

Distributed Systems Case Study Report
Distributed Version Control System in Advanced
Programming and its Workflow

A Case Study Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Bachelor of Technology

In
Computer Science Department

By
Indradhar Paka
Sai Venkat Lakamsani
Raghavendra Naga Varma

CSE-III
3rd year












BML MUNJAL
UNIVERSITY™

SCHOOL OF ENGINEERING AND TECHNOLOGY

BML MUNJAL UNIVERSITY GURGAON

November, 2020

TABLE OF CONTENTS

	ABSTRACT
	INTRODUCTION
	BACKGROUND
	DETAILED STUDY
	<ul style="list-style-type: none">▪ Version control systems in education▪ The Courses
	DATA ANALYSIS
	RESULTS
	<ul style="list-style-type: none">▪ Commit activity over the whole course▪ Commit activity during the sessions▪ Commit message analysis
	CONCLUSION
	ACKNOWLEDGEMENT
	REFERENCES

SUBMITTED TO

Dr. Satyendr Singh
Assistant Professor
Computer Science and Engineering Department
School of Engineering and Technology

PARTICULARS OF STUDENTS:

S No.	Name	Roll No.
1.	Indradhar Paka	1800315C203
2.	Raghavendra Naga Varma	1800241C203
3.	Sai Venkat Lakamsani	1800254C203

ABSTRACT

Distributed Systems is an essential course in computer science curriculum, which helps students to develop a mental model of how networks and connecting devices and distributed systems work. Distributed systems in daily life applications and large-scale plants are often complex, non-deterministic which makes them difficult for students to understand. One such concept is Distributed version control system.

In practice, DS course involve online classroom lectures describing high-level abstractions of the concepts, and student complete assignments to apply the material in a more concrete way. Depending on the assignments, this approach may leave students with only a theoretical understanding of DS concepts, which may be different from the actual way these concepts are implemented in a distributed system. What many students require is a practical knowledge of how distributed systems works and implementation to supplement the high-level presentations of concepts taught in class or presented in a textbook.

Introduction

We intend to investigate the adoption of the new trend within version control systems (VCS) — distributed version control systems (DVCS) — within the field of software engineering. The use of DVCS has increased in the last years, with the introduction of several implementations. The move from centralized version control to DVCS may not always be unproblematic due to conceptual differences. It is important to investigate the benefits and challenges of DVCS in order to let software projects make informed decisions in their choice of, and use of, version control systems.

Version Control Systems are essential tools in software development. Educational institutions offering education to future computer scientists should embed the use of such systems in their curricula in order to prepare the student for real life situations. The use of a version control system also has several potential benefits for the teacher. The teacher might, for instance, use the tool to monitor students' progress and to give feedback efficiently. This study analyzes how students used the distributed version control system Git in advanced programming related courses. We also have data from a second-year course, which enables us to compare between introductory level and master's level students. We found out that students do not use the system in an optimal way; they do not commit changes often enough and regard the version control system as file storage. They also often write commit messages which are meaningless. Further, it seems that in group work settings there is usually one dominant user of the system.

Background

Version control systems (VCSs) have a decades-long history in professional software engineering with early systems like Source Code Control System (SCCS) and Revision Control System (RCS) developed in the seventies and eighties respectively. These pioneering systems only supported storage of the versions on the file system, while later systems also allowed for remote and mostly centralized storage of the versions. The most well-known centralized systems are Concurrent Versions System (CVS) and Subversion (SVN). Currently, there is a trend towards the use of distributed version control systems (DVCS) where each user has a local copy of the repository which can be synchronized with other repositories. Systems such

as Git and Mercurial exemplify this type of present-day decentralized technology. These DVCSs enable flexible change tracking, reversibility, and manageable collaborative work, which are valuable for both small and large projects.

There are many arguments for incorporating VCSs into an educational setting. From a teacher's point of view, using VCSs increases the possibility of monitoring how students make progress with their assignments and eases the feedback process. The teacher could, for instance, include corrections and suggestions directly into the students' program code. More generally, educators acknowledge that the use of VCSs relates to effective teamwork and that it is a crucial skill to be taught to prepare a competent workforce for present-day distributed workplaces.

An educational concern of interest to us is how students use VCSs. This has been previously studied by Mierle et al. who investigated VCS usage patterns in a second-year course, hoping to find a correlation between an effective use of VCSs and study success. No clear patterns could be identified in the data which the authors attributed to the fact that beginner students climb their learning curve at different rates. These authors call for more research on VCS usage patterns in upper-year courses, which motivates the present study.

We have collected data about students' use of the distributed version control system (Git) from three different courses: Introduction to Software Engineering (second-year bachelor), Functional Programming (master's level), and Service oriented architectures and cloud computing for developers (master's level). A hypothesis arising from teacher observations during these courses is that students use VCS principally as a submission system rather than what it is intended to be. By this we mean that

- students commit at the end of the class sessions or right before the deadline, or there is only one commit per week/task,
- only one group member commits everything,
- students do not consider what file types to commit,
- overall, with no specific training, student do not use VCS efficiently.

We study these issues quantitatively exploring version control commit frequencies, commit sizes and the activity of individual students. Our specific research interest is the potential usage patterns identifiable in the commit log data of Git repositories.

Version control systems in education

From the study of Clifton et al. summarize that in educational settings VCSs have been adopted to enable more realistic software development experiences for students, as a tool to monitor or visualize team and individual contributions, and for noncoded artifacts such as creative writing.

From the study of Clifton et al. themselves, as well as many others, use a VCS for course management purposes. Further, some authors regard VCSs as a valuable tool to monitor and understand how students develop code. Unsurprisingly, one of the most usual educational targets appears to be courses with project work where VCSs both foster teamwork and facilitate course management tasks such as assessment and grading.

Study by Milentijevic et al. has gone as far as to propose a generalized model for the adoption of VCSs as support in a variety of project-based learning scenarios. All in all, we find that there is a consensus of the benefits of VCSs as an integral part of computing curricula, one key argument being that they measure up to the requirements of globally distributed workplaces.

There are also challenges in the educational use of VCSs. Reid and Wilson study states that who used the CVS system, report on the confusion in judging which of the students' assignment versions was the final one. Glassy found that students tend to put off working on assignments for as long as possible, even though a VCS is proposed to them with the hope of iterative work processes.

Issues of this kind relate to inefficient use of VCSs. Furthermore, Reid and Wilson study also noticed that some students mixed the functionalities of the CVS check out and update commands, and that also teaching assistants encountered problems if they had not properly familiarized themselves with the tool. These issues were due to a lack of a mental model of the VCS system used.

Yet another challenge Reid and Wilson study says that raise is increased teacher workload when repositories are initiated and managed by teachers.

In a more recent study, Xu points out that there can be a long and rough learning curve before students feel comfortable using Git.

Accordingly, study by Milentijevic et al. who used CVS, report that students find a VCS to be a useful tool after they are sufficiently familiar with it.

In a research paper by Glassy and Xu, the value of informative commit log messages is raised as a topic to be emphasized to the students. study of Rocco and Lloyd says that in turn observed that some student have difficulties in understanding what constitutes "a significant change" to be committed.

It is much more difficult to find systematic empirical studies on issues such as how frequently students make commits and how they share the work. Rocco and Lloyd study has found in their data that over 80% of a CS1 course population could adopt an iterative work process with the Mercurial system (50.0% did 7– 21 commits and 33.3% more than 21 commits).

On another course the authors defined a minimum commit frequency for one assignment and no requirements for the assignment that followed. With the first assignment, 75% of the students obtained a reasonable commit frequency, while with the latter this was 81%, altogether indicating that informing students of proper VCS usage can have a positive effect on their work processes.

The authors note that not only were the students able to grasp the basics of the VCS (Mercurial), but they tended to continue to take advantage of the tool later.

The present study focusing on the students' usage patterns with the Git system in both a second-year course and master's level courses complements the studies such as the ones by Rocco et al. and Mierle et al. & Reid and Wilson.

The Courses

Software Engineering (SE), a course assignment, and an end-of-course exam. The lectures introduce students to the basic concepts of software engineering, while the mandatory course assignment is the preparation of a project plan. The assignment is done in small groups and consists of four larger phases that need to be accepted by the lecturer. Mandatory supervision sessions on version control were arranged at the beginning of the course in order to encourage all the students to use the distributed version control system Git for the group assignment. The course had altogether 72 students in 33 groups (2.18 ± 0.76 students per group).

Functional programming (FP) implemented without traditional lectures and exams. The course is run in week-long cycles such that each week a new set of exercises is announced for the students. Students work in small groups and all their study time is devoted to programming the weekly exercises. Two contact sessions are held each week. The first one is devoted to supporting the students' work and answering their questions. During the second weekly contact session there is a review of the student-written code. Overall, the course emphasizes self-direction on the part of students, like recently discussed course models such as the flipped classroom. Git was proposed for students as their primary group work tool and all the exercises had to be returned via it. Thirty-six students were active in the course divided over 13 groups. (2.77 ± 0.89 students per group).

The last master's level courses studied, Service oriented architectures and cloud computing for developers (SOA&CC), introduces students to the use of digital services and the concept of cloud computing. A format like the FP Course During that part of the course students undertake independent group work on a set of assignments each week. Two weekly sessions are arranged for the group work and one mandatory contact session focusing on reflective program review is arranged at the end of each week. During the course Git is not only used as a version control system; it is also used as a tool to deploy code to Platform as a Service (PaaS) providers. Nine groups of students were formed with altogether 36 students (4 ± 0.82 students per group).

All three courses utilize the Faculty's YouSource1 system. Like staff members, students can use their university credentials to log in to this system and create projects and Git repositories to manage collaborative work. The projects and repositories can be defined to be either private or public and collaborators can be added to them with a variety of permissions. This system has been in use at the department since mid-2010 and has been used in many courses and research projects.

It should be noted that in the remainder of this report we are specifically concerned with the Git version control system, which belongs to the third generation of version control systems (DVCSs). Students are free in their choice of environment for interacting with the version control system. Students can for instance use the git command line tool, tools with a graphical user interface, or tools included in their integrated development environments.

Data Analysis

The Git repositories which students or course teachers created for the respective courses on the above-mentioned YouSource system are the source of all data analysis in this report. One limitation of this data source is that we cannot see any data related to branches which a student did not push to the central Git server.

However, if work of one of these so-called local branches got merged into a branch which is synchronized with the central Git server, we can see its history as well. Further, this limitation is of minor importance since we are mainly interested in how students use the version control system in group work settings.

Another limitation, which is inherent to the Git DVCS, is that we cannot know for sure whether time stamps on commits are truthful. It is technically possible to tamper with the date of the commits, but since there is no benefit for students to do so, we assume that the time stamps are correct.

To study our research hypothesis, we will perform five different analyses, the first four of which are based on commits to the repository and the last one on the content of the repositories. For each commit we extracted the number of insertions and number of additions in accordance with the short status log of each commit².

We added these two numbers together to form what we will call the number of changes of that commit. The tools used in the analysis have been developed by the authors and consume output produced by the diverse git commands.

For the first analysis we will, for each course, look at the commit activity over the whole course. To be concrete, we will visualize the commit activity by plotting the estimated probability density function of the total number of changes, i.e. for all students, over the span of the course. The density is estimated via the standard kernel density estimator, using a Gaussian kernel with bandwidth of 6 hours. The height of the plot then shows the relative likelihood of a commit at a specific point in time.

The second analysis focuses on students' activity during the implementation sessions. This is done only for the FP and SOA&CC courses since the SE course does not have distinct sessions during which students get time to implement their work. We use a similar method as in the first part but accumulate all commits that were made during the implementation sessions in the same plot. This plot shows when the students commit their code during the contact sessions. In the figure the far left of the x-axis represents the start of the session and the far right 15 minutes after the end. This is done in order to account for commits right after the sessions. In this case we use a bandwidth which is one tenth of the total length of the session.

In the third part we perform an analysis of the commit messages in the different courses by classifying them in three categories: useful, trivial, and nonsense. A message is placed in the nonsense category if its content is not anyhow related to what is being committed. An example of this type of messages are these which contain only a couple of random letters, needed because the git system does not allow for empty commit messages. A trivial message is one which has no information beyond what is immediately visible from the commit meta-

data. This category includes, for instance, a message consisting of a list of changed files or one saying that a given commit is a merge of two branches. All other commits are classified as useful. It should be noted that being in the useful class does not directly imply that the message is of high quality. It only means that the message is not trivial or nonsense. The classification was done manually by the respective teachers of the courses. We do not try to make a comparison between the courses, because the bias caused by having different raters is difficult to estimate.

In the fourth part we try to measure whether the version control system is used equally among the students in the group. If the system would be used by all students in a group, we would expect that the most active student in a group of n students performs $(1/n) * 100\%$ of the commits. To represent this number for all groups in the different courses we first find the students with the highest number of commits in their respective groups. Then we calculate their individual share in the total number of commits of their group. We then create an overview of the obtained percentages where we show different graphs for different group sizes since comparison among unequal group sizes would lead to biased results. It only makes sense to measure this for groups with more than one person. The SE course had a few single-person groups, hence only 28 groups from that course are included.

For the fifth and last part we investigate the types of files which students put under version control. First, teachers of each of the courses listed the file types and limits which they would expect a normal repository to contain. We started out from the files included in the HEAD of the master branch. For the FP and SOA&CC course we determined the type of each file using the BSD file3 command. The SE repositories required a manual analysis to decide the type of the files because the file command is unable to distinguish between the file types in use in the course. Then we counted the number of files of each file type. Then for each count, we compared it to the number of files expected by the teacher and any surplus was counted as garbage. The final number which we calculated for each group is the fraction of garbage in the total number of files.

Results

This section describes the results of our analyses. The first subsection shows the results for the analysis of the student activity during the whole course. In the second subsection, we focus on the implementation sessions only. The results of the analysis of the commit messages is shown in subsection three. Then we consider the activity distribution among students in the fourth subsection. Lastly, we look at the types of files which students submit to the version control system.

Commit activity over the whole course

The student activity in the SE, FP and SOA&CC courses is presented in the Figures 1, 2, and 3, respectively. In the SE course, we draw thick vertical lines to indicate the end of each of the four phases of the course assignment. As can be seen in that figure, there seems to be no correspondence between these deadlines and the student activity. In this course where VCS training was provided, students appeared to commit rather evenly throughout the course.

We attribute the activity spike at the start of this course to the students trying out and getting familiar with the version control system at the point in time of the training sessions.

In the graphs of the FP and the SOA&CC courses (figure 2 and 3), we indicated with thin vertical lines the sessions during which students get time in the classroom to work on the assignments. The thicker vertical lines indicate deadlines for the weekly assignments. The dates of the sessions are displayed on the x-axis in a month/day format. In contrast to the SE course, these graphs show a closer correlation between student activity and the implementation sessions and the deadlines. The graphs suggest that most of the work was committed during the contact sessions, which again suggests that students bring their work to the sessions to be committed there. This prompts us to study the student behavior during the sessions separately in the next section. It is also clearly visible that the students have a very low activity during the weekends.

Commit activity during the sessions

In the FP and SOA&CC courses students were more active during sessions than at other times. The graphs in figures 4 and 5 show the students activity during the sessions and 15 minutes after the session. We normalized the duration of the session (90 minutes) and the 15 minutes overtime between zero and one. Interestingly, we notice a similar behavior in both courses. There seem to be three periods of higher activity. The first moment of higher activity is in the beginning of the session after about 10 minutes. The second one, which last longer, is between 20 and 40 minutes after the start of the session and lastly, the activity peaks shortly after the session.

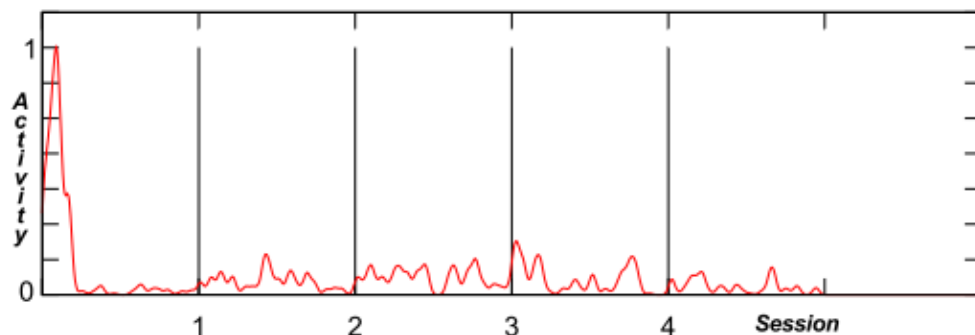


Figure 1 Commit activity during SE course

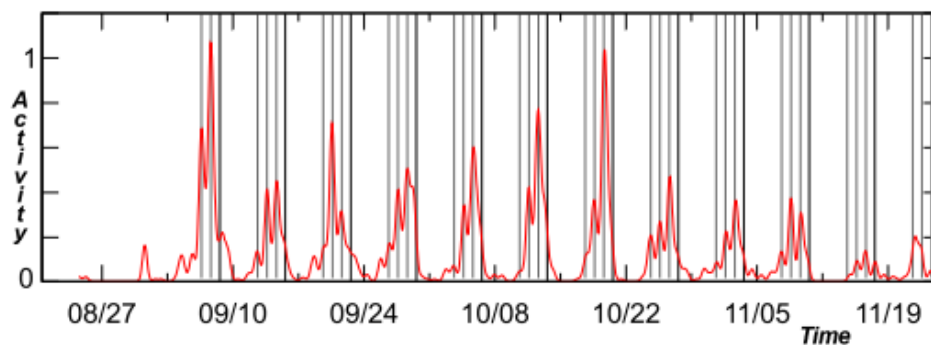


Figure 2 Commit activity during the FP course

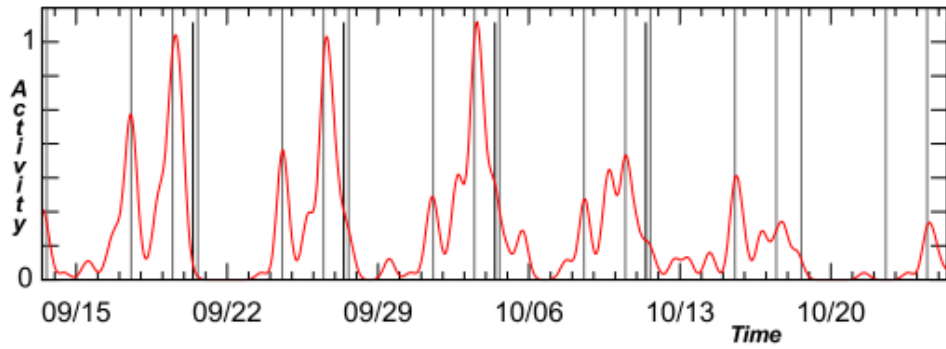


Figure 3 Commit activity during the SOA & CC course

The first period of activity is most likely because individual students have been implementing parts at home. These students then decide to commit only after receiving consent from other group members. This is an indication that students do not know how to use the version control system efficiently, as in principle they could have used a separate branch for their local development and merged their changes to their shared version of the exercises. Also, speculating based on student dialogue, some students might have feared 'losing face' by making their preliminary versions visible to others, including the teacher.

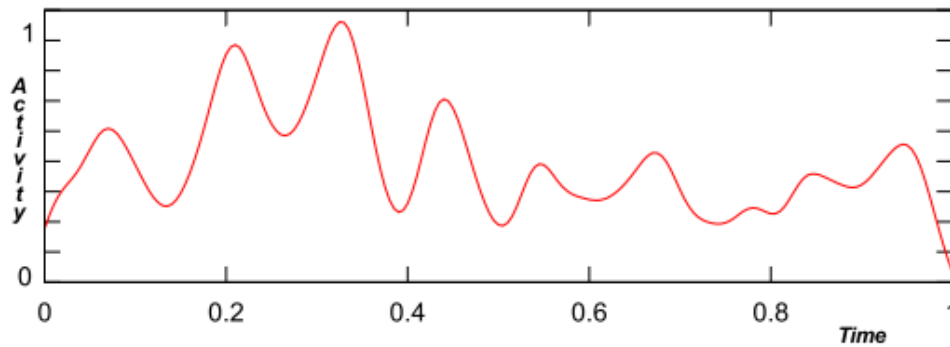


Figure 4 Commit activity during the sessions of the FP course

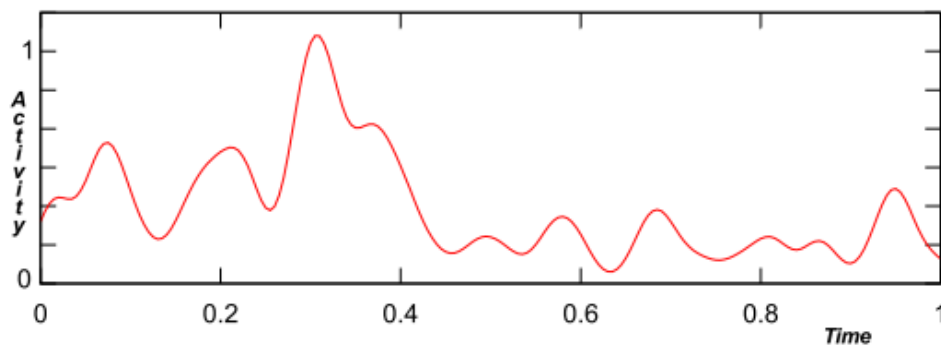


Figure 5 Commit activity during the sessions of the SOA&CC course

During the second period of activity students are using the system as they are supposed to, committing changes regularly. Then the activity drops for quite some time before reviving shortly after the session. We attribute this last peak to those groups who have been working during the whole session without committing many changes. At the end of the session they want to store their work for later continuation and decide to put all their work in the system.

Commit message analysis

The classification of the commit messages was performed for the SOA&CC and FP courses and yielded the results shown in table 1.

Table 1 Categorization of the commit messages per course

	USEFUL	TRIVIAL	NONSENSE
SE	996 (67%)	430(29%)	59(4%)
FP	1422(78%)	276(15%)	129(7%)
SOA&CC	289(74%)	65(17%)	37(9%)

In an ideal repository we would not find any trivial and nonsense messages. What we see from the table however is that there is a significant amount of these types of messages.

It is not visible from the table, but the teachers classifying the messages shared the opinion that the messages in the useful category were not all that descriptive. Some commit messages could be regarded as 'locally sensible', meaning that they could be useful for communication during a short time span, but offer not much for later inspection. Many of the commit messages are clear indicators that the students regard the system as an answer submission system. Examples include "Answer for week 12" and "exercise 4a". We also noticed some messages related to problems in using the git system. The amount was however not as significant as the teacher had expected. It is also observed that the quality of the messages is depending on the group, indicating that some groups use the messages for communicating, while others do not.

Conclusion

In this case study, we focused on students' usage patterns in advanced courses related to programming while using the distributed version control system Git. We first looked at when students commit their work during the course and in more detail at their committing pattern during the implementation sessions. We concluded that most students commit changes regularly during the implementation sessions, but do not commit changes of work which they have been doing before the session itself. Some groups commit rarely during the session and make a big commit at the end of the session. We did some effort in classifying commit messages and noticed that students do often write messages which are either trivial or even sheer nonsense. Further, we looked at how the usage of the system is divided inside groups and found out that the activity of the most active user in a group is significantly higher than what would be expected if each group member would use the system equally much. As the last part of our analysis we considered the types of files which students put under version control. We concluded that if students are not told explicitly that they should not include certain types of files, they will just do so.

Acknowledgement

We would like to thank our instructor for giving such case study as a project work. We have dig deeper into how the distributed systems work and version control systems works

and its workflow. And our University by supporting and encouraging to participate us in experiential learning without which this case study would not have possible.

References

1. Laadan, O., Nieh, J., Viennot, N.: Teaching operating systems using virtual appliances and distributed version control. In: Proceedings of the 41st ACM technical symposium on Computer science education. SIGCSE '10, New York, NY, ACM 480–484 (2010) 2.
2. Meneely, A., Williams, L.: On preparing students for distributed software development with a synchronous, collaborative development platform. In: Proceedings of the 40th ACM technical symposium on Computer science education. SIGCSE '09, New York, NY, ACM 529–533 (2009)
3. Mierle, K.B., Roweis, S.T., Wilson, G.V.: CVS data extraction and analysis: A case study. technical report utml tr 2004-002. Technical report (2004)
4. Reid, K.L., Wilson, G.V.: Learning by doing: Introducing version control as a way to manage student assignments. In: Proceedings of the 36th SIGCSE technical symposium on Computer science education. SIGCSE '05, New York, NY, ACM 272–276 (2005)
5. Clifton, C., Kaczmarczyk, L.C., Mrozek, M.: Subverting the fundamentals sequence: Using version control to enhance course management. SIGCSE Bull. 39(1) 86–90 (March 2007)
6. Hartness, K.T.N.: Eclipse and CVS for group projects. J. Comput. Sci. Coll. 21(4) 217–222 (April 2006)
7. Liu, Y., Stroulia, E., Wong, K., German, D.: Using CVS historical information to understand how students develop software. In: MRS 2004: International Workshop on Mining Software Repositories. (2004)
8. Lee, B.G., Chang, K.H., Narayanan, N.H.: An integrated approach to version control management in computer supported collaborative writing. In: Proceedings of the 36th annual Southeast regional conference. ACM-SE 36, New York, NY, ACM 34–43 (1998)
9. Glassy, L.: Using version control to observe student software development processes. J. Comput. Sci. Coll. 21(3) 99–106 (February 2006)
10. Xu, Z.: Using git to manage capstone software projects. 159–164 (2012)
11. Milentijevic, I., Ciric, V., Vojinovic, O.: Version control in project-based learning. Computers & Education 50(4) 1331–1338 (2008)
12. Rocco, D., Lloyd, W.: Distributed version control in the classroom. In: Proceedings of the 42nd ACM technical symposium on Computer science education. SIGCSE '11, New York, NY, ACM 637–642 (2011)
13. Tirronen, V., Isomöttönen, V.: Making teaching of programming learning-oriented and learner-directed. In: Proceedings of the 11th Koli Calling International Conference on Computing Education Research. Koli Calling '11, New York, NY, ACM 60–65 (2011)
14. Tirronen, V., Isomöttönen, V.: On the design of effective learning materials for supporting self-directed learning of programming. In: Proceedings of the 12th Koli Calling International Conference on Computing Education Research. Koli Calling '12, New York, NY, ACM 74–82 (2012)
15. Isomöttönen, V., Tirronen, V.: Teaching programming by emphasizing selfdirection: How did students react to active role required of them? ACM Transactions on Computing Education Research (accepted)
16. Isomöttönen, V., Tirronen, V., Cochez, M.: Issues with a course that emphasizes self-direction, submitted. (2013)
17. Parzen, E.: On estimation of a probability density function and mode. The annals of mathematical statistics 33(3) 1065–1076 (1962)