DOI: 10.28934/jwee21.34.pp113-133

ORIGINAL SCIENTIFIC PAPER

Exploring Possibilities of Integrating Version Control Platforms in Higher Education Through GitHub Data Analysis

Ana Milovanović [1]
University of Belgrade, Faculty of Organizational Sciences, Belgrade, Serbia
Danijela Stojanović [2]
Institute of Economic Sciences, Belgrade, Serbia
Dušan Barać [3]
University of Belgrade, Faculty of Organizational Sciences, Department of E-business, Belgrade, Serbia

A B S T R A C T

Working in the software development industry in the modern world cannot be imagined without integration with version control platforms. In addition to allowing copies of the code to be kept, the code versioning platforms provide the ability to control changes that occur in the code itself, so it can be determined which member of the development team made the change and when. This article aims to determine the possibilities of using code versioning platforms in an educational context to provide insights into the students' collaboration and dynamics of developing a software solution. Data was collected on the GitHub platform using the API of a public GitHub organization. Social coding activities performed on the GitHub platform were analyzed using the concepts of Social Network Analysis (SNA). The results of the analysis indicate that the teachers can use data generated on the code versioning platforms o better understand the

[1] Jove Ilića 154, 11000 Belgrade, Serbia, e-mail: am20205027@student.fon.bg.ac.rs

[2] Corresponding author, address: Zmaj Jovina 12, 11000 Belgrade, Serbia, e-mail: danijela.stojanovic@ien.bg.ac.rs, tel. +381 641380905

[3] Address: Jove Ilića 154, 11000 Belgrade, Serbia, e-mail: dusan@elab.rs

learning process and monitor the dynamics of project work. In this way, it is possible to follow how each team member contributes to the solution, understand the individual student activities, and monitor the state of the entire project.

KEY WORDS: *GitHub, SNA, education, version control, software development*

Introduction

One of the challenges that teachers are facing while conducting software engineering courses is monitoring students and their efforts while they work on developing software solutions. One approach to monitoring the collaboration and the results of software projects developed in teams of students relies on the integration of code versioning systems into the teaching flow. One of the most frequently used code versioning systems is Git, while the GitHub platform has already been used in software engineering courses (Calatrava Arroyo et al., 2020; Beckman et al., 2021). Further, GitHub enables gathering data for learning analytic through an API (Application Programming Interface), giving teachers insights into individual and collaborative performance. The collected data can be programmatically analyzed to investigate the possibility of applying the results to monitor the students' progress and collaboration.

El Mezouar et al. (2019) has already shown that SNA can be applied to GitHub data. In their analysis, by using the concept of the Pull-based network presented as a directed and weighted graph, network metrics were used to identify existing development team structures in 7,850 most popular projects on GitHub. Therefore, it is expected that SNA can be used to analyze GitHub data in education and provide data valuable insights into the learning process.

The goal of this research is to provide evidence and a concrete example of how the concepts of social networks analysis can be applied for the analysis of GitHub data to improve the educational process in the field of software development.

Version Control Platforms and SNA

Version control represents a system that tracks changes to user files. Files could be documents, layouts, images, files containing source code, etc (Chacon & Straub, 2020). Version Control System (VCS) allows users to control versions of a single file, multiple files, or an entire software project

by monitoring changes, as well as restoring them to one of the previous versions (Chacon & Straub, 2020).

The basic functionality of Git is related to version control. Each time a user commits or saves the state of their project, Git takes a snapshot of the entire file system and keeps a reference to that snapshot. If the files have not changed, Git will not save and store them again but create the reference to the files that have already been saved, thus showing the efficiency (Chacon & Straub, 2020).

Git is based on git commands for various operations (Somasundaram, 2013). The commands that are often used are (Blischak et al., 2016):

- pull: for retrieving commits from a remote repository and merging with a local repository,
- push: for sending commits from a local to a remote repository,
- merge: for updating files to include changes made in the new commits,
- stage: for preparation of files to be covered in the next commit and
- commit: for saving all changes made to staged files.

The Git repository is a database that contains all the information about the project and keeps a copy of it throughout its lifetime (Loeliger & McCullough, 2012).

GitHub is a platform through which users can upload an online copy of their Git repository. Hence, review, control, and cooperation with other users on a project are facilitated (Beer, 2018).

AlMarzouq et al. (2020) classify GitHub as a cloud platform for hosting FLOSS community software (free and open-source software). Also, they point out that researchers can observe the collaborative behavior of developers using GitHub. Git itself can provide data on changes in the source code, which lines of code have been modified, when and by whom. To get the metadata, Git commands are used: git log, git blame, and git diff. However, as stated by AlMarzouq et al. (2020), Git does not provide data on social coding interactions.

GitHub uses Git functionality and supports tools that turn coding into a social activity (AlMarzouq et al., 2020). Users represent sources of social coding content. They can be organizations or individuals. The social coding functionalities that can be found on the GitHub platform and which are listed in (AlMarzouq et al., 2020) are repositories where the shared code is located, pull requests, starring repositories, watching repositories, following

users, comments, discussions, bug reports, notifications, as well as code copy operations such as branching, forking, and cloning repositories.

Many of the aforementioned social coding functionalities that are characteristic of the GitHub platform can be observed. Based on that, it is concluded that SNA, i.e. social networks analysis, can be carried out over different entities that can be represented using graph theory, also known as network theory (Camacho et al., 2020), as nodes or relations of graphs (Jamali & Abolhassani, 2006).

Literature Review

There are numerous scientific papers on the improvement of the learning process in higher education through the application of modern information technologies and collaborative forms of learning (Radović-Marković et al., 2009; Bjelica & Pavlović, 2018; Stojanović et al., 2020; Slavinski et al., 2020; Radović-Marković et al., 2021; Stojanović & Domazet, 2020). Moreover, there are numerous research papers and experiments where the integration of a code versioning platform into the teaching flow has been tried and successfully implemented. One of the studies was conducted at the Universitat Politècnica de València (UPV) from 2019 through 2020, where 28 students in the Electronic and Automatic Engineering Degree (EAED) participated (Calatrava Arroyo et al., 2020). A new solution called ACTaaS (Assessment of Computational Thinking as a Service) was developed and applied while conducting the Computer Science course. The solution is based on Cloud technology. Code versioning, automatic testing of the code, as well as continuous integration and continuous deployment (CI/CD) tools, were used. Among other things, the solution included configuring Jenkins, creating a GitHub organization, and creating tasks in the GitHub Classroom environment that is part of the GitHub Education program. The GitHub Classroom assignment may contain an initial set of code with instructions.

The invitations to accept assignments were available to students through an LMS (Learning Management System) called Sakai. After accepting the assignment, a copy of the repository is created on the student's account. Task progress analysis is performed automatically by calling Jenkins unit tests after students push their code into their repository (Calatrava Arroyo et al., 2020).

The university courses conducted by Beckman et al. (2021) covered the study program in the field of Data Science. To begin with, the lecturers created GitHub Pro organizations (GitHub Education) where they added students as members of the organizations. In 2019, while teaching Introduction to R Programming course at Penn State University, lecturers began using the GitHub Classroom. The integration with the GitHub platform is similar to the one described in (Calatrava Arroyo et al., 2020). A Git repository has been created containing the initial setup where source code, instructions, grading tables, and documentation can be included (Beckman et al., 2021).

The study concludes that students quickly master the basics of code versioning and individual tasks, but that creating group tasks, which can also be created via the GitHub Classroom platform, is difficult due to conflicts when merging code. Numerous students solved the problem by working on one computer showing that collaboration through GitHub was challenging for students who had just started working in the collaborative environment (Beckman et al., 2021). To track student activity on a joint project, many instructors used each student's commit history. The authors point out that although the number of commits itself does not indicate the quality of the code, the lack of commits may indicate that the student did not participate in the work. The authors further suggest a combination of peer evaluation with commit history to obtain a complete picture (Beckman et al., 2021).

The study further addresses the issue of choosing the appropriate code versioning platform. The three most popular (Beckman et al., 2021) stand out: GitHub, GitLab, and Bitbucket. The reasons for choosing GitHub are the fact that it is used by most companies and organizations, rich API, and tools like GitHub Classroom.

In the study described in (Feliciano et al., 2016), it was decided that GitHub course materials and student papers would be publicly available, unlike the previous two cases (Calatrava Arroyo et al., 2020; Beckman et al., 2021). The advantages of the approach are the interactions and suggestions of people who were not working on the project. However, a group of the students did not consider the work on the repository to be good enough to be publicly available, and there were also concerns that such work was not of interest to the community (Feliciano et al., 2016). Although course materials were publicly available, Moodle LMS was used for

additional materials, such as grades and additional courses (Feliciano et al., 2016).

Zagalsky et al. (2015) point out an important difference between the GitHub platform and LMS systems such as Moodle and Sakai. GitHub was used as a Submission platform and as a way to host course content, features that were otherwise available in standard LMS systems. The difference was in the number of interactions supported by the GitHub platform. While traditional systems support reading and access to materials, GitHub provides the opportunity for active participation (Zagalsky et al., 2015).

There are different workflows that GitHub can support, and they differ in which users have a push privilege in the repository, i.e. which users can directly make changes to the repository code, and which must require permission to do so. The latter must fork a repository, work on the copy created on their account, and if they want the changes to be included in the main repository code, they must create a pull request in the main repository (AlMarzouq et al., 2020).

Pull request stands out as a central functionality that enables social coding. Changes to the code can be included (merge) in the repository code only after the owners of that repository, or users authorized to do so, have approved changes via the GitHub platform (AlMarzouq et al., 2020). Further communication takes place through comments resulting from the code review action performed by contributors involved as reviewers (Open source guide 2021a).

El Mezouar et al. (2019) investigate in detail the structure of teams that base their work on a pull-based development model. They researched the 7,850 most popular projects on GitHub. Investigating the efficiency of the procedure of reviewing pull requests, it was determined that the efficiency depends on technical, but also social factors. From the technical factors, the quality of the code stands out, while from the social ones, the connection that the project contributor has with the programmer who has the role of integrator and who is in charge of project maintenance can be singled out (El Mezouar et al., 2019).

The concept of pull-based networks is used when two programmers are connected when one of them integrates a pull request (integrator) created by the other (contributor) (El Mezouar et al., 2019). The Pull-based network is presented as a directed and weighted graph, and network metrics can be used to identify existing development team structures. The three most influential network metrics have been singled out that have the following

interpretation in the context of the pull-based model (El Mezouar et al., 2019):

1. Out-degree centralization measures the importance of contributors to the level of activity. The number of created pull requests can be taken as an example of activity levels. A high level of the beforementioned metric indicates that there are base contributors, and a low level indicates that all contributors participate equally.
2. Density indicates the connection of team members. If the density is higher, the team members are more strongly connected, i.e. have previous interactions with most other team members.
3. Reciprocity indicates the probability that the developer has both roles in the project, both the integrator and the collaborator. The high value of the beforementioned metric indicates that the programmer is more likely to have both roles.

The relationship between a collection of network metrics derived from pull-based networks (e.g., reciprocity, centralization, etc.) and a collection of performance metrics (e.g., response time) can be analyzed using linear regression. The results of previous studies have shown that network metrics can partly explain performance metrics (El Mezouar et al., 2019). Over one-third of the teams are dominated by a loosely connected structure where members have one role (single-role developers). It was found that the better-ranked teams were the more interconnected ones. There was a centralization of the team around key contributors and developers could be found both in the role of integrator and in the role of contributors. Thus, the existence of central programmers having both roles was associated with the closure of more pull requests (El Mezouar et al., 2019).

Methodology

There are various ways to collect data from the GitHub platform. In addition to the GitHub API (Application Programming Interface) analyzed in this paper, there are external repositories that take snapshots of data that the GitHub API has access to, which allows the user to freely search the data without the API's limitations. Examples of external repositories are GHTorrent and GHArchive (AlMarzouq et al., 2020).

The two stable versions of the GitHub API that are publicly available are: the GitHub REST API and the GraphQL API. For the research, the GitHub REST API will be examined. Further analysis will aim to

investigate the possibility of applying the results that the API returns to monitoring the work of students working together on the project.

The Axios API client was applied to obtain the data. Appropriate graphs were illustrated using the React JS library and React Google Charts data visualization library. Also, the Postman tool was used for endpoint calls testing. The tool is used in industry and education in various phases of API development and testing.

Endpoints provide real-time data on how to use the GitHub platform. In addition to different data on the number of commits and pull requests related to repositories, as well as the number of bytes of code written in different programming languages within the repository, user data can be obtained at different time intervals. By combining different calls made to the GitHub API and the data obtained in the response, different conclusions can be generated that are important for understanding the dynamics of joint work.

The limiting factor is the rate limit of 5000 requests per hour for an authenticated user, while for a non-authenticated user the number is significantly lower and amounts to 60 requests per hour. In the second case, the requests are related to the IP address and not to the user (GitHub docs 2021b).

As an example of a GitHub organization, the public GitHub organization Contentful was used to collect the data returned by the GitHub API, which is needed for further analysis.

Research Results

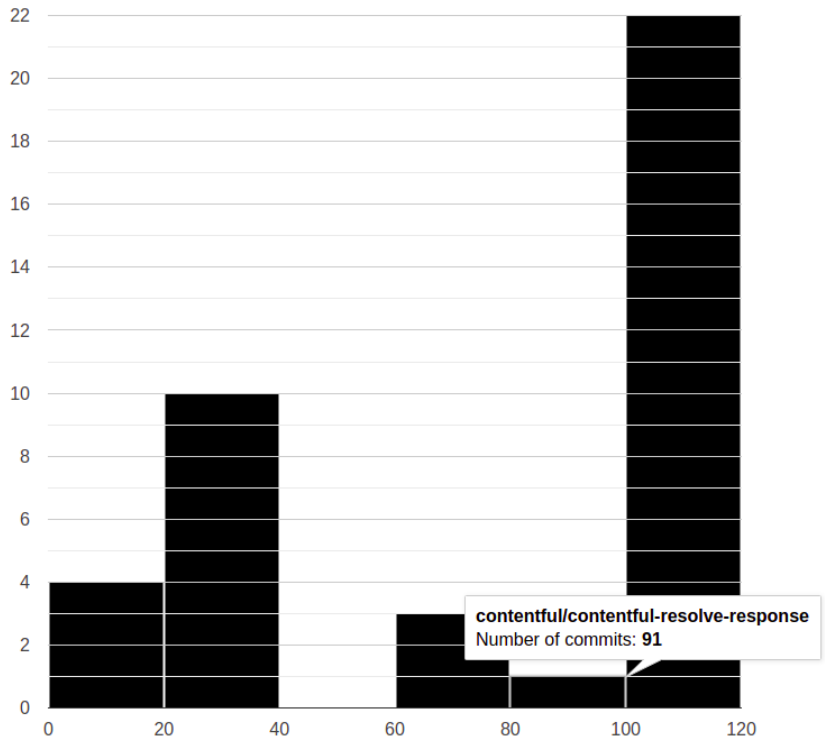
On a sample of 40 repositories, the Commits endpoint is called. The endpoint returns data on the commit of a particular repository, with the maximum number of commits that can be obtained by a single call to the endpoint being 100. By further processing of the data, all commits were counted according to the repository to which they belong, and the data thus obtained was used to make a histogram from the image below (Figure 1). The range of the number of commits is shown on the x-axis, and the number of repositories belonging to a particular range is shown on the y-axis.

Of the 40 repositories considered, four repositories have a range of 0-20 commits, 10 repositories have a range between 20-40 commits, and three repositories are in the range of 60-80 commits. The repository named *contentful-resolve-response* has 91 commits (Figure 1) and is the only one

in the range of 80-100 commits. The remaining 22 repositories have 100 or more commits. One of them is the *form-36* repository, which is analyzed in more detail.

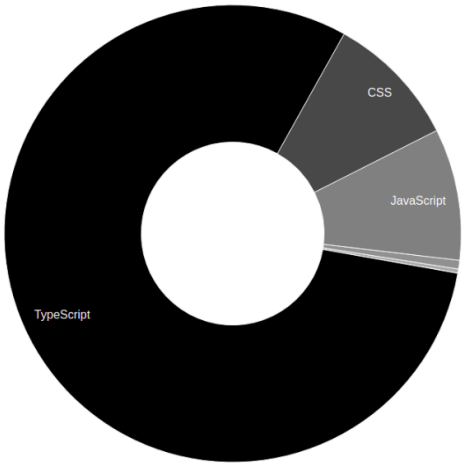
One way to examine the structure of the repository itself is to determine which programming languages are used. Calling the Languages endpoint gives a JSON object where the key is the name of the language, and the value is the number of bytes of code written in that language. For example, a repository named *form-36* was selected. The data obtained as output parameters are shown using a Pie chart. From the Figure 2, it can be seen that the most represented languages in the selected repository are: TypeScript (736,407 bytes, 80.3%), followed by CSS (88,672 bytes, 9.5%), JavaScript (85,377 bytes, 9.3%), SCSS (5,644 bytes, 0.6%) and Handlebars (2536 bytes, 0.3%). Shell (96 bytes) and HTML (71 bytes) are represented in a negligibly small percentage.

Figure 1: The sector of the business operations



Source: Authors

Figure 2: Languages distribution in the repository



Source: Authors

To monitor the activity of participants in group work on a repository, the Contributors endpoint can be called. Form-36 was again selected as the example repository. The endpoint returns Contributors sorted by the number of commits per contributor in descending order. As GitHub identifies collaborators by the author's email address, the endpoint groups collaboration numbers by GitHub user (GitHub docs 2021c).

There are currently a total of 60 collaborators in the observed repository. From the results obtained after the call of the endpoint, a tabular presentation was made containing three columns (Figure 3). The first column of *Username* contains the GitHub username of the contributor, which is applied to the generic name *user-number* to protect the privacy of the users. The second column, the *Number of commits*, contains the number of commits per contributor. The third column *Active user* (Figure 3) has a true/false value that is the result of a condition, which is added by application processing of the data. The value of the variable, i.e. the symbol in the table changes depending on whether the contributor has more than 10 commits or not. If the value is less than 10, the symbol "x" will appear showing that the user is less active (Figure 3).

The number of 10 commits was chosen to highlight those contributors who contribute the most to the solution. The number depends on the sample to which the check is applied. Figure 3 shows a sample of 30 contributors that are important for further analysis.

As the number of commits is often not a good indicator of contributor activity, and to obtain time-dependent data, another endpoint (Stats endpoint) concerning contributors will be tested using the same repository as in the previous case.

The endpoint also returns the total number of commits for each contributor and the Weekly Hash that contains the following information (GitHub docs 2021c):

- w – a Unix timestamp representing the beginning of the week
- a – the number of added or modified lines of code (Additions)
- d – the number of deleted lines (Deletions)
- c – the number of commits

Figure 1: Number of commits per Contributor

	Username	Number of commits	Active user
1	user-1	374	✔
2	user-2	239	✔
3	user-3	153	✔
4	user-4	143	✔
5	user-5	82	✔
6	user-6	60	✔
7	user-7	55	✔
8	user-8	53	✔
9	user-9	40	✔
10	user-10	38	✔
11	user-11	23	✔
12	user-12	14	✔
13	user-13	11	✔
14	user-14	11	✔
15	user-15	9	✗
16	user-16	9	✗
17	user-17	8	✗
18	user-18	6	✗
19	user-19	6	✗
20	user-20	6	✗
21	user-21	6	✗
22	user-22	6	✗
23	user-23	5	✗
24	user-24	5	✗
25	user-25	5	✗
26	user-26	5	✗
27	user-27	5	✗
28	user-28	5	✗
29	user-29	4	✗
30	user-30	4	✗

Source: Authors

From several values returned by the endpoint, one contributor was randomly selected. The analysis of the contributor's Weekly Hash is further approached. If in seven days, starting from the day representing the beginning of the week (mark w), there were no commits (c has a value of zero), the marks for the number of added and deleted lines also have a value of zero. To preview data more clearly, zero values have been removed from the Weekly Hash, leaving only those that have relevant data for further analysis.

Figure 4 illustrates the values that the Weekly Hash has for the selected contributor. The columns contain the values: w, c, a, and d, respectively. If we take into account the first row from Figure 4, the interpretation of the data obtained is as follows: in the week that started on December 2, 2018. the contributor made one commit in which the number of added or changed lines of code was equal to 109 and the number of deleted lines was 492.

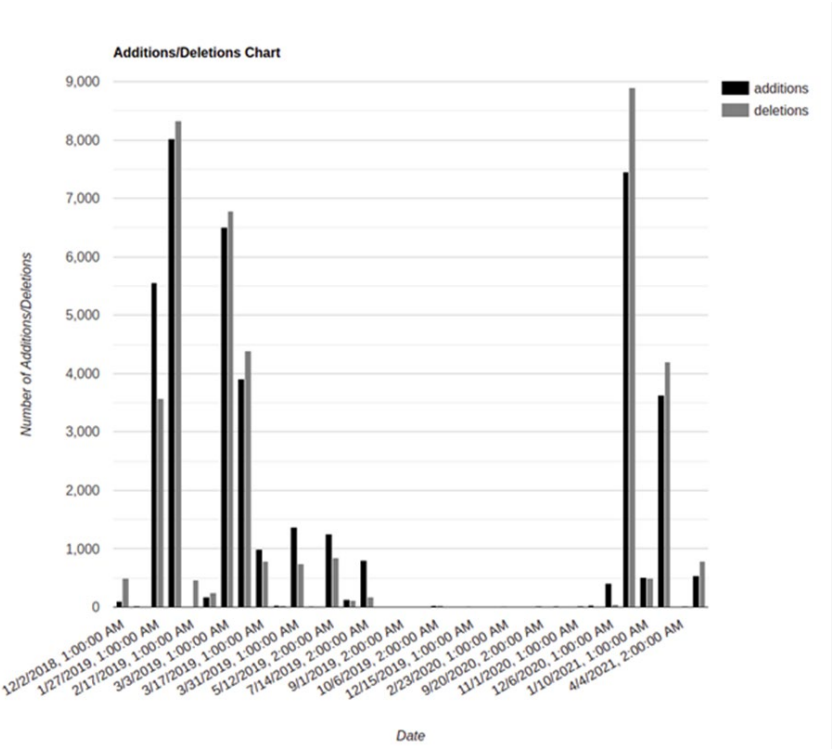
Figure 2: Weekly Hash of chosen Contributor

	Week	Number of commits	Number of additions	Number of deletions
1	12/2/2018, 1:00:00 AM	1	109	492
2	1/6/2019, 1:00:00 AM	1	18	0
3	1/27/2019, 1:00:00 AM	1	5,557	3,579
4	2/10/2019, 1:00:00 AM	23	8,022	8,325
5	2/17/2019, 1:00:00 AM	3	4	474
6	2/24/2019, 1:00:00 AM	8	181	255
7	3/3/2019, 1:00:00 AM	17	6,507	6,783
8	3/10/2019, 1:00:00 AM	6	3,909	4,392
9	3/17/2019, 1:00:00 AM	7	988	792
10	3/24/2019, 1:00:00 AM	1	31	28
11	3/31/2019, 1:00:00 AM	1	1,377	741
12	4/7/2019, 2:00:00 AM	3	14	11
13	5/12/2019, 2:00:00 AM	2	1,250	842
14	6/9/2019, 2:00:00 AM	3	133	114
15	7/14/2019, 2:00:00 AM	2	805	179
16	8/25/2019, 2:00:00 AM	1	3	3
17	9/1/2019, 2:00:00 AM	1	5	5
18	9/15/2019, 2:00:00 AM	1	7	6
19	10/6/2019, 2:00:00 AM	3	22	31
20	12/8/2019, 1:00:00 AM	1	3	1
21	12/15/2019, 1:00:00 AM	1	11	1
22	1/5/2020, 1:00:00 AM	1	8	0
23	2/23/2020, 1:00:00 AM	1	11	0
24	6/7/2020, 2:00:00 AM	1	4	4
25	9/20/2020, 2:00:00 AM	1	16	6
26	9/27/2020, 2:00:00 AM	1	16	7
27	11/1/2020, 1:00:00 AM	2	7	25
28	11/22/2020, 1:00:00 AM	2	33	6
29	12/6/2020, 1:00:00 AM	2	406	50
30	1/3/2021, 1:00:00 AM	11	7,450	8,895
31	1/10/2021, 1:00:00 AM	1	504	490
32	3/14/2021, 1:00:00 AM	3	3,626	4,197
33	4/4/2021, 2:00:00 AM	1	9	20
34	4/11/2021, 2:00:00 AM	1	539	794

Source: Authors

By displaying data that represent many added and deleted lines of code in different time intervals on the Column graph, the relationship between the numbers can be easily observed. Added lines of code are marked in black, and the number of deleted lines is marked in gray (Figure 5). It can be seen that the largest number of deleted lines of code, approximately 9000 (8,895), occurred in the week that began on 3.1.2021. It is also noticeable that the largest number of added or changed lines of code (8,022), occurred in the week that started on February 10, 2019.

Figure 5: Column Chart of Weekly Hash



Source: Authors

To conduct the SNA over data from GitHub, it is necessary to take into account social coding (AlMarzouq et al., 2020), i.e. social interactions among users that take place on the platform. The pull-based model as a way of integrating code stands out for its ability to analyze social activities. From creating a pull request, code review and commenting, to approving or rejecting the request and finally closing and integrating it (merge) or

deleting it, each of the mentioned activities involves several participants collaborating, exchanging opinions, making suggestions, and working together to solve problems or to make new functionalities of the software solution.

Such interactions have been noted in the form-36 repository of Contentful's GitHub organization. GitHub Pulls API was used to obtain relevant data. As it was noticed, some pull requests were made by bot programs that perform various tasks of code analysis and editing. Such requests were excluded from the data set in further analysis. The state field value of each analyzed pull request was open or closed, depending on whether the request was already integrated (closed) or still open for collaboration (open). The user field contains information about the user who created the pull request. The value of the author_association field is Contributor, Collaborator, or Member, depending on whether the user is, respectively, an associate who is part of the main development team, an associate who is not part of the team, or a member of a given organization.

To obtain all review activities for a particular pull request, the Reviews endpoint is called. The result is a series of JSON objects, and each object corresponds to the activity of one user that is related to a given pull request. In the state field, the user activity can be seen. Among the analyzed data, the following stand out: Approved indicates that the user has approved the changes contained in the request, Commented indicates that the user has left a comment (the user can also be the creator of the pull request) and Changes_Requested indicates that changes need to be made. Users who found themselves on a pull request as requested reviewers, but didn't have any activity on it were excluded from the analysis.

It was determined that there were 756 closed and seven open requests. After sorting the data, i.e. removing those requests made by bot programs, there were 695 requests left. Seventeen pull requests were singled out for the SNA.

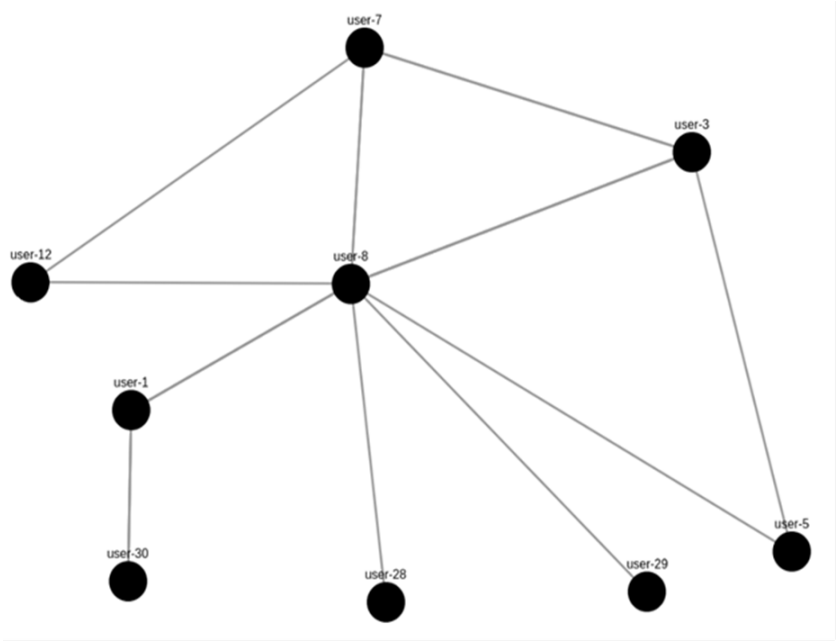
An overview graph was illustrated and centrality measures were applied. The total number of contributors within the analyzed repository is 60, while the total number of members of the organization is 34 at the time of analysis.

The undirected graph illustrated in Figure 6 was used for display. The relations in the graph are symmetrical (Cangalovic et al., 2014), it does not matter which node is the source and which the target. Graph nodes represent users who participated in social activity on the analyzed pull requests. Nine

nodes, i.e. nine users were obtained after analyzing 17 pull requests. Graph branches were created for each activity between the user who created the pull request and the user in the reviews overview (commenting requests, approving requests, or setting Changes_Requested status).

The graph is represented by the cytoscape.js package, which in addition to options for a visual representation of graphs, also supports metric calculation options such as those for centrality measures.

Figure 6: Undirected graph of SNA analysis



Source: Authors

From the very appearance of the graph, it can be concluded that the user with the username "user-8" (Figure 6) is in the center of the structure. The node corresponding to that user has a direct interaction with all nodes but one. Thus, it can be concluded that the user actively participates in pull-based social activities.

To better understand the position of the nodes in the graph, three measures of centrality are applied (Jovanović, 2017): Degree Centrality, Closeness Centrality, and Betweenness Centrality.

Degree Centrality indicates the degree i.e. the number of branches that are acquired in a given node for which centrality metric is calculated (Cangalovic et al., 2014; Jovanović, 2017). As the graph is not directed there is only one measure, the Degree. The measure was normalized in the range 0 to 1 to make it easier to compare the values obtained. As expected, according to the appearance of the graph (Figure 6), the largest number of branches is acquired in the node that is marked with the ordinal number one in the table below (Figure 7). The node has the highest value of the Degree Centrality metric's range. A node with the ordinal number nine has the lowest value of the metric because it is connected to the neighboring node using only one branch. Based on the Degree Centrality metric, it is possible to determine which members of the development team are the centers of the network, although the number of connections/branches often does not indicate the quality of contribution (Jovanović, 2017).

Figure 7: Centrality measures

	Username	degreeCentralityNormalized	closenessCentralityNormalized	betweennessCentrality
1	user-8	1	1	46
2	user-3	0.647	0.711	1
3	user-1	0.412	0.667	14
4	user-7	0.353	0.711	1
5	user-30	0.176	0.467	0
6	user-12	0.118	0.644	0
7	user-5	0.118	0.644	0
8	user-28	0.059	0.578	0
9	user-29	0.059	0.578	0

Source: Authors

Closeness Centrality represents the average distance of the node from all other nodes of the network. The higher the value of the metric is, the closer the node is to the center of the graph. The metric was also normalized in the range from 0 to 1. The node with ordinal number one has the highest value of the metric in the graph as well (Figure 7). Nodes with a high value of the Closeness Centrality metric (Figure 7) can be classified as influential members of the local group that affect the rapid dissemination of information in the network observed (Jovanović, 2017).

The high value of the Betweenness Centrality metric indicates that nodes often appear as intermediaries in the mutual communication of other nodes in the network (Jovanović, 2017). The node with ordinal number one

in the table has the highest value of 46 (Figure 7). It is followed by a node with ordinal number four whose value of the metric is 14, and then nodes with the ordinal number two and three whose value of the metric is one. Thus, the Closeness Centrality metric identifies potential burst points (Jovanović, 2017) of the network.

Conclusion

The paper discusses the possibility of using version control platforms in education while teaching courses that require the collaboration of several participants in the group projects, as well as individual activities. Also, the paper contributes to the field of learning analytics and presents a possible application of data science in the field of collaborative learning.

GitHub increases the possibility of collaboration and interaction, both among students and professors. Instead of submitting a ready-made solution at the end of the project completion deadline, students can solve the tasks iteratively. The iterative approach gives professors a better insight into the work and contribution of each student in group projects, but also when solving individual tasks. Hence, students achieve greater confidence and learn about the use of code versioning platforms that are widely used in the industry, while collaborating with their colleagues.

One way to achieve GitHub integration is the solution already mentioned in (Calatrava Arroyo et al., 2020; Beckman et al., 2021). GitHub Classroom environment within the GitHub Education program offers professors more control over the tasks they assign to students. After creating the organization and connecting it with the GitHub Classroom environment, it is possible to create a task that can be individually done or done in a group, and thus engage more students. The tasks themselves can have an initial repository setup and a description of the requirements. The assignment request can be set within the course on the LMS platform. By accepting the request, the repository that was previously added to the organization is automatically cloned to the student's account (Calatrava Arroyo et al., 2020).

The possibility of integrating the GitHub solution with learning management systems such as the Moodle platform needs to be further explored. Zagalsky et al. (2015) explore the possibility of replacing LMS platforms with version control platforms like GitHub. However, it is necessary to find a way to integrate them to benefit from both platforms.

Possible ways of employing the GitHub API are illustrated using the existing public GitHub organization, to analyze the work dynamics in group projects. Conclusions can be applied to monitoring the work dynamics of students that collaborate on group projects. The importance of the API is reflected in the flexibility it offers to the user.

The user chooses data by calling on different endpoints and then processes and filters it according to specific needs, draws conclusions, or displays the data.

Thus, the GitHub API tested in the paper has a role in collecting data on student activity. It is possible to perform an analysis of different groups of data and thus contribute to a better understanding of the state of individual activity, but also the state of the entire project. Data that indicates the state of the entire project are the number of commits and pull requests associated with the repository, as well as the number of bytes of code written in different programming languages within the repository. Data such as the number of commits, the number of added lines of code, and the number of deleted lines of code in time intervals can be good indicators of the participation of each student in group activity and act as one of the factors of forming student grades.

As the social coding functionalities are supported by the GitHub platform, the data are also suitable for SNA analysis. Students are represented as nodes in the graph, and the interactions they make on the GitHub platform can be represented by links. The pull-based model (El Mezouar et al., 2019) is of particular importance in the analysis, where interaction between students is achieved in the process of creating a pull request and the review process of that request. By applying different measures of centrality, nodes (i.e. students) that are in the center of the graph can be obtained and thus show the importance of their contribution to the project.

The possible recommendations for lecturers on how to use learning analytics:

- Include VCS into the teaching flow. One way to do it is by using the GitHub platform and GitHub Classroom environment.
- Communicate with the students about the new way of conducting the course. Teach about git and the use of the GitHub platform. Especially take into account the pull-based model.
- Use the number of commits combined with the number of added and deleted lines of code to determine students' activity.

- In order to apply SNA over the students' data from GitHub, social coding interactions of students in the platform should be considered, and measures of centrality applied.
- Include the data in the grading process, while evaluating student efforts e.g. by giving extra points to students with the high value of metrics;

The possibility of integrating platforms like GitHub into the teaching flow opens the door to new research in the field of education. The analysis conducted in the paper indicates that the data generated on code versioning platforms can be used to better understand the learning process and the dynamics of project work. Also, it is important to emphasize that a learning analytics approach developed in the paper can be employed in software development companies to assess the communication, collaboration, and contributions of developers.

As for future research, the application of machine learning can be considered, in order to better analyze the data about students, identify their characteristics, develop personalized e-learning environments or improve the monitoring of students' progress and results.

Acknowledgements

The paper is financed by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

References

[1] **AlMarzouq, M., AlZaidan, A., AlDallal, J.** 2020. Mining GitHub for research and education: challenges and opportunities. International Journal of Web Information Systems 16:451–473

[2] **Beckman, MD., Çetinkaya-Rundel, M., Horton, NJ., et al.** 2021. Implementing Version Control With Git and GitHub as a Learning Objective in Statistics and Data Science Courses. Journal of Statistics and Data Science Education 29:. https://doi.org/10.1080/10691898.2020.1848485

[3] **Beer, B.** 2018. Introducing GitHub: A Non-Technical Guide. O'Reilly Media

[4] **Bjelica, D., Pavlović, D.** 2018. Web Based Project Management Education in Student Population. In: Digital transformation: new challenges and business opportunities. Silver and Smith Publishers, London, pp. 189-213. ISBN 978-1-9993029-4-8

[5] **Blischak, JD., Davenport, ER., Wilson, G.** 2016. A Quick Introduction to Version Control with Git and GitHub. PLOS Computational Biology 12:. https://doi.org/10.1371/journal.pcbi.1004668

[6] **Calatrava Arroyo, A., Ramos Montes, M., Segrelles Quilis, JD.** 2020. A Pilot Experience with Software Programming Environments as a Service for Teaching Activities. Applied Sciences 11:. https://doi.org/10.3390/app11010341

[7] **Camacho, D., Panizo-Lledot, Á., Bello-Orgaz, G., et al.** 2020. The four dimensions of social network analysis: An overview of research methods, applications, and software tools. Information Fusion 63:. https://doi.org/10.1016/j.inffus.2020.05.009

[8] **Cangalovic, M., Manojlovic, V., Baltic, V.** 2014. Diskretne matematicke strukture. Fakultet organizacionih nauka, Newpress

[9] **Chacon, S., Straub, B.** 2020. Pro Git. Apress, Berkeley, CA

[10] **El Mezouar, M., Zhang, F., Zou, Y.** 2019. An empirical study on the teams structures in social coding using GitHub projects. Empirical Software Engineering 24:3790–3823. https://doi.org/10.1007/s10664-019-09700-1

[11] **Feliciano, J., Storey, M.A., Zagalsky, A.** 2016. Student experiences using GitHub in software engineering courses. In: Proceedings of the 38th International Conference on Software Engineering Companion. ACM, New York, NY, USA

[12] **Jamali, M., Abolhassani, H.** 2006. Different Aspects of Social Network Analysis. In: 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06). IEEE

[13] **Jovanović, J.** 2017. Softverska analiza drustvenih mreza, prezentacija „Analiza drustvenih mreza (2. blok)", Fakultet organizacionih nauka, Beograd

[14] **Loeliger, J., McCullough, M.** 2012. Version Control with Git: Powerful tools and techniques for collaborative software development. O'Reilly Media

[15] **Radović-Marković, M., Nelson-Porter, B., Omolaja, M.** 2009. The new alternative women's entrepreneurship education: e-learning and virtual universities. Journal of Women's Entrepreneurship and Education, 1(1-2), 1-12.

[16] **Radović-Marković, M., Vučeković, M., Nikitović, Z., Lapčević, G.** 2021. Learner creativity among entrepreneurship students in higher education through e-learning. International Journal of Entrepreneurship 25 (3), 1-7

[17] **Slavinski, T., Todorović, M., Vukmirović, V., Montenegro, A.M.** 2020. Women, Entrepreneurship and Education: Descriptive Bibliometric Analysis Based on SCOPUS Database. Journal of Women's Entrepreneurship and Education (3-4). pp. 181-201. ISSN 1821-1283

[18] **Somasundaram, R.** 2013. Git: Version Control for Everyone Beginner's Guide. Packt Publishing

[19] **Stojanović, D., Bogdanović, Z., Petrović, L,. Mitrović, S., Labus, A.** 2020. Empowering learning process in secondary education using pervasive technologies, Interactive Learning Environments, ISSN:1049-4820

[20] **Stojanović, D., Domazet, I.** 2020. Use of information technologies in educational purposes – Case from Serbia, Economic Analysis, 53(2), pp. 68-78 ISSN 1821-2573

[21] **Zagalsky, A., Feliciano, J., Storey, M.A., et al.** 2015. 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, New York, NY, USA

[22] **Open source guide.** 2021a. Anatomy of an open source project. In: https://opensource.guide/. https://opensource.guide/how-to-contribute/#anatomy-of-an-open-source-project. Accessed 8 May 2021

[23] **GitHub docs. 2021b**. Resources in the REST API. In: https://docs.github.com/. https://docs.github.com/en/rest/overview/resources-in-the-rest-api. Accessed 8 May 2021

[24] **GitHub docs. 2021c**. Repositories. In: https://docs.github.com/. https://docs.github.com/en/rest/reference/repos. Accessed 8 May 2021

Article history: Received: July 7th, 2021

Accepted: November 9th, 2021