International Journal of Information & Network Security (IJINS)
Vol.3, No.5, October 2014, pp. 292~307
ISSN: 2089-3299

Foundational Imperatives for Process Mining Learning Patterns in Floss Repositories

Patrick Mukala
Department of Computer Science, University of Pisa

Article Info

Article history:

Received Jun 12th, 2014
Revised Aug 20th, 2014
Accepted Aug 26th, 2014

Keyword:

Process Models;
Process Mining;
Social Networks;
Floss Environments;
Floss data;
Open Source;
learning processes

ABSTRACT

Free/Libre Open Source Software (FLOSS) projects enable groups of participants to work remotely and achieve projects of common purposes. While the phenomenon of FLOSS projects has generated considerable research interest, it still offers extensive potential worth exploring. In particular, in the context of learning, FLOSS communities have been established as environments where successful collaborative and participatory learning between participants occurs. The quality of FLOSS produced applications is indicative of their widespread use and popularity. Given this popularity, it is critical to explore their potential as learning environments. On the other hand, process mining has established itself as a novel approach for empirical analysis of event logs from data repositories. While many studies have provided invaluable insights in this direction, their results are mostly based on surveys and observation reports. In this paper, we lay a foundation to an important discussion that seeks to contribute in this context for the provision of empirical data for learning patterns from FLOSS repositories using process mining.

Corresponding Author:

Patrick Mukala,
Departement of Computer Science,
University of Pisa,
Largo Pontecorvo 5, 56127 Pisa PI
Email:patrick.mukala@gmail.com, mukala@di.unipi.it

1. INTRODUCTION AND MOTIVATION

This work is driven by the need to foster the understanding of learning patterns generated by knowledge acquisition mechanisms occurring in FLOSS environments. As evidence of consideration for FLOSS environments as teaching tools even in formal education setting [1],[2] mounts, there is a need to provide more empirical evidence in this regard and process mining appears to be an option worth exploring.

Since the mid-nineties, considerable work in the field of process mining has been conducted to develop techniques, design and implement algorithms for process models from event logs and based on these processes (activities or messages being exchanged between developers), a process model is produced [3]. A chronological report on the evolution of process modeling and approaches can be found in [4].

Given their widespread use and increasing success in tracing processes from recorded logs, we suggest that process mining techniques can be applied in software development process modeling specifically with FLOSS repositories where the relevant data is available. Paramount to this process is defining the purpose for mining. Hence, in this paper we discuss different routes through which learning occurs in FLOSS as a potential objective for process mining FLOSS repositories.

The rest of the paper is structured as follows: in section 2, we describe FLOSS environments with key references to the profile of FLOSS members, their roles and contributions as well as the general collaboration and activities that take place within these confines; in section 3

we describe FLOSS communities as learning environments and tools, section 4 introduces the concepts of process mining while in section 5 we lay the foundation to future way on how process mining can help in tracing learning patterns and conclude this discussion in section 6.

2. FLOSS ENVIRONMENTS

Free/Libre Open Source Software (FLOSS) environments are online communities made up of heterogeneous participants who remotely interact in the development of Open Source Software. The fundamental idea behind these environments is to enable free access to software source code to users whenever they wish to [5]. Given the aura around the concept and phenomenon of FLOSS, different terms are used to identify with it. In the literature, many terms are in use to describe the FLOSS phenomenon. Some of these terms include Free Software (FS), as used by Free Software Foundation (FSF), Libre Software (LS), Open Source Software (OSS) used by the Open Source Initiative (OSI), Free Open Source Software (FOSS), and Free/Libre/Open Source Software (FLOSS) are different terms used in reference with the concept of free software [5][6].

The term "FLOSS" as used in this paper, borrows its basic definition from the FLOSScom Project in [5] and [6]. In this work, it refers to " users' freedom to use, modify, distribute, or even sell the software with little obligations as in propriety or closed source software". In FLOSS environments, developers/participants make use of online platforms such as Sourceforge, Freshmeat or GitHub for all software development activities. Some common repositories and tools used include *de facto* versioning systems such as Concurrent Versions System (CVS) or Subversion (SVN), bug-tracking systems (BTS) and bug databases (e.g. Bugzilla)[5]. These repositories also make provision for different ways in which participants communicate and interact such as mailing lists, wikis, forums and Internet Relay Chats (IRC) [7][8][5][6].

The current status indicates that FLOSS has drastically changed software development and distribution [9]. With FLOSS communities, the internet enables a large number of remotely distributed individuals who work together towards software projects in a Bazaar model-like fashion [9]. These participants, through extensive collaboration, contribute in writing code, debugging, testing and integrating applications as well as any other required Software Engineering (SE) activities. Other activities in the communities range from support services such as product features suggestions, distribution, query handling, new members induction etc [5][6].

In recent years, these environments have been credited to deliver high-quality software products such as Linux, Apache, Sendmail, MySQL, PostgreSQL and Moodle among others [10][5]. Numerous studies have recently analyzed FLOSS environments and projects to determine the quality of OSS products. Specifically, these studies analyzed the activities, best practices and collaboration patterns within FLOSS environments in order to identify factors that enable the emergence of high quality OSS products. In [11], a number of FLOSS projects were reviewed using a set of quality metrics to reveal some prospects on the link between high quality and open source software practices. This study highlighted good project communication and management as key enablers. As revealed in [10], additional studies have been conducted by Coverity and its reports accentuate the high-quality of OSS products, which is compared to or even better than source proprietary software [10].

2.1 Participant Profiles

In order to study and explore FLOSS communities, we look at some data pertaining to personal features of the participants. These characteristics include age, qualification and professional background etc. The FLOSS Developers Survey [12] shows that the age of Open Source Software communities is indicative of a great interest in online collaborative software development. The survey indicates that individuals who become part of these communities are aged between 14 and 73 years old with a predominance of participants aged between 16 and 36 years [5] and an average age of 27.1 years. The study provides clear indications that the FLOSS community is in majority young, on average in the mid-twenties [12]. Also, this study has revealed the age at which participants join the communities. Participants join FLOSS communities at age ranging between 10 and 55 years [12] with 7 % of this proportion starting below the age of 16, 33% between 16 and

20 years old, an additional third of this population between 21 and 25, 25% older than 26 for an overall average of 22.9 years old starting age [5][12].

In terms of Education, the FLOSScom report [5][6] and the survey in [12] show that FLOSS members are people with higher education. The figures indicate that nine percent of the participants had a doctorate degree while 70% others boast a university degree. Studies also suggest that 17 % of the developers in FLOSS environments are high school diploma holders [13] with the rest either still schooling or having a lower educational background [13].

Furthermore, studies also reveal the professional background of these participants to better understand them. The FLOSS Developers Survey [12][13] reveals a professional structure for FLOSS community members with a great dominance for people with IT/Computer Science background as 83 percent of participants are representative of the sector. One third of the surveyed population is made up of software engineers, while students make up the second largest group with 16 % as trailed by programmers and IT consultants. The remainder of the group made up of Executives, marketing and product sales experts do not really have a significant impact on the professional structure of the FLOSS community [5][13].

2.2 Roles and Responsibilities of FLOSS members

The bulk of reports on FLOSS members profiling such as FLOSScom [5][6], FLOSSpol [12][13] and the analysis performed by Krishnamurthy in [14] among others have found that OSS members in these communities hold different roles that define their responsibilities and participation in the communities activities. Among, project initiators and the core development team remain at the heart of any development project in the community.

In [14], Krishnamurthy performed an analysis of the top 100 FLOSS communities of sourceforge.net and the findings of this work indicate that the core development team is made up of a small number of developers while the majority of the core team, referred to as the enhanced team, perform additional tasks such as feature suggestions, testing and query handling [14].

In their work on this subject, Crowston and Howison [7] developed a onion-like structure that depicts hierarchies among FLOSS members roles as shown in Figure 1a. The model is structured as follows:

- The core developers make the center of the model as they represent the engine behind any development project. These developers provide most of the code and monitor the entire development cycle of projects.
- The next ring represents the layer made up of co-developers. These are people who contribute to the project through patches such as bug fixes etc through the core developers.
- The following layer is composed of active users. These are members of the community who do not necessarily contribute to the project in terms of coding, but their support is made through testing and bug reporting.
- The outer layer of the model represents the passive users who do not leave any tangible trace of their participation whether through forums or mailing lists.

Figure 1: An organizational structure of a typical FLOSS community

This model suggests a progressive skills development process that can be observed in FLOSS. As highlighted in [7], participants increase their involvement in the project through a process of role meritocracy. This implies that passive users could move from their state of passiveness to active users, bug reporters until they become part of the core team [7] as described in Figure 2. All these roles represent crucial contributions required for the overall project's quality. However, in FLOSS environments, it is regarded as a reward and recognition of members' abilities and contributions to move from one state to another [7].

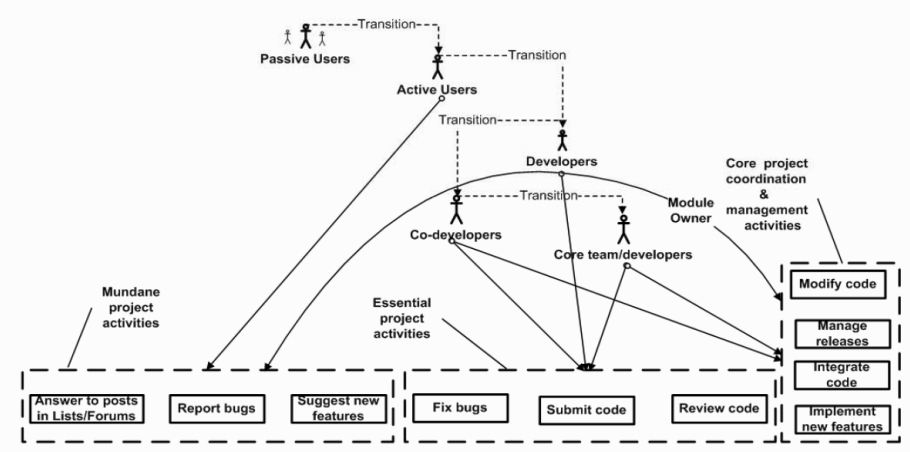

Figure 2: Role transition in FLOSS communities

2.3 Collaboration and Contribution activities of FLOSS members

In order for one to study learning patterns in FLOSS activities, it is helpful to understand how these activities occur. In analyzing collaboration and contribution of FLOSS members, [8] describe how collaboration and quality of FLOSS participants' contribution can be extracted from repositories. Cerone et al. argue that FLOSS repositories such as versioning systems, mailing lists and reporting systems can be examined to identify the identities of members involved in a particular communication channel, the topics of the interaction, the volume of data and information exchanged during the interaction [8]. These repositories can also provide the degree of participation in terms of number of code commits, bug fixing, email postings as well as reports and produced documentation [8]. In order to explain how collaboration and individuals contribution occurs, [8] propose an Individual-team interplay model adapted to FLOSS where interaction consists of posting related activities, the product of the task is delivered through commits made by individuals, or team leader's approval or release decision as shown in Figure 3.

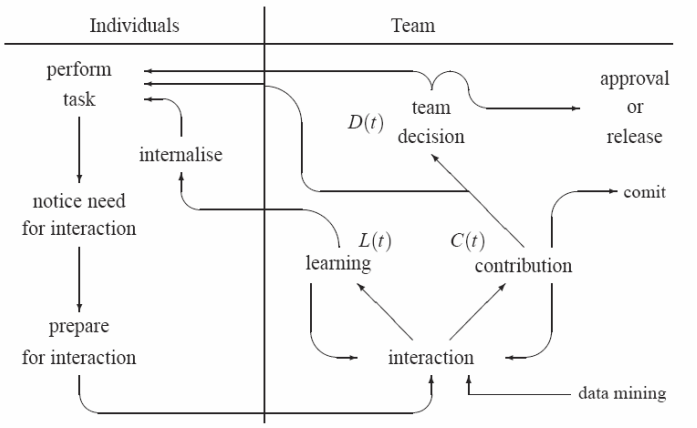

Figure 3: Individual Team interplay for FLOSS.

The model in figure 3 helps define metrics for collaboration effectiveness in FLOSS communities. Due to the evolving nature of OSS environments, [8] proposes taking measures for collaboration level consecutively at intervals of duration Δt. In order to get a sense of collaboration contribution, an average of the accumulated measures need to be calculated over the entire community lifetime.

In Figure 4, the following numbers are considered:

- $L(t)$ of learning activities,
- $C(t)$ of contributions,
- $D(t)$ of team decisions.

The numbers occur during the period $[t, t + \Delta t]$ for a given time t accounting for three stages of participation as described later in subsequent sections. In [10], the findings indicate that participation in a FLOSS project evolves through three stages: Understanding, Practice and Developing. In the first stage, the purpose of interaction and communication is solely to help capture, describe and understand contents with no production activity; in the second stage, communication is used to gradually propose new contents and any activities needed to defend the contents and provide feedback to other posted contents with production only at a trial and error process level; in the last stage, participants are able to get involved in development (software).

Furthermore, in order to study patterns of contribution from OSS members, one needs to trace participants' contribution from all repositories containing information about projects. FLOSS developers are characterized as mobile and tend to make use of different identities in their contributions to projects [15]. This poses a challenge for analysts that intend to track the location (where) and levels of participation (contribution) of such developers across interrelated repositories. A number of studies have investigated this issue and provide different mechanisms to study patterns of contribution of developers from multi-repositories perspective [15][16][17].

In [15], authors study the integration of data from multiple FLOSS repositories and present a methodology to study patterns of contribution of 502 developers in both SVN and mailing lists in 20 GNOME projects [15]. This approach can be summarized in the figure 4 below where [15] made use of repositories from the FLOSSMetrics project and proposed a methodology that can be applied when tracking users participation I both SVN and mailing lists. This means that the methodology helps identify developers who make contributions both by committing code to SVN and posting messages to mailing lists. More importantly, it helps establish that developers committing codes are the same ones posting messages for the same projects.

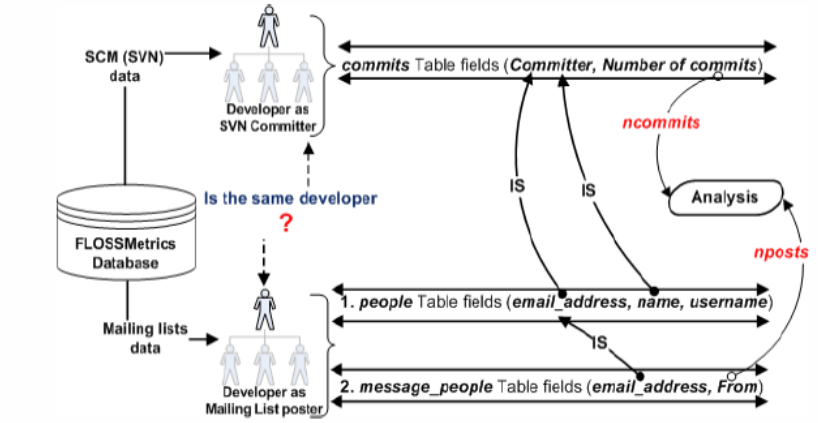

Figure 4: Methodology to identify developers from multiple repositories.

3. FLOSS COMMUNITIES AS LEARNING ENVIRONMENTS

An increasing number of studies have exploited the phenomenon of FLOSS and how learning occurs herein [5][6][18][19][1][20][21][22][2][10]. In most of these works and as highlighted in [8], findings are based on content data generated through surveys and questionnaires or through reports from observers who have been a part of the community for a defined period of time. We think in this paper that a new direction could be considered in attempting to fulfill the need to provide more objective observations through empirical analysis of data from the FLOSS repositories. Before that though, it is critical as we build the set the grounds that some of these works be succinctly reviewed.

3.1 Learning and Activity Patterns in FLOSS

In [10], Cerone studies the interplay between learning and activity patterns during members' participation and collaboration in Free/Libre Open Source Software (FLOSS) communities. This study notes the manner in which participants' activities facilitate a learning process that possibly occurs in FLOSS members through their participation in the communities' cycle of interactions and collaboration. The value of such analysis is , in the context of the current work, to provide useful insights on how to identify and analyze learning patterns in OSS environments at both the individual and community levels.

FLOSS communities are made up of heterogeneous groups of individuals, with different backgrounds, who organize themselves in an environment where they each play specific roles, with a set of responsibilities. These participants develop different levels of knowledge that determine their participation and contribution in the community and are driven by a large variety of intrinsic and extrinsic motivations [10]. The activities in these forums are motivated by the need to bolster reputation. The reputation that members build drive the emergence of driving personalities and forms of leadership [10] in the self-organization structure created in the community.

With the extensive peer-review and collaborative discussions, OSS projects can be considered as learning and development environments where knowledge is disseminated through constant discussions and put in practice through practical contributions to software development, code revision and testing [1].

To study the link between learning and activities, [10] review typical FLOSS contributors' roles in relation to basic activities. These roles include:

- **Observer:** This is the passive user that can perform a number of activities such as reading product reports, related documentation and user manuals. One would also assume that observers browse through the data in repositories, looking at conversations and flow of messages in discussion forums or mailing lists with no active participation;
- **Supporting user:** This role refers to the active user. This is the user that downloads the software releases from repositories and contributes via reporting bugs, providing feedback, helping new users, recommending the project to others, requesting new features, but does not produce artifacts;
- **Developer:** This role describes the co-developers and core team members who are at the centre of any development project. They actively ensure the life of the product through coding, updating software and managing related activities;
- **Tester :** This role resembles the supporting user but is distinct in that it focuses mostly on actively testing pieces of code, reporting and possibly fixing bugs;
- **Translator** : This user helps to widen the use and access to software and related documentation to different users from different countries and languages;

Based on the above roles and responsibilities of FLOSS contributors, four *basic activities* can be identified [10]: ***observe*** reports, documentation, tool manuals, data, posts, code; ***use*** code, tools; ***post*** questions, requests, advises, critics; ***commit*** software, documentation, artwork, bug reports, fixed code, translations.

The observer can formally be understood as performing two basic activities from the above four. He/she can ***observe*** and ***use***. As [10] notes, this user is learning as this is usually the main goal during the first stage of the participation projects. The active or supporting user in providing feedback, recommending software, reporting bugs performs another basic activity called ***post.*** One

can notice the active user's comments in discussion forums, his/her emails on the mailing lists etc. The remaining roles are both active and productive in the sense that not only do they actively participate in discussion forums, mailing lists, wikis etc , they also produce artifacts as a result of their contribution. Developers contribute software code and documentation while testers provide bug reports and fixed code and translators provide interpreted documentation. These are instances of the last basic activity called ***commit***. This is defined as the process of contributing artifacts to the FLOSS project in the forms of pieces of code, or software components. This activity can be accomplished directly if the committer has the credentials to do so or it can be done through a process spearheaded by the project leader or initiator [10].

The author argues that the way in which these activities are combined in order to define a participant activities pattern is bound to a number of factors including intrinsic and extrinsic motivations, maturity levels, technical and social skills. This yields a large heterogeneity of activity patterns at both individual level and community level [10]. Interactions and message exchange in discussion forums are triggered by ***post*** activities and this interaction process has two components [10]:

learning sub-process where knowledge exchange between an individual entity and the rest of the community increases knowledge acquisition at both the individual level and the team or community level;

contribution sub-process in which a contribution in the form of commit is a direct result of communication and message exchange.

Hence through this interaction process, one can identify and note the presence of a *collaborative learning* process, that consists of collaborative peer-review and criticism through which a wealth of knowledge is built. Therefore, "contribution, which is based on ***commit***, is the result of communication, which is based on ***post***, individual learning, which is based on ***observe*** and ***use***, and collaborative learning, which is based on ***post"***[10].

Furthermore knowledge exchange in FLOSS communities occurs in synergy. If an actor shares information about an effective method of implementing some aspect of the software for example, this process of putting his/her ideas into words will help him/her to shape and improve these ideas [23]. When participants engage in discussions with the community, they share ideas and learn from each others. This knowledge exchange is logged into FLOSS repositories and can serve as learning ground to future users.

Finally, Cerone identifies and distinguishes three stage that characterize the evolution of activity patterns and maturation of an OSS community participants in [10]:

Understanding : This is the initial stage of the learning process during which participants get involved in the projects by reviewing, communicating with the purpose of understanding contents without producing any tangible contribution. This is a critical stage as the participant accesses project repositories and exchanges emails and posts messages to get acquainted with the contents of the repositories.

Practicing : During this stage, the participant evolves from understanding to providing new contents in discussions forums, defending these contents and criticizing the existing materials.

Developing : at this stage, the participant is able to produce and review contribution from peers in terms of coding and software artifacts.

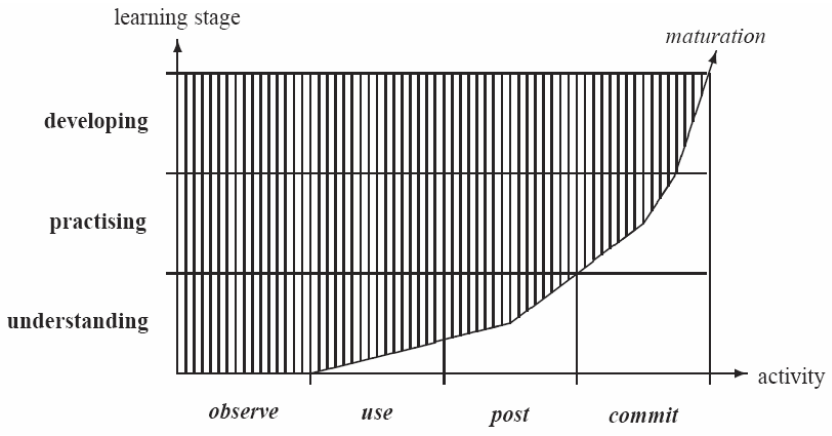

Figure 5: Learning Stages and contributor's activities in OSS communities

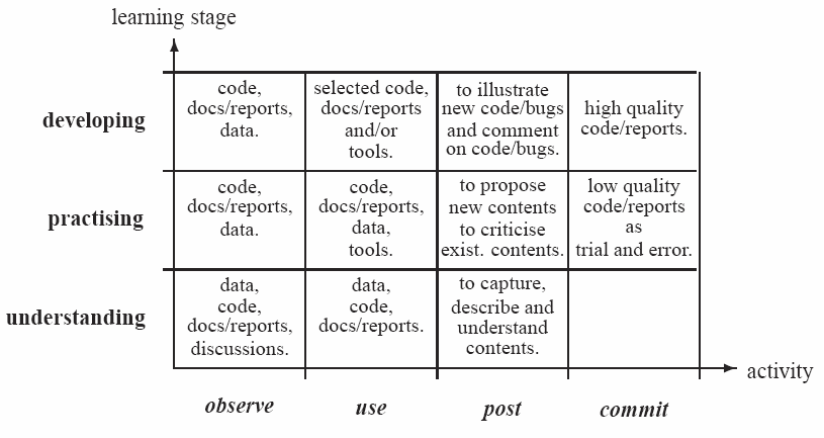

Figure 6: Learning Stages and Activity contents

3.2 Floss As E-learning Tools

As findings that support the occurrence of learning within FLOSS environments increase, practitioners in tertiary education have attempted to incorporate participation in FLOSS projects as a requirement for some Software Engineering courses [1]. A number of pilot studies have been conducted so as to evaluate the effectiveness of such an approach in traditional settings of learning [1][24][25].

FLOSS environments currently present an alternative approach that enables students to work on real-world problems for a more effective learning of software engineering as suggested by the joint IEEE/ACM CS undergraduate curriculum guidelines [29]. In this section we report on some results obtained from two pilot studies that were conducted for teaching Software Engineering at undergraduate and postgraduate levels [25]. We also look at findings from a similar experiment on a large scale for a longer period of time as presented in [24].

The first pilot study was conducted by Sowe and Stamelos in [25] and the objective was to assess whether FLOSS projects could be use as part of teaching Software Engineering courses as part of the curriculum in a formal learning. This study builds on an initial investigation conducted in [26] aimed at teaching software testing in order to help develop a methodology on how to assess student participation [25]. The study was conducted at Aristotle University in Greece where 15 students from a total of 150 enrolled students to "Introduction to Software Engineering" course participated in the study and 13 of whom completed. It consisted of three phases. The first phase was about introducing students to the FLOSS environments through lectures. During this period,

students had access to projects in FLOSS, they browsed through the projects and made a choice on which one to join; the second phase required students to participate in the chosen project in order to undertake a number of activities including finding and reporting bugs and possibly fixing them while the last phase was about students evaluation by their lecturers. At the completion of this study, two surveys were conducted among the students that took part in the pilot study, and the results indicated that most of them expressed the desire to prolong their participation in these FLOSS projects even after their graduation. Consequently, students continued to report on their activities to their lecturers after the conclusion of the pilot study [1].

The second study was conducted at postgraduate level by Jaccheri and Østerlie [27] . Their approach consisted of involving master's students to investigate and study the documentation and literature in OSS development and come up with possible research questions that can be solely addressed by participation in a FLOSS project. Students would then select a project related to the assignment and formulated research questions, then join the project as developers and also act as researchers in addressing the research questions. The results of this study are presented in [27] for a master student required to take part in a commercially controlled OSS project, the Netbean open source project, in order to study the various benefits that the project provides to firms [27]. The student was requested to determine "how the use of Software Engineering techniques, such as explicit planning, ownership, inspection and testing, affects the OSS project" [1].

In [24], authors present findings after four years of using FLOSS projects as tools to enhance software engineering teaching in formal learning. The experiments were conducted at the Aristotle University in Greece like in [25] on 408 junior students in Informatics. The respondents assumed different roles representing some aspects of software engineering such as requirements engineers, testers, developers, and designers/analysts. This study involved students enrolled for two courses namely "Introduction in Software Engineering" (ISE) and "Object-Oriented Analysis" (OOA) offered as part of the curriculum in a 4-year degree program. With about 150 enrolling every semester, the number of students participating in the experiment increased every semester over the duration of the study. Just like in the first pilot study, students were introduced to some background information on FLOSS and allowed time to get acquainted to the concept before they join a project of their choice to work on the assignments. In the first course, the assignments require students to adopt the roles of requirements engineer, tester, or developer, while they can choose to be either designer or analyst for the second course. This diversity of roles is intended to allow students the opportunity to experience FLOSS environments from different perspectives. This experience is vital for the entire class as these students regularly report to their classes on their activities. Furthermore, students were not restricted to specific roles as they were allowed to take up different responsibilities over the course of their involvement in FLOSS; testers could become developers and assist in fixing and providing code etc… However they were not allowed to play the same roles in the same projects.

The findings of this experiment indicate that students were able to accommodate to the new environment and successfully carry out their activities although working on real problems in a real world context present both challenges and motivations for them [24]. The results over the course of this investigation were positive as students could perform their tasks and undertake FLOSS activities and most of them indicated their desire to continue being part of the communities they joined after they completion of the assignments. They finally indicated that participation in FLOSS environments fosters skills development and knowledge acquisition.

In these studies, the freedom given to students for the selection of the project is key in ensuring that they finally complete their task. FLOSS environments are heterogeneous communities of "volunteers" and this constitutes an important aspect of this approach that needs to be preserved. Participants are endowed with different levels of skills and competency and this drives the decision for project choice.

4. PROCESS MINING TECHNIQUES FOR KNOWLEDGE DISCOVERY

Process mining is used as a method of reconstructing processes as executed from event [28]. These logs can be generated from process-aware information systems such as Enterprise Resource Planning (ERP), Workflow Management (WFM), Customer Relationship Management (CRM), Supply Chain Management (SCM), and Product Data Management (PDM) [3]. The logs contain records of events such as activities being executed or messages being exchanged on which process mining techniques can be applied in order to discover, analyze, diagnose and improve processes, organizational, social and data structures [29]. In [3], the goal of process mining is described to be the extraction of information on the process from event logs using a family of a-posteriori analysis techniques. These techniques enable the identification of sequentially recorded events where each event refers to an activity and is related to a particular case (i.e., a process instance). They also can help identify the performer or originator of the event (i.e., the person/resource executing or initiating the activity), the timestamp of the event, or data elements recorded with the event.

Current process mining techniques evolved from the work done in [28] where the purpose was to generate a workflow design from recorded information on workflow processes as they take place. Assuming that from event logs, each event refers to a task (a well-defined step in the workflow), each task refers to a case (a workflow instance), and these events are recorded in a certain order. The work in [28] combines the techniques from machine learning and Workflow nets in order to construct Petri nets that provide a graphical but formal language for modeling concurrency as seen in the figure below.

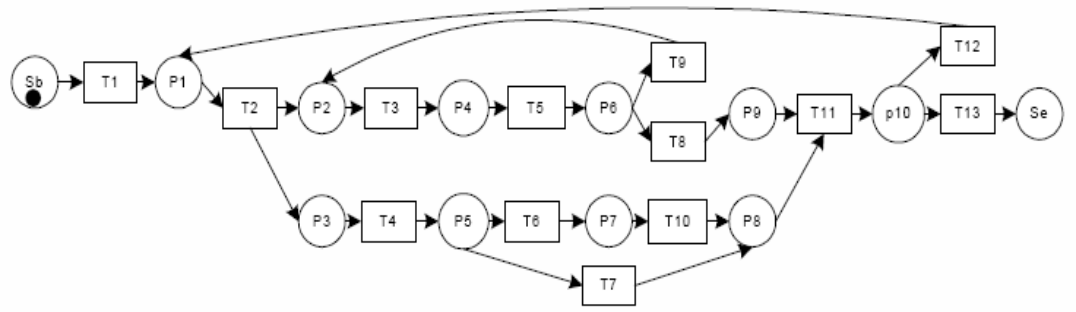

Figure 7: example of a workflow process modeled as a Petri net

The preliminaries of process mining can be explained starting with the α-algorithm of which formalization is given below.

Let W be a workflow log over T. α(W) is defined as follows.

1. $T_W = \{ t \in T \mid \exists_{\sigma \in W}\, t \in \sigma \}$,
2. $T_I = \{ t \in T \mid \exists_{\sigma \in W}\, t = first(\sigma) \}$,
3. $T_O = \{ t \in T \mid \exists_{\sigma \in W}\, t = last(\sigma) \}$,
4. $X_W = \{ (A,B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B}\, a \rightarrow_W b \wedge \forall_{a1,a2 \in A}\, a_1 \#_W a_2 \wedge \forall_{b1,b2 \in B}\, b_1 \#_W b_2 \}$,
5. $Y_W = \{ (A,B) \in X \mid \forall_{(A',B') \in X} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B') \}$,
6. $P_W = \{ p_{(A,B)} \mid (A,B) \in Y_W \} \cup \{i_W, o_W\}$,
7. $F_W = \{ (a,p_{(A,B)}) \mid (A,B) \in Y_W \wedge a \in A \} \cup \{ (p_{(A,B)},b) \mid (A,B) \in Y_W \wedge b \in B \} \cup \{ (i_W,t) \mid t \in T_I \} \cup \{ (t,o_W) \mid t \in T_O \}$, and
8. $\alpha(W) = (P_W, T_W, F_W)$.

The sequence of execution of the α-algorithm goes as follows [30]: the log traces are examined and in the first step, the algorithm creates the set of transitions *(T_W)* in the workflow, (Step 2) the set of output transitions (T_I) of the source place, and (Step 3) the set of the input transitions (T_O) of the

sink place. In steps 4 and 5, the α-algorithm creates sets (X_W and Y_W, respectively) used to define the places of the mined workflow net. In Step 4, it discovers which transitions are causally related. Thus, for each tuple (A, B) in X_W, each transition in set A causally relates to all transitions in set B, and no transitions within A (or B) follow each other in some firing sequence. Note that the OR-split/join requires the fusion of places. In Step 5, the α-algorithm refines set XW by taking only the largest elements with respect to set inclusion. In fact, Step 5 establishes the exact amount of places the mined net has (excluding the source place iW and the sink place oW. The places are created in Step 6 and connected to their respective input/output transitions in Step 7. The mined workflow net is returned in Step 8 [30].

From a workflow log, four important relations are derived upon which the algorithm is based. These are $>_W$, $\rightarrow_W$, $\#_W$, and $||_W$ [30]. In order to construct a model such as the one in Figure 7 on the basis of a workflow log, the latter has to be analyzed for causal dependencies [31]. For this purpose, the Log-based ordering relations notation is introduced:
Let W be a workflow log over T, i.e., $W \in P(T*)$. Let $a, b \in T$:

- $a >_W b$ if and only if there is a trace $\sigma = t_1t_2t_3 \ldots t_{n-1}$ and $i \in \{1,\ldots,n-2\}$ such that $\sigma \in W$ and $t_i = a$ and $t_{i+1} = b$,
- $a \rightarrow_W b$ if and only if $a >_W b$ and $b >_W a$,
- $a \#_W b$ if and only if $a >_W b$ and $b >_W a$, and
- $a ||_W b$ if and only if $a >_W b$ and $b >_W a$.

Considering the workflow log W = {ABCD, ACBD, AED}, relation $>_W$ describes which tasks appeared in sequence (one directly following the other). Clearly, $A >_W B$, $A >_W C$, $A >_W E$, $B >_W C$, $B >_W D$, $C >_W B$, $C >_W D$, and $E >_W D$. Relation $\rightarrow_W$ can be computed from $>_W$ and is referred to as the (direct) causal relation derived from workflow log W. $A \rightarrow_W B$, $A \rightarrow_W C$, $A \rightarrow_W E$, $B \rightarrow_W D$, $C \rightarrow_W D$, and $E \rightarrow_W D$. Note that $B \rightarrow_W C$ because $C >_W B$. Relation $_W$ suggests potential parallelism.

5. PROPOSED APPROACH

In light with all that has been said above, we propose an analysis of learning patterns in FLOSS environments through the application of process mining algorithms and Social Network analysis on recorded data to be extracted from repositories (SVN, emails and forums of discussion). An analysis of these repositories for learning patterns in participants' activities and message exchange will result in constructing Process Models that can be validated for conformance or learning patterns discrepancy with existing approaches as depicted in Figure 8. Hence, the motivation of this discussion is predicated on 3 important hypotheses:

[H1] Floss communities are indeed possible learning environments

[H2] FLOSS repositories can be analyzed using process mining and network analysis tools and techniques for learning patterns

[H3] New methods and supporting systems (Algorithms) for learning process identification in FLOSS can be developed and evaluated.

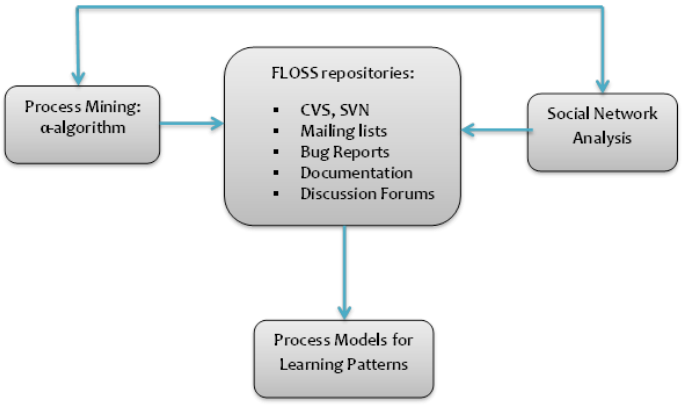

Figure 8: Generation of Process Models from Floss repositories using Process and Social Mining

In our approach, we hope to provide answers to questions related to as suggested in [8]:

1. how communication and development enable a natural *learning process* ;
2. how the linkage between learning process and basic activities drives evolution of activity patterns and maturation of participants in FLOSS;
3. how activity patterns can be analyzed to identify the presence of *learning patterns*

The implementation of the approach follows a non-sequential seven-step conceptual framework that will serve as our methodology. The steps include define learning vocabularies, determine Learning Processes, generate "semantic-oriented" event logs, generate and Interpret corresponding process models, build and interpret related Graphs, evaluate rate and levels of maturation for each found process and finally verify the hypotheses as formulated.

A certain level of iteration is to be maintained in the execution of this framework. This means that the steps don't necessarily have to be performed or undertaken strictly in a sequential fashion; however as the needs arise, one can still go back and forth in order to update or adjust the deliverables as required.

In step 1, the idea is to design and agree on a common definition of concepts or key words and phrases that one can use to identify whether or not learning has taken place during an interaction between FLOSS members. These keywords will guide the rest of the process in that they will help trace a learner's trail in mailing list, SVN, bug trackers and even forums in order to rate the progression of the learning process if any. Step 2, determining learning processes lay the foundation that will guide conclusions one can make about learning patterns in FLOSS in our defined context. In eLearning or virtual learning communities, there are some strategies that are put in place to explain the learning exchange channels. These identify the different avenues through which learning can occur. We refer to such categorization as learning processes. In the context of our work, we have chosen and adapted only two learning processes in FLOSS communities with the second learning process (Directed Learning) unfolding from 2 perspectives in 4 different formats. These are:

- Undirected Learning: This process can also be referred to as Peer-2-Peer or Reflective Learning. This learning is assumed to take place between any numbers of participants. In this process, any participant can be both a receiver (Learner) and a sender (Teacher). At this level, the assumption is that learning occurs between mates with a diversified expertise background who learn from each other.
- Directed Learning: This process refers to involvement of more knowledgeable participants or expert members in helping less expert members to develop their skills with some level of guidance or supervision. The occurrence of the process is twofold:

- Pulling: This is the process where a less expert on any topic would initiate a need to learn by reaching out to the more advanced that can culminate in a supervised or guided learning process. This can also in turn occur following 4 formats as follow:
  - Modeling: In this process, the guide (sender) activities and actions are systematically monitored and observed by the receiver. This can happen as the receiver aims to emulate the sender given the latter's reputation on their FLOSS contribution. An example could be tracking the sender's commits in SVN, their comments on mailing lists etc.;
  - Coaching: As the term explains, this involves giving direct monitoring and guidance to the requester's and observing his/her performance;
  - Scaffolding: In this process, the sender analyses and determines the receiver's level of capacity and allows him/her the opportunities to acquire knowledge accordingly. For example, supplying materials (tutorials etc.) on specific problems and a solution approach etc. based on the requester's background.
  - Fading: This process depicts involving a requester in practical execution of tasks for skills acquisition. However, as the requester's performance matures, the sender gradually gives them autonomy to apply their skills.
- Pushing: This is the type of directed learning that occurs when the sender takes the initiative to make available opportunities of knowledge acquisition for requesters. Just like the pulling, this process can also be understood in 4 formats: Modeling, Coaching, Scaffolding and Fading

The understanding and analysis of these learning processes will also depend on the learning stages as discussed in the previous sections. Having discussed the learning avenues or learning processes, in this work we will define them in light with the learning stages manifested through participants or more specifically requester's and sender's activities.

Hence, this layer of the framework can be summarized as follows in figures 9 and 10. Each represents the major categorization of learning processes: Undirected and Directed.

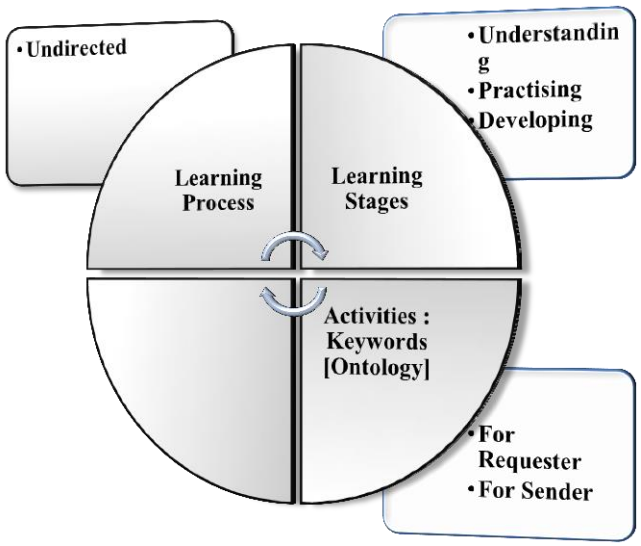

Figure 9: Graphical Representation of first learning process category.

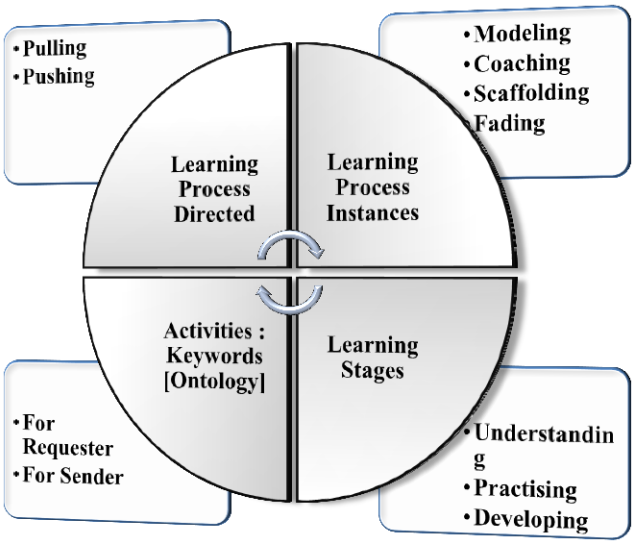

Figure 10: Graphical Representation of second learning process category.

Therefore, given this classification, one can identify nine learning processes, namely Undirected/Reflective Learning, Directed-Pulling-Modeling, Directed-Pulling-Coaching, Directed-Pulling-Scaffolding, Directed-Pulling-Fading, Directed-Pushing-Modeling, Directed-Pushing-Coaching, Directed-Pushing-Scaffolding and Directed-Pushing-Fading.
For each of these stages, we look to model the activities for both parties involved in the learning processes through the three stages of learning.

The third step of the methodology entails the preprocessing task of process mining. This is related to the generation of the log that will be used to construct the process models. We refer to the event logs as semantic oriented because the events we will return will be based on the contents' meaning of the message exchanged. This will be guided by the use of the ontologies to be developed. The following step concerns the generation and analysis of the corresponding process models that explain how and what kind of activities govern the learning processes in FLOSS repositories. Next, we build directed graphs where necessary to explain the flow of learning processes and maturation levels of participants' activities. This is achieved by using social network analysis. Social network Analysis techniques can help us in locating knowledge brokers. Knowledge brokers are hubs or experts that are active across different FLOSS repositories. Also, it can help us in establishing the exchange of mails or interactions between senders (teachers) and receivers (learners) and also identify clusters of nodes that could explain undirected or even directed learning. The last two steps entail the evaluation of learning process execution as well as some insights on dynamism behind learning process suitability based on the levels of participants' activities maturation. The levels of maturation can be studied and understood using Social Network Analysis. This is done by looking at the degree of participation exhibited by the learners for example in forms of number of emails (weighted tie). Thus social network analysis offers an important mechanism for the discovery of key individuals in FLOSS communities, and help us determine who is who, who is interacting with who, how much individuals are contributing to community discourse.

6. CONCLUSION

FLOSS communities are an important phenomenon that has raised interest from different perspectives of research. Given the quality of FLOSS products, the environments appear to provide learning opportunities for participants. While this aspect has been investigated with reports such as in [5][6], we believe that is a need for empirical to support such findings. An important remark from [8] emphasizes that in such reports, 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. We hope that this work will help support evidence about learning in FLOSS environments through empirical analysis of FLOSS data. This can be accomplished through extracting information from FLOSS repositories in order to identify patterns, progress, evolution and achievements within participatory groups [8].

From their immense work on workflow logs and process mining, [31] highlight two fundamental reasons for process mining on event logs. The first purpose that the method serves is to discover how people and procedures really work. An example could be the understanding of flow of patients in a hospital. While information about activities in these environments is available, the actual underlying process is lacking. The second reason is the use of process mining for Delta analysis. This implies a comparison of results from the actual process with some predefined process. With the existence of descriptive or prescriptive models that specify how procedures are assumed to occur or how people in organizations are expected to work, comparison with the results of the actual process can help in detecting discrepancies in order to improve the process. We hope that the results of our work can help in certifying the "existing models" that describe how learning occurs or provide new insights that can help enrich this area of research.

REFERENCES

[1] Cerone, A. K., & Sowe, S. K. (2010). Using Free/Libre Open Source Software Projects as E-learning Tools. *Electronic Communications of the EASST*, *33*.

[2] 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

[3] van der Aalst, W. M., Rubin, V., Verbeek, H. M. W., van Dongen, B. F., Kindler, E., & Günther, C. W. (2010). Process mining: a two-step approach to balance between underfitting and overfitting. *Software & Systems Modeling*,*9*(1), 87-111

[4] van der Aalst, W. M., van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G., & Weijters, A. J. M. M. (2003). Workflow mining: a survey of issues and approaches. *Data & knowledge engineering*, *47*(2), 237-267.

[5] 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.

[6] Glott, R., Meiszner, A., & Sowe, S. K. (2007). „FLOSSCom Phase 1 Report: Analysis of the Informal Learning Environment of FLOSS Communities".*FLOSSCom Project*.

[7] Aberdour, M. (2007). Achieving quality in open-source software. *Software, IEEE*, *24*(1), 58-64.

[8] Cerone, A., Fong, S., & Shaikh, S. A. (2011). Analysis of collaboration effectiveness and individuals' contribution in FLOSS communities. *Proc. OpenCert*, *44*, 48.

[9] Raymond, E. S. (2008). *The Cathedral & the Bazaar: Musings on linux and open source by an accidental revolutionary*. O'Reilly.

[10] Cerone, A. (2011). Learning and Activity Patterns in OSS Communities and their Impact on Software Quality. *ECEASST*, *48*.

[11] Halloran, T. J., & Scherlis, W. L. (2002, May). High quality and open source software practices. In *2nd Workshop on Open Source Software Engineering*.

[12] Ghosh, R.A., Glott, R,. Krieger, B. and Robles, G., 2002. Free/Libre and Open Source Software: Survey and Study.

[13]Ghosh, R., & Glott, R. (2008). FLOSSPOLS Skill Survey Report, 2005

[14] Krishnamurthy, S. (2002). Cave or community?: An empirical examination of 100 mature open source projects. *First Monday*.

[15] Sowe, S. K., & Cerone, A. (2010). Integrating Data from Multiple Repositories to Analyze Patterns of Contribution in FOSS Projects. *Electronic Communications of the EASST*, *33*.
[16] Hassan, A. E. (2008, September). The road ahead for mining software repositories. In *Frontiers of Software Maintenance, 2008. FoSM 2008.* (pp. 48-57). IEEE.
[17]Xu, J., Gao, Y., Christley, S., & Madey, G. (2005, January). A topological analysis of the open souce software development community. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on* (pp. 198a-198a). IEEE.
[18]Weller, M., & Meiszner, A. (2008). FLOSSCom Phase 2: Report on the effectiveness of a FLOSS-like learning community in formal educational settings. *FLOSSCom Project*.
[19] 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.
[20]Fernandes, S., Cerone, A., & Barbosa, L. S. Analysis of FLOSS communities as learning contexts.
[21]Fernandes, S., Barbosa, L. S., & Cerone, A. FLOSS Communities as Learning Networks.
[22]Fernandes, S., Cerone, A., Barbosa, L., & Papadopoulos, P. FLOSS in Technology--Enhanced Learning.
[23] 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.
[24]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*, 1-17.
[25]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.
[26]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.
[27] 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.
[28] Weijters, A. J. M. M., & Van der Aalst, W. M. P. (2001, October). Process mining: discovering workflow models from event-based data. In *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001)* (pp. 283-290).
[29] De Weerdt, J., Schupp, A., Vanderloock, A., & Baesens, B. (2012). Process Mining for the multi-faceted analysis of business processes—A case study in a financial services organization. *Computers in Industry*.
[30]de Medeiros, A. K. A., van der Aalst, W. M., & Weijters, A. J. M. M. (2003). Workflow mining: Current status and future directions. In *On the move to meaningful internet systems 2003: Coopis, doa, and odbase* (pp. 389-406). Springer Berlin Heidelberg.
[31] Van der Aalst, W., Weijters, T., & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *Knowledge and Data Engineering, IEEE Transactions on*, *16*(9), 1128-1142.

BIBLIOGRAPHY OF AUTHOR

Patrick Mukala is currently a PhD Candidate in Computer Science at the University of Pisa in Italy. Holding a Masters degree in Software Engineering as well as a Masters degree in Information Networks from the Tshwane University of Technology in Pretoria (South Africa), his doctoral work spans from process and data mining in FLOSS repositories to complex networks and web mining.