外文文献翻译

**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

**摘要**

本文介绍了三个Moodle插件，Online Judge Assignment、Moss Plagiarism和Github Assignment，它们是为CS1、数据结构、算法、编译器和操作系统等涉及编程的课程设计的。 Online Judge 扩展了文件分配的高级上传，以针对预定义的测试用例即时编译和运行提交。根据通过测试用例的数量和权重，它可以自动对提交的提交进行评分。没有达到最高等级的学生可以一次又一次地改进和上传他们的程序以获得更高的成绩。它支持40多种编程语言。 Moss 是一个抄袭插件。它比较相同/相关作业中的所有提交，列出最相似的配对，并让教师确认谁作弊。它的底层引擎旨在解析源代码，但该插件扩展了这种能力以支持 pdf、doc、docx 和 odt 文档。 Github 作为一个赋值类型的插件，可以同步来自 github.com 中 git 仓库的提交，并将它们存储为提交。因此，教师可以详细跟踪学生项目并直接在 Moodle 中对其进行评分。此外，对于基于团队的项目，每个组使用一个共享 repo，插件可以统计每个组成员的工作量百分比。实践证明，这些插件可以显着减少教师的劳动，提高学生的学习积极性，让编程课程更有效率。所有这些都是开源和免费的。欢迎您从 http://git.io/hmdl 下载它们。

**关键词**

Moodle、在线评委、Moss、抄袭、Github、团队项目

**介绍**

Moodle 是一个很棒的通用平台，具有许多构建在线课程的内置工具。它适用于任何人、任何机构和任何学科的几乎所有类型的课程。但是不同的课程通常有不同的要求。在计算机科学或软件工程系，大部分课程都与编程有关，老师们希望 Moodle 能够提供以下功能：

首先，在线评委提交成绩作业。在线评审系统可以编译和执行提交，并使用预先构建的数据对其进行测试以判断正确性。这种方法在很多编程竞赛中得到广泛应用，如ACM国际大学生编程竞赛（ACM-ICPC，2012）和TopCoder（Inc.，2012）。还有许多线上评测网站（维基百科，2012 年）提供了大量的问题集。它们是如此有吸引力，以至于许多学生主动在它们身上练习编程技巧。自然地，线上评测应用于课程并已被证明是成功的（Kurnia et al., 2001）（Cheang, Kurnia, Lim, &

哦，2003 年）。

其次，检测作业中的抄袭行为。我们的大部分课程都有超过 100 名学生。有的教师甚至需要在一门课上指导400多名学生。教师不可能手动发现抄袭。 Turnitin（2010 年 8 月）已与 Moodle 很好地集成，但价格昂贵且不适合源代码。有多种为代码设计的工具（Hage、Rademaker 和 Vugt，2010 年），Moss（Aiken，2012 年）（Schleimer、Wilkerson 和 Aiken，2003 年）是最受欢迎的一种。

第三，跟踪学生项目的进程。有些课程要求学生在 4-16 周。通常设置1-3个检查点，学生进行展示，教师进行评价。有两个问题很难解决：

1 教师不知道检查点之间发生了什么。 学生会有拖延症吗？

2 老师不知道学生讲的分工是否属实。 小组中是否有人只做很少甚至什么都不做？

Github.com 使跟踪项目的整个历史成为可能。 但是 github 不是为课程管理而设计的。 导师、学生、小组和成绩最好在 Moodle 中进行管理，Moodle 应该将学生的活动与 github 同步，并以更方便的界面呈现给教师。

本文介绍了 Moodle 的 Online Judge Assignment、Moss Plagiarism 和 Github Assignment 插件。 我们从 2006 年开始在 C/C++ 编程语言 (CS1) 课程中使用它们，从 2008 年开始在操作系统课程中使用它们。也有许多来自不同国家的用户。 反馈表明，它们可以为教与学带来极大的便利、公平和效率。

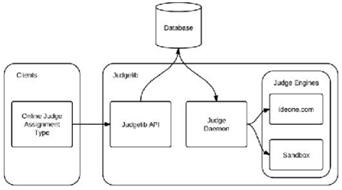
**在线评测(Online Judge)**

Moodle Online Judge 包含两个插件：

1 本地插件实现了一个名为 Judgelib 的在线评测库。它包含三个模块：判断引擎、判断守护进程和库API。

2 分配类型插件提供 UI 并调用 Judgelib 来判断提交。

图 1 是 Moodle Online Judge 的架构。客户端（我们只实现了一种客户端——在线评测分配类型）通过API向judgelib提交评委任务。每个任务都有源代码、测试数据、资源限制以及应该使用哪个评测引擎。任务被写入数据库并标记为 ONLINEJUDGE\_STATUS\_PENDING。评测系统守护进程始终驻留在内存中，并定期检查数据库中是否有待处理的任务。当发现一个待处理任务时，将其标记为 ONLINEJUDGE\_STATUS\_JUDGING 并发送到指定的判断引擎。判断引擎需要一些时间来判断并返回结果。然后将任务的状态设置为结果。最后触发一个 onlinejudge\_task\_judged 事件，让客户端知道结果。评测系统守护进程很棘手。它可以在并行模式下工作。因此同时运行多个判断守护进程是安全的。



**图 1：在线评测架构**

作业类型扩展自Moodle官方高级上传文件作业类型，继承了大部分标准功能，如上传文件、限时、手动评分、反馈等。 在其设置页面，教师可以选择编程语言，设置内存和CPU时间限制。 它的设置块可以指导教师重新判断所有提交或配置测试用例。 每个作业可能有几个测试用例，包括输入数据、输出数据、等级和错误反馈。 提交的材料针对每个案例分别进行评判。 如果一个案例通过，则其成绩累加到最终成绩。 否则，反馈将作为提示显示给学生（见图 2）。



**图2：提交的在线评测结果**

评测引擎被设计为子插件。 因此，很容易集成更多的引擎。 现在有两个评测引擎：sandbox 和 ideone。 Sandbox 使用 libsandbox(Liu, 2012) 作为在 Linux 上执行 C/C++ 提交的受限环境。 它可以在运行时监视目标程序的行为。 当程序的执行超过特定的时间或内存限制，或者调用了不允许的系统调用时，sandbox会立即终止它。 Ideone 引擎通过调用 ideone.com 的 Web 服务支持 40 多种编程语言（例如 Java、Perl、Python、C#）。 它比sandbox更安全，但延迟更长。 此外，ideone 只允许一个免费帐户每月提交 2000 次。 但他们正在考虑为 Moodle Online Judge 的用户提供大量配额。

**Moss 查重**

Moss 的官方网站 (Aiken, 2012) 说：

Moss（用于衡量软件相似性）是一种用于确定程序相似性的自动系统。 迄今为止，Moss 的主要应用是检测编程课程中的抄袭。 自 1994 年开发以来，Moss 在这个角色上一直非常有效。

moss 背后的算法是对其他作弊检测算法（至少，对我们已知的那些）的重大改进。

它支持数十种程序语言：C、C++、Java、C#、Python、Visual Basic、JavaScript、FORTRAN、ML、Haskell、Lisp、Scheme、Pascal、Modula2、Ada、Perl、TCL、Matlab、VHDL、Verilog、Spice、 a8086 汇编、MIPS 汇编和 HCL2。

Moss 作为服务与用 Perl 编写的客户端程序一起工作。该客户端可以 (a) 上传文件； (b) 指定上传文件的编程语言； (c) 指定一组文件作为骨架代码，不会被视为抄袭； (d) 设置给定段落在被忽略之前可能出现的最大次数。成功后，它将打印一个 URL 并退出。在任意网页浏览器中访问该网址都可以得到抄袭检测结果页面，该页面并排显示可能的作弊代码并突出显示相似的代码段。

在应用 Moodle 之前，Moss 已经在我们的课程中使用了 2 年。因此，集成它们的要求很简单。 Moss 插件的第一个版本是作为一个块实现的。 Moodle 2 引入了Plagiarism API（“Plagiarism API”，2012）之后，迁移升级到了Moodle 2。新版本不仅继承了Moodle Plagiarism API 和Moss 的所有能力，还引入了四大显着特性。

第一个功能是确认。 Moss 的结果仅供参考。教师必须手动检查结果以确认谁作弊。 Moss 插件会在每次提交时添加确认状态。教师可以查找潜在抄袭的排序列表（图 3），然后单击查看并排比较。如果他们确定一定有人作弊，可以按下确认按钮，相关学生将通过Moodle消息系统得到通知。他们还可以轻松地在同一页面中对已确认的学生进行惩罚。当然，如果他们改变主意，还有不确认按钮（例如学生给出合理的解释。在图3中，第一对99％相似度的人未确认，因为他们中的一个偶尔上传了另一个人的文件。他展示了自己的文件，并且可以解释其中的每一个细节，以证明它是属于他的。）



图 3：Moss 查重插件结果页面

第二个功能是支持 pdf、doc、docx、rtf 和 odt 文档。尽管在主页中从未提及，但 Moss 的客户端支持纯文本。所以 Moss 查重插件从常见的文档格式中解析出纯文本，让 Moss 完成剩下的工作。

第三个特点是历史文件。 Moss 引擎从不长时间存储文件。老师们通常会在新学期开始时重新设置课程，以清理所有未注册学生的提交。 Moss 插件可以保留这些文件，并将在以后的测量中包含它们。此外，不同课程中具有相同moss标签（由教师设置）的作业也可以一起测量。

第四个特点是多配置。由于文件分配的高级上传允许多文件提交，因此该插件能够为不同的文件使用不同的配置。例如，一个作业需要三个提交文件。一个是报告文件，另一个是Java 源代码文件。教师可以设置\*.doc、\*.pdf和\*.docx为纯文本，\*.java为Java源代码。结果页面可以显示两个配置的结果一起或单独显示。

与 turnitin(“Plagiarism Prevention Turnitin,” 2012) 和 crot (Butakov, 2012) 插件上传文件后立即显示结果不同，Moss 仅在指定时间（默认为作业的截止日期）进行检测，因为 Moss 要求所有文件必须一起提交。另一个弱点是 Moss 不进行网络搜索。仅测量提交给 Moodle 的文件。但这并不是什么大问题，因为一个以上的学生抄袭同一个来源是很常见的。他们无法躲避Moss。

**Github 作业**

Github.com 不仅是开发人员的绝佳网站，也是教师和学生的有用工具。一些导师要求他们的学生在github上托管毕业项目。以便主管可以观看项目以了解最新更新并评论提交以提供建议。对于主管来说，一切都比以前更容易，更值得信赖。对于学生来说，公共项目可以向他们未来的雇主展示令人印象深刻的展示。因此，github 也应该对课程项目有所帮助。

但是，github 不是为课程设计的。 Github 作业类型插件填补了课程和 github 之间的空白。它允许教师在不离开 Moodle 的情况下跟踪项目。特点是：

1 收集学生的 github 用户名和存储库 URL。

2 如果作业处于分组模式，则组成员共享同一个存储库。否则，每个人都有他/她自己的存储库。

3 同步所有存储库的所有提交历史并缓存日志。

4 统计小组成员的工作量百分比。

5 按可选排序顺序（名称、上次更新日期）列出所有存储库。

6 指向教师和学生可能关心的 github 页面的直接链接。

7 继承了作业类型的标准功能，例如评分。

图 4 显示了一个名为 taskmanager-for-linux-chengjisihan 的项目。 两名小组成员为此做出了贡献。

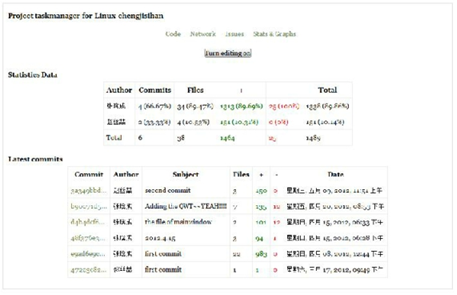


图4：github作业插件的项目状态页面

这个插件有一个简单的架构。 除了用户界面，它还提供了一个命令行同步脚本，它调用 git 命令和 Github API (Duplessis, 2012) 来获取数据。 Git 工作目录缓存在 Moodle 临时目录中，日志存储在数据库中。 CLI 脚本可以在任何时间段通过 crontab 运行。

现在，该插件仅支持公共存储库。 Github 有一个教育计划，该计划为教师提供免费帐户，以根据课程需要创建尽可能多的私人存储库。 由于可能有些老师或学生希望代码是私有的，因此插件将在下一个版本中支持私有存储库。

**案例分析**

Online Judge、Moss Plagiarism 和 Github Assignment 插件为我们的课程做出了重大贡献。 它们可以一起使用或单独使用。 一些老师表示，他们改变了编程课程的教与学方式。 而且，大多数学生也很乐意接受他们。

**在线评测**

2006 年，Online Judge 插件首次应用于计算机科学和软件工程学生的C 编程课程。现在它已广泛应用于其他专业和课程以及其他机构和国家。一些用户甚至贡献了波兰语和葡萄牙语翻译。

我校每学期有2500多名学生选修C程序设计语言课程，授课教师不超过10人。有的教师要同时指导400多名学生。但学校只支持每位教师1-4名助教。在没有 Online Judge 的日子里，作业少了，只能给一部分评分，因为老师负担不起。但是最近几年，一个拿到通过的学生要编写50多个程序是很正常的。并且所有的程序都得到了很好的评分。尽管教师必须花时间设计作业和测试用例，但作业可以轻松地在教师之间共享并在将来重复使用。因此，老师们比以往任何时候都更快乐。甚至有人在网上添加了“无限”的在线评测作业，并宣布只有完成最多作业的学生才能获得100分等等。

大多数学生也喜欢征服在线评测作业的过程。 他们将其视为在线冒险游戏。 如果成绩不满意，他们有机会重试。 随着他们的代码变得更好，他们获得了更多的技能、经验和更高的成绩。 最后，最高等级将利用他们并满足他们的可实现性。 在课程论坛中，我们可以轻松找到有关他们享受的帖子。 有人甚至需要更多的任务。 去年，每位学生平均提交一份作业的时间为 6.97 次。 最坚持的学生为一项作业提交了 816 次。

一位教授从 2010 年春季开始使用 Online Judge。 在此之前的四个学期，她的课程平均成绩为75.3、67.2、67.8和70.3。 在使用 Online Judge 后的接下来的学期中，平均成绩分别为 89.6、86.6 和 83.7。 她说她是在2010年春天才降低难度的。 之后，难度和数量都增加了很多。 数字显示教学质量显着提高。

**Moss抄袭**

抄袭是几乎所有课程的大问题。自从纸质作业消失以来，复印一直是零成本。抄袭的来源包括同学、高年级学生和网络。有研究生估计，只有不到10%的本科生在毕业前从来没有抄袭过。

在我们大学，Moss Plagiarism插件主要用于C/C++编程和操作系统课程。所有 C 提交都是纯原始源代码，因此默认设置运行良好。操作系统的作业要求学生修改Linux内核并撰写报告。因此，在提交中，有包含 Linux 代码库和学生补丁的测试报告和源代码文件。所以每个分配都有两个配置。 (a) \*.c 文件被测量为 C 语言并使用 Linux 代码库作为基本文件，因此它们在结果中被忽略。 (b) \*.pdf、\*.doc 和 \*.docx 文件以纯文本形式测量。

Moss给出多达 250 个潜在的抄袭对，无论检测到相似的百分比和多少行。其中只有一部分可以是真正的抄袭。所以确认步骤是最敏感的。教师必须仔细检查结果列表顶部的配对。 Moss 的默认排序顺序足够正确，因此通常可以安全地忽略教师认为不是作弊的第一对之后的所有对。据报道，只有极少数学生能够成功挑战确认。

很多同学都说，Moss给了他们很大的压力。由于只有少数课程使用此插件，因此这些课程的老师在课程的前半场经常会抱怨。然而，中场休息后，当学生觉得自己在压力下确实学到了一些东西时，他们开始热爱和支持老师。 2010年秋季学期，在OS课程结束时，有一个调查问“您是否支持在以下课程中使用Moss Plagiarism？”有91.7%的学生选择了“是”。

**Github 作业**

Github Assignment 插件仍在开发中，于 2012 年 3 月在课程软件设计与开发实践 III 中首次使用。 SDDP是一系列课程。他们要求 200 多名本科生在 16 周内以 2-5 个成员规模的小组为单位设计和开发一个软件项目。一项调查显示，在SDDP I&II中，只有19.27%的群体可以一直关注项目，而49.4%的群体

在最后的 1-4 周才开始了他们的工作。此外，许多学生抱怨有些小组成员没有为项目做出贡献，但他们可以得到与其他人相同的成绩。 SDDP III 正试图以 Moodle 的方式改变一些东西。

最初的学生和导师（教师或研究生）被添加到 Moodle 组中。然后在分群模式下创建了一个名为“每周报告”的论坛和一个github作业。论坛最高分设置为20，作业设置为80。

学生必须每周在论坛上发帖，报告他们在过去一周中做了什么以及他们无法解决的问题。导师应阅读帖子，参与讨论，并根据作品质量对帖子进行评分。因此，学生不断地做某事；导师知道进度和问题很快得到解决。

在 Github 作业中，学生必须在第一周设置 github 存储库和帐户，然后在接下来的几周内提交并推送代码到 github。 Git 日志已同步并显示在作业中。导师可以阅读日志，点击 github 链接，享受 github 功能，如内嵌评论和统计图。课程导师关注所有群体。因此，他通常可以查看按最后提交日期排序的存储库列表，以了解学生去了哪里。

在最后一周，学生们进行了答辩和演示。 导师在github作业中直接聆听，提问并评分。 他还检查了 git 日志和统计信息，以了解究竟是谁做了什么。

对于大多数学生来说，这是他们第一次接触 Git 和 Github。 除了一些指向 ProGit 和 Github 文档的链接外，没有关于它们的培训。 学生必须自学。 在本课程结束时，一项调查询问“git 怎么样？” 36.14% 的学生选择“它有帮助”，27.71% 的学生选择“麻烦多于解决方案”，36.14% 的学生选择“我不在乎”。 所以应该有一些培训来解决下次使用中的麻烦。 另一个调查问题是“哪个麻烦最常让你说F字？” 前两个麻烦是“Cooperation problem”和“Push rejected”。 因此，培训应更加关注这些问题。

**结论**

所有这些插件都以 GPL 许可证发布，可以在 http://git.io/hmdl 上免费下载。 自第一个版本以来，我们收到了来自世界各地许多教师、管理员和研究人员的许多反馈、建议和补丁。 他们和我们的经验证明，这些插件可以使涉及编程的课程更加高效和有效。

Moodle 2.3 更改了分配模块。 Online Judge 和Github 将迁移到它。 并且还有很多改进正在计划中。 我们希望不断升级它们并使它们免费。

**参考文献**

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>

**致谢**

我们的工作得到了哈尔滨工业大学计算机科学与技术学院和软件学院的资助。 Github Assignment 的大部分代码是Fu Jianyu写的。 Shi Xing和Liu Qiqing为在线裁判做出了巨大贡献。 Li Zhijun, Che Wanxiang, Guo Ping, Zhao Wei, Zhang Wei and Li Hanjing的经验对本文有所贡献。