Using GitHub in the Classroom – a Collaborative Learning Experience

Csaba-Zoltán Kertész
Department of Electronics and Computers
Transilvania University
Brașov, Romania
csaba.kertesz@etc.unitbv.ro

***Abstract*—The most important skills employers are looking for in engineering are creativity, ability to work in teams and critical thinking. However the most used teaching methods at universities, based on individual assignments or group projects while do address the creativity, fall back on the critical thinking and even in team work skill. Collaborative learning was long suggested as a good alternative for enhancing these skills and recent evolution in social coding sites and collaborative software development give a new opportunity to employ such teaching method. This paper presents the results of a collaborative learning experiment carried out by using GitHub in laboratory assignments, where the main focus was on the direct interaction of the students on each other's learning process. Results were analyzed in comparison with traditional methods and also perspectives of enhancing the learning experience in the vision of current workforce requirements.**

***Keywords*—collaborative learning, version control, social coding, GitHub**

I. Introduction

Collaborative working environment has already a long history especially in software development but also in other fields of engineering. With the advent of social media, this collaborative development process was also heavily influenced giving birth to a series of social coding sites like GitHub, Bitbucket, GitLab, etc. Many open source projects are hosted by these sites, but also an increasing number of companies deploy their codebase on such platforms.

Many studies point out the benefits of using social coding, like the higher interaction between the developers leading to better code review, continuous learning and more efficient tracking of project evolution [1]. One of the most popular social coding sites is GitHub with over 11 million users and more than 28 million projects hosted[1], most of it being open source projects, but a strong commercial presence is also detected, many companies using it a cloud storage for their software development projects. [2] Most important aspect outlined in the studies is the collaborative nature of working on GitHub [3], [4], [5].

Using git or other version control systems in classrooms already proved some advantages for the students like better tracking of student evolution and also exposing students to standard industry tooling in software development [6]. Using cloud solutions instead of local repository servers is one natural choice for faculty as presented in [7].

Using different frameworks for collaborative learning have also been widely studied [8], [9]. It is generally thought that collaborative learning methods increase the critical thinking and the teamwork spirit [10], skills which are now heavily sought on in the industry. Most of the frameworks for these type of learning however do not consider collaborative framework used in industry.

Lately however, given the high impact in the industry, GitHub has also started emerging as a collaborative platform used in education, many teachers already employing it in the classroom in various scenarios [11]. To help this GitHub is offering an educational package[2], allowing a number of private repositories on request for GitHub organizations created by educational institutions.

In the following an experimental teaching method is presented which was carried out at the Department of Electronics and Computers, Transilvania University, for the Operating Systems laboratory, where GitHub was employed as a collaborative platform for students to do their homework and classroom assignments. The methodology employed is presented in detail as well as the results after the assessment of students at the end of semester and a survey on students' opinions about this collaborative method.

II. GitHub Workflow in the Class

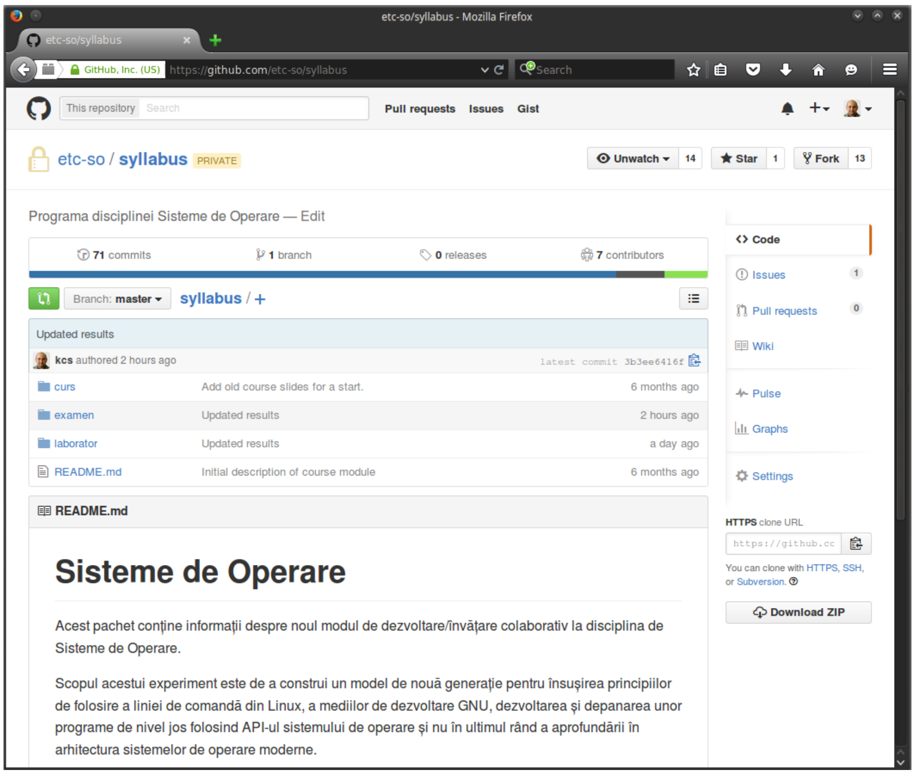
For the Operating Systems class a GitHub organization was created with the name *etc-so*, with one repository for the syllabus and several other repositories, one for each chapter in the laboratory textbook. Students were able to access these repositories for reading the teaching materials and checking out the examples.

They also solved their assignments by contributing back to these repositories in the form of examples. Version controlling via git and the pull-requests on GitHub were used in the process. They were also able to directly interact each other by raising issues and commenting on the repositories.

As an example the front page of the Syllabus repository is presented on Fig.1.

[1]https://github.com/about/press

[2]https://education.github.com/



Programa disciplinei Sisteme de Operare — Edit

71 commits 1 branch 0 releases 7 contributors

Branch: master syllabus / +

Updated results

kcs authored 2 hours ago latest commit 3b3ee6416f

curs	Add old course slides for a start.	6 months ago
examen	Updated results	2 hours ago
laborator	Updated results	a day ago
README.md	Initial description of course module	6 months ago

README.md

Sisteme de Operare

Acest pachet conține informații despre noul modul de dezvoltare/învățare colaborativ la disciplina de Sisteme de Operare.

Scopul acestui experiment este de a construi un model de nouă generație pentru însușirea principiilor de folosire a liniei de comandă din Linux, a mediilor de dezvoltare GNU, dezvoltarea și depanarea unor programe de nivel jos folosind API-ul sistemului de operare și nu în ultimul rând a aprofundării în arhitectura sistemelor de operare moderne.

Fig. 1. Home screen of the Operating Systems course Syllabus on GitHub

A. *Git usage*

Because students are already required to work mostly in Linux command line as the main mean of executing their laboratory assignments (either bash scripting or compiling/running Linux console applications), a pure command line git usage was also favored for the GitHub interaction.

To access the laboratory assignment each student forked the given laboratory repository on GitHub and cloned their own fork locally with the command:

```
git clone https://github.com/username/reponame
```

obviously with the student's username and the repository name replaced accordingly. This command creates a local folder with the same name as the repository and copies there all the currently available file from the repository.

Every repository contains two folders, one for the examples from the laboratory textbook and one for exercises, where the student can create their own application.

After creating a new file or modifying an existing exercise, it can be committed into version control with the commands:

```
git add filename
git commit
```

the *add* command stages the new file, while *commit* inserts it into version control. This two-step mechanism of adding new code into version control may appear cumbersome but it has some advantages when working with large projects. For small laboratory exercises it also proved to be advantageous as without prior knowledge of other version control systems, the students got used to the local staging concept and the possibility for one more step to review exactly what should be committed or not.

After committing to the local repository the changes have to be pushed to GitHub with the command:

```
git push
```

after which the commit also appears on the student's private fork, giving the opportunity to create a new pull-request to the original repository. After the changes were also accepted by the teacher, they were integrated into the main repository and all the student could synchronize their local copies with the commands:

```
git remote add upstream http://github.com/etc-
    so/reponame
git fetch upstream
git checkout master
git merge upstream/master
```

the first command has to be issued only once in a local repository, and it will make a link to the original repository. After that changes from there can be retrieved with the *fetch* command and the local modifications merged together with those changes. The checkout of master branch was necessary because it was recommended to the students to create local branches:

```
git branch newbranch
git checkout newbranch
```

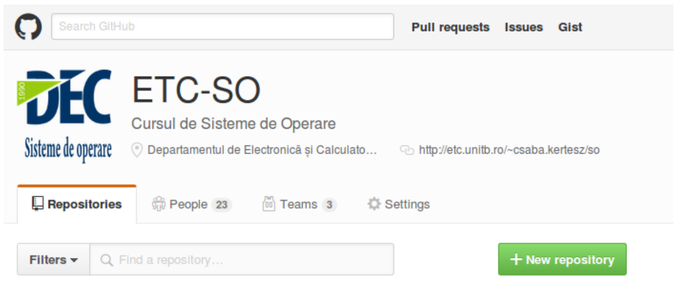


Fig. 2. Page header of *etc-so* organization

so all their changes for a single task end up in a separate branch and be committed to GitHub also on new branch with the command:

```
git push -u origin newbranch
```

This form of pushing also needed to be issued only once while on the branch, subsequent commits can be pushed simply without any arguments.

Separate branches on the GitHub repository for every single task allowed the students to create pull-requests for their solution and, while it was checked by the teacher, to start working on some other tasks simply by creating new branches.

B. GitHub usage

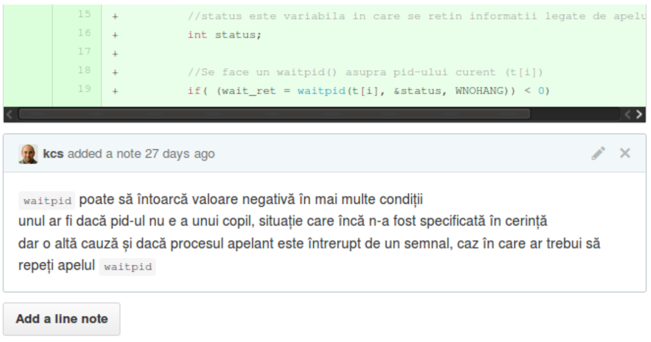
The GitHub educational program is working to the best if new organizations are created for each class. Any faculty member can create organizations and request entering the educational program. The request is usually accepted within a week so this must be taken into account when creating it for a new class.

The organization can be set up to have a logo and some descriptive text as it is presented on Fig.2. The same page is also used for creating new repositories and managing peoples and teams.

Students were added to the repository at the first meeting after they created their own GitHub account, using the people management tab by simply inserting their username. Two teams were created to micromanage the students' permissions: one team with read/write permission and one team with read-only permission. A third team was reserved for the teachers with administrative persmission.

This permission management was necessary to allow the students a full exploration of the GitHub world and working with git. An empty repository called *scratchpad* was specifically created for this purpose with full read-write permission to all students. During the first classes the usage of git and GitHub was demonstrated on this repository with the students being able to push directly into it, fork it, create pull-requests which were integrated by each other and also experiment with advanced git features. This proved to be useful because of the initial higher learning curve on software versioning.

All other repositories were accessible read-only for the students so teachers could have a control on their submitted work. Obviously the *Syllabus* repository was also read-only



```
+           //status este variabila in care se retin informatii legate de apelu
+           int status;
+
+           //Se face un waitpid() asupra pid-ului curent (t[i])
+           if( (wait_ret = waitpid(t[i], &status, WNOHANG)) < 0)
```

kcs added a note 27 days ago

`waitpid` poate să întoarcă valoare negativă în mai multe condiții
unul ar fi dacă pid-ul nu e a unui copil, situație care încă n-a fost specificată în cerință
dar o altă cauză și dacă procesul apelant este întrerupt de un semnal, caz în care ar trebui să repeți apelul `waitpid`

Fig. 3. Adding a line note to a commit

containing all the laboratory textbooks. These were added in their original electronic form from the existing learning platform, mostly in pdf files, which were rendered viewable online by the GitHub engine. However additional learning material was added using GitHub flavored markdown text[3]. This markdown allows for simple formatting of text documents which in turn are nicely rendered by GitHub on the online platform.

The markdown syntax is used throughout the GitHub pages, in comments, issues, pull-requests and due to it's easy to use nature it become quite popular among students.

The most important part of activity from the point of view of the laboratory assignments is undoubtedly the pull-request interface of GitHub. Every student made private fork of the repository of the laboratory chapter they were working on and cloned locally to their machines. After completing an exercise they pushed back their work to the forked repository and on the GitHub page of that fork they created a new pull-request towards the master branch on the original repository.

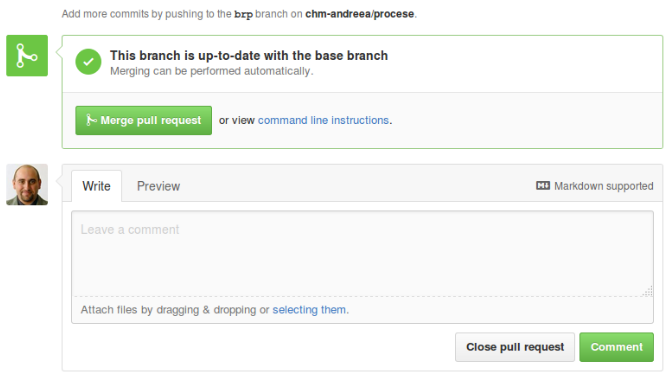
The teacher due to the writing permission on the original repository got an email notification each time a pull-request was sent so he could review the code quickly after the creation of the pull-request. The code review took place on 2 levels:

- GitHub offers a nice diff view of the changed code which can be studied online right at the pull-request page. Usual programming languages are also highlighted making it very easy to look out for obvious programming errors
- the committed files were fetched locally to the teacher computer and compiled/ran to check out conformance to the functionality requirements.

The online diff viewer of GitHub also makes possible to add line-wise comments wherever errors were found (Fig.3). This proved to be a handy feature as initially students' code did have a lot of good practice issues (like not handling returned error codes) which did strike out fast and directly commenting on the offending lines offered a very fast feedback and interaction between the students and the teacher.

Checking out the functionality of the student's work requires some local work. This was done mostly by cloning the forked

[3]https://help.github.com/articles/github-flavored-markdown


Add more commits by pushing to the `brp` branch on **chm-andreea/procese**.

This branch is up-to-date with the base branch
Merging can be performed automatically.

or view command line instructions.

Leave a comment

Attach files by dragging & dropping or selecting them.

Fig. 4. Commenting and merging a pull-request

repository for which the teacher had read access (all forked repositories of the private organization repository had this feature) and could run the program locally.

If all the code was acceptable and local running also proved satisfactory, the pull-request was merged into the master branch. This was accomplished through the GitHub interface which offers a straightforward commenting and merging possibility on each pull-request (Fig.4). Commenting on the whole pull-request was also used when observations on the whole code was to be made. Also if the pull-request was rejected it was explained in detail the motives behind the rejection. This usually happened if the code was not matching the requirements or it was already submitted by others.

A couple of time some amendments from the teacher was necessary in order to point out good code practice. In such cases a local clone of the original repository was used on the teacher's computers and the changes from the student's fork were added with the command:

```
git fetch https://github.com/student/reponame
git checkout branchname
```

and after some local amendments it was merged back to the master

```
git commit -a
git checkout master
git merge branchname
git push
```

pushing the local changes containing the student's commit from the pull-request to the master branch on the GitHub repository also automatically closed the pull-request.

The other GitHub feature mainly used in the laboratory work was the issue tracking interface. Although its original intent was to point out to the program author any issues found in the programs or to request new features to be implemented, this interface was used mostly to create new laboratory assignments for the students by the teacher and for the students to ask any questions.

Initially the teacher created a series of issues with different exercises and assignments to the students. Issues could be made general, on which any student could work and also there is a possibility to assign certain issues individually to a student.

Every issue automatically got a number on GitHub, which could be referred later on from comments, making possible linking discussion to these issues. To achieve this the markdown sequence of `#1` was used with the number representing the number of issue in the GitHub repository. This made it easy to cross-reference issues from pull-requests as well. Because most of the pull-requests were made to solve such issues this linking made easy to track which student solved their assignment.

Issue numbers inserted into the commit message together with a keyword like *close*, *fix* or *resolve* also closed the issue together with merging the pull-request into the original repository.

Another use of the issue tracking was to add the possibility for students to ask questions, create new tasks for fellow students (for example by breaking assignments into smaller tasks) and to report problems found in laboratory exercises. This issues were monitored by the teacher assigning different tags to them. These colorful tags marking the issue as: *bug*, *feature request*, *question*, *help wanted*, *duplicate* or *invalid*. Later these tags served in the assessment of student activity on GitHub.

III. Assessment of Student Activity

Final assessment of the students was done based on their activity on GitHub. This assessment included not just the quality of code committed as their homework, but also contributions on other pull-requests, commenting fellow students' code, pull-requests and issues.

To check out the students' activities a Python script was created which accesses the GitHub API[4]. The API can be accessed with a simple *requests* object to either the *issues*, *pulls* or *commits* databases with optional filtering and returns a JSON object containing all available information, like author, date, tags, status, etc.

An example script for getting such list is presented below:

```
api_url = "https://api.github.com/"
get_url = "repos/{}/{}/{}"
def github_get(get_code, **params):
   res = []
   url = api_url + get_url.format(org, repo,
       get_code)
   while url:
      r = requests.get(url, auth=auth, params=
          params)
      if r.status_code != 200:
         print('Status␣code␣{}␣received'.
             format(pr.status_code))
         print(pr.content)
         sys.exit(1)
      links = dict((m[1],m[0]) for m in
          link_rel.findall(r.headers.get('link'
          , '')))
      url = links.get('next')
      res.extend(r.json())
   return res
```

[4]https://developer.github.com/v3/

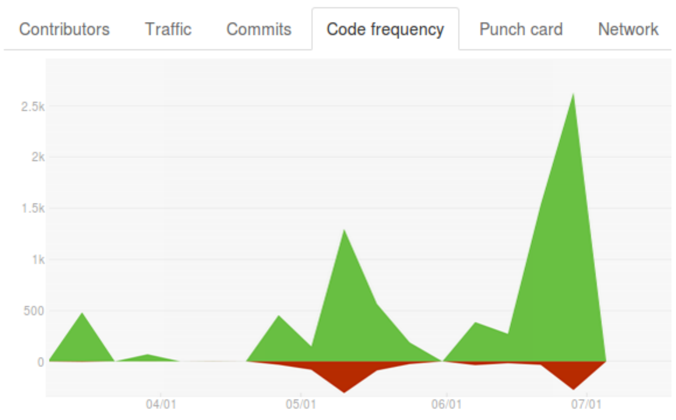


Fig. 5. Code frequency graph

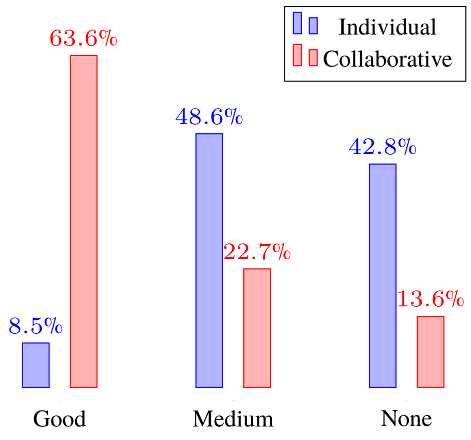


Fig. 6. Comparison of student homework quality between the traditional individual assignment and collaborative work on GitHub

the result being a Python dictionary representing the JSON object. This dictionary was further parsed to extract all comments made by the students, the merged pull-requests and any valid issue raised which was subsequently solved.

The actual assessment for each student was a subjective one based on analyzing the code submitted and offering bonuses for quality comments and good issues. This analysis had to be done manually by passing through all comments and issues and filtering out those unrelated to the laboratory. Being a social coding site most of the comments were alike other social media sites, with many of them thanking for answers or supporting some questions. These while do indicate interaction were left out from the final consideration and only those comments which pointed out valuable pieces of information were counted for.

The results of these assessment were continuously made available to the students during the semester to reinforce their implication in the collaborative style of work.

Indeed the high learning curves of all technology involved like version control and using remote repositories showed an initial low activity which steadily increased towards the end of the semester, and example for this is shown on the code frequency graph offered by GitHub for one of the repositories (Fig.5).

By the end of the semester due to the continuous reinforcement most students got used to the collaborative nature of the laboratory work, adding ever more comments, sometimes giving feedback much faster than the teacher to pending pull-requests.

This instant feedback on code (either from the teacher or from colleagues) also led to an increase of code quality in the homework of the students. Compared to previous years when exercises were solved individually as homework assignment and evaluated during laboratory sessions, the code proved to be much better, leading to better grading for the homework part.

A pleasant surprise was that after a while, when students had already increased their collaborative style, they started interacting also with the teaching material. First by pointing out errors and later offering new materials via pull-requests to the syllabus.

Looking back at Fig.1 taken at the end of the semester one can easily see that a repository primarily intended to hold the teaching materials and a weekly report on progress by the students and as such with only a few prospective commits initially, ended up with 7 contributors and 71 commits. About one third of these commits can be accorded directly to the students contributing by pull-requests and many others to solving issues reported by the students. New additions to the teaching material revolved especially around using git and GitHub, some of the student not being happy with only the bare necessaries for the laboratory, they also found some case where more advanced use of git was needed and also documented it. Other types of additions were scripts (mostly bash scripts) which made laboratory work much easier. The teacher considered these contributions as the highest achievement of collaborative style learning and contributing students were graded accordingly.

Overall by the end of the semester 268 commits were pushed to the main repository by merging 138 pull-requests and 228 issues were reported. This lead to an average of higher student activity than in previous years with individual homeworks. Also to be noted the hidden benefit of full accountability of each student's contribution. While in previous years many students turned in similar of effectively the same solution to their homework assignments making it hard to determine their actual contribution if any, using shared version control it was much easier to attribute every line of source code to a student. As a personal opinion, the teacher considers a single line of source code where a students corrects a fellow student's mistake just as valuable as a student turning in a complete solution using unverifiable sources (also possibly copying from each other).

While the group of students involved in this learning method (22 in total) is a statistically irrelevant sample to compare with traditional results, a comparison is made on Fig.6 between the ratio of students with good laboratory activity, some activity not sufficient for passing the laboratory and with absolutely no activity versus the activity and quality of homeworks turned in in the previous years using traditional methods. Neither the sampling, neither the comparison between different groups of students and different generation is useful to take a conclusion, however the trends are showing encouraging. A note of precaution when interpreting this comparison, the high jump in good quality code is mostly because of the online feedback on pull-requests resulting in revised code before it was merged, while individual homework solving has direct feedback only at evaluation.

IV. Survey Results

At the end of semester the students were surveyed about their opinion of the use of GitHub. The survey contained open question about the preference for collaborative platforms against individual homework assignments and about the advantages/disadvantages of both methods. The survey was anonymous, however a correlation between the activity level on GitHub was checked.

15 out of 22 students preferred the collaborative platform, pointing out some advantages they observed during their activity:

- learning from each others faults
- getting help from colleagues much faster, even if they are at distance
- learning a development platform commonly used in industry
- making them aware of different solutions to the same problem leading to better decisions on how to implement certain tasks.

On the downside almost everyone noticed the higher learning curve, meaning more work had to be invested into learning both version controlling and GitHub platform and also focusing on the assignments from the class. Another disadvantage they noticed was the relative difference between them in picking up the new platform and methodology leading to some discrepancies between their activity comparing to each other.

There was a strong correlation between the students dissatisfied with collaborative approach and the lack of any activity during the semester. Knowing the high rate of failing to solve any homework in previous years it is inconclusive whether these students would have performed better with individual assignments.

However there were some valid points in the difficulties pointed out by these students: the lack of any prior knowledge in version control or programming in general and not being used to such teaching method is a strong deterrent. Increasing the occurrence of collaborative methods would definitely increase the effectiveness of it.

V. Conclusion

A collaborative learning method was implemented successfully for the Operating Systems laboratory using GitHub as collaborative framework. The methodology and the involved software tools were completely new to the students, and while having initial difficulties in accommodating with it, the end results showed improved progress with both the given laboratory assignment and ancillary competences.

The students formed a more critical approach to software developed by them and their colleagues and even to the teaching materials, helping improving not just their own software skills but the overall group learning. These skills together with the inherent group working skills form an added advantage to the bare software knowledge and competences offered by the laboratory by itself.

References

[1] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, “Social coding in github: Transparency and collaboration in an open software repository,” in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, ser. CSCW ’12, Seattle, Washington, USA, 2012, pp. 1277–1286.

[2] K. Peterson, “The github open source development process,” May 2013.

[3] A. Lima, L. Rossi, and M. Musolesi, “Coding together at scale: Github as a collaborative social network,” in *Proc. of ICWSM*, Ann Arbor, USA, 2014.

[4] Y. Yoshikawa, T. Iwata, and H. Sawada, “Collaboration on social media: Analyzing successful projects on social coding,” *arXiv preprint arXiv:1408.6012*, 2014.

[5] E. Kalliamvakou, D. Damian, K. Blincoe, L. Singer, and D. German, “Open source-style collaborative development practices in commercial projects using github,” in *Proceedings of the 37th International Conference on Software Engineering (ICSE’15)*, Florence, Italy, 2015.

[6] J. Kelleher, “Employing git in the classroom,” in *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*, Hammamet, Tunisia, Jan 2014, pp. 1–4.

[7] J. Lawrance, S. Jung, and C. Wiseman, “Git on the cloud in the classroom,” in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE ’13, Denver, Colorado, USA, 2013, pp. 639–644.

[8] N. Mora, S. Caballe, T. Daradoumis, D. Ganan, and L. Barolli, “Providing cognitive and social networking assessment to virtualized collaborative learning in engineering courses,” in *Intelligent Networking and Collaborative Systems (INCoS), 2014 International Conference on*, Salerno, Sept 2014, pp. 463–468.

[9] R. Zhao and C. Zhang, “A framework for collaborative learning system based on knowledge management,” in *Education Technology and Computer Science, 2009. ETCS ’09. First International Workshop on*, vol. 1, Wuhan,Hubei, March 2009, pp. 733–736.

[10] A. A. Gokhale, “Collaborative learning enhances critical thinking,” *Journal of Technology Education*, vol. 7, no. 1, Fall 1995.

[11] A. Zagalsky, J. Feliciano, M.-A. Storey, Y. Zhao, and W. Wang, “The emergence of github as a collaborative platform for education,” in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 2015, pp. 1906–1917.