Mining Learning Processes from FLOSS Mailing Archives

Patrick Mukala, Antonio Cerone, Franco Turini

Mining Learning Processes from FLOSS Mailing Archives

Patrick Mukala[1], Antonio Cerone[1 2] and Franco Turini[1]

[1] Department of Computer Science, University of Pisa, Pisa, Italy
{mukala,cerone,turini}@di.unipi.it
[2] IMT Institute for Advanced Studies Lucca, Italy
antonio.cerone@imtlucca.it

Abstract. Evidence suggests that Free/Libre Open Source Software (FLOSS) environments provide unlimited learning opportunities. Community members engage in a number of activities both during their interaction with their peers and while making use of these environments. As FLOSS repositories store data about participants' interaction and activities, we analyze participants' interaction and knowledge exchange in emails to trace learning activities that occur in distinct phases of the learning process. We make use of semantic search in SQL to retrieve data and build corresponding event logs which are then fed to a process mining tool in order to produce visual workflow nets. We view these nets as representative of the traces of learning activities in FLOSS as well as their relevant flow of occurrence. Additional statistical details are provided to contextualize and describe these models.

Keywords: FLOSS learning processes, learning activities in Open Source, Mining Software Repositories, Process Mining, Semantic Search

1 Introduction

Currently a number of studies provide evidence that suggests the existence of learning opportunities in FLOSS environments [1,10,12−17,23]. As part of this substantiation, FLOSS communities have been established as environments where successful collaborative and participatory learning between participants occurs [14,16,17].

Moreover, the levels of interest as well as the aura created around the occurrence of learning within FLOSS have attracted practitioners in tertiary education to consider incorporating participation in FLOSS projects as a requirement for some Software Engineering courses [12,14,27,32-35]. A number of pilot studies have been conducted in order to evaluate the effectiveness of such an approach in traditional settings of learning [10−13,20]. To aid in this endeavor, in our previous study, we put an emphasis on how learning occurs in terms of phases [2,19]. To this end, it has been proposed that a typical learning process in FLOSS occurs in three main phases: Initiation, Progression and Maturation. In each phase, a number of activities are executed though

adfa, p. 1, 2011.

interactions between Novices and Experts. A Novice is considered as any participant in quest of knowledge while the knowledge provider is referred to as the Expert. Fig. 1 depicts the categorization of the learning phases with the Initiation Phase synonymously corresponding to understanding on the x axis as a learning stage, while Progression and Maturation correspond to practicing and developing respectively. The gray area in Fig. 1 represents the progression with regards to users as they progressively perform the types of activities on the y axis.

In this paper, we present an approach for mining these learning phases from FLOSS data. For illustrative purposes, we detail our approach and present the results for the understanding (Initiation) phase, which is at the bottom of the scale in Fig.1. In this phase, FLOSS participants get involved in the projects by reviewing and communicating with the purpose of understanding contents without producing any tangible contributions. Initiation is a critical stage as the participant accesses project repositories and exchanges emails and posts messages seeking information and posting any requests. Fig.1 also shows how, in the practicing and developing phases, the participants' activities gradually move from simply using to posting and making significant contributions through commits [2].

FLOSS repositories, such as CVS, Bug reports, mailing archives, Internet relay chats etc., contain all traces of participants' activities as they work in these environments. Singh, Nichols and Twidale [6] argue that the FLOSS environment typically includes discussion forums or mailing lists to which users can post questions and get help from developers or other users. These forums are unrestricted and act as a learning environment for novices and experts alike. While many studies have provided invaluable insights in this direction, their results are mostly based on surveys and observation reports [7,8,22−24,27]. Our paper proposes to contribute in this context by studying learning activities from FLOSS repositories using process mining. In particular, the paper focuses on tracing and visualizing the learning activities as well as their flow of occurrence collected in mailing archives of a FLOSS platform called OpenStack [28].

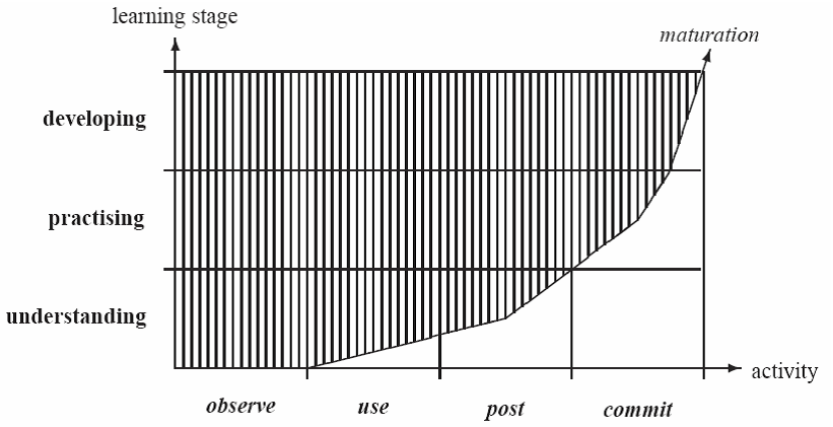

Fig. 1. Learning Stages and participants' learning progression in OSS communities [2]

Our major contribution is the approach used in analyzing the data, the application of process mining and mostly the discovered empirical evidence of learning activities' traces. The rest of the paper is structured as follows. Section 2 provides preliminary details on mining the data and constructing the log and succinctly describes the Initiation Phase of the learning process. In Section 3 we discuss the data collection and analysis and then present the empirical results. Section 4 concludes the paper.

2 Preliminaries: Mining Data and Catalog of Key Phrases

In order to identify activities and construct the event logs needed for our analysis, we undertake a number of tasks. The first task is analyzing the contents of emails. Text mining appears to be the most direct solution for this task as we need to analyze the contents of a post/email and deduct a corresponding activity. Current text mining tools such as Carrot2, GATE, OpenLP, RapidMiner and KH Coder appear not to be appropriate for the kind of analysis we want to conduct. Tracing learning activities requires semantic interpretation of email contents and this could not be achieved by using any of these tools.

Therefore, we considered making use of Semantic search with MS SQL as a fit alternative. Semantic search improves search by understanding the contextual meaning of the terms and tries to provide the most accurate answer for a given text document. However, this also requires the use of key phrases to steer the search [18]. Our choice of key phrases is based on a number of studies conducted in FLOSS with regards to the kinds of questions and answers that are asked in FLOSS communication environments [3,4,5,30].We start from this categorization, following questions and responses categories; then we deduct a number of key phrases. We try as much as we can to include all the identified key phrases and expressions within the context of identifying learning activities and establishing the learning process across its three phases, although in this paper we only present the details of the first phase.

We make use of previous findings [3,4], a formal model of learning activities in FLOSS communities [19], as well as lexical semantics to draw a catalog of key phrases with respect to our endeavor. Lexical semantics builds from synonyms of terms and their homonyms to derive the meaning of words in specific contexts. Hence, making use of semantic search is paramount and promises to capture the meaning of message contents as much as possible in identifying activities. Fig. 2 presents a catalog that contains the key phrases that semantically identify activities as categorized according to the participants' roles in the Initiation Phase of the learning process.

Principal activities gravitate around observing and making contacts in the Initiation Phase of the learning process [19]. Ideally, this step constitutes an opportunity for the Novice to ask questions and get some help depending on the requests while the Expert intervenes at this point to respond to such requests.

STATES	GLOBAL KEYWORDS	PARTICIPANTS	ACTIVITIES	KEYPHRASE/CONDITIONAL ACTIVITY
Observation	"problem", "help", "error"	NOVICE	FormulateQuestion	If PostQuestion = true
			IdentifyExpert	"How did you do this", "I saw your code", "I need your help", "this does not work for me", "-1", "is this possible to do this", "can this be done?", "very helpful", "very well"
			PostMessage	If IdentifyExpert = true
			PostQuestion	"How can I do…?" "How to?", "don't understand how", "could help?", " what is wrong?", "my code is not running", "code not executing", "question", "How to", "what is wrong", "where can I", "Any ideas how to solve this problem?", "I have tried doing", " search for this", "but have had little luck", "any help?", "any suggestions?", "everything I could", "new to the"
			CommentPost	"does not work", "not executing", "this does not work for me", "do not know what is wrong with my code", "here is my code", "in short my problem", "step by step", "details provided", "works as follows", "I want it to", "expect it to" ,"my question is like this", "what I mean"
		EXPERT	ReadMessages	If CommentPost = true
			ReadPost	If CommentPost = true
			ReadSourceCode	"syntax error", "maybe you should…", "it seems to work for me""do not know what is wrong with the code" or "not sure it can"') or "running your code"
			CommentPost	"system details needed", "more details needed", "more problems details needed", "more details needed of what is on the screen", "Did it work before?", "provide exact step by step details"
ContactEstablishement	"can I get your help", "can you help", "send question", "contact details", "send email", "send file", "more details"	NOVICE	ContactExpert	If SendDetailedRequest = true
			SendDetailedRequest	"actually the code is like this…" "I tried this", "I don't know how"," I don't understand how?" "can you help", "your help", "you explain","as you asked","so my question is", "I wanted to know","what I meant is","my screenshot looks","I get this error","how do I fix this"
		EXPERT	ContactNovice	If CommentPost/SendFeedback = true
			CommentPost/SendFeedback	"does not work", "not executing", "maybe you should…", "it seems to work for me", "this does not look right" "you need to delete this..", "the syntax is not correct", "send me your code", "what is your problem?", "this works for me", "Did it work before", "I think it should work"

Fig. 2. Catalog of Key Phrases for Initiation Phase

On the one hand, a Novice seeking help can execute a number of activities. These include *FormulateQuestion*, *IdentifyExpert*, *PostQuestion*, *CommentPost* or *PostMessage*, *ContactExpert* and *SendDetailedRequest*. On the other hand, the main activities as undertaken by the Expert during the same period of time include *ReadMessages* on the mailing lists/Chat messages, *ReadPost* from forums, *ReadSourceCode*, as any participant commits code to the project, or *CommentPost*, *ContactNovice* and *CommentPost*.

In order to conduct our analysis, we need to identify the most appropriate repository in this regard. The main criteria in making such a decision lies on the existence of some form of communication exchange between FLOSS members on any candidate repository. Mailing Archives contain email messages between FLOSS members about discussions on topics relevant to the community. Some of these topics involve general questions or specific requests about files, pieces of code or even the use of new plugins etc. Hence, these Mailing Archives provide adequate details to track activities and explain their flow of occurrence in the Initiation Phase. Moreover, it is worth noting that the same approach can be applied to mine the remaining phases on other repositories such as source code or commits.

3 Data Collection and Analysis

The FLOSS platform used in our analysis is OpenStack [28]. According to Wikipedia, "OpenStack is a free and open-source software cloud computing software platform. Users primarily deploy it as an infrastructure as a service (IaaS) solution. The technology consists of a series of interrelated projects that control pools of processing, storage, and networking resources throughout a data center—which users manage through a web-based dashboard, command-line tools, or a RESTful API that is released under the terms of the Apache License" [28].

We considered this platform mainly due to the availability of data about email archives and also because it is still an active platform. This database is made up of 7 tables that store data pertaining to compressed files (source_code file, bugs), the mailing lists as per group discussions and topic of interests, the number of messages exchanged as well as details of the individuals involved in these exchanges as shown in Table 1.

This repository contains exactly 54762 emails exchanged between 3117 people who are registered on 15 distinct mailing lists. These emails were sent during a period of time spanning from 2010 to 2014. The length of the messages considered is of typical email length specifically with an average of 3261 characters, the longest email was of 65535 characters and the shortest message yields a single character length.

In order to analyze this data set, we make use of process mining techniques. The key in Process Mining is to identify events. An event is a tuple made up essentially of case ID, performer, activity and any relevant attributes we need for our analysis. In our case, we include the phase of the learning process, date as well as the role (Novice, Expert). Other key components include the catalog of key phrases as shown in Fig. 2 and the data set. Based on all these elements, we generated our event log, which is the set of all identified events.

Table 1. Details of Mailing archives elements from OpenStack

Tables	Records
dbo.compressed_files	401
dbo.mailing_lists	15
dbo.mailing_lists_people	4434
dbo.messages	54762
dbo.messages_people	54762
dbo.people	3117
dbo.people_upeople	3117

An event E is a sextuple (t, a, p, d, s, r) such that: t is the case in the event and can be either a topic on emails or an issue number on code and bug reports; a is the activity; p is the participant; d is the relevant date of occurrence; s is the state of the learning process; and r is the participant's role in the process.

Moreover, we refer to the catalog introduced earlier to retrieve the mappings between key phrases, activities, states and participants. Let c_1, c_2 and c_3 be catalogs respectively for Initiation, Progression and Maturation. We distinguish between key phrases for activities and states. We refer to key phrases for states as *gl_key* (global keys) while the key phrases that help distinguish activities are referred to as *lc_key* (local keys). We define catalogs as sextuples (C, c_i, *gl_key*, *state*, *lc_key*, *activity*, *role*) such that: C is the set of all our catalogs, $c_i \in C$ is a single catalog, *gl_key* is the key phrase for the identification of a state, *state* is the state as it appears in the catalog, *lc_key* is the key phrase used to identify an activity, *activity* is the corresponding activity in the catalog, and *role* is the role as it appears in the catalog. Using such information, we generate the event log to be analysed through process mining.

3.1 Process Mining Mailing Archives

In order to process mine these records, we choose Disco (Discover Your Processes) [29], an appropriate tool for analyzing the identified events and providing efficient visualizations to demonstrate the workflow of occurrence of activities in these processes. Disco is a toolkit for process mining that enables the user to provide a preprocessed log specifying case, activities, originator and any other needed attributes. The tool performs automatic process discovery from the log and outputs process models (maps) as well as relevant statistical data.

In essence, Disco applies process mining techniques in order to construct process models based on available logging data that is organized into an event log. This logging data is all the details about transactions that can be found in log file or transac-

tion databases. Therefore, an event log can take a tabular structure containing all recorded events that relate to executed business activities [29].

Making use of Disco, we produced the Process Models representing the occurrence of learning activities as documented by their corresponding email messages.

For simplicity, we choose to represent the models through a graphical workflow as well as the statistical information as provided by Disco. Disco offers the possibility for a process model to be represented with frequency metrics that explain the flow of occurrence of events.

The main objective of the frequency metrics is the depiction of how often certain parts of the processes have been executed. We can distinguish three levels of frequency: absolute frequency, case frequency and maximum repetitions. We consider these metrics to model learning activities executed by both the Novices and Expert.

Additional details regarding the numerical measures such as events over time, active cases during this given period of time, case variants, the number of events per case as well as case duration could be plotted as needed. However, for simplicity and effectiveness, we represent only major statistical details that are most representative of the presence, impact and occurrence of learning activities in FLOSS over the chosen period of time.

3.2 Empirical Results

Before we unpack details about process models for both the Novice and Experts, we give some crucial details about the overall Initiation Phase. It should be noted that in Fig. 3 and 4 the numbers, the thickness of the arcs or edges, and the coloring in the model illustrate how frequent each activity or path has been performed. For the purpose of this paper, we retained the topic of emails, the message itself, the people involved in exchanging these emails, the resulting activities and classification of where such activities fall in our defined learning curve to build events and produce the event log used for model extraction.

The analysis of the Initiation Phase of the learning process is carried out on data that refer to the period between the 11th of November 2010 and the 6th of May 2014. During this time, we note that a total of 123401 events were generated. An event represents a tuple made up of the case (in this context, the discussion topic), the email senders as well as the relevant learning activities. With about 565 cases, a total of 14 activities are executed with an average time per case of 69.9 days while the median duration is of 57.8 days.

We can also point out that participants in quest for knowledge claim the majority of activities with a total of 122838 amounting to 99.54% of all executed activities at this point in contrast with Experts who intervene at a lower rate of 0.36% with 440 activities, slightly ahead of people doing something other than exchanging knowledge with 123 activities.

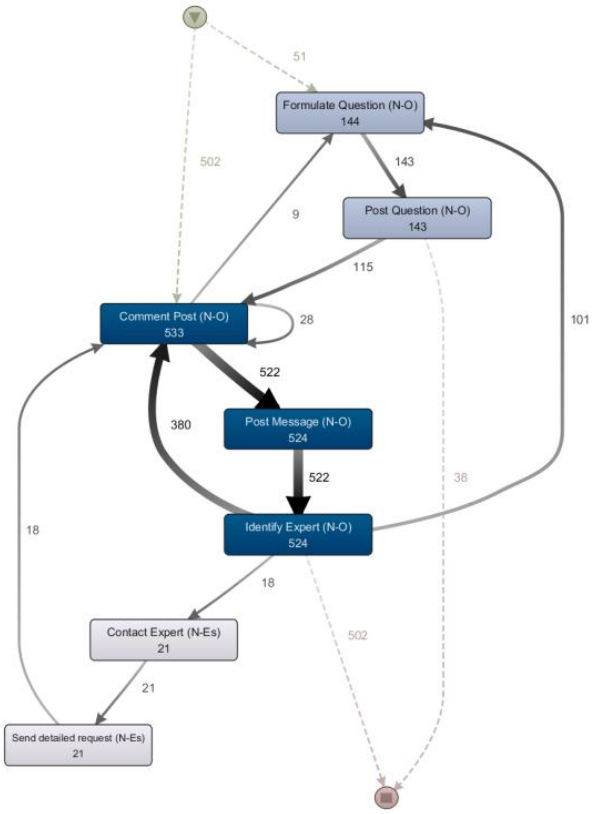

Fig. 3. Process Model for Novice–Per frequency [Initiation Phase] in Mailing Lists

The process model depicted in Fig. 3 represents a workflow for all the activities performed by the Novice during the first phase of the learning process. On average, how often an activity has been executed in this process by the Novice as well as how often an activity links to another (path) can be noted through the numbers, the thickness of the arcs or edges, and the coloring in the model. We note that the Novice in OpenStack has engaged in a number of learning activities throughout this period of time. Fig. 3 demonstrates that in 51 cases the process would start from formulating a question, posting the question, commenting on post (and this has occurred about 27 times), posting a message, which indicates that an expert has been identified, contacting that expert and sending a detailed request to the expert through commenting on a post. The numerical argument between the transitions from one activity to another indicates how many times on average this has happened.

Moreover, Fig. 3 also shows that in 502 cases, a Novice starts by commenting on a post first, then formulates a question and follows the process as explained above. Sometimes (101 times in the depicted process map), after identifying an expert, a Novice could go back to formulating another question (or follow-up questions) or even go back to just commenting on the post as part of the interaction.

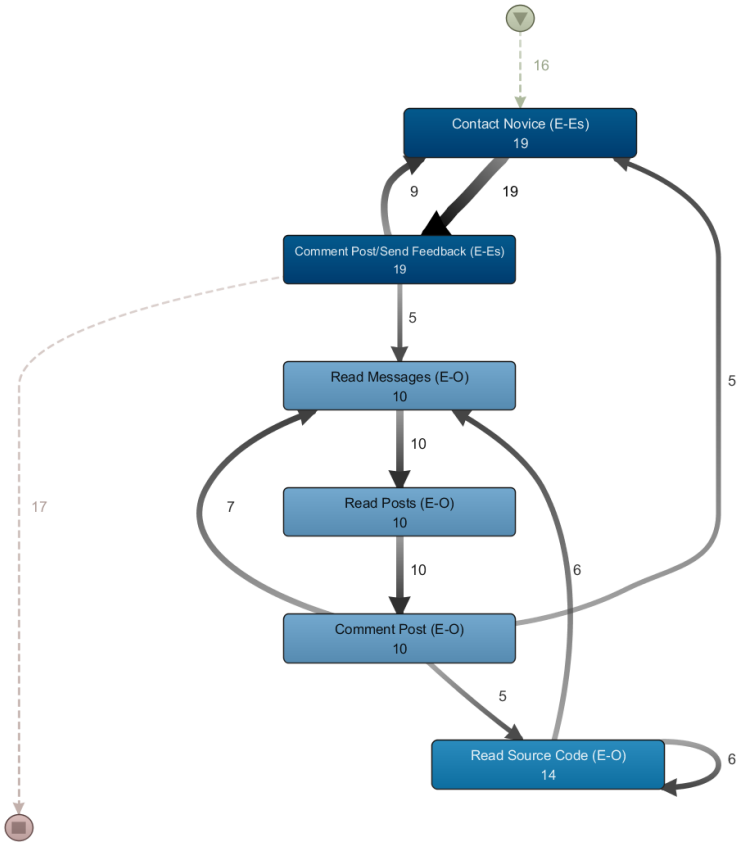

Fig. 4. Process Model for Expert–Per frequency [Initiation Phase] in Mailing Lists

The process model depicted in Fig. 4 represents a workflow for all the activities performed by the Expert during the first phase of the learning process. One should note that 6 main activities are undertaken by the Expert. In some cases, the Expert

would contact the Novice, by commenting on a post or giving feedback regarding a request from the Novice, then read messages and posts, commenting on these posts as well as reading source code, especially if the Novice's requests have to do with source code. The assumption here is that the Expert is referred to as such because of the nature of the reaction activity. Every time an Expert comments, it is in response to a Novice request or to request further details on an already posted question.

In some instances, the Expert would go back to reading messages after commenting on a post or reading source code, or sometimes contacting the Novice again after commenting on a post involved in the exchange. In some instances, on average in 9 cases, the Expert goes back to contacting the Novice after providing feedback. This is where further details about the request or clarification might be requested.

4 Conclusion

FLOSS environments appear to provide learning opportunities for participants. While this aspect has been previously investigated [7,8], we believe that there is a need for empirical support for such findings. An important remark [9] emphasizes that in such previous work, content data had been collected using surveys and questionnaires or through reports from observers who have been part of the community for a defined period of time. Research suggests that a growing number of participants are highly motivated to engage in these platforms through discussions and email exchange. These interactions produce massive volumes of data that contain evidence of learning. Since these learning activities are not directly observable in the repositories, we made use of semantic search in MS SQL to identify activities from message texts and constructed the event log that we then analyzed with the help of the Disco process mining tool.

Using a combination of text mining techniques (semantic search), key phrases and a set of rules, we believe that our approach has provided insights on how to find traces of interaction and learning activities in FLOSS email messages. We can thus say that it is feasible to trace these activities and that process mining can play a catalyst role in identifying these activities in FLOSS. Our work aimed to give evidence about the existence of learning processes in FLOSS environments through the empirical analysis of OpenStack Mailing Archives.

Our results demonstrate how these learning processes are extracted and how each activity fits within the global picture. For a Novice, we can note that the process in some cases spans from formulating a question, posting the question, commenting on post, posting a message, which indicates that an expert has been identified, contacting that expert and sending a detailed request to the expert through commenting on a post. In most cases, a Novice starts by commenting on a post first, and then formulates a question before following the process as described above. Six main activities are performed by the Expert starting from contacting the Novice, through commenting on a post or giving feedback regarding a request from the Novice. The Expert will then read messages and posts, commenting on these posts as well as reading source code, especially if the Novice's requests have to do with source code. In some instances, the

Expert would go back to reading messages after commenting on a post or reading source code, or sometimes contacting the Novice again after commenting on a post involved in the exchange. In some other instances the Expert goes back to contacting the Novice after providing feedback. This is where further details about the request or clarification might be requested.

Finally, more experiments using this approach can be conducted in order to trace activities in the next two phases. These phases include the Progression and Maturation phases. Fig. 1 indicates how FLOSS contributors start by just observing, in the Initiation Phase, and gradually evolve to more tangible activities such as posting and committing software artifacts and source code [2]. The types of activities in the Progression phase include *Revert*, *Post* and *Apply*. After the Expert makes contact, the Novice provides additional details about the previous request through activity *Revert.* The Expert gets further involved by providing guidance and the required help to the Novice through *Post,* while the Novice implements the new acquired knowledge through *Apply*. During the Maturation phase, activities include *Analyze, Commit, Develop, Review* and *Revert*. In this phase, the Novice's acquired skills are expressed through the execution of advanced activities such as producing new code, reviewing new commits and providing assistance during discussions, thus gradually performing a transition from Novice to Expert [19]. The Expert performs the same activities with an emphasis on transferring knowledge and providing help when needed rather than applying new skills.

References

1. Sowe, S. K., & Stamelos, I. (2008). Reflection on Knowledge Sharing in F/OSS Projects. In Open Source Development, Communities and Quality (pp. 351-358). Springer US.
2. Cerone, A. (2011). Learning and activity patterns in OSS communities and their impact on software quality. In Proc. of the 5th International Workshop on Foundations and Techniques for Open Source Software Certification. (OpenCert 2011). Vol 48 of Electronic Communications of the EASST. The European Association of Software Science and Technology.
3. Lakhani, K. R., & Von Hippel, E. (2003). How open source software works: "free" user-to-user assistance. Research policy, 32(6), 923-943.
4. Steehouder, M. F. (2002). Beyond technical documentation: users helping each other. In Professional Communication Conference, 2002. IPCC 2002. Proceedings. IEEE International (pp. 489-499). IEEE
5. Singh, V., Twidale, M. B., & Nichols, D. M. (1899, December). Users of open source software-how do they get help?. In *hicss* (pp. 1-10). IEEE.
6. Singh, V., Twidale, M. B., & Rathi, D. (2006, January). Open source technical support: A look at peer help-giving. In System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on (Vol. 6, pp. 118c-118c). IEEE
7. Glott, R., SPI, A. M., Sowe, S. K., Conolly, T., Healy, A., Ghosh, R., ... & West, D. FLOSSCom-Using the Principles of Informal Learning Environments of FLOSS Communities to Improve ICT Supported Formal Education.
8. Glott, R., Meiszner, A., & Sowe, S. K. (2007). "FLOSSCom Phase 1 Report: Analysis of the Informal Learning Environment of FLOSS Communities".FLOSSCom Project.
9. Cerone, A., Fong, S., & Shaikh, S. A. (2011). Analysis of collaboration effectiveness and individuals' contribution in FLOSS communities. In Proc. of the 5th International Workshop on Foundations and Techniques for Open Source Software Certification. (OpenCert 2011). Vol 48 of Electronic Communications of the EASST. The European Association of Software Science and Technology, 2012.
10. Papadopoulos, P. M., Stamelos, I. G., & Meiszner, A. (2013). Enhancing software engineering education through open source projects: Four years of students' perspectives. Education and Information Technologies, 18(2), 381-397.

11. LeBlanc, R. J., Sobel, A., Diaz-Herrera, J. L., & Hilburn, T. B. (2006). Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. IEEE Computer Society Press.
12. Sowe, S. K., & Stamelos, I. G. (2007). Involving software engineering students in open source software projects: Experiences from a pilot study. Journal of Information Systems Education, 18(4), 425.
13. Cerone, A. K., & Sowe, S. K. (2010). Using Free/Libre Open Source Software Projects as E-learning Tools. In Proc. of the 4th International Workshop on Foundations and Techniques for Open Source Software Certification. (OpenCert 2010), Vol. 33 of Electronic Communications of the EASST. The European Association of Software Science and Technology.
14. Fernandes, S., Cerone, A., & Barbosa, L. S. (2014). Analysis of FLOSS Communities as Learning Contexts. Proc. of the 7th International Workshop on Foundations and Techniques for Open Source Software Certification. (OpenCert 2013), Vol. 8368 of Lecture Notes in Computer Science, pages 405–416, Springer.
15. Fernandes, S., Cerone, A., & Barbosa, L. S. (2013). FLOSS Communities as Learning Networks. Int. Journal of Information and Education Technology, 3(2), pages 278–281, IACSIT Press, April 2013.
16. Fernandes, S., Cerone, A., Barbosa, L. S., & Papadopoulos, P. M. (2014). Proc. of the 1st International Symposium on Innovation and Sustainability in Education (InSuEdu 2012), Vol. 7991 of Lecture Notes in Computer Science, pages 121– 132. Springer.
17. Meiszner, A., Glott, R., & Sowe, S. K. (2008). Free/Libre Open Source Software (FLOSS) communities as an example of successful open participatory learning ecosystems. UPGRADE, The European Journal for the Informatics Professional, 9(3), 62-68.
18. Larson, B. (2012). Delivering Business Intelligence with Microsoft SQL Server 2012. McGraw-Hill Osborne Media.
19. Mukala, P., Cerone, A., & Turini, F. (2014). An Abstract State Machine (ASM) Representation of Learning Process in FLOSS Communities. In SEFM 2014 Workshops, Vol. 8938 of Lecture Notes in Computer Science (pp. 227-242). Springer.
20. Meiszner, A., Mostaka, K., & Syamelos, I. (2009). A hybrid approach to Computer Science Education–A case study: Software Engineering at Aristotle University.
21. Fernandes, S., Cerone, A., & Barbosa, L. (2014). A preliminary analysis of learning awareness in FLOSS projects. Proc. of the 1st International Symposium on Innovation and Sustainability in Education (InSuEdu 2012), Vol. 7991 of Lecture Notes in Computer Science, pages 133–139. Springer.
22. Dillon, T., & Bacon, S. (2006). The potential of open source approaches for education. FutureLab Opening Education Reports, http://www. futurelab. org. uk/resources/publicationsreports-articles/opening-education-reports/Opening-Education-Report200.
23. Sowe, S. K., Stamelos, I., & Deligiannis, I. (2006). A framework for teaching software testing using F/OSS methodology. In Open Source Systems (pp. 261-266). Springer US.
24. Sowe, S. K., Stamelos, I., & Angelis, L. (2006). An Empirical Approach to Evaluate Students Participation in Free/Open Source Software Projects. InIADIS International Conference on Cognition and Exploratory Learning in Digital Age CELDA 2006 (pp. 304-308).
25. Jaccheri, L., & Osterlie, T. (2007, May). Open Source software: A source of possibilities for software engineering education and empirical software engineering. In Emerging Trends in FLOSS Research and Development, 2007. FLOSS'07. First International Workshop on (pp. 5-5). IEEE.
26. Papadopoulos, P. M., Stamelos, I. G., & Meiszner, A. (2013). Enhancing software engineering education through open source projects: Four years of students' perspectives. *Education and Information Technologies*, *18*(2), 381-397.
27. Meiszner, A., Stamelos, I., & Sowe, S. K. (2009). 1st international workshop on:'Designing for participatory learning'Building from open source success to develop free ways to share and learn. In *Open Source Ecosystems: Diverse Communities Interacting* (pp. 355-356). Springer Berlin Heidelberg.
28. OpenStack. (2014, November 3). In Wikipedia, The Free Encyclopedia. Retrieved 15:48, November 10, 2014, from http://en.wikipedia.org/w/index.php?title=OpenStack&oldid=632224644
29. Günther, C. W., & Rozinat, A. (2012, September). Disco: Discover Your Processes. In *Proc. of the Demonstation Track of BPM 2012*, Vol. 940 of CEUR Workshop Proceedings, CEUR-WS.org (pp. 40-44).
30. Elliott, M. S., & Scacchi, W. (2005). Free Software Development: Cooperation and Conflict in. Free/open source software development, 152.