PROCESS-BASED ASSISTANCE METHOD FOR LEARNER ACADEMIC ACHIEVEMENT

Joffrey Leblay[1], Mourad Rabah[1], Ronan Champagnat[1] and Samuel Nowakowski[2]
[1]*L3i Laboratory - University of La Rochelle, Pôle Science et Technologie avenue Michel Crépeau, 17000 La Rochelle, France*
[2]*LORIA - University of Lorraine, Campus Scientifique, 615 Rue du Jardin-Botanique, 54506 Vandœuvre-lès-Nancy, France*

ABSTRACT

How can we learn to use properly business software, digital environments, games or intelligent tutoring systems (ITS) ? Mainly, we assume that the new user will learn by doing. But what about the efficiency of such a method? Our approach proposes an answer by introducing on-line coaching. In learning process, learners may need guidance to help them in their academic achievement. In this paper, we introduce a process-based assistance method to provide this help. Our method proposes to build a model using process mining upon the observations collected during previous users' experiences with the considered application. It represents the steps chaining and their impacts on the states of the overall process and is used to recommend the most suitable step to guide the current user or learner. We implemented our coaching approach in the La Rochelle University Institute of Technology jury decision system to show its relevance.

KEYWORDS

Process Mining, Recommendation, Learning, Traces-based System

1. INTRODUCTION

The aim of our work is to assist users involved with a business process, by guiding them in order to find the best way to complete successfully the process. Without guidance, users can fail at reaching business process objective, which may have severe consequences on the overall process or the information system, such as the loss of sensitive data, hampering service-providing processes of a company, etc. To prevent such failing situations, we can observe previous users' behaviors to extract chains of steps that represent the organization of activities within the application and their impacts on the application state. Both information serve to build, step by step, a way to reach the goal from the actual state of the business process. Therefore, we can recommend the next activity to perform in order to achieve the given goal.

This issue has been studied in some previous works such as Cordier et al. (2013), Ho et al. (2018), Toussaint & Luengo (2015), to name a few. In Toussaint & Luengo (2015), the authors introduced a method to supervise surgeon student during their practical work and tried to prevent their potential errors by using a data model that identifies the potential steps that could describe an error. The data model, called process model, represents a schedule of steps done by the users (Van der Aalst & Weijters 2004, Polyvyanyy et al. 2017). In our case, we try to guide users to achieve their goals even if they make some errors. Therefore, our approach suggests a corrective way, if necessary. In Ho et al. (2018), the authors introduced a method to drive the users to take the right decisions according to some information extracted from a data model that describes the possible activities to perform to reach the goal and their impact. This data model is an input to the method, in the sense that it is built a priori by field experts. In our case, our method discovers the data model from the information before recommending the next step. In Cordier et al. (2013), the authors defined a data model based on traces that are interaction logs between the users and the software in previous executions to help the users take decisions. Such a help can be provided by presenting a ranking of the most used activities. We borrowed from Cordier et al. (2013) the data model and adapted it to our context to develop a process-based learning assistance method that we aim to use in ITSs.

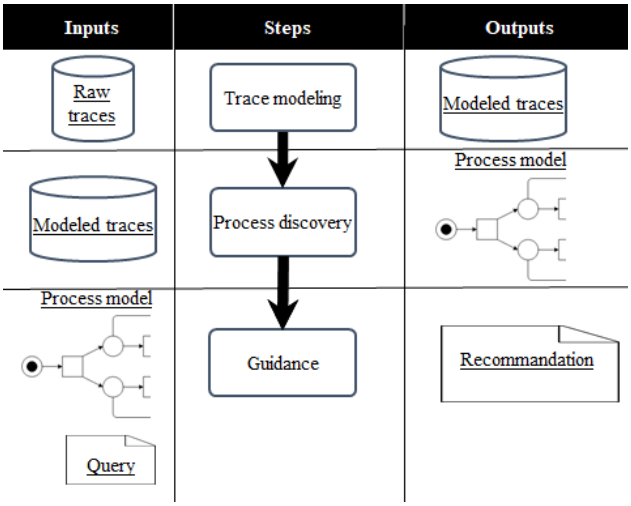

Figure 1. Process-based learning assistance method steps

In this paper, we describe our process-based assistance method that searches in given data, such as traces, for information about the organization of activities on the application state. It consists in three steps as follows: (i) we transform the given traces for process model discovery, (ii) use the transformed traces to obtain the process model, and (iii) calculate the most efficient path to the defined goal and recommend it to the user. We have applied our method on the jury decision system of La Rochelle University Institute of Technology to recommend to the students the best semester validation path to achieve their studies and get the qualification of the Institute of Technology. The obtained results show that our learning assistance approach provides relevant recommendations.

Section 2 describes our process-based learning assistance method. Section 3 presents our experimentation and discusses the results obtained. Section 4 concludes the paper and presents some future works.

2. PROCESS-BASED LEARNING ASSISTANCE SYSTEM

Our approach explores system's traces to discover the activities' organization and past users learning processes that are seen as business processes. Then, it uses the obtained model to guide current learners through the application or the learning environment. As illustrated in figure 1, our method consists of three steps: (i) trace modeling, (ii) process discovery and (iii) guidance of the user.

2.1 Trace Modeling

This step transforms raw traces collected by the system into modeled ones corresponding to the entry data model of our approach. We can see raw traces as data, where each entry is a tuple of values of attributes with different domain of definition. For example, let "`A,Case 1,2,1`" be a row of a set of traces in CSV[1] format. The first and second attributes are string type, and the third and fourth attributes are integers.

More formally, let T be a set of traces. Each tuple $t \in T$ is such that $t = (v_1,..., v_n)$ where $\forall i \in \{0,...,n\}$, $v_i \in \mathbb{D}_i$ *and* $\mathbb{D}_i$ being the domain of definition of the attribute r_i associated to v_i (Cordier et al. 2013, Rozinat & van der Aalst 2008). The set of all attributes is denoted R. We associate to them two functions: (i) $f_R(r_1,...,r_j)$, the projection of T on attributes $r_k \mid 1 \leq k \leq j \leq n$ as it is defined in the relational algebra for databases and (ii) $f_T(t_i)$, the selection of row (tuple) t_i in T with $i = 1,...,|T|$.

We have to identify the information we need from the raw traces, such as the organization of the observed activities and their impacts. Therefore, we use Event Log (Leemans et al. 2014) as a trace model. We introduce hereafter only the needed tools for the Event Log model; a more complete presentation can be found in Verbeek et al. (2010). The model describes the schedule of users' process by using the organization

[1] Comma Separated Values text format.

of activities and their impacts. Therefore, this model has to identify: (i) the activity performed inside the observed data, (ii) an identifier that defines to which sequence the observation belongs. We consider that the traces are ordered chronologically according to their activities achievement time. Furthermore, we have to identify the impact of the activity by extracting the subset of data that describes the state of the process used through the application before or after the activity is performed.

Each modeled trace is an entry $e = (a,l,o)$ with an activity $a \in \mathcal{A}$, a sequence identifier $l \in L$ and a set of data that describes the impact of the considered activity $o \in O$. We consider the set of entries $\mathcal{E} = \mathcal{A} \times L \times O$ and we can get or set the values of an entry $e \in \mathcal{E}$ with the following three functions:

$$f_a : \mathcal{E} \rightarrow \mathcal{A} \qquad e \rightarrow f_a(e) = a$$

$$f_l : \mathcal{E} \rightarrow L \qquad e \rightarrow f_l(e) = l$$

$$f_o : \mathcal{E} \rightarrow O \qquad e \rightarrow f_o(e) = o$$

In the trace modeling step, we use the knowledge about the location of activity, sequences identifier and the other data, to model the raw traces. To transform the raw traces into event logs, we need to match attributes of event logs with attributes of raw traces. Each attribute of event log is a Cartesian product, denoted $\prod$, of a subset of raw traces attributes such that $\mathcal{A} = f_R(r_1,...r_n)$, $r_i \in R_a$, $1 \leq i \leq n$, $R_a \subseteq R$, $L = f_R(r_1,...,r_n)$, $r_i \in R_l$, $1 \leq i \leq n$, $R_l \subseteq R$, and $O = f_R(r_1,...r_n)$, $r_i \in R_o$, $1 \leq i \leq n$, $R_o \subseteq R$.

With functions f_a, f_l and f_o, we create each entry e from the raw traces. We use formulas (1) to associate activity (f_a), sequence identifier (f_l) and other data (f_o) to an entry, as follows:

$$f_a(e) \leftarrow \prod_{r \in R_a} f_R(r) \cap f_T(t) \qquad f_l(e) \leftarrow \prod_{r \in R_l} f_R(r) \cap f_T(t) \qquad f_o(e) \leftarrow \prod_{r \in R_o} f_R(r) \cap f_T(t) \qquad (1)$$

Example 1: Table 1 shows some modeled traces corresponding to raw traces in the form of "`A,Case 1,2,1`" presented earlier. To transform the entry t of raw traces into the entry e of modeled traces, we associate the first attribute to R_a (the subset on activity attributes), the second one to R_l (the subset of sequence identification attributes), the third and fourth attributes as R_o (the subset of other attributes). Hence, the first line $e_1 = (A,Case1,\{2,1\})$ is obtained by considering the activity $a =$ *"A"*, its sequence identifier $l =$ *"Case1"* and the other data $o = (2,1)$.

For $\mathcal{E} = \mathcal{A} \times L \times O$ the overall sets of the above example are: $\mathcal{A} = \{A,B,C,D,E,F\}$, $L = \{Case1,...,Case5\}$ and $O = \{\{2,1\},\{2,2\},\{2,3\},\{1,4\},\{1,1\},\{1,2\}\}$

Table 1. Modeled traces

Order identifier	Activity	Sequence identifier	Other data 1	Other data 2	Order identifier	Activity	Sequence identifier	Other data 1	Other data 2
e_1	Task A	Case 1	2	1	e_{10}	Task D	Case 2	1	4
e_2	Task A	Case 2	2	1	e_{11}	Task E	Case 5	1	1
e_3	Task A	Case 3	2	1	e_{12}	Task C	Case 4	2	2
e_4	Task B	Case 3	2	2	e_{13}	Task D	Case 1	1	4
e_5	Task B	Case 1	2	2	e_{14}	Task C	Case 3	2	3
e_6	Task C	Case 1	2	3	e_{15}	Task D	Case 3	1	4
e_7	Task C	Case 2	2	2	e_{16}	Task B	Case 4	2	3
e_8	Task A	Case 4	2	1	e_{17}	Task F	Case 5	1	2
e_9	Task B	Case 2	2	3	e_{18}	Task D	Case 4	1	4

2.2 Process Discovery

From the modeled traces, we extract the schedule of activities as a business process that reaches a given goal. We admit that the business process always ends even if the users goal is not necessarily reached, and whatever the path the execution takes. Hence, failing is seen as a process reachable goal. This is the soundness property of a process model as defined in Leemans et al. (2014). We need an algorithm that verifies that property and among the existing process discovery algorithms that we tested Inductive Miner (IM) (Leemans et al. 2014) is the only one able to extract a sound process model in finite time for all given modeled traces.

IM provides sound structured workflow-net from traces. A structured workflow-net is a representation of activities organization. It is a variant of Petri-net model. In this paper, we introduce the needed notion for our method. A complete introduction to Petri-nets and workflow-nets can be found in Desel & Esparza (2005),

Reisig & Rozenberg (1998), Murata (1989), van der Aalst (2016). We consider a process model constituted with a set of transitions, a set of places and a set of directed arcs. The set of transitions represents a potential activity performed, the set of places represents the state of the process and the set of directed arcs represents the condition of utilization of an activity if the arc links a place to a transition, and a realization of the condition otherwise. A structured workflow-net, or simply SWF-net, is a tuple $(P,W,\mathcal{F})$ where:

- P is a finite set of places, with an input place i (no arc to it) and an output place u (no arc from it);
- $\mathcal{F} \subseteq (P \times W) \cup (W \times P)$ is a set of directed arcs, called the flow relation,
- W is a finite set of transitions such that $P \cap W = \emptyset$, and all transitions W can reach any other transitions in the case of adding an arc from u to i.

The process model in Figure 2 is a SWF-net corresponding to the modeled traces of Table 1 and respecting the soundness property. This type of model offers standard building blocks such as AND-split, AND-join, OR-split and OR-join. They are used to model sequential, conditional and parallel routing (Russell et al. 2016). The sequential routing is a succession of activities such as E and F in Figure 2, the conditional routing is a choice between some activities such as A and E, and the parallel routing is some activities used in any order such as B and C. These routings allow to schedule the activities and are used by the process mining algorithm to extract the process model.

To check if a condition is met, we add to the model an overlay, named marking, which describes the number of token on each place. A token represents an achieved condition and counts only zero or one token per place since we consider a condition cannot be completed more than once.

A marked SWF-net is a pair (M,s), where $M = (P,W,\mathcal{F})$ is a SWF-net and where $s \in \mathbb{N}^{|P|}$ is a bag over P denoting the marking of the net. Let M be a SWF-net, x and y be two places of M, w be a transition of M, $x\mathcal{F}w$ denote an arc from x to w and $w\mathcal{F}y$ denote an arc from w to y. Let us define the following notations:

- $\mathcal{F} = \{x\mathcal{F}w, w\mathcal{F}y\}$ denotes the set of directed arcs;
- $s = [x]$ denotes a marking s over M that has a token on the place x but not on y. A place is in a marking if it has at least one token in the marking;
- $x \in s$ denotes the fact that x has at least one token on marking s. Otherwise, it is denoted by $x \notin s$,
- $s(x) = 1$ and $s(y) = 0$ denote the number of token in a place;
- If all places before a transition are marked then the transition can be fired and $(M,s)[w\rangle(M,[y])$ denotes the firing of w.

Let us add to M one transition w', one place z and two arcs $y\mathcal{F}w'$ and $w'\mathcal{F}z$. Then let denote:

- $(M,[x])[w,w'\rangle(M,[z])$ the fact that firing w and w' or $(M,[x])[\sigma\rangle(M,[z])$ if we consider a sequence $\sigma = \langle w,w' \rangle$;
- $[z] \in [M,[x]\rangle$ the fact that the marking $[z]$ is reachable from the marking $[x]$ in the marking M.

IM process mining algorithm extracts process models from modeled traces as SWF-net. We provide modeled traces $\mathcal{E}$ obtained above and it returns the process model $M = (P,W,\mathcal{F})$. This algorithm works in one recursive step. It observes the sequences in the given set of modeled traces and find the set of activities $\mathcal{A}$. Then it makes a cut into the activities to split the set into two other subsets. The cut is done if and only if $\mathcal{A}_1 \neq \mathcal{A}_2 | \mathcal{A}_1 \cup \mathcal{A}_2 = \mathcal{A}$. After that, it searches for a routing that explains the link between the two parts of the split. There is four possible cuts or routings: exclusive choice cut, sequence cut, parallel cut and redo loop cut. To find the right routing, the algorithm uses a model, which describes the routing with some rules found in the considered sequences. Then this step is recursively done as long as the sequences, i.e. the subset of activities, have more than one activity inside. A more complete introduction to IM can be found in Leemans et al. (2014).

Example 2: Let use the IM algorithm on the traces in Table 1 where $\mathcal{A} = \{A,B,C,D,E,F\}$. We first perform a parallel cut to obtain $\mathcal{A}_1 = \{A,B,C,D\}$ and $\mathcal{A}_2 = \{E,F\}$. Then, we apply the algorithm on the two resulting subsets. For instance, with the sequence $\langle E,F \rangle$, which defines $\mathcal{A}_2$ and is constituted with the entries e_{11} and e_{17} of $\mathcal{E}$, IM will perform a sequence cut to obtain $\mathcal{A}_3 = \{E\}$ and $\mathcal{A}_4 = \{F\}$ and do not go farther since $|\mathcal{A}_3| = |\mathcal{A}_4| = 1$. Then, it searches for the cut on $\mathcal{A}_1$ and so on to definitively stop when there is no more recursive cut to run.

To sum up, the process discovery step uses the event log obtained from the previous step to extract the scheduling of the activities by using a process-mining algorithm. To find the scheduling, we use the IM algorithm. The next step is about how to use the process model to find a path to guide the user.

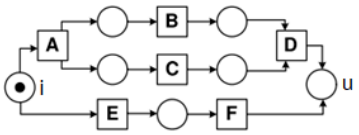

Figure 2. Process model corresponding to the modeled traces of Table 1

2.3 Guidance of the User

The aim of this step is to find the path to reach a given goal from the current progression of the user in the system. The guidance step uses the process model and waits for a query that only contains an incomplete sequence σ_x. This query is made when a user performs an activity according to the studied overall business process. The guidance step verifies if the sequence σ_x follows the process model by checking if a marking s exists after firing the sequence of activities, such that $(M,[i])[\sigma_x\rangle(M,s)$, where $M = (P,W,F)$ is the process model and i its input place. If no marking s exists, the query is skipped. If the marking exists, we try to find a sequence σ_z that reaches u the output place of the model, such that $(M,s)[\sigma_z\rangle(M,[u])$. We search for a sequence that reaches the output with the lower number of activities, using a path-finding algorithm.

The path-finding algorithm is a way to obtain a path in a model that describes a relation between entities, such as our model that describes the schedule of activities. Some path finding algorithms using different approaches have been tested. We consider the number of activities as quality of a sequence because we just need a path that completes the process model. Therefore, Dijkstra's algorithm (Heineman et al. 2016) is enough to solve our problem by searching for the shortest path in a set of sequences to the goal. This algorithm iterates three steps:

1. We consider a list of candidate sequences (at the beginning it only contains the requested sequence). For each sequence, we search for all firable transitions and make a list of possible resulting sequences;
2. From the resulting list, we keep the shortest sequence, i.e. with the lower number of activities, not already in the candidate list. We choose one randomly if several and add it to the candidate sequences list;
3. We check if the kept sequence reaches the output place u, this sequence is returned and the algorithm stops, otherwise, we continue with the next iteration.

The returned path describes the succession of activities completing the process model. This path contains the sequence σ_z that reaches u from the marking s corresponding to the query sequence σ_x: $(M,[i])[\sigma_x\rangle(M,s)[\sigma_z\rangle(M,[u])$. The final candidate sequences list is kept for the next query.

Example 3: Let us consider the process model obtained in the previous section from the modeled data of Table1. Let us also consider a query with sequence $\sigma_{[i]}$, the starting empty sequence with no activity inside, the marking $s = [i]$. First iterations of the algorithm are:

Iteration 1 the candidate sequences' list contains only the empty sequence $\sigma_{[i]}$. The firable transitions are A and E. The resulting sequences are σ_A containing only the activity A and σ_E containing only the activity E. Both are 1-activity length and suppose that we randomly choose σ_E that is added to the candidate sequences list. u is not reached, the execution moves on.

Iteration 2 the candidate sequences' list contains $\sigma_{[i]}$ and σ_E. The firable transitions are A and F. The resulting sequences are σ_A containing only the activity A and σ_{EF} containing E and F. σ_A is the shortest and is added to the candidate sequences list. u is not reached.

Iteration 3 the candidate sequences' list contains $\sigma_{[i]}$, σ_A and σ_E. The firable transitions are B, and F. The resulting sequences are σ_{AB} containing A and B, σ_{AC} containing A and C, σ_{EF} containing E and F, all of same 2-activities length. Let suppose that we randomly choose σ_{EF} that is added to the candidate sequences list. However, firing F reaches the output place u, the execution stops, and the sequence σ_{EF} is returned.

3. EXPERIMENTAL RESULTS

The context of our experiment is the academic jury decision taken at the end of each semester at the University Institute of Technology of La Rochelle (France). The institute delivers a two-year technical degree that requires the validation of 4 semesters. We apply our method on the jury delivery data in order to identify students' progression, lack of learning and courses with high fail grade. The process considered in this case study is the semester validation process followed by the students. This process can be analyzed with different granularity points of view. For instance, we can consider the courses as activities for the most accurate point of view and the semesters' validation decision for the most generic one.

For our experimentation, we consider the semesters' decisions as activities in order to have an overall view of student academic achievement (or fail). This point of view considers the chaining of decision the students got. Thanks to this, it is possible to identify the learners who need remediation to reach their qualification. However, it is also possible to look at the courses that the learner fails at. This increases the accuracy of the provided recommendation.

3.1 Context of the La Rochelle Institute of Technology

In the La Rochelle Institute of Technology, some students have difficulties to achieve their semesters due to their initial academic lacks in some specific courses or to their insufficient motivation. Our first goal is to use our process-based method on jury statistical data to identify the fail paths and the no classical success path thanks to the process model. Then the results could be correlated with learners' marks to give them a more precise recommendation (but this part is out of the scope of this paper).

French Institutes of Technology have normalized format for jury decisions. Each semester (numbered 1 to 4) may be: *V* for the automatic validation if student results meets the academic requirements; *C−* or *C+* for a validation with compensation between two successive semesters (*C−* denotes the compensation by the previous semester and *C+* by the next one); *N* for a no validation caused by not sufficient academic results; *J* for a validation granted by the jury when the academic requirements are not met but student's results are close to what expected and *E* if the student fails and is not allowed to pursue for any reason. Each decision is followed by the concerned semester level number. For instance, *V//S1* means that the student had sufficient results and the first semester was automatically validated. An additional code *DEM* is also possible and denotes student's resignation.

The considered data in our study was provided, after anonymization, by the Computer Science Department of La Rochelle Institute of Technology and concerns 6 years of jury decisions (2007-2013). Each year, about 100 students start the first semester in September. For each semester, the studies are organized in two Learning Units (LU) each of which includes several Teaching Units (TU). To validate a semester a student must have at least 10/20 overall average mark and at least 8/20 average mark in each Learning Unit.

3.2 Technical Process

The row traces format is: "`LearnerId, Decision, Semester, Date, EvalAvr, EvalLU1, EvalLU2, EvalTU11, ... EvalTU1n, EvalTU21,...EvalTU2m`". ***LearnerId*** is the learner identifier assigned after the preliminary anonymization step. ***Decision*** is the jury decision with one of values in *{V, J,N,C+,C−,A,E,DEM}*. ***Semester*** is the semester level. ***Eval**** are student marks during the considered semester where ***EvalAvr*** is the overall average mark, ***EvalLU1*** and ***EvalLU2*** are average marks in the two semester's Learning Units and ***EvalTU**** are the detailed marks in each Teaching Unit that we do not consider in for this experiment.

The raw traces are transformed into modeled traces according to the Event Log Model (cf. 2.2). The model needs the sequence identifier and the activity. We instantiate the sequence identifier with ***LearnerId*** and the activity with the couple ***Decision***, ***Semester***. The remaining row data attributes represent other attributes of the trace model.

For the sake of clarity, we prefer to present the Markov model[2] corresponding the SWF-net instead of the SWF-net[3] itself. Nevertheless, the guidance step is applied on the SWF-net as stated in section 2.3. Figure 3 shows the obtained model. Each state of the Markov model represents the most frequent decision and each arrow points out the most probable activity after the considered one. The model is a second order Markov model shows the possible preceded and followed linking. Furthermore, the presented model is a compacted model, i.e. all the states do not appear. The states with lower incidence are grouped together.

After that, we use the models, such as the Markov model, to compute the sequences as explained in Section 2.3. The sequences are verified to know if they respect the French Institutes of Technology decision rules previously presented. If they do, we consider our method able to recommend a way to get qualification.

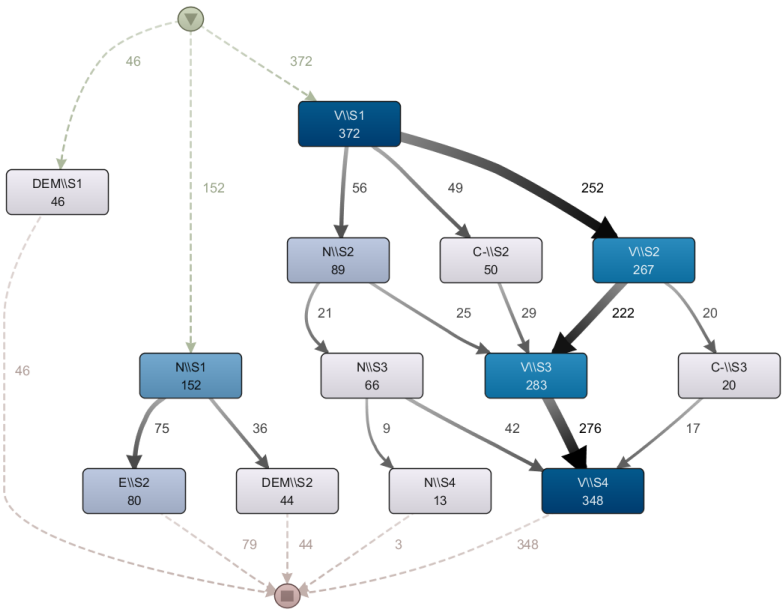

Figure 3. Chaining of Jury decision for each semester[2]

3.3 Discussion

The model in Figure 3 shows the frequency of activities used in the studied process. We can see that globally validating activities are the most frequent ones. However, the second and third level semesters validate decisions are less frequent than the first and the fourth one. This means that there are alternative ways to get the qualification. That also means that efforts should be made on students learning process in these two semesters. We can also observe the most frequent failing paths and that the students mainly fail after first or second semesters. Assistance should be provided by the Computer Science Department to help weak students from the first semester.

Furthermore, our method automatically warns the student and his teachers if fail path is detected. And the method recommends the closest success path towards the qualification.

[2] Obtained by the software Disco (https://fluxicon.com/disco/)
[3] Obtained by using IM algorithm from the ProM process mining software (http://www.promtools.org)

4. CONCLUSION

We have introduced a process-based assistance method to help learners in their academic achievement. The developed method provides learners a progression sequence to follow in order to reach the academic overall objective. Our method uses the process mining on previous learners' results in order to build the process model that serves to guide current learners. The method identifies the sequences performed by the learners in the raw traces through a trace modeling. Then, builds the process model based on the scheduling of identified activities extracted from the modeled traces. This model is used to determine the sequence that describes a chaining of activities the learner should follow by applying a path-finding algorithm. We have made some experimentation to show the applicability and the relevance of our method on the La Rochelle Institute of Technology case study.

The short-term perspectives are twofold. On the presented case study, we seek to improve the efficiency of our method by rising the granularity of our data and consider correlating of the overall semester results with student's lectures results. For instance, we want to analyze student's academic failures to find to which extent a bad result in maths or programming course impacts the overall jury decision. And use this analysis to recommend an appropriate remediation to the learner. The second perspective is to check how our method can be generalized to be used on applications based on enterprise information system. It is also possible to improve our method by finding some other sources of knowledge to enrich the mined model.

REFERENCES

Cordier, A., Lefevre, M., Champin, P.-A., Georgeon, O. L. & Mille, A., 2013, Trace-Based Reasoning-Modeling Interaction Traces for Reasoning on Experiences, *Florida Artificial Intelligence Research Society Conference (FLAIRS)*, Florida, USA, pp. 363–368.

Desel, J. & Esparza, J., 2005, Free choice Petri nets, *Cambridge university press*.

Heineman, G. T., Pollice, G. & Selkow, S., 2016, Algorithms in a nutshell: A practical guide, *O'Reilly Media, Inc.*

Ho, H. N., Rabah, M., Nowakowski, S. & Estraillier, P., 2018, Trace-Based Multi-Criteria Preselection Approach for Decision Making in Interactive Applications like Video Games, *In The Digital Turn in Higher Education*, Springer, pp. 211–234.

Leemans, S. J. J., Fahland, D. & Van Der Aalst, W. M. P., 2014, Discovering block-structured process models from incomplete event logs, *International Conference on Applications and Theory of Petri Nets and Concurrency*, Tunis, Tunisia, pp. 91–110.

Murata, T., 1989, Petri nets: Properties, analysis and applications, *In Proceedings of the IEEE* Vol. 77, No. 4, pp 541–580.

Polyvyanyy, A., Van Der Aalst, W. M. P., Ter Hofstede, A. H. M. & Wynn, M. T., 2017, Impact-driven process model repair, *ACM Transactions on Software Engineering and Methodology (TOSEM)* Vol. 25, No.4, pp 1–28.

Reisig, W. & Rozenberg, G. 1998. *Lectures on petri nets I: basic models: advances in petri nets*, Springer Science & Business Media.

Rozinat, A. & van der Aalst, W. M. P., 2008, Conformance checking of processes based on monitoring real behavior, *Information Systems* Vol. 33, No. 1, pp. 64–95.

Russell, N., van der Aalst, W. M. P. & ter Hofstede, A. H. M. (2016), *Workflow patterns: the definitive guide*, MIT Press.

Toussaint, B.-M. & Luengo, V., 2015, Mining surgery phase-related sequential rules from vertebroplasty simulations traces, *Conference on Artificial Intelligence in Medicine in Europe, Springer International Publishing*, Pavia, Italy, pp. 35–46.

Van der Aalst, W., 2016, *Data Science in Action*, Process Mining, Springer.

Van der Aalst, W. M. P. & Weijters, A., 2004, Process mining: a research agenda, *Computers in industry* Vol. 53, No. 3, pp. 231–244.

Verbeek, H. M. W., Buijs, J. C. A. M., Van Dongen, B. F. & Van Der Aalst, W. M. P., 2010, XES, XESame, and ProM 6, *Information Systems Evolution, Springer*, pp. 60–75.