Mining student repositories to gain learning analytics

An experience report

Gregorio Robles
GSyC/LibreSoft
Universidad Rey Juan Carlos
Madrid, Spain
grex@gsyc.urjc.es

Jesús M. González-Barahona
GSyC/LibreSoft
Universidad Rey Juan Carlos
Madrid, Spain
jgb@gsyc.urjc.es

Pre-print version of Gregorio Robles, Jesús M. González-Barahona: **Mining student repositories to gain learning analytics. An experience report.** EDUCON 2013: 1249-1254.

Abstract—**Engineering students often have to deliver small computer programs in many engineering courses. Instructors have to evaluate these assignements according to the learning goals and their quality, but ensure as well that there is no plagiarism. In this paper, we report the experience of using mining software repositories techniques in a multimedia networks course where students have to submit several software programs. We show how we have proceeded, the tools that we have used and provide some useful links and ideas that other lecturers may use.**

Keywords-component; mining software repositories; software analytics; e-learning; plagiarism; automatic assessment;

I. Introduction

Many engineering courses include in their syllabus programming tasks that students have to submit. These tasks are usually small programs that have to fulfill certain criteria. Sometimes students have to create these programs from scratch, but there are as well assignments where students make use of some code that is provided by the instructors and that they have to modify.

Once submitted, instructors have to evaluate these tasks according to some rules: if the programs fulfill some functional requirements and, usually, some other characteristics such as the quality of the code. In addition, instructors have to check if students have done their work on their own or if they have copied someone else's work, a circumstance known as plagiarism. If it is a group assignment, the additional problem exists of verifying if all the components of the team have been active. Nonetheless, instructors generally face the problematic that they only have access to the student's final output, the submitted assignment. The process followed by the student to obtain the final result is in general not taken into consideration due to the difficulty to gain access to such information.

In this paper, we present our experience from introducing techniques from the mining software repositories (MSR) research field in this type of scenarios. MSR is a field that "analyzes the rich data available in software repositories to uncover interesting and actionable information about software systems and projects"[1]. MRS has been a very active software engineering field for the last ten years with a specific working conference on the topic, but that has had great impact in other areas such as program comprehension, software processes, empirical software engineering, automated software engineering, among others. A modern term for MSR is software analytics.

We have used as a case study a third year computer networks course where students have to submit several programs to show the possibilities that MSR techniques introduce in the educational environments.

The contributions of this paper are following:

1. It presents experiences and ideas related to MSR that have already been used in educational environments.

2. It introduces MSR techniques that can be used in programming course assignments. Hence, methodologies and tools are presented and evaluated.

3. It provides and discusses an educational experience of using such techniques in an educational environment by the authors of the paper.

The structure of this paper is as follows: in the next section, we will show the related research. Then, we will introduce the methodology and tools used in the subject used as case study. The fourth section contains the evaluation of our educational experience, while the following one discusses the method, its benefits and limitations. Finally, conclusions are drawn.

II. Related Research

We have grouped the related research into four different categories: first, we show experiences of using version control systems in education. Then, the use in teaching of the next-generation distributed version control systems is presented. A third group of papers discusses some approaches to obtain integrated development environments for educational environments. Finally, we present those papers that have a similar point of view to this study: the study and analysis of

[1] http://msrconf.org

information obtained from students to enhance the learning experience and improve the learning outcomes.

A. Version control system

The use of versioning systems in educational environments has been discussed in several papers in the research literature.

Glassy [1] reports an educational experience using the Subversion version control system, presenting benefits and drawbacks. Among the first ones he noticed that version control systems are used in industrial scenarios, and that students acquire skills that will be valuable in future by learning them. In addition, instructors could track the student work better, and whether the student progressed incrementally or waited until the deadline. On the negative side, their experience showed that introducing version control systems did not change the temporal behavior of students by its own; milestones had to be introduced so that the work was more distributed and not shifted towards the final deadline. In addition, the use of versioning systems supposed more work on the side of instructors and students as new concepts had to be introduced and understood by students.

Milentijevic et al. [2] report the experience of the use of a version control system to support project-based learning (PBL) environment. The study shows that activities that are sometimes difficult to perform in PBL-learning such as mentoring and monitoring are improved considerably, as instructors can control the process and how students cooperate. In addition, they note that the solution is cost-effective using free software and that the inclusion of web interfaces lowers some adoption barriers both by students and by fellow instructors.

Jones [3] explores the use of versioning systems in evaluating the contributions of individual students in group coding assignments.

B. Distributed versioning systems

A new generation of versioning systems has become widely used in the last years, gaining in popularity tools such as git or Mercurial.

Rocco and Lloyd [5] discuss the benefits of using distributed versioning system in education against the older, centralized systems. In the experience of the authors, the simplicity and velocity with which students can set up a new project is a major advantage, aside from the fact that a lot of concepts usually not related to the subject can be omitted. Submission can also be simplified to the extreme, as students can just zip their repositories and submit them in a single file without requiring any (centralized) infrastructure with its related issues: space, permissions, security, etc.

Laadan et al. [6] use a distributed version control system in an operating systems course. The authors note that the use of an advanced distributed system has several advantages.

A first one was the optimization of the repository space, as all the assignments included modifications to the Linux kernel, but only a single copy of it was required. They note that being distributed, offline work is not problematic as each repository is self-contained and independent. Nonetheless, the lecturers had to use logging of the submissions to the central repository as the students may tamper tamper with submission dates and times.

C. Integrated environments

A third set of efforts is devoted to the creation of integrated environments for the fulfillment of programming tasks by students.

Chen and Marx [7] report an integrated teamwork enablement and management system that they have labeled ITEAM (from Integrated Teamwork Enablement And Management). This system groups Course Management System (CMS), Source Configuration Management (SCM) and public teleconference services into a unique platform.

Helmick presents an integrated online courseware for computer science courses [8], a web-based online courseware system for the management of computer science courses developed by the Miami University. By means of using a versioning system, the system allows rapid feedback. It uses PMD[2] to check the style of the submitted Java code and has capabilities for automatic grading. Authentication mechanisms among the various components are shared using LDAP.

Inspired by the now defunct Google Wave communication service, Vandeventer and Barbour created a real-time, collaborative IDE for enhanced learning in computer science, which they named CodeWave [9]. CodeWave combines features of IDEs such as syntax highlighting with others such as integrated messaging and logged playback, a feature that puts the granularity of the changes at the keystroke level. The authors state that by adding this logging feature, instructors are able to "answer the question of whether or not a student is commenting during development or waiting until afterward", among others. Workload statistics that include the percentage of code contributed per user, the amount of time editing and the distribution of ownership are also offered.

D. Software learning analytics

The final set of papers is the one that is closer to the perspective of this paper. In them, the main goal is centered in the knowledge that can be acquired from the analysis of data and traces left by students when doing their programming assignments.

Liu et al. present a way to track the progress of students using historical information from a versioning system, presenting and analyzing the information, so that the instructor obtains them in a variety of forms [10]. They provide in their study some insight into the type of information that could be gathered from applying such techniques on the student's assignments. The approach that this study presents is similar to the one in this paper, although its infrastructure is limited to obtaining information only from the versioning system logs.

Mierle et al. [4] have studied over 200 student assignments that used a versioning system and have analyzed if there is any correlation between some measures such as student behavior or code quality and the grades that students have obtained. The

[2] http://pmd.sourceforge.net/

results obtained showed that no predictor was found to be stronger than simple lines-of-code-written.

Poncin et al. mine students capstone projects, a group project in industry that students have to take at the end of their studies [11]. The authors note that while "traditionally such projects solely focused on the software product to be developed, in more recent work importance of the development process has been stressed". They use therefore two tools: FRASR [12], that extracts information from the software repositories and integrates it in a unique log, and ProM [13], that analyzes the previous log and offers several visualizations to understand the process.

III. Infrastructure

The course for which we have set-up a softwarelearning analytics environment is a third-year multimedia networks course. In this course, students learn about multimedia protocols such as Session Initiation Protocol (SIP), Real-time Transport Protocol (RTP), Real-time Streaming Protocol (RTSP), Synchronized Multimedia Integration Language (SMIL) and streaming multimedia over IP networks.

The theory is complemented with several practical lessons, for which students have to create small programs that handle with these protocols.

The technologies that are used are:

- Python: Python has been selected as programming language as it is simple and high-level language that allows students to create complex programs without major effort. Students attending this course have basic programming knowledge from previous courses, although they have not been introduced to object-oriented programming.
- git: We use this distributed versioning system for the programming assignments. In the first practical lesson we introduced the program and its basics: creating a repository, adding and committing files, inspecting the log, etc. Each programming assignment requires as first task to build a new, local repository. The instructors collect the repositories automatically from the student's home directories at our teaching laboratories for their evaluation.
- pep8: PEP 8[3] is the name of the proposal that contains a style guide for Python. One of the goals of the course is that students get used to this style guide. Python scripts may be checked against the style guide by means of the "pep8" command line tool.
- wireshark[4]: Wireshark is a graphical network protocol analyzer.

As the programs that students have to create mainly include the communication between clients and servers using the above mentioned protocols, the assignments usually include a live capture with the result of a scenario.

As instructors, we have tried to automatize the process of retrieving, analyzing and evaluating the assignments as much as possible. The complete process has been divided in several steps, and is described next:

1. Retrieval

The retrieval of the assignments is done through a web interface, that the sysadmins of the computer labs offer. This allows to collect specific directories in all the homes of the students at a specified time. These directories are copied to a location in the home of the instructor.

2. Preprocess

The preprocess step consists of several substeps for each student. The output of the preprocess step is a text file per student with all the information n a structured way, suitable for being parsed in a later step.

a) Cloning of the repository

We use git to retrieve the working copy that contains the last version of the student's program. Instructors provide a check script for students so that they can ensure that they have done this step correctly.

b) Checking if the files with the assignment exist are have been correctly named

A Python script checks if the files that are included in the working copy correspond to the ones specified in the assignment. The number of files delivered should be exactly the same as specified. The checking script provided by the instructors includes as well some instructions to allow students to verify if they follow these rules.

c) Checking if the style guide has been followed (with pep8)

All Python files are assessed against pep8. We use the "-q --statistics" parameters to obtain statistical information for the compliance of the scripts. As feedback for the student we also run pep8 with following parameters: "--repeat --show-source --statistics".

d) Evaluating the quality of the code

Almost all programming languages have programs that evaluate some quality attributes of the code in a static manner (i.e., they look exclusively at the source code). The most notable program for Python is PyChecker. But in our experiment we have used a different one, Pylint[5], as it offers some more features, such as verifying if declared interfaces are really implemented. Pylint can be configured to omit some tests (in our case, we omit following conventions and warnings: C0103, W0231, W0621) and offers a numerical mark at the end summarizing the quality of the script. We use this mark as a factor of the final grade for the submission.

e) Extraction of some software metrics

[3] http://www.python.org/dev/peps/pep-0008/

[4] http://www.wireshark.org

[5] http://www.logilab.org/project/pylint

A number of traditional software metrics is obtained from the source code, such as lines of code, number of classes, number of functions/methods, McCabe complexity measures etc. In addition, we look for docstrings and to what extent classes and functions/methods have been documented in the source code. All this is done with the help of the pymetrics[6] script.

f) Retrieving of the git log and analysis

There is plenty of information that can be obtained from the analysis of the log of a versioning system. This is because, in addition to the commit message, the versioning systems store information on the committer (and author, if they are different), the timestamp, the file, etc. Mining versioning logs has been a major research are of the MSR field and there are several tools that have been designed to perform this task. In our approach, we use CVSAnalY [14].

g) Treatment of the wireshark capture

A task included in many computer network programming assignments is a network capture with the traffic that the agents (clients and servers) have exchanged. There are several programs that students may use for the capture, as for instance wireshark, from the GUI, or tcpdump, from the command line. These captures are stored in a specific binary format, called libpcap, which is difficult to parse. With the help of a command-line script, called tshark, we transform the pcap (packet capture), an application programming interface (API) for capturing network traffic, captures into the Packet Details Markup Language (PDML), an XML specification.

PDML files can therefore be easily parsed and analyzed.

In the evaluation of the assignments, we parse them to obtain information such as the correct exchange of protocol-consistent messages, the use of the correct IP addresses and ports, etc.

3. Plagiarism detection

There exist several programs with the objective of assessing student programming assignments for unauthorized copying (plagiarism). Although it is not free software, we have made use of the MOSS[7] tool, which can be used (gratis) by instructors via a web interface, or paying a license on a local machine.

We plan to use a different tool, similarity_tester[8], in the future for this task, in order to have a completely free software infrastructure.

Plagiarism detection tools usually offer a percentage of similarity among programs. Above a given threshold (which depends on the assignment), assignements get marked as suspicious. In our quest for copied assignements, we use as well programs from previous courses.

4. Functional assessment

The functional assessment is concerned with the requirements that a software program has to fulfill. We therefore use a type of black box testing, based on the specifications of the program to be evaluated, so that we provide inputs and examine the output produced by the program. A battery of tests has been designed from the guidelines for each of the assignments.

5. Post-process

The text file with information from all previous steps is parsed, and final grades for the assignment are calculated. This script also serves to obtain a text file with feedback for the student, with input information from all the steps. Students have therefore a detailed report of their assignment, how it has been graded and how they could improve their programming.

Instructors get, in addition, of a report of the whole process, including assignements suspicious of plagiarism, and errors during the whole process.

6. Creation of a personalized exam

The system is able to automatically create a personalized exam for each student based on the submitted assignment. Questions are introduced into a database and displayed with a web interface. This personal exam has as main goal to verify that the student is the original author of the assignment, but it serves as well as a way of ascertain to what extent the student is aware of what he/she has been done. The reason for the second goal is that in programming tasks students often lose the big picture in favor of the programming details. Therefore, the student is asked several type of questions:

a. Code snippets: The students gets displayed code snippets of his/her own assignment and from other students, and is asked if she/he recognizes the code as her/his own, and in that case, from what source code file it has been extracted.

b. Black box questions: In a similar fashionto functional assessment, the student is asked for the outputs that the program would produce given a specified input.

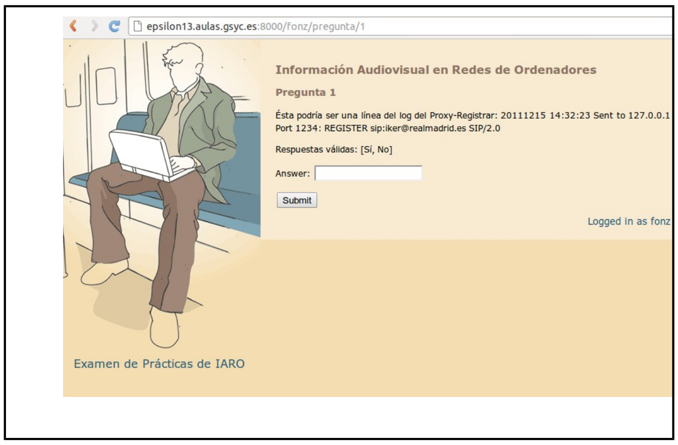
c. Questions about specific scenarios: The student is presented with a change to a part of the code orto the configuration, and is asked for the consequences this would result.

Personalized exams take from 10 to 20 minutes and can be done simultaneously by many students. Usually, we perform these exams only for the last assignment, which is a larger program. A screenshot of the personalized exam is shown in Figure 1.

[6] http://sourceforge.net/projects/pymetrics/

[7] http://theory.stanford.edu/~aiken/moss/

[8] http://packages.debian.org/similarity-tester


epsilon13.aulas.gsyc.es:8000/fonz/pregunta/1

Información Audiovisual en Redes de Ordenadores

Pregunta 1

Ésta podría ser una línea del log del Proxy-Registrar: 20111215 14:32:23 Sent to 127.0.0.1 Port 1234: REGISTER sip:iker@realmadrid.es SIP/2.0

Respuestas válidas: [Sí, No]

Answer:

Figure 1. Screenshot of a personalized exam (in Spanish).

IV. Experience Report

The infrastructure has been used for the submission of 5 assignments during a four month period. We have used it in two consecutive courses so far with relative success.

Four out of the five assignments where small programs that account for 50 to 100 lines of Python code for which students had two weeks (around 10 hours of total work by the student). The fifth one is the final assignment, a more complex program that comprehends between 300 and 600 lines of code for which one month is allocated (around 35-30 hours of total work by students).

A. Learning experience for students

The experience with this method shows that students welcome feedback and introduce enhancements in the following assignments. From the perspective of the students, this is the most satisfactory output of our method as they are provided with high detail feedback for each assignment. This affects non-functional and functional requirements.

Regarding non-functional requirements, we have seen that they make better use of git, introduce better git log comments and become aware of the benefits of the continuous development model that a versioning system offers. In addition, they get used to the style guide with large sauces. Even if we have observed that they only pass the pep8 tool just before the submission (even in the final assignments), we have found that the intermediate state of the code improvesnotably in the last assignments. The code submitted in the last assignments is of better quality as well, measured from the quality attributes (such as short try-except exception statements, readable variable names, etc.) considered from pylint.

Regarding functional requirements, we had the experience from previous years that students have difficulties in understanding that (computer network) standards provide over a limited amount of possibilities in the communication exchange: the semantics and syntactics of the protocols have to be followed in detail, as minor changes may produce a non-standard exchange and hence an error. As a result of using automatic tests already in the first assignments, they understand this problem early in the semester.

B. Experience by instructors

Although we, as instructors, promised ourselves complete automatization of our experiment, we are aware that this is hardly possible. For each of the assignments there has been always manual inspection and manual evaluation on our part. This is because there are always cases that are outsiders, such as wrong submissions, wrong code structure, etc. that are difficult to integrate in a completely automated environment. The good news, however, is that this is only necessary for a small amount of cases, specifically for those student assignments that after manual inspection by the instructors are noted to be suspicious of having being evaluated wrongly. Out of the 164 total submissions in the first year, this accounted for 46 cases, while in the current semester out of the 110 submissions this has only been the case for 22 of them. Part of the improvement is due to assignment instructions better documented.

We have also noted that students require some time to get used to the environment. In this way, the use of git is introduced prior to the first submission. Even if a script is provided that allows students to verify if the assignment has been submitted correctly, students are asked to submit it through traditional means (a file up-load in the Moodle course) just in case they did not use git in a proper way.

V. Conclusions and Discussion

In this paper, we have presented the experience of using an almost automated infrastructure that allows to gather software analytics data from programming assignments by students.

The first benefit that offers such an approach is that, even if it has been shown to be not fully automatable, it is scalable to the point that it allows for continuous evaluation for large groups of students. We have seen that having continuous evaluation offers some benefits that can be seen in the learning results at the end of the semester. Basically, students obtain continuous feedback of their work, and thanks to the tools used this feedback is very detailed.

One of the main outcomes of our experience as instructors is that no evaluation system of this complexity can be made completely automatic. We have seen that refining the infrastructure is an iterative process, and that there will always be elements that will be outside the means of instructors. There are some efforts that can be done to minimize the manual parts of the process, the most important one being that students attend the instructions properly.

In addition, the domain of the programming tasks is very important. For instance, our approach is suited specifically on multimedia computer network programming assignments. This implies to have specific functional requirements that are completely different to be met than in other domains. If the learning goals or the area are different, other tools and processes may be better choices than the ones presented in this paper.

Although our assignments have to be done on an individual basis, our method could be increased easily (and we think that with success) to group projects. It is our understanding that much of our approach could be reused as is, and that the main efforts should be to integrate means to analyze the communication exchange among the team members. Nonetheless, if properly used, the log analysis of the versioning system offers detailed information of the amount of work by each participant in the group, as some research papers have pointed out [2, 3, 10].

We have built our infrastructure integrating external tools. A positive side effect is that this makes students learn many of the tools that they may use in their professional careers, such as a distributed versioning system or a style-checking tool.

A possibility that we have thought of is using an integrated development environment (IDE) such as Eclipse and use or develop external plugins to achieve our goals. By using an IDE we could augment the collection of information such as it is done in CodeWave, at the point of logging the keystrokes that students perform. The problem of this solution is that it is too IDE-centric, meaning that it is difficult to integrate all the external tools. We plan to keep on working on our insfrastructure in the near future.

Acknowledgment

This research has been partially supported by eMadrid, S2009/TIC-1650, "Investigación y Desarrollo de tecnologías para el e-learning en la Comunidad de Madrid" funded by the Region of Madrid. We would like to thank all the students from the Escuela Técnica Superior de Ingeniería de Telecomunicación who have participated in this teaching experience and have provided feedback and suggestions.

References

[1] Louis Glassy. 2006. Using version control to observe student software development processes. J. Comput. Small Coll. 21, 3 (February 2006), 99-106.

[2] Ivan Milentijevic, Vladimir Ciric, and Oliver Vojinovic. 2008. Version control in project-based learning. Comput. Educ. 50, 4 (May 2008), 1331-1338. DOI=10.1016/j.compedu.2006.12.010 http://dx.doi.org/10.1016/j.compedu.2006.12.010

[3] Curt Jones. 2010. Using subversion as an aid in evaluating individuals working on a group coding project. J. Comput. Small Coll. 25, 3 (January 2010), 18-23.

[4] Keir Mierle, Kevin Laven, Sam Roweis, and Greg Wilson. Mining Student CVS Repositories for Performance Indicators. In 2nd Int. Workshop in Mining Software Repositories, pp. 41-46, May 2005.

[5] Daniel Rocco and Will Lloyd. 2011. Distributed version control in the classroom. In Proceedings of the 42nd ACM technical symposium on Computer science education (SIGCSE '11). ACM, New York, NY, USA, 637-642. DOI=10.1145/1953163.1953342 http://doi.acm.org/10.1145/1953163.1953342

[6] Oren Laadan, Jason Nieh, and Nicolas Viennot. 2010. Teaching operating systems using virtual appliances and distributed version control. In Proceedings of the 41st ACM technical symposium on Computer science education (SIGCSE '10). ACM, New York, NY, USA, 480-484. DOI=10.1145/1734263.1734427 http://doi.acm.org/10.1145/1734263.1734427

[7] Zhixiong Chen and Delia Marx. 2007. ITEAM integrated teamwork enablement and management. J. Comput. Sci. Coll. 22, 6 (June 2007), 117-125.

[8] Michael T. Helmick. 2007. Integrated online courseware for computer science courses. In Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE '07). ACM, New York, NY, USA, 146-150. DOI=10.1145/1268784.1268828 http://doi.acm.org/10.1145/1268784.1268828

[9] Jason Vandeventer and Benjamin Barbour. 2012. CodeWave: a real-time, collaborative IDE for enhanced learning in computer science. In Proceedings of the 43rd ACM technical symposium on Computer Science Education (SIGCSE '12). ACM, New York, NY, USA, 75-80. DOI=10.1145/2157136.2157160 http://doi.acm.org/10.1145/2157136.2157160

[10] Liu, Y, E. Stroulia, K. Wong, and D. German, "Using CVS historical information to understanding how students develop software," in 1st Int. Workshop in Mining Software Repositories, pp. 32-36, May 2004.

[11] Wouter Poncin, Alexander Serebrenik, and Mark van den Brand. Mining student capstone projects with FRASR and ProM. SPLASH '11 Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion

[12] Wouter Poncin, Alexander Serebrenik, and Mark van den Brand. Process mining software repositories. In European Conf. on Softw. Maintenance and Reeng., pages 7–15. IEEE, 2011.

[13] Boudewijn F. van Dongen, Ana Karla Alves de Medeiros, Eric Verbeek, Ton Weijters, and Wil M. P. van der Aalst. The ProM framework: A new era in process mining tool support. In

[14] Int. Conf. on App. and Theory of Petri Nets, volume 3536 of Lecture Notes in Computer Science, pages 444–454. Springer, 2005.

[15] Gregorio Robles, Stefan Koch and Jesús M. González-Barahona. Remote analysis and measurement of libre software systems by means of the CVSAnalY tool. 2nd ICSE Workshop on Remote Analysis and Measurement of Software Systems. May 2004