are not satisfying, they have the chance to retry. As their codes get better, they get more skills, experiences and higher grades. At last, the maximum grades will draws upon them and satisfy their

achievability. In the course forums, we can easily find posts about their enjoyments. Somebody even required more assignments. In last year, the average submitting time per student for one assignment was 6.97. The most persistent student submitted 816 times for one assignment.

One professor began to use Online Judge since 2010 spring. In the four semesters before that, the average grades of her courses had been 75.3, 67.2, 67.8 and 70.3. In the following semesters after using Online Judge, the average grades were 89.6, 86.6 and 83.7. She said that she decreased difficulty only in 2010 spring. After that, both difficulty and quantity were increased much. The numbers show the significant improvement on quality of teaching and learning.

Moss Plagiarism

Plagiarism is a big problem for nearly all courses. Since the dead of paper-based homework, copying has been zero cost. The sources of plagiarism include classmates, upperclassmen and Internet. Some graduate students estimated that there are only no more than 10% undergraduate students have never do plagiarism before graduation.

In our university, Moss Plagiarism plugin is mainly used in C/C++ Programming and Operating Systems courses. All C submissions are pure original source code so the default setting works well. The assignments of Operating Systems require students to modify Linux kernel and write reports. Therefore, in submissions, there are test reports and source code files which contain Linux code base and students' patches. So each assignment has two configures. (a) *.c files are measured as C language and using Linux code base as base files so that they are ignored in results. (b) *.pdf, *.doc and *.docx files are measured as plain text.

Moss gives out up to 250 potential plagiarism pairs no matter what the percentage and how many lines are detected similar. Only part of them can be real plagiarism. So the confirmation step is the most sensitive. Teachers must carefully check the pairs in the top of the result lists. The default sort order of Moss is correct enough so that it is usually safe to ignore all pairs following the first pair that teachers believe it is not a cheating. As reports, only a very little number of students could successfully challenge the confirmations.

Many students said that Moss gave them so much pressure. Since only a few courses are using this plugin, teachers of these courses are common to get complain during the first halftime of courses. However, after the halftime, when students feel that they do learn something under the pressure, they begin to love and support teachers. In 2010 autumn semester, at the end of OS course, a survey asked "Do you support to use Moss Plagiarism in the following courses?" There are 91.7% students chose "Yes".

Github Assignment

Github Assignment plugin is still under development and was first used in March, 2012 in the course Software Design and Development Practice III. SDDP is a series of courses. They require the 200+ undergraduate students to design and develop a software project in 2-5 member-sized groups in 16 weeks. A survey shows that, In SDDP I & II, only 19.27% groups can pay attentions on projects through all time, and 49.4% groups started their jobs even in the last 1-4 weeks. Moreover, many students complained that some group members did not contribute to the projects but they can get the same grade as others. SDDP III is trying to change something in Moodle way.

First students and tutors (teachers or graduate students) were added into Moodle groups. Then one forum called "Weekly reports" and one github assignment were created in separated group mode. The forum's maximum grade was set to 20, and the assignment was set to 80.

Students must weekly post in the forum to report what they have done in the past week and what were the problems they could not solve. Tutors should read posts, join discussion, and grade posts upon the quality of works. Therefore, students keep doing something; tutors know the progress and problems are solved quickly.

In the Github assignment, students must setup github repositories and account in the first week, then commit and push code to github in following weeks. Git logs were synchronized and shown in the assignment. Tutors could read logs, click the github links and enjoy the github features such as inline comment and statistic graphs. Instructor of the course concerned about all groups. So he could usually check the repository list sorted by last commit date to learn where the students had gone to.

In the last week, students made presentations and demos. The instructor listened, asked questions and graded directly in the github assignment. He also checked the git logs and statistics to know who exactly did what.

For most students, it was the first time they met Git & Github. There was no training about them except some links to ProGit and Github documents. Students must learn by their selves. At the end of this course, a survey asked "How about git?" 36.14 % students chose "It's helpful", 27.71 % chose "More troubles than solutions", and 36.14% chose "I don't care". So there should be some training to cover troubles in the next usage. Another survey question asked "Which trouble is the most often to make you say the F-word?" The top two troubles are "Cooperation problem" and "Push rejected". So the training should pay more attentions on these problems.

Conclusion

All these plugins are published in GPL license and free to download at http://git.io/hmdl . Since the first release, we have got many feedbacks, suggest and patches from many instructors, administrators and researchers around the world. Their experience and ours proved that these plugins can make courses involving programming more efficient and effective.

Moodle 2.3 changed the assignment module. Online Judge and Github will be migrated to it. And there are also many improvements under plan. We'd like to keep upgrading them and make them free.

References

ACM-ICPC. (2012). The ACM-ICPC International Collegiate Programming Contest. Retrieved from http://icpc.baylor.edu/

Aiken, A. (2012). Moss: A System for Detecting Software Plagiarism. Retrieved from http://theory.stanford.edu/~aiken/moss/

August, U. (2010). Turnitin Moodle Direct Integration Instructor User Manual. Integration The Vlsi Journal. iParadigms, LLC.

Butakov, S. (2012). Crot Plagiarism Checker. Retrieved from http://moodle.org/plugins/view.php?plugin=plagiarism_crot

Cheang, B., Kurnia, A., Lim, A., & Oon, W.-C. (2003). On automated grading of programming assignments in an academic institution. Computers & Education, 41(2), 121-131.

Duplessis, T. (2012). PHP GitHub API. Retrieved May 12, 2012, from https://github.com/ornicar/php-github-api

Hage, J., Rademaker, P., & Vugt, N. e V. (2010). A comparison of plagiarism detection tools. Sciences-New York. Retrieved December 16, 2010, from http://www.cs.uu.nl/research/techreps/repo/CS-2010/2010- 015.pdf

Inc., T. (2012). TopCoder. Retrieved July 4, 2012, from http://www.topcoder.com/

Kurnia, A., Lim, A., & Cheang, B. (2001). Online Judge. Computers & Education, 36(4), 299-315. Retrieved from http://www.sciencedirect.com/science/article/B6VCJ-42SPV9S-2/2/cc5dbbe6b0537f1043d2d08086c6238c

Liu, Y. (2012). Sandbox Libraries. Retrieved from http://sourceforge.net/projects/libsandbox/ Plagiarism API. (2012). . Retrieved from http://docs.moodle.org/dev/Plagiarism_API Plagiarism Prevention Turnitin. (2012). . Retrieved May 14, 2012, from http://docs.moodle.org/22/en/Plagiarism_Prevention_Turnitin

Schleimer, S., Wilkerson, D. S., & Aiken, A. (2003). Winnowing: local algorithms for document fingerprinting. Proceedings of the 2003 ACM SIGMOD international conference on Management of data (pp. 76-85). San Diego, California: ACM Press.

Wikipedia. (2012). Online judge. Wikipedia. Retrieved July 4, 2012, from http://en.wikipedia.org/wiki/Online_judge

Acknowledgement

Our works are funded by School of Computer Science and Technology, and School of Software of Harbin Institute of Technology. Most code of Github Assignment was written by Fu Jianyu. Shi Xing and Liu Qiqing contributed much to Online Judge. Li Zhijun, Che Wanxiang, Guo Ping, Zhao Wei, Zhang Wei and Li Hanjing's experiences give contribution to this paper.