ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάλυση Διαδικασιών στη Δημόσια Διοίκηση με χρήση Εξόρυξης Διαδικασιών

ΚΑΤΣΙΚΗΣ ΔΙΟΝΥΣΙΟΣ

Εισηγητής:
Μπουσδέκης Αλέξανδρος

ΑΘΗΝΑ
2022-2023

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Μιαούλης Γεώργιος, Ομότιμος Καθηγητής	Μπαρδής Γεώργιος, Επίκουρος Καθηγητής	Μπουσδέκης Αλέξανδρος, Πανεπιστημιακός Υπότροφος
Georgios MIAOULIS Digitally signed by Georgios MIAOULIS Date: 2023.03.26 16:27:13 +03'00' (Υπογραφή)	Georgios Bardis Digitally signed by Georgios Bardis Date: 2023.03.30 13:31:34 +03'00' (Υπογραφή)	Alexandros Bousdekis Digitally signed by Alexandros Bousdekis Date: 2023.03.26 08:28:57 +03'00' (Υπογραφή)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Κατσίκης Διονύσιος του Ιωάννη, με αριθμό μητρώου 46690 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου.

Ο Δηλών,
Κατσίκης Διονύσιος

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία με θέμα «Ανάλυση Διαδικασιών στη Δημόσια Διοίκηση με χρήση Εξόρυξης Διαδικασιών», πραγματοποιήθηκε στο πλαίσιο των διπλωματικών εργασιών του τμήματος Μηχανικών Πληροφορικής και Υπολογιστών.

Με την ολοκλήρωση της, θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον επιβλέπων καθηγητή μου, κύριο Αλέξανδρο Μπουσδέκη, για την πολύτιμη επιστημονική καθοδήγηση και υποστήριξη του καθ' όλη τη διάρκεια εκπόνησης της εργασίας μου.

Ακόμη, θα ήθελα να ευχαριστήσω την οικογένεια μου για την στήριξή της καθ' όλη τη διάρκεια των σπουδών μου.

ΠΕΡΙΛΗΨΗ

Η διπλωματική εργασία ασχολείται με την ανάλυση και την αξιολόγηση των επιχειρησιακών διαδικασιών στη Δημόσια Διοίκηση, με έμφαση σε Οργανισμούς Τοπικής Αυτοδιοίκησης με τη χρήση τεχνικών εξόρυξης διαδικασιών. Τα δεδομένα των πληροφοριακών συστημάτων σε μορφή event logs αναλύονται προκειμένου να εξαχθούν και να αξιολογηθούν τα μοντέλα των διαδικασιών. Με αυτόν τον τρόπο είναι δυνατό να εντοπιστούν αδύναμα σημεία και να βελτιωθούν οι διαδικασίες. Η προσέγγιση της διπλωματικής εφαρμόζεται στις Τεχνικές Υπηρεσίες ενός Ολλανδικού Δήμου με τη χρήση της βιβλιοθήκης της Python pm4py.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: επιχειρησιακές διαδικασίες, εξόρυξη διεργασιών, ανακάλυψη διαδικασίας, ανάλυση δεδομένων, python, pm4py

ABSTRACT

The thesis deals with the analysis and evaluation of Business Process in Public Administration, with an emphasis on Local Government Organizations using Process Mining techniques. Information systems data in the form of event logs are analyzed in order to extract and evaluate Process Models. In this way it is possible to identify weak points and improve processes. The thesis approach is applied to the Technical Services of one Dutch Municipality using the Python library pm4py.

KEY WORDS: business process, process mining, process discovery, data analysis, python, pm4py

Περιεχόμενα

Κατάλογος Εικόνων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Γενικά

1.1.1 Εξόρυξη Διεργασιών (Process Mining)

Η εξόρυξη διεργασιών είναι ένας κλάδος που συνυπάρχει ανάμεσα στην επιστήμη των δεδομένων και την επιστήμη των διεργασιών. Χρησιμοποιεί τεχνικές και αλγόριθμους για την ανακάλυψη, παρακολούθηση και τη βελτίωση διαδικασιών, μέσω της εξαγωγής γνώσης από τα αρχεία καταγραφής συμβάντων. Προσφέρει γνώσεις οι οποίες είναι αντικειμενικές και βασισμένες σε γεγονότα που βοηθούν στον έλεγχο, την ανάλυση και τη βελτίωση των υπαρχουσών επιχειρηματικών διαδικασιών, απαντώντας τόσο σε ερωτήσεις που σχετίζονται με τη συμμόρφωση όσο και σε ερωτήσεις που σχετίζονται με την απόδοση. Η βάση της εξόρυξης διεργασιών είναι τα ακατέργαστα δεδομένα, αλλά χρειάζεται να έχουν συγκεκριμένη δομή προκειμένου να χρησιμοποιηθούν ως είσοδος για αλγόριθμους ανακάλυψης διεργασιών.

Σε αντίθεση με την εξόρυξη δεδομένων, όπου χρησιμοποιείται ένας τεράστιος όγκος δεδομένων για τον εντοπισμό και την πρόβλεψη προτύπων, η εξόρυξη διεργασιών εστιάζει περισσότερο σε αλληλουχίες διασυνδεδεμένων δεδομένων και δημιουργεί μια αμερόληπτη ολοκληρωμένη προβολή των διαδικασιών.

Τα τελευταία χρόνια, όλο και περισσότερες επιχειρήσεις ενδιαφέρονται για την ανάλυση και την βελτιστοποίηση των διαδικασιών τους, με αποτέλεσμα ο κλάδος του Process Mining να γνωρίζει μεγάλη ανάπτυξη.

1.1.2 Αρχείο καταγραφής συμβάντων (Event log)

Σε ένα αρχείο καταγραφής συμβάντων, κάθε συμβάν αναφέρεται σε μια περίπτωση, μια δραστηριότητα και ένα χρονικό σημείο. Μπορεί να θεωρηθεί ως μια συλλογή περιπτώσεων, όπου κάθε περίπτωση μπορεί να θεωρηθεί ως ίχνος/ακολουθία γεγονότων.

Τα αρχεία καταγραφής συμβάντων μπορεί να προέρχονται από:

- Σύστημα βάσης δεδομένων(π.χ. δεδομένα ασθενών σε νοσοκομείο)
- Υπολογιστικό φύλλο τιμών διαχωρισμένων με κόμμα(CSV files)
- Αρχείο καταγραφής συναλλαγών
- Ένα επαγγελματικό σύστημα ERP(SAP, Oracle κτλπ.)
- Ένα αρχείο καταγραφής μηνυμάτων(IBM middleware)
- Ένα ανοιχτό API που παρέχει δεδομένα από ιστότοπους ή μέσα κοινωνικής δικτύωσης

Τα γεγονότα παρατίθενται μαζί με τα χαρακτηριστικά τους σε ένα αρχείο καταγραφής συμβάντων(event log). Τα χαρακτηριστικά που αναφέρονται συνήθως είναι το όνομα, το case id, η χρονική σήμανση ώρας έναρξης και λήξης, αλλά και χαρακτηριστικά του συμβάντος που καταγράφονται από το σύστημα πληροφορικής. Το case id αντιστοιχίζει κάθε δραστηριότητα σε μία συγκεκριμένη περίπτωση, έτσι όλες οι δραστηριότητες με ίδιο case id σχηματίζουν ένα ίχνος(trace) στο αρχείο καταγραφής συμβάντων.

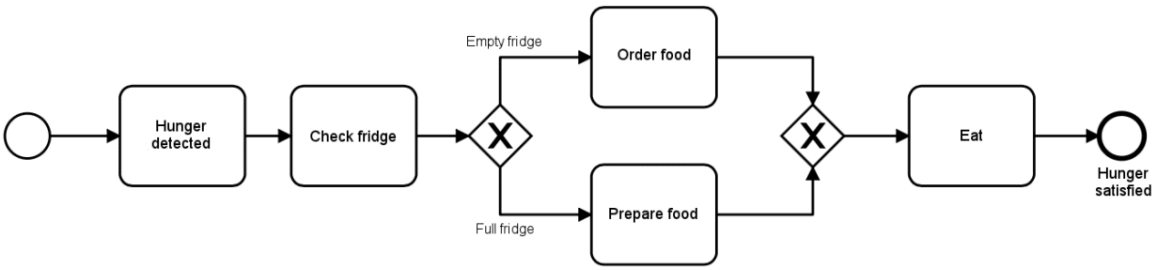
1.1.3 XES Standard

Το πρότυπο XES ορίζει μια γραμματική για μια γλώσσα που βασίζεται σε ετικέτες και στόχος της είναι να παρέχει μια ενοποιημένη και επεκτάσιμη μεθοδολογία για την καταγραφή συμπεριφορών των συστημάτων μέσω event logs και event streams που ορίζονται στο πρότυπο XES. Σκοπός αυτού του προτύπου είναι να παρέχει μια αναγνωρισμένη μορφή XML για την ανταλλαγή δεδομένων μεταξύ πληροφοριακών συστημάτων.

1.1.4 Μοντέλο διαδικασίας (Process model)

Η δημιουργία ενός μοντέλου διαδικασίας (process model) είναι ένα σημαντικό μέρος της διαχείρισης επιχειρηματικών διαδικασιών. Είναι μια γραφική αναπαράσταση που περιγράφει την αναμενόμενη συμπεριφορά μιας διαδικασίας και στόχος της είναι η ανακάλυψη δυνατότητων βελτιστοποίησης.

Τα μοντέλα αυτά, επεξεργάζονται τη γνώση και τις υπάρχουσες διαδικασίες και μέσα από αναλύσεις μπορούν να εντοπιστούν διπλά βήματα διαδικασιών ή σημεία συμφόρησης.



Εικόνα 1: Παράδειγμα ενός process model που δημιουργήθηκε με το μοντέλο BPMN για την διαδικασία "eliminate hunger"

1.1.5 Ευφυΐα επιχειρηματικής διαδικασίας (Business Process Intelligence)

Πρόκειται για την επιστήμη που συνδυάζει:

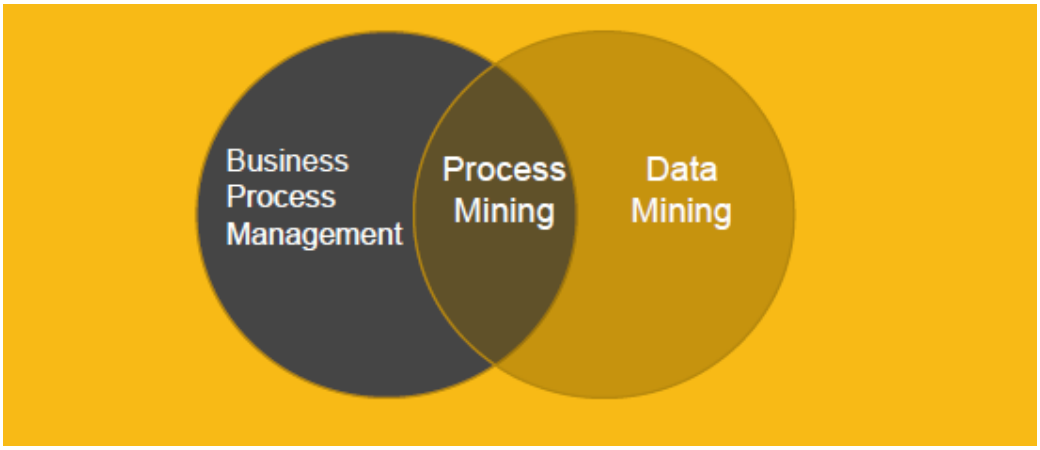
- Επιχειρηματική ανάλυση
- Εξόρυξη δεδομένων
- Οπτικοποίηση δεδομένων
- Εργαλεία και υποδομές δεδομένων

ώστε να βοηθήσει τους οργανισμούς να λαμβάνουν αποφάσεις βάσει δεδομένων.

Αποτελείται από ένα σύνολο μεθοδολογιών, διαδικασιών, αρχιτεκτονικών και τεχνολογιών που μετατρέπουν τα ακατέργαστα δεδομένα σε χρήσιμες πληροφορίες που χρησιμοποιούνται για τη βελτίωση της απόδοσης, τη λήψη αποφάσεων και την προγνωστική ανάλυση.

Οι διαδικασίες αυτές περιλαμβάνουν:

- Data mining: Με τη χρήση βάσεων δεδομένων, στατιστικών και μηχανικής μάθησης, ανακαλύπτει τάσεις σε μεγάλα σύνολα δεδομένων.
- Reporting: Μοιράζοντας την ανάλυση των δεδομένων στους ενδιαφερόμενους, ώστε να βγάλουν συμπεράσματα και να λάβουν αποφάσεις.
- Performance metrics and benchmarking: Σύγκριση τωρινών δεδομένων απόδοσης, με δεδομένα προηγούμενων αναλύσεων, για την παρακολούθηση της απόδοσης σε σχέση με τους στόχους που έχουν οριστεί.
- Descriptive analysis: Χρήση της preliminary data analysis ώστε να μάθουμε το τι συνέβη.
- Querying: Θέτοντας συγκεκριμένες ερωτήσεις για τα δεδομένα, το BI (Business Intelligence) αντλεί τις απαντήσεις από τα datasets.
- Statistical analysis: Παίρνουμε τα αποτελέσματα από την descriptive analysis και τα διερευνούμε περισσότερο χρησιμοποιώντας στατιστικά στοιχεία όπως το πώς συνέβη αυτή η τάση αλλά και γιατί.
- Data visualization: Μετατροπή της data analysis σε οπτικές αναπαραστάσεις όπως γραφήματα και ιστογράμματα.
- Visual analysis: Διερεύνηση των δεδομένων μέσω οπτικής αφήγησης.
- Data preparation: Συγκέντρωση πολλαπλών πηγών δεδομένων, προσδιορισμός διαστάσεων και μετρήσεων, προετοιμάζοντας τα για ανάλυση.



Εικόνα 2: Συσχέτιση Business Process Management – Process Mining – Data Mining

1.2 Αντικείμενο και Στόχοι της Διπλωματικής Εργασίας

Στόχος της συγκεκριμένης εργασίας είναι να εξεταστεί η συμβολή του process mining στη Δημόσια Διοίκηση και συγκεκριμένα σε διαδικασίες χρηματοδότησης σε Οργανισμούς Τοπικής Αυτοδιοίκησης. Θα επεξεργαστούμε ένα dataset το οποίο περιέχει δεδομένα σχετικά με αιτήσεις οικοδομικών αδειών, σε μορφή event logs, ενός Ολλανδικού Δήμου σε διάστημα τεσσάρων ετών, με σκοπό την ανάλυση και την αξιολόγηση των επιχειρησιακών διαδικασιών.

Αρχικά θα γίνει εισαγωγή των δεδομένων με τη χρήση της συνάρτησης **pm4py.read_xes**. Προκειμένου να είναι πιο εύκολα επεξεργάσιμα θα εφαρμόσουμε κατάλληλα φίλτρα όπως:

- **pm4py.get_start_activities / pm4py.get_end_activities**
- **pm4py.get_event_attribute_values**
- **pmp4py.filter_time_range**
- **pm4py.filter_event_attribute_values**
- **pm4py.filter_between**

Στη συνέχεια θα εφαρμόσουμε διάφορες τεχνικές εξόρυξης διεργασιών, δίνοντας έμφαση στους αλγόριθμους ανακάλυψης διαδικασιών ώστε να εξάγουμε συμπεράσματα για το ποιος αλγόριθμος μας δίνει τα καλύτερα αποτελέσματα. Αυτό θα το πετύχουμε χρησιμοποιώντας τις παρακάτω συναρτήσεις:

- **pm4py.discover_dfg**
- **pm4py.discover_performance_dfg**
- **pm4py.discover_petri_net_alpha**
- **pm4py.discover_petri_net_inductive**
- **pm4py.discover_heuristics_net**
- **pm4py.discover_petri_net_heuristics**

Επίσης θα εφαρμόσουμε έλεγχο συμμόρφωσης και τέλος θα αξιολογήσουμε το dataset ως προς την καταλληλότητα, την ακρίβεια, τη γενίκευση και την απλότητα. Μέσα από αυτά μπορούμε να εντοπίσουμε αδύνατα σημεία και να κάνουμε τις απαραίτητες βελτιώσεις. Για την εξαγωγή αυτών των αποτελεσμάτων είναι απαραίτητες οι παρακάτω συναρτήσεις:

- **pm4py.conformance_diagnostics_token_based_replay**
- **pm4py.conformance_diagnostics_alignments**
- **pm4py.fitness_token_based_replay**
- **pm4py.fitness_alignments**
- **pm4py.precision_token_based_replay**
- **pm4py.precision_alignments**
- **generalization_evaluator.apply**
- **simplicity_evaluator.apply**

1.3 Διάρθρωση της Διπλωματικής Εργασίας

Ξεκινώντας από το 2ο κεφάλαιο, θα προσεγγίσουμε σε θεωρητικό επίπεδο τις έννοιες Διαχείριση Επιχειρησιακών Διαδικασιών και Εξόρυξη Διεργασιών. Θα δώσουμε τους ορισμούς, θα εξετάσουμε το γιατί είναι τόσο σημαντικές στην εποχή μας, θα αναφέρουμε τα πλεονεκτήματα καθώς και τους κινδύνους που κρύβουν οι λάθος χειρισμοί αυτών αλλά και τα εμπόδια τα οποία πρέπει να ξεπεραστούν ώστε να έχουμε τα βέλτιστα αποτελέσματα σε επίπεδο οργανισμών. Επίσης, στο ίδιο κεφάλαιο θα αναφερθούμε στις προκλήσεις που υπάρχουν σε Διαδικασίες Δημόσιας Διοίκησης, κάνοντας και μια αναφορά στην Δημόσια Διοίκηση της Ελλάδας.

Στο κεφάλαιο 3, θα αναλύσουμε την ερευνητική μεθοδολογία που θα ακολουθήσουμε. Περιληπτικά, θα ασχοληθούμε με εισαγωγή και φιλτράρισμα δεδομένων, με ανακάλυψη διαδικασίας, με αλγόριθμους ανακάλυψης διαδικασιών, με έλεγχο συμμόρφωσης και με ανάλυση απόδοσης.

Στο 4ο κεφάλαιο θα γίνει εφαρμογή των παραπάνω σε διαδικασίες χρηματοδότησης σε Οργανισμούς Τοπικής Αυτοδιοίκησης. Πρώτα θα γίνει η υλοποίηση και στη συνέχεια θα διεξαχθούν τα πειραματικά αποτελέσματα τα οποία θα συνοδεύονται από τα σχετικά σχόλια.

Τέλος στο 5ο κεφάλαιο, εφόσον έχουν ολοκληρωθεί οι παραπάνω αναλύσεις, θα είμαστε σε θέση να εξάγουμε συμπεράσματα και να αναφερθούμε σε μελλοντική μελέτη.

Κεφάλαιο 2ο: Βιβλιογραφική Επισκόπηση

2.1 Διαχείριση Επιχειρησιακών Διαδικασιών (Business Process Management)

2.1.1 Τι είναι;

Η διαχείριση επιχειρησιακών διαδικασιών (BPM) είναι μια δομημένη προσέγγιση για τη βελτίωση των διαδικασιών που χρησιμοποιούν οι οργανισμοί ώστε να ολοκληρώσουν τη δουλειά τους, να εξυπηρετήσουν τους πελάτες τους και να δημιουργήσουν επιχειρηματική αξία.

Βοηθάει τους οργανισμούς να πετύχουν τους στόχους τους, ένας τέτοιος στόχος μπορεί να είναι η αύξηση των κερδών ή η προώθηση της ποικιλομορφίας του εργατικού δυναμικού. Χρησιμοποιεί διάφορες μεθόδους, με σκοπό να βελτιώσει μια επιχειρηματική διαδικασία, αναλύοντας την, μοντελοποιώντας τον τρόπο με τον οποίο λειτουργεί σε διαφορετικά σενάρια, εφαρμόζοντας αλλαγές, παρακολουθώντας τη νέα διαδικασία και βελτιώνοντας συνεχώς την ικανότητα της να οδηγεί στα επιθυμητά επιχειρηματικά αποτελέσματα.

Είναι ένας ευρύς κλάδος και αρκετά δυναμικός, δεδομένου ότι οι οργανωτικοί ρόλοι, οι κανόνες, οι τακτικές, οι επιχειρηματικοί στόχοι και άλλα στοιχεία που περιλαμβάνει, αλλάζουν συνεχώς. Με τον καιρό, οι επιχειρηματικές διαδικασίες, σε ορισμένες εταιρείες, έγιναν πολύ μεγάλες και πολύπλοκες προκειμένου να γίνουν διαχειρίσιμες, χωρίς τη βοήθεια αυτοματοποιημένων εργαλείων. Γι' αυτό το λόγο δημιουργήθηκε η ανάγκη της ανάπτυξης προϊόντων λογισμικού BPM για την υποστήριξη μεγάλης κλίμακας επιχειρηματικής αλλαγής. Αυτές οι τεχνολογίες BPM έχουν γνωρίσει μεγάλη εξέλιξη, με γνώμονα τις προόδους στην τεχνητή νοημοσύνη, τη μηχανική μάθηση και άλλες παρόμοιες τεχνολογίες που παρέχουν νέους τρόπους για την ανακάλυψη, το σχεδιασμό, τη μέτρηση, τη βελτίωση και την αυτοματοποίηση των ροών εργασίας.

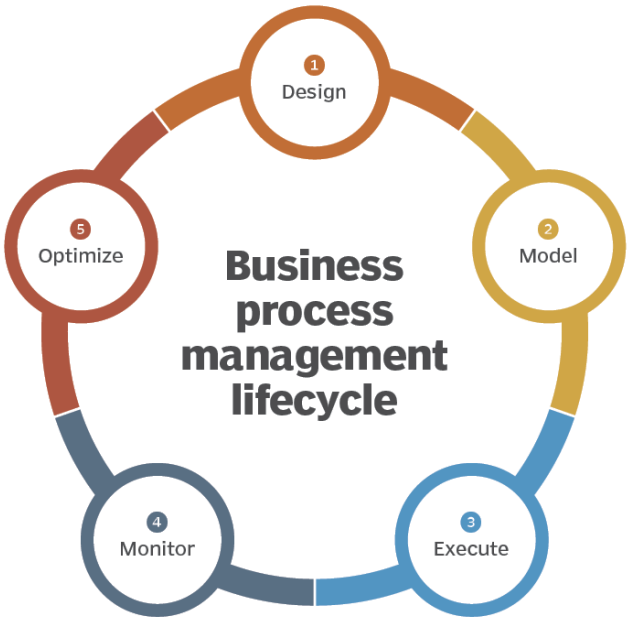
2.1.2 Γιατί είναι σημαντική;

Είναι αρκετά σημαντική, διότι ορισμένες επιχειρηματικές δραστηριότητες μπορεί να συνεπάγονται εκατοντάδες ή και χιλιάδες εργασίες και τις εγκρίσεις που απαιτούνται για την ολοκλήρωση τους. Συνήθως περιλαμβάνουν άτομα, συστήματα πληροφορικής και άλλα μηχανήματα που ανήκουν στην επιχείρηση. Μια καλά σχεδιασμένη επιχειρηματική διαδικασία χωρίζει αυτές τις εργασίες σε δομημένα, επαναλαμβανόμενα βήματα που μπορούν να ακολουθήσουν οι εργαζόμενοι για να παράγουν συνεπή αποτελέσματα. Τα επαναλαμβανόμενα βήματα βοηθούν τους οργανισμούς να προβλέψουν τους πόρους που απαιτούνται, μειώνοντας τον κίνδυνο υποκατανομής ή υπερκατανομής πόρων. Η μέτρηση των βημάτων αποκαλύπτει αδύναμους κρίκους και σημεία συμφόρησης, υποδεικνύοντας τον δρόμο για πιθανές βελτιώσεις. Αντιθέτως, μια κακώς σχεδιασμένη ή κακοδιαχειριζόμενη επιχειρηματική διαδικασία μπορεί να βλάψει μια εταιρεία, εμποδίζοντας την παραγωγικότητα και την αποδοτικότητα της.

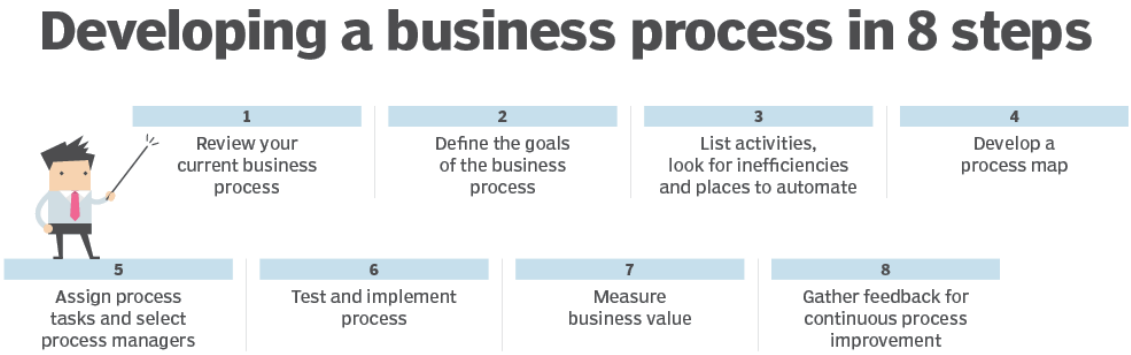
2.1.3 Κύκλος ζωής μιας επιχειρηματικής διαδικασίας

Οι ειδικοί περιγράφοντας ένα έργο BPM, αναφέρονται σε πέντε φάσεις:

- **Σχεδίαση:** Ανάλυση της υπάρχουσας διαδικασίας, ώστε να δούμε τι μπορούμε να βελτιώσουμε και στη συνέχεια, ανάπτυξη επιχειρηματικής διαδικασίας, χρησιμοποιώντας την τυποποίηση και την αυτοματοποίηση, όπως θα έπρεπε ιδανικά να υπάρχει.
- **Μοντελοποίηση:** Κοιτώντας πως λειτουργεί η επανασχεδιασμένη επιχειρηματική διαδικασία σε διαφορετικά σενάρια.
- **Υλοποίηση:** Εκτελούμε τις βελτιώσεις, χρησιμοποιώντας τα παραπάνω.
- **Παρακολούθηση:** Παρακολούθηση των βελτιώσεων ώστε να δούμε την απόδοση τους.
- **Βελτιστοποίηση:** Συνεχίζουμε να βελτιώνουμε το BPM σε συνεχή βάση.



Εικόνα 3: Κύκλος ζωής ενός BPM



Εικόνα 4: Παράδειγμα ανάπτυξης ενός BPM σε 8 στάδια

2.1.4 Πλεονεκτήματα

Η δομημένη προσέγγιση της BPM βελτιώνει την ποιότητα της εργασίας και τη λειτουργική αποτελεσματικότητα. Ένα καλά εκτελεσμένο πρόγραμμα BPM μπορεί να:

- εξαλείψει τη σπατάλη
- μειώσει τα σφάλματα
- εξοικονομήσει χρόνο
- βελτιώσει τη συμμόρφωση
- αυξήσει την ευελιξία
- υποστηρίξει τον ψηφιακό μετασχηματισμό
- βοηθήσει στην παροχή καλύτερων προϊόντων και υπηρεσιών στους πελάτες

Επίσης μπορεί να αποτελέσει ένα αποτελεσματικό εργαλείο διαχείρισης για τους ακόλουθους λόγους:

- Η εστίαση στην τυποποίηση των διαδικασιών μειώνει τον κίνδυνο ανθρώπινου λάθους
- Τα ενσωματωμένα αναλυτικά στοιχεία δίνουν στους διαχειριστές μεγαλύτερη ορατότητα σχετικά με την απόδοση της διαδικασίας και τους βοηθούν να εντοπίσουν τα σημεία συμφόρησης
- Τα εργαλεία αυτοματισμού αυξάνουν την αποτελεσματικότητα και επιτρέπουν στους υπαλλήλους να επικεντρωθούν σε εργασίες που απαιτούν την ανθρώπινη παρέμβαση και τεχνογνωσία

2.1.5 Εμπόδια

Προκειμένου να μεγιστοποιηθούν τα οφέλη του BPM, οι οργανισμοί πρέπει να είναι προετοιμασμένοι για τις προκλήσεις που μπορεί να αντιμετωπίσουν καθώς και να έχουν ένα σχέδιο για την αντιμετώπισή τους.

Παρακάτω θα παραθέσουμε ορισμένα εμπόδια τα οποία είναι οι πιο συνηθισμένοι λόγοι αποτυχίας ενός BPM:

- Έλλειψη εκτελεστικής συμμετοχής και υποστήριξης
- Ασαφείς επιχειρηματικοί στόχοι
- Ανεπαρκείς υποδομές δοκιμών
- Σύγχυση σχετικά με την επιλογή του σωστού εργαλείου
- Κρυφές διαδικασίες ευάλωτες σε βλάβες
- Κακή ορατότητα και ιχνηλασιμότητα της διαδικασίας

2.1.6 Κατηγορίες

Επειδή το BPM είναι τόσο ευρύ και περιλαμβάνει τη χρήση διάφορων εργαλείων, οι ειδικοί το χωρίζουν στις παρακάτω κατηγορίες:

- **System-centric:** Εστιάζει σε διαδικασίες που περιλαμβάνουν ροές εργασίας σε επιχειρηματικά συστήματα που λειτουργούν χωρίς μεγάλη ανθρώπινη παρέμβαση και ενσωματώνονται σε εταιρικές εφαρμογές.
- **Human-centric:** Επικεντρώνεται σε διαδικασίες που χειρίζονται οι άνθρωποι. Περιλαμβάνουν διαδικασίες που διαθέτουν χαρακτηριστικά σχεδιασμένα για ανθρώπινη αλληλεπίδραση, όπως ένα καλά σχεδιασμένο περιβάλλον εργασίας χρήστη και ειδοποιήσεις.
- **Document-centric:** Επικεντρώνεται σε έγγραφα, όπως η διαδικασία μορφοποίησης, υπογραφής και επαλήθευσης συμβάσεων.

2.2 Εξόρυξη Διεργασιών (Process Mining)

2.2.1 Σύγκριση εξόρυξης διεργασιών με τη διαχείριση επιχειρηματικών διαδικασιών

Η εξόρυξη διεργασιών βρίσκεται στη διασταύρωση της διαχείρισης επιχειρηματικών διαδικασιών(BPM) και της εξόρυξης δεδομένων(data mining). Ενώ η εξόρυξη διεργασιών και η εξόρυξη δεδομένων λειτουργούν με δεδομένα, το εύρος κάθε συνόλου δεδομένων διαφέρει. Η εξόρυξη διεργασιών χρησιμοποιεί συγκεκριμένα δεδομένα καταγραφής συμβάντων(event logs) για τη δημιουργία μοντέλων διεργασιών που μπορούν να χρησιμοποιηθούν για την ανακάλυψη, σύγκριση ή βελτίωση μιας υπάρχουσας διαδικασίας.

Η εξόρυξη διεργασιών ακολουθεί μια προσέγγιση η οποία βασίζεται περισσότερο στα δεδομένα. Αντιθέτως το BPM, το οποίο δέχεται μια πιο χειροκίνητη διαχείριση, συλλέγει δεδομένα πιο ανεπίσημα μέσω εργαστηρίων και συνεντεύξεων και στη συνέχεια χρησιμοποιεί λογισμικό για να τεκμηριώσει αυτή τη ροή εργασίας ως χάρτη διαδικασίας. Δεδομένου ότι τα δεδομένα που ενημερώνουν αυτούς τους χάρτες διαδικασίας είναι πιο ποιοτικά, η εξόρυξη διεργασιών φέρνει μια πιο ποσοτική προσέγγιση σε ένα πρόβλημα διαδικασίας, περιγράφοντας λεπτομερώς την πραγματική διαδικασία μέσω δεδομένων συμβάντων.

2.2.2 Γιατί είναι σημαντική;

Η αύξηση των πωλήσεων σε μια εταιρεία δεν είναι ο μόνος τρόπος δημιουργίας εσόδων. Η μείωση του λειτουργικού κόστους μπορεί να αυξήσει την απόδοση της επένδυσης. Η εξόρυξη διεργασιών βοηθά τις επιχειρήσεις να μειώσουν αυτά τα κόστη, ποσοτικοποιώντας τις αδυναμίες στα λειτουργικά τους μοντέλα, επιτρέποντας στους χειριστές τους να λαμβάνουν αντικειμενικές αποφάσεις σχετικά με την κατανομή των πόρων. Η ανακάλυψη αυτών των σημείων συμφόρησης όχι μόνο μπορεί να μειώσει το κόστος και να επιταχύνει τη βελτίωση της διαδικασίας, αλλά μπορεί να οδηγήσει σε περισσότερη καινοτομία, ποιότητα και καλύτερη διατήρηση των πελατών.

2.2.3 Εμπόδια

Δεδομένου ότι η πρόκειται για έναν σχετικά νέο κλάδο, έχει κάποια εμπόδια τα οποία χρειάζεται να αντιμετωπίσει. Μερικά από αυτά είναι:

- **Ποιότητα δεδομένων:** Προκειμένου να ενεργοποιηθεί η εξόρυξη διεργασιών, απαιτείται η εύρεση, η συγχώνευση και ο καθαρισμός των δεδομένων. Τα δεδομένα ενδέχεται να είναι διανεμημένα σε διάφορες πηγές δεδομένων, μπορεί να είναι ελλιπή ή θορυβώδη, καθώς και να περιέχουν διαφορετικές ετικέτες. Τα παραπάνω παίζουν καθοριστικό ρόλο για το αποτέλεσμα που θα εξάγει ένα μοντέλο διαδικασίας.
- **Μετατόπιση εννοιών:** Μερικές φορές οι διαδικασίες αλλάζουν καθώς αναλύονται, με αποτέλεσμα την εννοιολογική μετατόπιση.

2.2.4 Τύποι εξόρυξης διεργασιών

Υπάρχουν τρείς τύποι εξόρυξης διεργασιών:

1) **<u>Ανακάλυψη διεργασιών(Process discovery):</u>** Σε αυτό τον τύπο εξόρυξης διεργασιών, οι αλγόριθμοι δημιουργούν ένα μοντέλο διαδικασίας που βασίζεται σε ήδη υπάρχοντα δεδομένα συμβάντων. Υπάρχουν διάφοροι αλγόριθμοι ανακάλυψης διεργασιών, όπως οι Alpha Miner, Heuristic Miner και Inductive Miner.
2) **<u>Συμμόρφωση διαδικασίας(Process conformance):</u>** Ένα υπάρχον μοντέλο συγκρίνεται με ένα event log της ίδιας διαδικασίας, έτσι μπορούν να εντοπιστούν οι αποκλίσεις μεταξύ των δύο, ελέγχοντας αν η πραγματικότητα που καταγράφεται στο αρχείο, συμμορφώνεται με το μοντέλο και αντίστροφα.
3) **<u>Ενίσχυση διαδικασίας(Process enhancement):</u>** Στόχος είναι η βελτίωση ή η επέκταση ενός υπάρχοντος μοντέλου διαδικασίας. Αυτό επιτυγχάνεται είτε τροποποιώντας ένα μοντέλο ώστε να αντικατοπτρίζει καλύτερα την πραγματικότητα, είτε επεκτείνοντας ένα μοντέλο προσθέτοντας πληροφορίες, όπως είναι ο χρόνος διεκπεραίωσης ή συχνότητες, που δεν έχουν αποτυπωθεί μέχρι στιγμής στο μοντέλο.

2.3 Προκλήσεις σε Διαδικασίες Δημόσιας Διοίκησης

Η Δημόσια Διοίκηση σαν επιστημονικός κλάδος έχει ως σκοπό να αναλύσει και να ερμηνεύσει τη συνεργασία των πολιτικών και διοικητικών θεσμών αλλά και των στελεχών του κράτους ώστε να πετύχει τη δημιουργία προγραμμάτων και αποφάσεων για την επίλυση δημόσιων ζητημάτων. Όπως αναφέραμε προηγουμένως, εξαιτίας του ανταγωνισμού στο χώρο των επιχειρήσεων, έχει αναπτυχθεί η ανάγκη βελτίωσης της απόδοσής τους ενσωματώνοντας καινοτόμες ιδέες και τις τελευταίες τεχνικές διαχείρισης προκειμένου να παραμείνουν ανταγωνιστικές και να εξασφαλίσουν την επιβίωση τους. Τα παραπάνω ισχύουν για όλους τους τομείς οργανώσεων.

2.3.1 Δημόσιος Τομέας

Ωστόσο, έρευνες δείχνουν ότι οι οργανισμοί του Δημόσιου τομέα υστερούν σε αυτό το κομμάτι, καθώς οι οργανωτικές αλλαγές είναι εφικτές μέχρι ένα ορισμένο όριο. Εάν θέλουμε να γίνουν βελτιώσεις στο Δημόσιο τομέα θα χρειαστεί να γίνουν ριζικές αλλαγές στην εκτέλεση των επιχειρηματικών διαδικασιών και στις οργανωτικές δομές. Αυτό προϋποθέτει την ενοποίηση επιχειρηματικών διαδικασιών, αυτοματοποίηση δραστηριοτήτων και εξάλειψη ορισμένων περιττών.

Είναι απαραίτητη η λογική οργάνωση των ανθρώπων, των υλικών, της ενέργειας, του εξοπλισμού και των διαδικασιών σε δραστηριότητες οι οποίες στοχεύουν στο να παράγουν ένα συγκεκριμένο αποτέλεσμα που θα επιφέρει βελτιώσεις σε μέτρα απόδοσης κόστους, ποιότητας, εξυπηρέτησης και ταχύτητας. Εάν τα παραπάνω δε γίνουν με σωστό τρόπο, το αποτέλεσμα μπορεί να διαταράξει τις λειτουργίες, να παρεμποδίσει την παραγωγικότητα και να σταματήσει την ανάπτυξη του οργανισμού. Ο ανασχεδιασμός επιχειρησιακών διαδικασιών περιλαμβάνει την ανακάλυψη, το σχεδιασμό και την ανάπτυξη επιχειρηματικών διαδικασιών, αλλά και τον εκτελεστικό, διοικητικό και εποπτικό έλεγχο τους προκειμένου να διασφαλιστεί η συμμόρφωση με τους επιχειρηματικούς στόχους. Για να εξασφαλίσουμε την επιτυχία ενός τέτοιου ανασχεδιασμού, είναι σημαντικό να γίνει επιλογή μιας επίσημης μεθοδολογίας.

2.3.2 Συντελεστές επιτυχίας ενός ανασχεδιασμού διαδικασιών

Η αρχική διαδικασία θα χρειαστεί να επανασχεδιαστεί σύμφωνα με το σημερινό περιβάλλον του οργανισμού και όχι με τις παραδόσεις του. Ορισμένοι συντελεστές επιτυχίας είναι η στρατηγική ευθυγράμμιση, το επίπεδο επενδύσεων στις τεχνολογίες πληροφορικής, η μέτρηση επιδόσεων αλλά και το επίπεδο εξειδίκευσης των εργαζομένων.

Επίσης, σημαντικό ρόλο βάσει δυναμικών δυνατοτήτων αλλά και βάσει της τεχνολογίας των εργασιών παίζουν:

- οι οργανωτικές αλλαγές
- ο διορισμός ιδιοκτητών διαδικασιών
- οι αλλαγές
- η χρήση ενός συστήματος συνεχούς βελτίωσης
- η τυποποίηση διαδικασιών
- η αυτοματοποίηση
- η πληροφόρηση
- η κατάρτιση
- η ενδυνάμωση των εργαζομένων

Κάποιοι επιπλέον παράγοντες είναι η δημιουργία ενός κλίματος οργανωτικής αλλαγής, η αναθεώρηση ανταμοιβής, η αποτελεσματική επικοινωνία και η ανθρώπινη συμμετοχή.

Αναλύοντας περισσότερο τους συντελεστές, σύμφωνα με ειδικούς μπορούμε να διακρίνουμε δύο κατηγορίες:

- Τους εξωτερικούς παράγοντες στους οποίους συγκαταλέγονται ο μισθός, οι παροχές, η ασφάλιση, τα μπόνους και οι υπερωρίες.
- Τους εσωτερικούς παράγοντες στους οποίους περιλαμβάνονται οι ευθύνες, ο ομαδικός σχεδιασμός, το αναπτυξιακό πρόγραμμα και η αυτονομία στην εργασία.

Επίσης οι οργανισμοί πρέπει να δώσουν ιδιαίτερη έμφαση στην ομαδική εργασία, το συνεργατικό περιβάλλον, στην ανάπτυξη ενός συστήματος διαχείρισης ποιότητας, στην ικανοποιητική ανταμοιβή, στην σωστή διαχείριση των αλλαγών αλλά και στη χρήση νέων τεχνολογιών στα συστήματα πληροφορικής. Αντιθέτως, κάποιοι κρίσιμοι παράγοντες που πρέπει να αποφευχθούν είναι η αντίσταση στην αλλαγή, οι άκαμπτες ιεραρχίες, οι αλληλεπικαλύψεις των πρωτοβουλιών και το ευρύ πεδίο δραστηριοτήτων.

2.3.3 Δημόσια Διοίκηση στην Ελλάδα

Μια Δημόσια διοίκηση περιλαμβάνει όλα τα νομικά πρόσωπα όπου είναι οργανωμένα τα διοικητικά όργανα που προβλέπουν οι κανόνες δικαίου. Τα διοικητικά όργανα είναι υπεύθυνα να θεσμοθετούν κανόνες δικαίου, να ασκούν εκτελεστική εξουσία χωρίς να υπάγονται στη νομοθετική ή στη δικαστική και έχουν στόχο τη δημιουργία κυβερνητικής πολιτικής. Επίσης, ευθύνη τους είναι η παραγωγή και η διακίνηση των αγαθών, η παροχή υπηρεσιών και η ολοκλήρωση των έργων. Ακόμη ένας σημαντικός τομέας ευθύνης είναι η προστασία της εθνικής άμυνας και η διατήρηση της δημόσιας τάξης.

Στην Ελλάδα υπάρχουν τρεις κατηγορίες συστημάτων διοικητικής οργάνωσης:

- Το συγκεντρωτικό σύστημα

Σε αυτό το σύστημα, τα όργανα δημόσιας διοίκησης που βρίσκονται στην πρωτεύουσα της χώρας είναι υπεύθυνα για την άσκηση της διοικητικής οργάνωσης σε ολόκληρη την επικράτεια. Οι Κεντρικές Κρατικές Υπηρεσίες καθορίζουν τη γενική κατεύθυνση, τον συντονισμό και τον έλεγχο νομιμότητας των περιφερειακών οργάνων, ορίζοντας συγκεκριμένους φορείς που έχουν ως στόχο τη διαμόρφωση της πολιτικής του υπουργείου, τον προγραμματισμό του και την παροχή οδηγιών προς τα περιφερειακά όργανα.

- Το αποκεντρωτικό σύστημα

Σύμφωνα με το αποκεντρωτικό σύστημα, τα όργανα είναι υπεύθυνα για τις υποθέσεις που αφορούν τη διοικητική τους περιφέρεια, ενώ τα κεντρικά όργανα έχουν αρμοδιότητα πάνω σε θέματα όπως η γενική κατεύθυνση, ο συντονισμός και ο έλεγχος νομιμότητας των περιφερειακών οργάνων.

- Η αυτοδιοίκηση

Όσον αφορά το σύστημα διοικητικής αυτοδιοίκησης, οι διοικούντες δεν είναι κρατικά όργανα αλλά ξεχωριστά. Οι υποθέσεις ορίζονται και εκτελούνται από δημοκρατικά εκλεγμένα όργανα αυτοδιοικούμενων οργανισμών υπό την κρατική εποπτεία και αφορούν συγκεκριμένο τόπο και υπάρχει ανάγκη γρήγορης επίλυσης των τοπικών ζητημάτων, όπου παρουσιάζονται.

Στο κομμάτι του ανασχεδιασμού των διαδικασιών, κατά κύριο λόγο οι κερδοσκοπικοί οργανισμοί αναλαμβάνουν τα έργα αλλαγής επιχειρηματικών διαδικασιών, ωστόσο ορισμένοι μη κερδοσκοπικοί οργανισμοί όπως οι οργανισμοί του Δημόσιου Τομέα αναλαμβάνουν σημαντικά έργα ανασχεδιασμού.

Σε ένα τέτοιο έργο πρέπει να εξεταστούν τέσσερα κύρια κριτήρια:

- η ριζοσπαστικότητα του έργου
- η δομημένη διαδικασία
- η εστίαση στον πελάτη
- η δυνατότητα ενεργοποίησης της πληροφορικής

Κεφάλαιο 3ο: Ερευνητική Μεθοδολογία

3.1 Εισαγωγή δεδομένων

Η εξόρυξη διεργασιών είναι αδύνατη χωρίς τα κατάλληλα αρχεία καταγραφής συμβάντων. Ανάλογα με την τεχνική εξόρυξης που χρησιμοποιείται, οι απαιτήσεις μπορεί να διαφέρουν. Πρέπει να εξαχθούν δεδομένα από μια ποικιλία πηγών δεδομένων, π.χ. βάσεις δεδομένων, αρχεία καταγραφής μηνυμάτων, αρχεία καταγραφής συναλλαγών και συστήματα διαχείρισης εγγράφων. Κατά τη συγχώνευση και την εξαγωγή δεδομένων, τόσο η σύνταξη όσο και η σημασιολογία παίζουν σημαντικό ρόλο.

3.2 Φιλτράρισμα δεδομένων (Filtering data)

Μόλις εξάγουμε τα αρχεία καταγραφής συμβάντων από τα δεδομένα, το επόμενο βήμα είναι να τα φιλτράρουμε. Το φιλτράρισμα είναι μια επαναληπτική διαδικασία. Η βιβλιοθήκη pm4py μας δίνει τη δυνατότητα, να εφαρμόσουμε ορισμένα φίλτρα όπως:

- Φιλτράρισμα με βάση το χρονικό πλαίσιο - Filter on timeframe
- Φιλτράρισμα στις τιμές των χαρακτηριστικών - Filter on attribute values
- Φιλτράρισμα στις τιμές των αριθμητικών χαρακτηριστικών - Filter on numeric attribute values
- Ενδιάμεσο φίλτρο - Between filter
- Φίλτρο μεγέθους περίπτωσης - Case size filter
- Φίλτρο εκ νέου επεξεργασίας - Rework filter
- Φίλτρο απόδοσης μονοπατιών - Paths performance filter

Με βάση το φιλτραρισμένο αρχείο καταγραφής, μπορούν στη συνέχεια να εφαρμοστούν οι διαφορετικοί τύποι τεχνικών εξόρυξης διεργασιών οι οποίοι είναι η ανακάλυψη (discovery), συμμόρφωση (conformance) και βελτίωση(enhancement).

3.3 Ανακάλυψη διαδικασίας (Process discovery)

Ένας αλγόριθμος ανακάλυψης διεργασιών ξεκινά με σάρωση των δραστηριοτήτων που περιγράφονται στο αρχείο καταγραφής συμβάντων και σιγά-σιγά δημιουργεί ένα μοντέλο που αντιπροσωπεύει καλύτερα την προηγούμενη συμπεριφορά. Μια αόρατη ακολουθία δραστηριοτήτων θεωρείται σαν κάτι καινούριο. Όταν ο αλγόριθμος φτάσει στο τέλος του αρχείου καταγραφής συμβάντων, έχει δει όλες τις πιθανές παραλλαγές της συμπεριφοράς που επιδεικνύει το σύστημα, και έτσι εξάγει ένα μοντέλο διαδικασίας που αντιπροσωπεύει τη συμπεριφορά του συστήματος.

Περιγράφει την οπτικοποίηση μιας διαδικασίας βάσει δεδομένων και αποτελεί μέρος του Process Mining. Οι αλγόριθμοι Process Discovery αναζητούν ένα κατάλληλο μοντέλο διεργασιών που να περιγράφει τη σειρά των γεγονότων/δραστηριοτήτων που εκτελούνται κατά την εκτέλεση μιας διεργασίας. Αυτό το πετυχαίνουν αναγνωρίζοντας μοτίβα μέσα στο αρχείο καταγραφής συμβάντων.

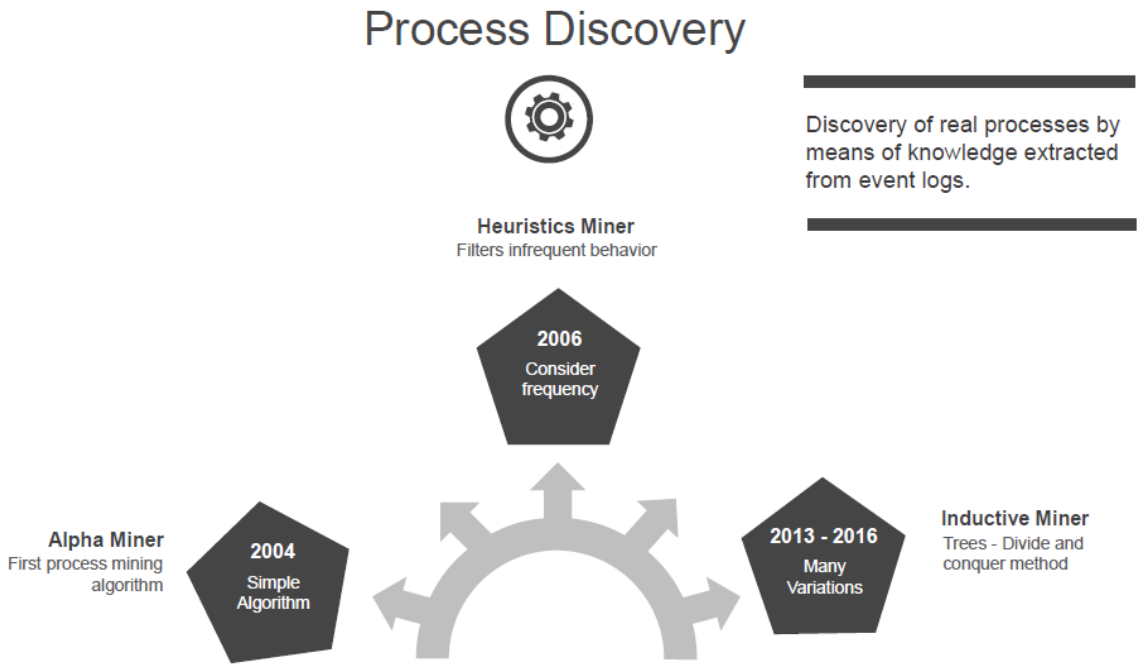
Η ανακάλυψη διεργασιών πρέπει να πληροί τέσσερις βασικές προϋποθέσεις:

- Το μοντέλο που θα προκύψει πρέπει να είναι εύκολα κατανοητό.
- Το τελικό μοντέλο θα πρέπει να αποτυπώνει την πραγματική συμπεριφορά του συστήματος.
- Θα πρέπει να παρέχει ορισμένες εγγυήσεις, όπως να προσφέρει μια καλά ορισμένη συμπεριφορά, χωρίς ανωμαλίες (π.χ. αδιέξοδα).
- Ο αλγόριθμος εξόρυξης διεργασιών θα πρέπει να αντιμετωπίζει τα event data τα οποία είναι ανεπαρκή.

Υπάρχουν 4 αλγόριθμοι που εφαρμόζονται στο Process Discovery:

- Alpha Miner
- Alpha+ Miner
- Inductive Miner
- Heuristic Miner

Αφού εφαρμοστούν οι τεχνικές ανακάλυψης διεργασιών, λαμβάνουμε σαν αποτέλεσμα ένα μοντέλο διαδικασίας καθώς και στατιστικά στοιχεία των περιπτώσεων που αποτελούν μέρος του αρχείου καταγραφής συμβάντων. Η αναπαράσταση και η ακρίβεια του μοντέλου που ανακαλύφθηκε εξαρτώνται τόσο από την τεχνική που χρησιμοποιήθηκε για την ανακάλυψη όσο και από τον τύπο της οπτικοποίησης που επιλέχθηκε.

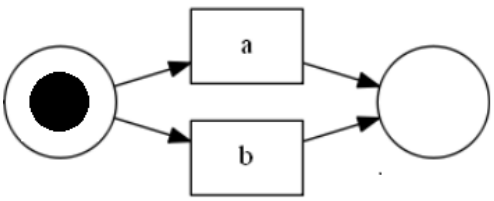


Εικόνα 5: Process Discovery

3.4 Petri net

Το petri net εφευρέθηκε το 1939, από τον Γερμανό Carl Adam Petri, σε ηλικία 13 ετών, με σκοπό την απεικόνιση των χημικών διεργασιών. Είναι ένα κατευθυνόμενο διμερές γράφημα, στο οποίο οι κόμβοι αντιπροσωπεύουν μεταβάσεις και θέσεις και έχουν ένα σχετικό βάρος. Τα τόξα συνδέουν σημεία με μεταβάσεις και αντίστροφα. Μια μετάβαση μπορεί να ενεργοποιηθεί αν κάθε μια από τις θέσεις εισόδου περιέχει έναν αριθμό από tokens που είναι τουλάχιστον ίσος με το βάρος του τόξου που το συνδέει. Όταν ενεργοποιείται μία μετάβαση, τότε τα tokens αφαιρούνται από τις θέσεις εισόδου ανάλογα με το βάρος του τόξου και προστίθενται στις θέσεις εξόδου ανάλογα με το τόξο εξόδου.

Οι αλγόριθμοι ανακάλυψης διαδικασιών που υλοποιούνται στην pm4py επιστρέφουν ένα petri net μαζί με μία αρχική σήμανση και μία τελική σήμανση. Μια αρχική σήμανση, είναι η αρχική κατάσταση εκτέλεσης μιας διαδικασίας και η τελική σήμανση είναι μια κατάσταση που πρέπει να επιτευχθεί στο τέλος της εκτέλεσης της διαδικασίας.



Εικόνα 6: Ένα απλό Petri net

Στην εικόνα 6, οι κύκλοι αναφέρονται σε μέρη, τα ορθογώνια σε μεταβάσεις και η μαύρη κουκίδα αντιπροσωπεύει ένα token. Οι μεταβάσεις αναφέρονται μόνο ως γράμματα. Εάν το δίκτυο πρόκειται να χρησιμοποιηθεί στην πράξη, οι δραστηριότητες πρέπει να ονομάζονται επαρκώς.

Το Petri net μπορεί να περιέχει πολλαπλές μεταβάσεις με την ίδια ετικέτα, καθώς και μεταβάσεις με κενή ετικέτα, οι οποίες αναφέρονται ως αόρατες ή αθόρυβες.

3.5 Directly-Follows Graph (DFG)

Τα μοντέλα διεργασιών που δημιουργούνται με τη χρήση Petri net, έχουν μια καλά καθορισμένη σημασιολογία. Μία διεργασία ξεκινά από τις θέσεις που περιλαμβάνονται στην αρχική σήμανση(initial marking), και τελειώνει στις θέσεις που περιλαμβάνονται στην τελική σήμανση(final marking).

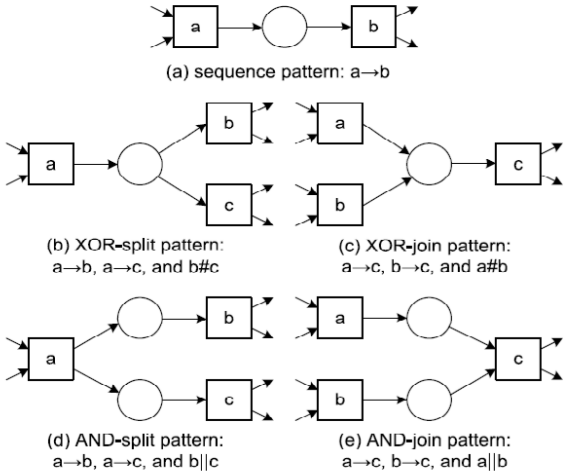
Τα γραφήματα Directly-Follows, ακολουθούν μια άλλη τεχνική. Είναι γραφήματα, όπου οι κόμβοι αντιπροσωπεύουν τα συμβάντα/δραστηριότητες στο αρχείο καταγραφής συμβάντων, και οι κατευθυνόμενες ακμές υπάρχουν μεταξύ των κόμβων εάν υπάρχει τουλάχιστον ένα ίχνος στο αρχείο, όπου το συμβάν/δραστηριότητα πηγής, ακολουθείται από το συμβάν/δραστηριότητα στόχο. Εκτός από τις κατευθυνόμενες ακμές, είναι εύκολο να αναπαραστήσουμε μετρήσεις όπως είναι η συχνότητα (μετρώντας πόσες φορές το συμβάν/δραστηριότητα πηγή, ακολουθείται από το συμβάν/δραστηριότητα στόχο), και η απόδοση (π.χ. ο μέσος όρος του χρόνου που μεσολάβησε μεταξύ των δύο συμβάντων).

3.6 Αλγόριθμοι ανακάλυψης διαδικασιών (Process Discovery Algorithms)

3.6.1 Alpha Miner

Πρόκειται για έναν από τους πιο γνωστούς αλγόριθμους, που χρησιμοποιείται στην εξόρυξη διεργασιών, με στόχο την ανακατασκευή της αιτιότητας από ένα σύνολο γεγονότων, μετατρέποντας το αρχείο καταγραφής συμβάντων σε ένα δίκτυο ροής διεργασιών με βάση τις σχέσεις μεταξύ των διαφόρων δραστηριοτήτων που περιέχονται στο αρχείο. Ο Alpha miner μπορεί να βρει:

- Το μοντέλο Petri Net όπου όλες οι μεταβάσεις είναι ορατές και μοναδικές και αντιστοιχούν σε γεγονότα όπως είναι οι δραστηριότητες.
- Το initial marking το οποίο περιγράφει την κατάσταση του μοντέλου Petri Net όταν ξεκινάει μια εκτέλεση.
- Το final marking το οποίο περιγράφει την κατάσταση του μοντέλου Petri Net όταν τελειώνει μια εκτέλεση.



Εικόνα 7: Απεικόνιση απλών προτύπων διαδικασίας και σχέσεων μεταξύ δραστηριοτήτων

Ο αλγόριθμος Alpha miner, ωστόσο έχει ορισμένα μειονεκτήματα:

- Δεν μπορεί να χειριστεί βρόχους μήκους ένα και δύο
- Δεν μπορεί να εντοπίσει αόρατες και διπλές εργασίες
- Το μοντέλο που θα ανακαλυφθεί ενδέχεται να μην είναι βάσιμο
- Είναι αδύναμο έναντι του θορύβου

3.6.2 Alpha+ Miner

Βελτιωμένη εκδοχή του αλγορίθμου Alpha, με τη δυνατότητα να ανακαλύπτει βρόχους με μήκος 1 ή 2.

3.6.3 Inductive Miner

Ο αλγόριθμος Inductive miner χρησιμοποιεί τρία κύρια βήματα που εκτελούνται αναδρομικά:

Εικόνα 8: Αναδρομή του inductive miner

Η βασική ιδέα του αλγορίθμου είναι η ανίχνευση μιας «τομής» στο αρχείο καταγραφής. Αυτή η τομή μπορεί να βασίζεται σε διαδοχικότητα βρόχου, παραλληλότητα βρόχου ή και περικοπή βρόχου.

Αρχικά, ο αλγόριθμος υπολογίζει ένα dfg (directly follows graph) με βάση το αρχείο καταγραφής. Χρησιμοποιώντας το dfg, εκτελεί ανίχνευση τομής(cut detection). Εάν η ανίχνευση κοπής είναι επιτυχής, ο αλγόριθμος επιστρέφει έναν cut-operator και ένα cut-partition. Με βάση τον επιστρεφόμενο cut-operator και το cut-partition, χωρίζει το αρχείο καταγραφής σε έναν ή περισσότερους υποκαταλόγους(sub logs). Αυτό το βήμα ονομάζεται log splitting. Στη συνέχεια, ο αλγόριθμος εφαρμόζεται αναδρομικά σε κάθε sub log.

Κατά τη διαδικασία του cut detection, ο αλγόριθμος αναζητά μια περικοπή με την ακόλουθη σειρά:

- Exclusive choice (XOR) cut
- Sequence cut
- Concurrent (parallel) cut
- Loop cut

Εάν βρεθεί μια τομή, η ανίχνευση ολοκληρώνεται. Για παράδειγμα, εάν εντοπιστεί μια τομή exclusive choice, η ανίχνευση σταματά και δεν θα ανιχνευθεί άλλη τομή.

Ανάλογα με τον τύπο τομής που βρέθηκε, εφαρμόζεται μία διαφορετική λειτουργία διαχωρισμού. Υπάρχουν τέσσερις λειτουργίες:

- Exclusive choice split
- Sequence split
- Concurrent split
- Loop split

Σε περίπτωση που δεν εντοπιστεί καμία τομή, ο αλγόριθμος εφαρμόζει fall through. Υπάρχουν έξι κατηγορίες fall through που εφαρμόζονται με την ακόλουθη σειρά:

- **Empty traces:** Εφαρμόζεται όταν το αρχείο καταγραφής συμβάντων περιέχει κενά ίχνη.
- **Activity once per trace:** Ισχύει εάν μια δραστηριότητα εμφανίζεται ακριβώς μια φορά σε κάθε ίχνος ενός αρχείου καταγραφής συμβάντων.
- **Activity concurrent:** Προσπαθεί να βρει μια τομή αφήνοντας εκτός μια δραστηριότητα από το αρχείο καταγραφής συμβάντων, και επαναλαμβάνει για όλες τις υπόλοιπες δραστηριότητες. Δημιουργείται ένα νέο αρχείο καταγραφής συμβάντων.
- **Strict tau loop:** Ο αλγόριθμος κατασκευάζει ένα δεύτερο αρχείο καταγραφής, με βάση το υπάρχον αρχείο, διαχωρίζει τα ίχνη σε κάθε εμφάνιση μιας end activity που ακολουθείται από μία start activity. Τα ίχνη που προκύπτουν αποθηκεύονται στο αρχείο καταγραφής.
- **Tau loop:** Η συμπεριφορά είναι παρόμοια με την παραπάνω, με τη διαφορά ότι εδώ ο αλγόριθμος διαχωρίζει τα ίχνη μόνο σε κάθε εμφάνιση μιας start activity.
- **Flower model:** Εφαρμόζεται για οποιοδήποτε αρχείο καταγραφής, το οποίο δε περιέχει κενά ίχνη.

3.6.4 Heuristic Miner

Πρόκειται για έναν αλγόριθμο που δρα στο Directly-Follows Graph, παρέχοντας έναν τρόπο αντιμετώπισης του θορύβου και εύρεσης κοινών construct μεταξύ δύο δραστηριοτήτων. Η έξοδος του, είναι ένα Heuristics Net, δηλαδή ένα αντικείμενο που περιέχει τις δραστηριότητες και τις μεταξύ τους σχέσεις. Το Heuristics Net μπορεί στη συνέχεια να μετατραπεί σε ένα Petri Net.

Σε αντίθεση με τον Alpha miner, λαμβάνει υπόψη τη συχνότητα των γεγονότων και των ακολουθιών. Χρησιμοποιεί πολλές παραμέτρους για να φιλτράρει σπάνια μονοπάτια και συμβάντα, το οποίο είναι χρήσιμο για μεγάλα αρχεία καταγραφής συμβάντων.

3.7 Έλεγχος συμμόρφωσης (Conformance checking)

Επόμενο βήμα είναι η εφαρμογή του ελέγχου συμμόρφωσης. Ο έλεγχος συμμόρφωσης είναι μια τεχνική σύγκρισης ενός μοντέλου διεργασίας με ένα αρχείο καταγραφής συμβάντων της ίδιας διαδικασίας. Ο στόχος αυτής της τεχνικής είναι να γίνει έλεγχος για το εάν το αρχείο καταγραφής συμβάντων συμμορφώνεται με το μοντέλο, και αντίστροφα.

Μπορεί να χρησιμοποιηθεί για την εύρεση αποκλίσεων, την εύρεση αδυναμιών ή τη διόρθωση μοντέλων που δεν πληρούν ορισμένες από τις προϋποθέσεις ενός ποιοτικού μοντέλου. Ένα βασικό χαρακτηριστικό του ελέγχου συμμόρφωσης είναι η καταλληλότητα (fitness), η οποία είναι μια τιμή που δείχνει πόσο καλά ταιριάζει το ίχνος ή το αρχείο καταγραφής συμβάντων στο μοντέλο. Μπορεί να πάρει τιμή μεταξύ 0 (το ίχνος ή το αρχείο καταγραφής δε ταιριάζει καθόλου με το μοντέλο) και το 1 (το ίχνος ή το αρχείο καταγραφής ταιριάζει απόλυτα στο μοντέλο).

Εφαρμόζονται δύο κύριες τεχνικές conformance checking:

- Token-based replay
- Alignments

3.7.1 Token-based replay

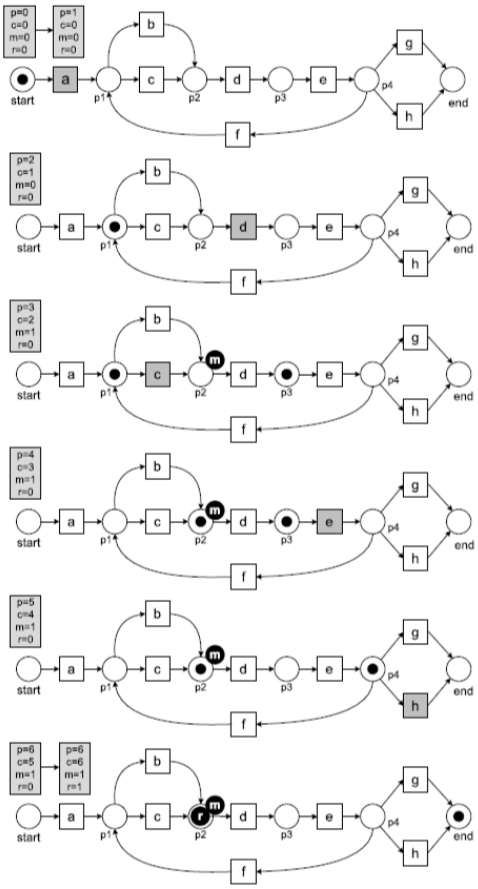
Ένα ίχνος ταιριάζει με το μοντέλο, εάν κατά την εκτέλεση του, οι μεταβάσεις μπορούν να ενεργοποιηθούν χωρίς να χρειάζεται να εισαχθεί κανένα token που λείπει.

Για κάθε ίχνος, υπάρχουν τέσσερις τιμές που πρέπει να καθοριστούν:

- Produced tokens - p
- Remaining tokens - r
- Missing tokens - m
- Consumed tokens – c

Με βάση αυτά, μπορεί να προκύψει μια φόρμουλα, όπου ένα Petri net (n) και ένα ίχνος (t) δίνοντας ως είσοδος : $\mathbf{fitness(n, t)=\frac{1}{2}(1-\frac{r}{p})+\frac{1}{2}(1-\frac{m}{c})}$. Για να εφαρμοστεί ο τύπος σε ολόκληρο το αρχείο καταγραφής, τα p,r,m και c υπολογίζονται για κάθε ίχνος, συνοψίζονται και τελικά τοποθετούνται στον παραπάνω τύπο στο τέλος.

Αρχικά τα p,r,m,c έχουν τιμή 0 και όλες οι θέσεις είναι κενές. Στη συνέχεια, το πρώτο token παράγεται στην αρχική θέση. Το p αυξάνεται κατά 1. Όταν εκτελείται μια μετάβαση, καταναλώνει ένα token από κάθε θέση εισόδου. Αυτό αυξάνει τον μετρητή των token που καταναλώθηκαν κατά ένα δεδομένο αριθμό θέσεων. Στη συνέχεια, δημιουργείται ένα token σε κάθε θέση εξόδου. Αυτό αυξάνει τον αριθμό των παραγόμενων token. Όταν ένα token εισέρχεται στην τελική θέση, καταναλώνεται από το περιβάλλον. Εάν, μετά την εφαρμογή του token-based replay, τα m και r έχουν την τιμή 0, το ίχνος επαναλήφθηκε τέλεια. Όσο μεγαλύτερος είναι ο αριθμός των m και r, τόσο χαμηλότερο είναι το fitness του μοντέλου.



Εικόνα 9: Παράδειγμα εφαρμογής Token-based replay

Η παραπάνω τεχνική έχει πλεονεκτήματα καθώς και μειονεκτήματα.

Τα πλεονεκτήματα της είναι:

- Πρόκειται για έναν απλό και μη απαιτητικό αλγόριθμο.
- Επιτρέπει τη βασική διάγνωση με βάση τα remaining και missing tokens (εάν πολλά remaining tokens συγκεντρώνονται σε ένα μέρος, αυτό σημαίνει ότι το μοντέλο αναμένει μετάβαση που δεν συμβαίνει στο αρχείο καταγραφής, ή το αντίθετο, μπορεί να σημαίνει ότι η δραστηριότητα εμφανίζεται συχνά στο αρχείο καταγραφής, αλλά δεν επιτρέπεται από το μοντέλο).

Τα μειονεκτήματα της είναι:

- Το fitness μπορεί μερικές φορές να είναι πολύ υψηλό εάν υπάρχουν αποκλίσεις. Το Petri net «πλημμυρίζεται» από tokens και επιτρέπει οποιαδήποτε συμπεριφορά.
- Μπορεί να εφαρμοστεί μόνο σε Petri nets.
- Εάν ένα case δεν ταιριάζει, η προσέγγιση δεν δημιουργεί μια αντίστοιχη διαδρομή μέσω του μοντέλου.

Το Token-based replay επιτρέπει τη λήψη λεπτομερών πληροφοριών σχετικά με μεταβάσεις που δεν εκτελέστηκαν σωστά ή δραστηριότητες που βρίσκονται στο αρχείο καταγραφής και όχι στο μοντέλο. Συγκεκριμένα, οι εκτελέσεις που δεν ταιριάζουν με το μοντέλο, αναμένεται να έχουν μεγαλύτερο χρόνο διεκπεραίωσης.

Τα διαγνωστικά που παρέχονται από την pm4py είναι τα ακόλουθα:

- Ανάλυση απόδοσης στις μεταβάσεις που εκτελούνται με ένα τρόπο που δε ταιριάζει με το μοντέλο.
- Ανάλυση απόδοσης για τις δραστηριότητες που δεν περιλαμβάνονται στο μοντέλο.
- Root cause analysis για τις αιτίες που οδηγούν σε ακατάλληλη εκτέλεση των μεταβάσεων.
- Root cause analysis των αιτιών που οδηγούν στην εκτέλεση δραστηριοτήτων που δεν περιέχονται στο μοντέλο διαδικασίας.

3.7.2 Alignments

Η τεχνική Alignment-based replay στοχεύει στο να βρει μία από τις καλύτερες ευθυγραμμίσεις μεταξύ του ίχνους και του μοντέλου. Για κάθε ίχνος, η έξοδος ενός alignment είναι μια λίστα ζευγαριών, όπου το πρώτο στοιχείο είναι ένα συμβάν και το δεύτερο είναι μια μετάβαση. Για κάθε alignment θα μπορούσε να ορισθεί η παρακάτω ταξινόμηση:

- **Sync moves**: Η ταξινόμηση του συμβάντος αντιστοιχεί στην ετικέτα μετάβασης. Σε αυτή την περίπτωση, τόσο το ίχνος όσο και το μοντέλο προχωρούν με τον ίδιο τρόπο κατά την επανάληψη.
- **Move on log**: Για ζευγάρια όπου το δεύτερο στοιχείο είναι », αντιστοιχεί σε μια κίνηση επανάληψης στο ίχνος που δεν συμμορφώνεται με το μοντέλο. Αυτό σηματοδοτεί ότι το fitness δεν είναι το επιθυμητό, και υπάρχει μια απόκλιση μεταξύ του ίχνους και του μοντέλου.
- **Move on model**: Για ζευγάρια όπου το πρώτο στοιχείο είναι », αντιστοιχεί σε μια κίνηση επανάληψης στο μοντέλο που δεν συμμορφώνεται με το ίχνος. Για κινήσεις στο μοντέλο μπορούμε να έχουμε την ακόλουθη διάκριση:
    - Κινήσεις στο μοντέλο που περιλαμβάνει κρυφές μεταβάσεις, σε αυτή την περίπτωση η κίνηση είναι κατάλληλη.
    - Κινήσεις στο μοντέλο που δεν περιλαμβάνει κρυφές μεταβάσεις, σε αυτή την περίπτωση, η κίνηση είναι ακατάλληλη και σηματοδοτεί μια απόκλιση μεταξύ του ίχνους και του μοντέλου.

Τα πλεονεκτήματα αυτής της τεχνικής είναι τα παρακάτω:

- Παρέχει πιο λεπτομερή διαγνωστικά.
- Παρέχει δυνατότητα ρύθμισης μέσω της συνάρτησης κόστους.
- Παρέχει πιο ακριβή διαγνωστικά από τον αλγόριθμο token-based replay. Τα remaining token ενδέχεται να παρέχουν παραπλανητικά διαγνωστικά λόγω προηγούμενων αποκλίσεων, τα οποία μπορούν να καλύψουν μεταγενέστερες αποκλίσεις.
- Μπορεί να χρησιμοποιηθεί οποιοδήποτε μοντέλο με αρχική και τελική κατάσταση, επομένως ο αλγόριθμος είναι ανεξάρτητος από το μοντέλο.

3.8 Ανάλυση απόδοσης (Performance analysis)

Τελευταίο βήμα είναι η ανάλυση απόδοσης. Πρόκειται για μια ανάλυση η οποία βοηθάει στην κατανόηση της απόδοσης, παρέχοντας αντικειμενικές πληροφορίες. Έχει ως σκοπό την βελτιστοποίηση των υπαρχουσών τεχνικών, καθώς και την εκμάθηση καινούριων.

3.8.1 Αξιολόγηση (Evaluation)

Είναι δυνατό να συγκριθεί η συμπεριφορά που περιέχεται στο αρχείο καταγραφής και η συμπεριφορά που περιέχεται στο μοντέλο, ώστε να δούμε αν και πως ταιριάζουν. Υπάρχουν τέσσερα κύρια κριτήρια:

- **Fitness (καταλληλότητα):** Ο υπολογισμός της καταλληλότητας έχει στόχο να υπολογίσει πόση από τη συμπεριφορά στο αρχείο καταγραφής, γίνεται αποδεκτή από το μοντέλο διαδικασίας. Αυτό μπορεί να γίνει είτε με τη μέθοδο του token-based replay, είτε με την μέθοδο alignments.

- **Precision (ακρίβεια):** Η ακρίβεια υπολογίζεται συγκρίνοντας τη συμπεριφορά που υπάρχει στο μοντέλο, με τη συμπεριφορά που επιτρέπεται από το αρχείο καταγραφής. Η ακρίβεια είναι υψηλή εάν το μοντέλο δεν επιτρέπει πολύ μεγαλύτερη συμπεριφορά από αυτή που καταγράφεται στο αρχείο καταγραφής. Αντίθετα είναι χαμηλή, αν το μοντέλο επιτρέπει.

- **Generalization (γενίκευση):** Στις δύο προηγούμενες περιπτώσεις, την καταλληλότητα και την ακρίβεια, λαμβάνεται υπόψη μόνο η σχέση μεταξύ του αρχείου καταγραφής και του μοντέλου διαδικασίας. Ωστόσο, το αρχείο καταγραφής συμβάντων περιέχει ένα μέρος όλης της πιθανής συμπεριφοράς που επιτρέπεται από το σύστημα. Επομένως, η γενίκευση (generalization) θα πρέπει να υποδεικνύει εάν το μοντέλο δεν «υπερπροσαρμόζεται» στη συμπεριφορά που φαίνεται στο αρχείο, και περιγράφει το πραγματικό σύστημα. Ένα μοντέλο είναι γενικό εάν τα στοιχεία του μοντέλου επισκέπτονται αρκετά συχνά κατά τη διάρκεια μιας λειτουργίας επανάληψης. Στην περίπτωση που το μοντέλο ταιριάζει υπερβολικά, είναι πιθανό να υποστηρίζει παραλλαγές όπως είναι ο θόρυβος. Με αυτό τον τρόπο γίνεται πολύ συγκεκριμένο και επομένως αποτυγχάνει να αποτυπώσει την πραγματική δομή των δεδομένων. Έτσι, ένα μοντέλο με καλή γενίκευση έχει σκοπό να εμποδίσει την υπερπροσαρμογή.

- **Simplicity (απλότητα):** Η διάσταση της απλότητας αξιολογεί πόσο εύκολα κατανοητό είναι ένα μοντέλο διαδικασίας για έναν άνθρωπο. Αυτή η διάσταση επομένως δεν σχετίζεται άμεσα με την παρατηρούμενη συμπεριφορά, αλλά μπορεί να εξετάσει το μοντέλο μεμονωμένα. Εφόσον, μια συμπεριφορά μπορεί να περιγραφεί με διαφορετικούς τρόπους, χρησιμοποιώντας διαφορετικά μοντέλα, η επιλογή του απλούστερου είναι το καλύτερο. Ένα μοντέλο έχει καλή απλότητα εάν δεν έχει πολυπλοκότητα και είναι εύκολο να κατανοηθεί από έναν άνθρωπο.

3.9 Directly-Follows Graphs

Ένα επιπλέον βήμα που θα μας βοηθήσει στην οπτικοποίηση των αποτελεσμάτων μας είναι τα γραφήματα Directly-Follows, καθώς και τα δέντρα διεργασίας.

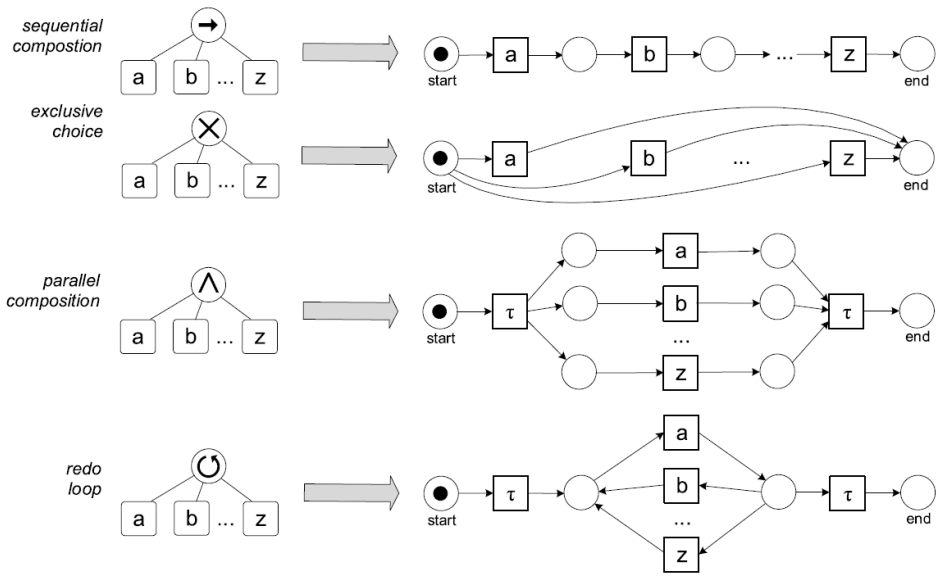
Τα γραφήματα Directly-Follows μπορεί να περιέχουν έναν τεράστιο αριθμό δραστηριοτήτων και μονοπατιών, γι' αυτό το λόγο, ορισμένα δεν αντιπροσωπεύουν την πραγματικότητα. Ωστόσο, για να το αποφύγουμε αυτό, μπορούμε να φιλτράρουμε τις δραστηριότητες και τα μονοπάτια του γραφήματος, διατηρώντας ένα υποσύνολο της συμπεριφοράς του.

3.10 Δέντρα διεργασίας (Process Trees)

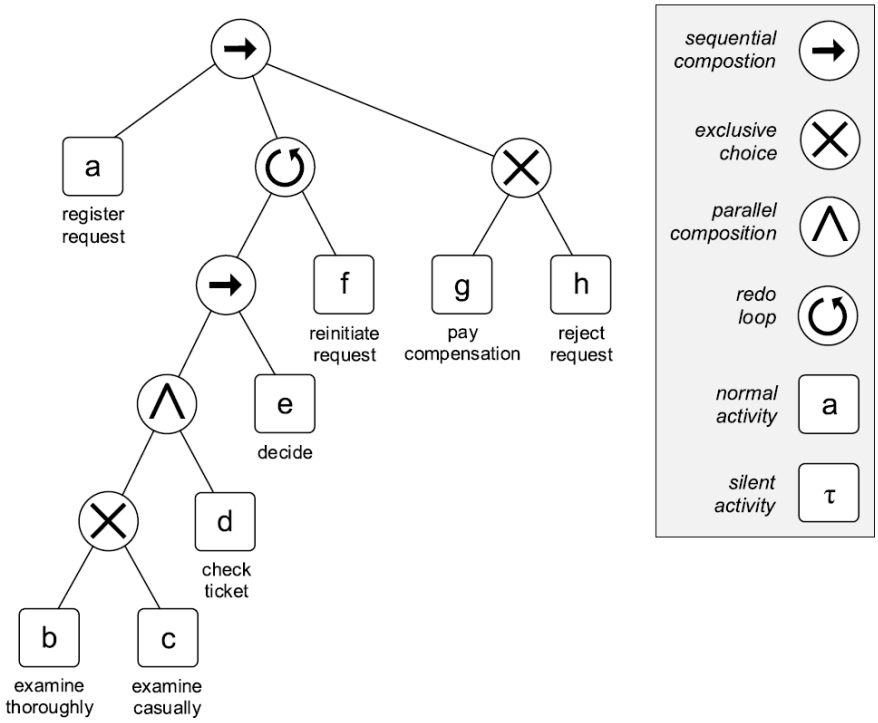
Πρόκειται για ένα δέντρο, όπου τα φύλλα είναι σχολιασμένα με δραστηριότητες ή σιωπηλές δραστηριότητες. Οι εσωτερικοί κόμβοι ενός δέντρου διεργασίας σχολιάζονται με τελεστές που καθορίζουν τη σειρά με την οποία μπορούν οι δραστηριότητες να εκτελεστούν.

Υπάρχουν τέσσερα είδη τελεστών:

- Διαδοχική σύνθεση (sequential composition - sequence)
- Αποκλειστική επιλογή (exclusive choice - xor)
- Παράλληλη σύνθεση (parallel composition – parallel)
- Επανάληψη βρόχου (redo loop – loop)



Εικόνα 10:Απεικόνιση τελεστών ενός Process Tree



Εικόνα 11: Παράδειγμα ενός Process Tree

Κεφάλαιο 4ο: Εφαρμογή σε διαδικασίες χρηματοδότησης της Τοπικής Αυτοδιοίκησης

Οργανισμοί τοπικής αυτοδιοίκησης (Ο.Τ.Α.) ονομάζονται τα νομικά πρόσωπα δημοσίου δικαίου καθένα από τα οποία έχει συσταθεί σε συγκεκριμένη εδαφική περιφέρεια και έχει ως σκοπό τη διοίκηση των τοπικών ζητημάτων, τα δε όργανά του εκλέγονται με καθολική ψηφοφορία από τους δημότες της περιφέρειας.

Οι βασικές αρχές συγκρότησης και λειτουργίας των Ο.Τ.Α. τίθενται από το Σύνταγμα, σύμφωνα με το οποίο:

- Οι Ο.Τ.Α. είναι υπεύθυνοι για τη διοίκηση των τοπικών υποθέσεων. Μάλιστα σε περίπτωση σύγκρουσης αρμοδιοτήτων μεταξύ Ο.Τ.Α. και κεντρικής διοίκησης, συντρέχει υπέρ των πρώτων τεκμήριο αρμοδιότητας. Επίσης η κεντρική διοίκηση μπορεί να αναθέσει στους Ο.Τ.Α. και δικές της αρμοδιότητες.
- Οι Ο.Τ.Α. χαίρουν διοικητικής και οικονομικής αυτοτέλειας και οι αρχές τους εκλέγονται με καθολική μυστική ψηφοφορία.
- Η εποπτεία του κράτους στους Ο.Τ.Α. περιορίζεται αποκλειστικά στον έλεγχο νομιμότητας των αποφάσεων και πράξεων τους. Επίσης το κράτος είναι υποχρεωμένο να λαμβάνει όλα εκείνα τα μέτρα που απαιτούνται για την εξασφάλιση της οικονομικής αυτοτέλειας των Ο.Τ.Α., καθώς και να τους ενισχύει οικονομικά όταν τους μεταβιβάζει δικές του αρμοδιότητες.

4.1 Υλοποίηση

4.1.1 Dataset

Το dataset από το οποίο θα αντλήσουμε τα δεδομένα μας είναι το παρακάτω:

- https://data.4tu.nl/articles/dataset/BPI_Challenge_2015_Municipality_1/12709154/1

Πρόκειται για ένα dataset που περιλαμβάνει δεδομένα σχετικά με τις αιτήσεις οικοδομικών αδειών ενός δήμου σε διάστημα τεσσάρων ετών.

Αποτελείται από:

- 1199 cases
- 52217 events
- 398 event classes

Τα cases που περιέχονται στο αρχείο καταγραφής περιέχουν πληροφορίες για την κύρια αίτηση καθώς και διαδικασίες ενστάσεων σε διάφορα στάδια. Επίσης, παρέχει πληροφορίες σχετικά με τον πόρο που πραγματοποίησε την εργασία καθώς και για το κόστος. Τα events αναγνωρίζονται με ένα κωδικό καθώς και μια ετικέτα στα αγγλικά. Κάθε activity code αποτελείται από τρία μέρη: 2 ψηφία, μία ποικιλία χαρακτήρων και άλλα 3 ψηφία. Τα δύο πρώτα ψηφία καθώς και οι χαρακτήρες υποδηλώνουν την υποδιεργασία στην οποία ανήκει η δραστηριότητα. Για παράδειγμα το '01_HOOFD_xxx' δηλώνει την κύρια διαδικασία και το '01_BB_xxx' δηλώνει την υποδιεργασία "objections and complaints" ("Beroep en Bezwaar" στα Ολλανδικά). Τα τελευταία τρία ψηφία υποδεικνύουν τη σειρά με την οποία εκτελούνται οι δραστηριότητες.

BPIC15_1.xes

```
        <string key="case_type" value="UNKNOWN"/>
    </global>
    <global scope="event">
        <string key="dateFinished" value="UNKNOWN"/>
        <string key="org:resource" value="UNKNOWN"/>
        <date key="time:timestamp" value="1970-01-01T00:00:00+01:00"/>
        <string key="lifecycle:transition" value="UNKNOWN"/>
        <string key="question" value="UNKNOWN"/>
        <string key="concept:name" value="UNKNOWN"/>
        <string key="monitoringResource" value="UNKNOWN"/>
        <string key="activityNameNL" value="UNKNOWN"/>
        <string key="activityNameEN" value="UNKNOWN"/>
    </global>
    <classifier name="Activity classifier" keys="concept:name lifecycle:transition"/>
    <classifier name="Activity classifier activityNameNL" keys="activityNameNL lifecycle:transition"/>
    <classifier name="Activity classifier activityNameEN" keys="activityNameEN lifecycle:transition"/>
    <classifier name="Resource" keys="org:resource"/>
    <float key="meta_org:different_resources_standard_deviation" value="0.906"/>
    <int key="meta_org:different_resources_min" value="1"/>
    <int key="meta_concept:named_events_total" value="52217">
        <int key="01_HOOFD_110" value="539"/>
        <int key="01_BB_770" value="157"/>
        <int key="11_AH_II_055" value="5"/>
        <int key="01_HOOFD_590" value="1"/>
        <int key="01_HOOFD_470" value="247"/>
        <int key="11_AH_II_050" value="3"/>
        <int key="08_AWB45_075" value="31"/>
        <int key="08_AWB45_080" value="3"/>
        <int key="01_BB_775" value="105"/>
        <int key="07_OPS_080_2" value="2"/>
        <int key="07_OPS_080_0" value="3"/>
        <int key="07_OPS_080_1" value="2"/>
        <int key="01_HOOFD_101" value="843"/>
        <int key="01_HOOFD_465" value="83"/>
        <int key="01_HOOFD_100" value="388"/>
        <int key="11_AH_II_040" value="25"/>
        <int key="08_AWB45_060" value="50"/>
        <int key="01_HOOFD_460" value="83"/>
        <int key="08_AWB45_180" value="4"/>
```

Εικόνα 12: Απόσπασμα από το dataset που θα χρησιμοποιήσουμε

Αρχικά, εισάγουμε τα δεδομένα και στη συνέχεια δημιουργούμε μία συνάρτηση με την οποία θα βρούμε και θα εμφανίσουμε πληροφορίες για το πρώτο *trace*, το πρώτο *event* από το πρώτο *trace* και τα *start/end activities*.

```
# importing the dataset
log = pm4py.read_xes('D:\Programs\PyCharm\Datasets\BPI\input_data\BPIC15_1.xes')

#print start/end activities of the log
start_end_activities(log)
```

```
def start_end_activities(file):
    start_activities = pm4py.get_start_activities(file)
    end_activities = pm4py.get_end_activities(file)
    print(log[0]) #print the first trace of the log
    print(log[0][0])#print the first event of the first trace
    print("Start activities: {}\nEnd activities: {}".format(start_activities, end_activities))
```

Εικόνα 13: Εισαγωγή δεδομένων και κάλεσμα συνάρτησης start_end_activities

```
D:\Programs\PyCharm\CvLab\venv\Scripts\python.exe D:/Programs/PyCharm/ProcessMining/bpi15.py
parsing log, completed traces :: 100%|██████████| 1199/1199 [00:04<00:00, 247.17it/s]
{'attributes': {'endDate': datetime.datetime(2014, 7, 18, 0, 0, tzinfo=datetime.timezone(datetime.timedelta(seconds=7200))), 'caseStatus'
  'O', 'SUMleges': 157.09305, 'last_phase': 'Beschikking verzonden', 'case_type': '557669', 'concept:name': '10009138',
 'Responsible_actor': '560464', 'parts': 'Bouw,Reclame', 'termName': 'Termijn bezwaar en beroep 1', 'endDatePlanned': datetime.datetime
 (2014, 7, 18, 0, 0, tzinfo=datetime.timezone(datetime.timedelta(seconds=7200))), 'startDate': datetime.datetime(2014, 4, 11, 0, 0,
 tzinfo=datetime.timezone(datetime.timedelta(seconds=7200))), 'requestComplete': 'TRUE', 'IDofConceptCase': '10009280'}, 'events':
 [{'question': 'EMPTY', 'dateFinished': '2014-04-14 00:00:00', 'dueDate': datetime.datetime(2014, 4, 17, 9, 30, 39, tzinfo=datetime
 .timezone(datetime.timedelta(seconds=7200))), 'action_code': '01_HOOFD_010', 'activityNameEN': 'register submission date request',
 'planned': datetime.datetime(2014, 4, 15, 9, 30, 39, tzinfo=datetime.timezone(datetime.timedelta(seconds=7200))), 'time:timestamp':
 datetime.datetime(2014, 4, 11, 0, 0, tzinfo=datetime.timezone(datetime.timedelta(seconds=7200))), 'monitoringResource': '2670601',
 'org:resource': '9264148', 'activityNameNL': 'registratie datum binnenkomst aanvraag', 'concept:name': '01_HOOFD_010',
 'lifecycle:transition': 'complete'}, '..', {'monitoringResource': '560912', 'org:resource': '560912', 'activityNameNL': 'instellen
 besluitfase:besluit vergunnen verzonden', 'concept:name': '01_HOOFD_510_2a', 'question': 'EMPTY', 'dateFinished': '2014-06-04 00:00:00',
 'action_code': '01_HOOFD_510_2a', 'activityNameEN': 'start decision phase decision permitting sent', 'planned': datetime.datetime(2014,
 6, 5, 8, 41, 21, tzinfo=datetime.timezone(datetime.timedelta(seconds=7200))), 'lifecycle:transition': 'complete', 'time:timestamp':
 datetime.datetime(2014, 6, 5, 0, 0, tzinfo=datetime.timezone(datetime.timedelta(seconds=7200)))}]}
```

Εικόνα 14: Εκτύπωση πρώτου trace

Παρατηρούμε ότι για κάθε ίχνος, λαμβάνουμε μια πληθώρα πληροφοριών. Ορισμένες από τις πιο σημαντικές, που θα μας χρειαστούν στην συγκεκριμένη εργασία είναι:

- action_code
- activityNameEN
- org:resource
- concept:name
- case:endDate
- lifecycle:transition

```
{'question': 'EMPTY', 'dateFinished': '2014-04-14 00:00:00', 'dueDate': datetime.datetime(2014, 4, 17, 9, 30, 39, tzinfo=datetime.timezone
 (datetime.timedelta(seconds=7200))), 'action_code': '01_HOOFD_010', 'activityNameEN': 'register submission date request', 'planned':
 datetime.datetime(2014, 4, 15, 9, 30, 39, tzinfo=datetime.timezone(datetime.timedelta(seconds=7200))), 'time:timestamp': datetime
 .datetime(2014, 4, 11, 0, 0, tzinfo=datetime.timezone(datetime.timedelta(seconds=7200))), 'monitoringResource': '2670601',
 'org:resource': '9264148', 'activityNameNL': 'registratie datum binnenkomst aanvraag', 'concept:name': '01_HOOFD_010',
 'lifecycle:transition': 'complete'}

Process finished with exit code 0
```

Εικόνα 15: Εκτύπωση πρώτου event

Το πρώτο event του αρχείου καταγραφής, αφορά τη δραστηριότητα με **activityNameEN**: ‘register submission date request’, με **action_code**: ‘01_HOOFD_010’. Επίσης, έχουμε το org:resource: ‘9264148’ καθώς και dateFinished: ‘2014-04-14 00:00:00’. Τέλος παρατηρούμε ότι το lifecycle:transition έχει τιμή ‘complete’.

```
D:\Programs\PyCharm\CvLab\venv\Scripts\python.exe D:/Programs/PyCharm/ProcessMining/bpi15.py
parsing log, completed traces :: 100%|██████████| 1199/1199 [00:04<00:00, 249.28it/s]
Start activities: {'01_HOOFD_010': 1138, '01_HOOFD_065_0': 1, '01_HOOFD_080': 1, '01_HOOFD_011': 34, '01_HOOFD_030_2': 6,
 '01_HOOFD_195_2': 1, '05_EIND_020': 1, '01_HOOFD_065_2': 4, '01_HOOFD_490_3': 1, '11_AH_II_010': 1, '11_AH_II_040b': 7,
 '11_AH_II_037_2': 1, '01_HOOFD_490_5': 1, '02_DRZ_010': 2}
End activities: {'01_HOOFD_510_2a': 52, '01_HOOFD_510_2': 132, '01_HOOFD_490_5a': 25, '01_HOOFD_515': 3, '08_AWB45_090_2':
  22, '01_HOOFD_809': 8, '01_HOOFD_811': 7, '01_HOOFD_490_5': 22, '08_AWB45_020_1': 5, '01_HOOFD_510_4': 16,
 '05_EIND_025': 6, '01_HOOFD_809c': 3, '01_BB_770': 33, '01_HOOFD_815': 20, '01_HOOFD_814': 8, '01_HOOFD_516a': 9,
 '01_HOOFD_530a': 29, '01_HOOFD_015': 5, '01_HOOFD_030_2': 30, '01_HOOFD_530b': 2, '01_HOOFD_820': 51, '08_AWB45_052': 1,
 '05_EIND_030_2': 5, '08_AWB45_070_5': 1, '08_AWB45_020_2': 8, '08_AWB45_045': 4, '01_HOOFD_510_1': 6, '01_HOOFD_810': 20,
  '08_AWB45_090_2a': 4, '01_HOOFD_375': 1, '01_HOOFD_510_3': 2, '08_AWB45_025': 4, '04_BPT_021': 1, '08_AWB45_020_0': 2,
 '05_EIND_065': 4, '06_VD_060_2': 4, '04_BPT_029': 1, '01_HOOFD_061': 2, '01_HOOFD_060': 10, '01_HOOFD_530': 295,
 '05_EIND_060': 3, '01_HOOFD_790': 21, '05_EIND_010': 2, '01_HOOFD_800': 24, '01_HOOFD_120': 2, '01_HOOFD_490_3': 13,
 '01_HOOFD_100': 7, '08_AWB45_070_1': 3, '05_EIND_050': 1, '01_HOOFD_040': 9, '01_HOOFD_195_1': 1, '01_HOOFD_065_1': 3,
 '01_BB_670_2': 1, '01_BB_540': 5, '08_AWB45_090_1': 3, '01_HOOFD_030_1': 6, '05_EIND_040': 1, '06_VD_060': 3,
 '01_HOOFD_020': 3, '08_AWB45_030': 1, '05_EIND_070': 2, '01_HOOFD_520': 26, '01_HOOFD_065_2': 1, '01_HOOFD_010': 1,
 '04_BPT_005': 1, '01_HOOFD_180': 3, '04_BPT_030': 2, '02_DRZ_030': 1, '01_HOOFD_250': 1, '01_HOOFD_110_1': 1,
 '01_HOOFD_110_2': 2, '01_HOOFD_110': 1, '01_HOOFD_505': 3, '01_HOOFD_260': 1, '01_HOOFD_101': 17, '01_HOOFD_490_1': 5,
 '02_DRZ_020_1': 1, '01_HOOFD_490_2': 2, '01_HOOFD_200': 1, '01_HOOFD_050': 1, '01_HOOFD_250_2': 2, '01_HOOFD_805': 1,
 '01_BB_755': 3, '01_BB_680': 1, '10_UOV_065': 1, '06_VD_060_1': 1, '01_HOOFD_250_1': 1, '08_AWB45_010': 1, '03_GBH_005':
1, '01_HOOFD_516': 82, '01_BB_775': 8, '01_HOOFD_099': 2, '02_DRZ_020_2': 2, '01_HOOFD_519': 30, '02_DRZ_030_2': 1,
 '08_AWB45_090_3': 2, '01_HOOFD_193': 1, '01_HOOFD_510_0': 1, '01_HOOFD_495': 1, '01_BB_635': 1, '08_AWB45_090_4': 1,
 '05_EIND_030_0': 1, '01_HOOFD_532': 1, '14_VRIJ_060_2': 1, '08_AWB45_051_0': 1, '08_AWB45_070_4': 1, '10_UOV_030': 1,
 '08_AWB45_150': 1}
```

Εικόνα 16: Start/End activities

Στην παραπάνω εικόνα, έχουμε δύο εκτυπώσεις οι οποίες περιέχουν τις αρχικές δραστηριότητες(start activities) και τις τελικές δραστηριότητες(end activities) αντίστοιχα. Το αρχείο μας περιέχει 14 αρχικές δραστηριότητες και 101 τελικές δραστηριότητες.

4.1.2 Φιλτράρισμα με βάση το χρονικό πλαίσιο (Filtering on timeframe)

Επόμενο βήμα είναι η εφαρμογή φίλτρων επάνω στα δεδομένα μας, αρχικά φιλτράροντας με βάση το χρονικό πλαίσιο και στη συνέχεια με βάση τις τιμές των χαρακτηριστικών. Με το πρώτο φίλτρο, σκοπός μας είναι να εξάγουμε μόνο τα traces που περιέχονται σε ένα συγκεκριμένο χρονικό διάστημα, αυτό το πετυχαίνουμε ορίζοντας δύο μεταβλητές(date1, date2) οι οποίες αντιστοιχούν στον αρχικό και τελικό χρόνο αντίστοιχα. Αποθηκεύουμε το αποτέλεσμα στο αρχείο **filtered_timeframe.csv**.

```
def filter_timeframe(file, date1, date2):
    filtered_timeframe = pm4py.filter_time_range(log, date1, date2, mode='traces_contained')
    dataframe = pm4py.convert_to_dataframe(filtered_timeframe)
    dataframe.to_csv('D:/Programs/PyCharm/Datasets/BPI/output_data/filtering/filtered_timeframe.csv')
```

Εικόνα 17: Συνάρτηση filter_time_range

A	B	C	D	E	F	G	H
	question	dateFinished	dueDate	action_co	activityNa	planned	time:timestamp
0	EMPTY	26/6/2014 0:00	2014-06-2	01_HOOFI	register su	2014-06-2	2014-06-15 00:00:00+02:00
1	FALSE	26/6/2014 0:00		01_HOOFI	OLO mess	2014-06-2	2014-06-26 00:00:00+02:00
2	EMPTY	26/6/2014 0:00		01_HOOFI	phase app	2014-06-2	2014-06-26 00:00:00+02:00
3	EMPTY	23/6/2014 0:00	2014-06-2	01_HOOFI	register su	2014-06-2	2014-06-20 00:00:00+02:00
4	FALSE	23/6/2014 0:00		01_HOOFI	OLO mess	2014-06-2	2014-06-23 00:00:00+02:00
5	TRUE	23/6/2014 0:00		01_HOOFI	send conf	2014-06-2	2014-06-23 00:00:00+02:00
6	FALSE	23/6/2014 0:00	2014-06-3	02_DRZ_0	forward to	2014-06-2	2014-06-23 00:00:00+02:00
7	EMPTY	23/6/2014 0:00		01_HOOFI	send conf	2014-06-2	2014-06-23 00:00:00+02:00
8	EMPTY	23/6/2014 0:00		01_HOOFI	phase app	2014-06-2	2014-06-23 00:00:00+02:00
9	FALSE	26/6/2014 0:00		04_BPT_0(	MER requi	2014-06-2	2014-06-26 00:00:00+02:00
10	Uitgebreic	26/6/2014 0:00		04_BPT_0	activities	2014-06-2	2014-06-26 00:00:00+02:00
11	FALSE	26/6/2014 0:00		04_BPT_0	phased ap	2014-06-2	2014-06-26 00:00:00+02:00
12	FALSE	26/6/2014 0:00		01_HOOFI	create pro	2014-06-2	2014-06-26 00:00:00+02:00
13	TRUE	26/6/2014 0:00		01_HOOFI	publish	2014-06-2	2014-06-26 00:00:00+02:00
14	FALSE	26/6/2014 0:00		01_HOOFI	create sub	2014-06-2	2014-06-26 00:00:00+02:00
15	FALSE	26/6/2014 0:00		01_HOOFI	procedure	2014-06-2	2014-06-26 00:00:00+02:00
16	FALSE	26/6/2014 0:00		01_HOOFI	create pul	2014-06-2	2014-06-26 00:00:00+02:00
17	FALSE	26/6/2014 0:00		04_BPT_0(	regular pr	2014-06-2	2014-06-26 00:00:00+02:00
18	EMPTY	26/6/2014 0:00		04_BPT_0	start WAB	2014-06-2	2014-06-26 00:00:00+02:00
19	EMPTY	26/6/2014 0:00		01_HOOFI	registratic	2014-06-2	2014-06-26 00:00:00+02:00
20	EMPTY	26/6/2014 0:00		04_BPT_0	read field	2014-06-2	2014-06-26 00:00:00+02:00
21	EMPTY	26/6/2014 0:00		01_HOOFI	enter sen	2014-06-2	2014-06-26 00:00:00+02:00

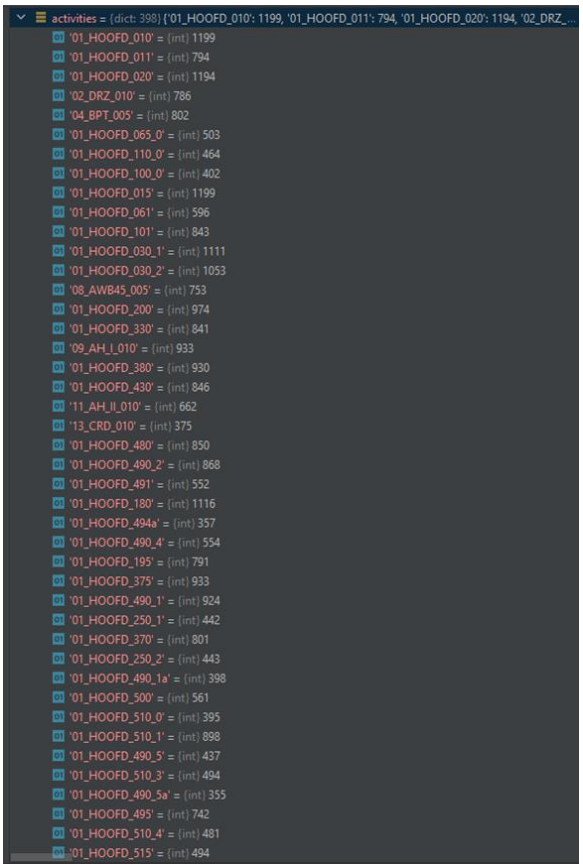
Εικόνα 18: Παράδειγμα χρήσης φίλτρου timeframe για τον μήνα Ιούνιο του 2014

4.1.3 Φιλτράρισμα με βάση τις τιμές χαρακτηριστικών (Filter on attribute values)

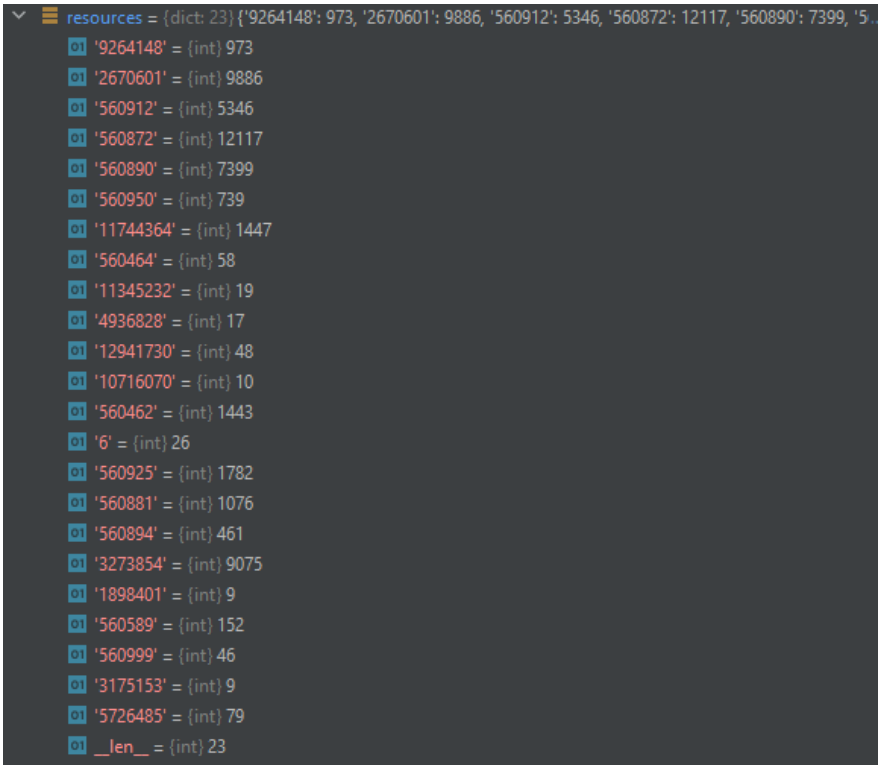
Σε αυτό το κομμάτι, αρχικά μέσω μιας συνάρτησης βρίσκουμε και τυπώνουμε τις δραστηριότητες(activities) και τις πηγές(resources). Στη συνέχεια δημιουργούμε δύο συναρτήσεις που μας επιτρέπουν να κάνουμε αναζήτηση μέσα στο event log, η μία θα δέχεται σαν όρισμα το resource που μας ενδιαφέρει και η άλλη το name. Με αυτό τον τρόπο εξάγουμε τα επιθυμητά αποτελέσματα και τα αποθηκεύουμε σε ένα αρχείο της μορφής csv.

```python
def get_attribute_values(file):
    activities = pm4py.get_event_attribute_values(log, "concept:name")
    resources = pm4py.get_event_attribute_values(log, "org:resource")
    print("Activities contained on the log: {}\nResources contained on the log: {}".format(activities, resources))


def filter_on_resource(file, resource):
    filter_resource = pm4py.filter_event_attribute_values(log, "org:resource", [resource], level="event", retain=True)
    dataframe = pm4py.convert_to_dataframe(filter_resource)
    dataframe.to_csv('D:/Programs/PyCharm/Datasets/BPI/output_data/filtering/filter_resource.csv')


def filter_on_name(file, name):
    filter_concept_name = pm4py.filter_event_attribute_values(log, "concept:name", [name], level="event", retain=True)
    dataframe = pm4py.convert_to_dataframe(filter_concept_name)
    dataframe.to_csv('D:/Programs/PyCharm/Datasets/BPI/output_data/filtering/filter_concept_name.csv')
```

Εικόνα 19: Συναρτήσεις get_attribute_values, filter_on_resource, filter_on_name



Εικόνα 20: Δραστηριότητες που περιέχονται στο αρχείο καταγραφής



Εικόνα 21: Πηγές που περιέχονται στο αρχείο καταγραφής

Παρατηρώντας τα αποτελέσματα, βλέπουμε ότι το αρχείο καταγραφής περιέχει 398 δραστηριότητες και 23 πηγές. Στη συνέχεια θα κάνουμε χρήση της συνάρτησης **filter_event_attribute_values** της pm4py και δίνοντας σαν όρισμα αρχικά resource = “2670601” και στη συνέχεια name = “01_HOOFD_101”, παίρνουμε τα παρακάτω αποτελέσματα:

	monitoringResource	org:resource	activityNameNL	concept:name
0	2670601	2670601	aanvraag volledig	08_AWB45_005
1	2670601	2670601	brief in behandeling versturen	01_HOOFD_200
2	2670601	2670601	procedure verandering	01_HOOFD_330
3	2670601	2670601	artikel 34 WABO van toepassing	09_AH_I_010
4	2670601	2670601	weigeringsgrond	01_HOOFD_380
5	2670601	2670601	belanghebbenden zienswijzen vragen	01_HOOFD_430
6	2670601	2670601	aanhoudingsgrond van toepassing	11_AH_II_010
7	2670601	2670601	coördinatie van toepassing	13_CRD_010
8	2670601	2670601	van rechtswege	01_HOOFD_480
9	2670601	2670601	besluitdatum voorafgaand aan beschikking	01_HOOFD_490_2
10	2670601	2670601	beschikking voorleggen	01_HOOFD_491
11	2670601	2670601	procedure verandering	01_HOOFD_180
12	2670601	2670601	fase beschikking gereed	01_HOOFD_494a
13	2670601	2670601	genereren beschikking omgevingsvergunning	01_HOOFD_490_4
14	2670601	2670601	fase aanvraag ontvankelijk	01_HOOFD_195
15	2670601	2670601	fase advies bekend	01_HOOFD_375
16	2670601	2670601	aanmaken besluit omgevingsvergunning	01_HOOFD_490_1
17	2670601	2670601	behandelen deelzaken inhoudelijkheid	01_HOOFD_250_1
18	2670601	2670601	inhoudelijke beoordeling afgerond	01_HOOFD_370
19	2670601	2670601	deelzaken inhoudelijkheid afgerond	01_HOOFD_250_2
20	2670601	2670601	instellen besluitstatus	01_HOOFD_490_1a
21	2670601	2670601	aanvraag volledig	08_AWB45_005
22	2670601	2670601	termijn aanleveren ontbrekende gegevens	08_AWB45_010
23	2670601	2670601	brief opvragen ontbrekende gegevens aanmaken	08_AWB45_020_0
24	2670601	2670601	procedure verandering	01_HOOFD_180
25	2670601	2670601	opvragen ontbrekende gegevens	08_AWB45_020_1
26	2670601	2670601	fase aanvullende gegevens gevraagd	08_AWB45_025
27	2670601	2670601	invoeren verzenddatum opvragen ontbrekende gegevens	08_AWB45_020_2
28	2670601	2670601	aanvullende gegevens ontvangen	08_AWB45_030
29	2670601	2670601	registreren ontvangst aanvullende gegevens	08_AWB45_040
30	2670601	2670601	fase aanvullende gegevens ontvangen	08_AWB45_045
31	2670601	2670601	procedureverandering na volledigheid	01_HOOFD_196
32	2670601	2670601	brief in behandeling versturen	01_HOOFD_200
33	2670601	2670601	procedure verandering	01_HOOFD_330
34	2670601	2670601	artikel 34 WABO van toepassing	09_AH_I_010
35	2670601	2670601	weigeringsgrond	01_HOOFD_380
36	2670601	2670601	belanghebbenden zienswijzen vragen	01_HOOFD_430

Εικόνα 22: Αποτελέσματα συνάρτησης filter_on_resource με όρισμα "2670601" (filter_resource.csv)

Για τη συγκεκριμένη πηγή, εξάγουμε όλες τις δραστηριότητες που περιλαμβάνει με τη σειρά που εκτελέστηκαν ώστε να δούμε αν περνάει από τις επιθυμητές μεταβάσεις.

	monitorin	org:resou	activityNameNL	concept:name	question	dateFinished
0	2670601	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	14/4/2014 0:00
1	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	17/4/2014 0:00
2	2670601	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	17/4/2014 0:00
3	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	25/4/2014 0:00
4	560912	560912	registratie datum publicatie	01_HOOFD_101	EMPTY	29/4/2014 0:00
5	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	29/4/2014 0:00
6	560890	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	1/5/2014 0:00
7	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	1/5/2014 0:00
8	2670601	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	15/5/2014 0:00
9	2670601	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	15/5/2014 0:00
10	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	12/5/2014 0:00
11	4676453	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	12/5/2014 0:00
12	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	12/5/2014 0:00
13	4676453	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	12/5/2014 0:00
14	2670601	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	15/5/2014 0:00
15	2670601	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	15/5/2014 0:00
16	2670601	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	15/5/2014 0:00
17	2670601	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	19/5/2014 0:00
18	560890	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	19/5/2014 0:00
19	2670601	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	22/5/2014 0:00
20	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	22/5/2014 0:00
21	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	23/5/2014 0:00
22	4676453	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	26/5/2014 0:00
23	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	2/6/2014 0:00
24	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	2/6/2014 0:00
25	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	2/6/2014 0:00
26	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	2/6/2014 0:00
27	4676959	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	5/6/2014 0:00
28	4676959	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	5/6/2014 0:00
29	4676959	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	5/6/2014 0:00
30	2670601	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	5/6/2014 0:00
31	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	6/6/2014 0:00
32	2670601	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	12/6/2014 0:00
33	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	16/6/2014 0:00
34	2670601	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	19/6/2014 0:00
35	2670601	560872	registratie datum publicatie	01_HOOFD_101	EMPTY	20/6/2014 0:00
36	2670601	9264148	registratie datum publicatie	01_HOOFD_101	EMPTY	23/6/2014 0:00

Εικόνα 23: Αποτελέσματα συνάρτησης filter_on_name με όρισμα "01_HOOFD_101" (filter_concept_name.csv)

Αντίστοιχα εδώ, λαμβάνουμε όλες τις περιπτώσεις στις οποίες περιέχεται το συγκεκριμένο concept:name. Παρατηρούμε ότι το concept:name αντιστοιχεί σε ένα μοναδικό activityName. Σε αυτό το κομμάτι, εστιάζουμε στο dateFinished ώστε να δούμε πότε ολοκληρώθηκε η συγκεκριμένη δραστηριότητα για κάθε περίπτωση.

4.1.4 Between filter

Με αυτό το φίλτρο, μετατρέπουμε το αρχείο καταγραφής ορίζοντας δύο διαδικασίες, τη δραστηριότητα πηγή και την δραστηριότητα στόχο. Σκοπός μας είναι να αναλύσουμε τη συμπεριφορά του αρχείου καταγραφής στο διάστημα που μεσολαβεί μεταξύ των δύο διαδικασιών, λαμβάνοντας αποτελέσματα για το χρόνο διεκπεραίωσης, ποιες δραστηριότητες περιλαμβάνονται και το επίπεδο συμμόρφωσης. Ορίζουμε σαν διαδικασία πηγή την διαδικασία “register submission date request" και σαν διαδικασία στόχο την “request complete”. Θέλουμε να δούμε πόσες μέρες χρειάζεται να περάσουν από τη στιγμή που θα γίνει η αίτηση μέχρι τη στιγμή που θα ολοκληρωθεί, αλλά και ποιες διεργασίες μεσολάβησαν.

```
def filter_between(file, act1, act2):
    filtered_log = pm4py.filter_between(file, act1, act2)
    dataframe = pm4py.convert_to_dataframe(filtered_log)
    dataframe.to_csv('D:/Programs/PyCharm/Datasets/BPI/output_data/filtering/filter_between.csv')
```

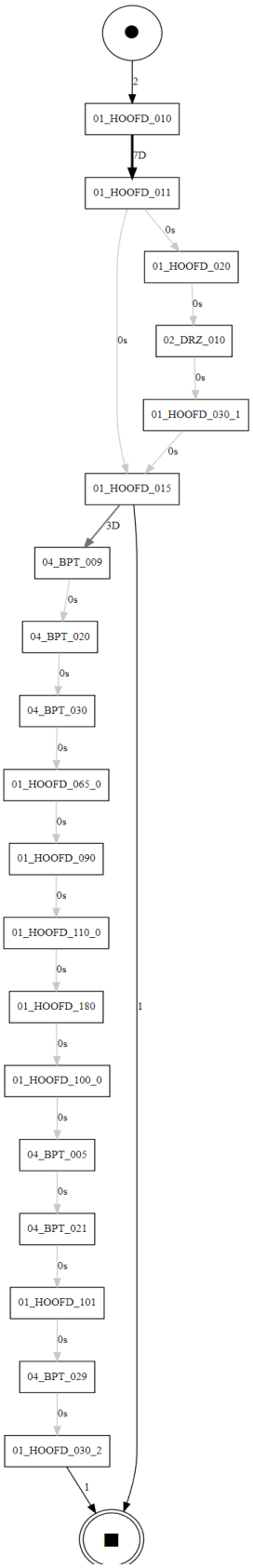
Εικόνα 24: Συνάρτηση filter_between

4.1.5 Ανακάλυψη διαδικασίας – Directly-follows Graph

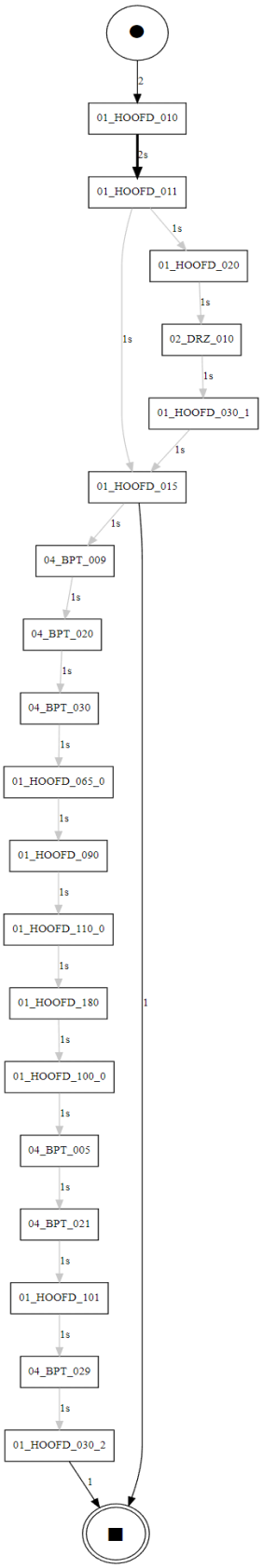
Επόμενο βήμα είναι να δημιουργήσουμε δύο συναρτήσεις. Η πρώτη θα χρησιμοποιεί την συνάρτηση **pm4py.discover_dfg()** της βιβλιοθήκης pm4py, η οποία δέχεται σαν είσοδο ένα αρχείο καταγραφής συμβάντων και στη συνέχεια εξάγει το Directly-Follows Graph το οποίο περιέχει τις συχνότητες των δραστηριοτήτων. Τέλος, εμφανίζει το γράφημα στην οθόνη και το αποθηκεύει σε ένα αρχείο της μορφής .svg. Η δεύτερη συνάρτηση χρησιμοποιεί την **pm4py.discover_performance_dfg()**, η διαφορά με την πρώτη είναι ότι εμφανίζει τις επιδόσεις των άκρων σε κάθε βήμα.

```
def directly_follows_graph(file):
    count = 1
    dfg, start_activities, end_activities = pm4py.discover_dfg(file)
    pm4py.view_dfg(dfg, start_activities, end_activities)
    pm4py.save_vis_performance_dfg(dfg, start_activities, end_activities,
                           '../Datasets/BPI/output_data/miners/export_dfg_'+str(count)+".svg")


def performance_directly_follows_graph(file):
    count = 1
    performance_dfg, start_activities, end_activities = pm4py.discover_performance_dfg(file)
    pm4py.view_performance_dfg(performance_dfg, start_activities, end_activities)
    pm4py.save_vis_performance_dfg(performance_dfg, start_activities, end_activities,
                           '../Datasets/BPI/output_data/miners/export_performance_dfg_'+str(count)+".svg")
```

Εικόνα 25: Συνάρτηση directly_follows_graph και performance_directly_follows_graph



Εικόνα 26: Εξαγόμενο Directly-Follows Graph (export_dfg_1.svg)



Εικόνα 27: Εξαγόμενο Performance Directly-Follows Graph (export_performance_dfg_1.svg)

Βλέπουμε ότι αρχική διεργασία είναι η **01_HOOFD_010**, η οποία ακολουθείται από την **01_HOOFD_011.** Έπειτα γίνεται διαχωρισμός σε δύο ίχνη τα οποία οδηγούν στην **01_HOOFD_015.** Από εκεί διακρίνουμε πάλι δύο ίχνη, ένα που οδηγεί απευθείας στην τελική διεργασία, και ένα το οποίο εκτελεί 13 διεργασίες μέχρι να φτάσει στο τέλος. Τα πιθανά μονοπάτια που δημιουργούνται είναι 4.

Στο πρώτο, ξεκινώντας από την αρχική διεργασία και φτάνοντας στην τελευταία διακρίνουμε ότι περνάει από τις παρακάτω διεργασίες:

- register submission date request
- OLO messaging active
- phase application received

Στο δεύτερο:

- register submission date request
- OLO messaging active
- send confirmation receipt
- forward to the competent authority
- send confirmation receipt
- phase application received

Στο τρίτο:

- register submission date request
- OLO messaging active
- phase application received
- MER required
- activities regular procedure
- phased application
- create procedure confirmation
- publish
- create subcases completeness
- procedure change
- create publication document
- regular procedure without MER
- start WABO procedure
- registration date publication
- read field phased application
- enter send date acknowledgement

Στο τέταρτο:

- register submission date request
- OLO messaging active
- send confirmation receipt
- forward to the competent authority
- send confirmation receipt
- phase application received
- MER required
- activities regular procedure
- phased application
- create procedure confirmation
- publish
- create subcases completeness
- procedure change
- create publication document
- regular procedure without MER
- start WABO procedure
- registration date publication
- read field phased application
- enter send date acknowledgement

Πρέπει να σημειωθεί ότι με τον χάρτη διεργασιών (process map) που βασίζεται στο DFG είναι πολύ δύσκολο να συμπεράνουμε την πραγματική εκτέλεση μιας διαδικασίας. Γι' αυτό το λόγο, είναι πιο συχνή η χρήση των process trees.

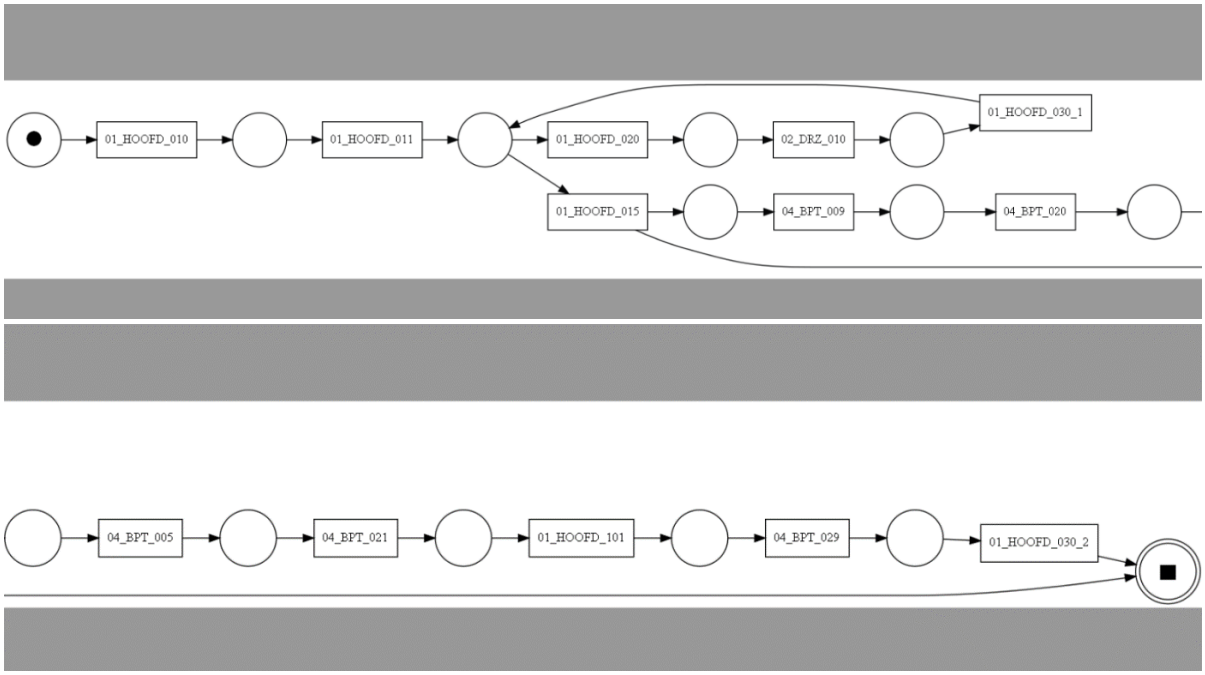
4.1.6 Alpha Miner

Στη συνέχεια, θα εφαρμόσουμε τους αλγόριθμους ανακάλυψης διαδικασιών, ξεκινώντας από τον Alpha miner.

Δημιουργώντας τη συνάρτηση η οποία θα δέχεται σαν είσοδο ένα event log και με τη χρήση της συνάρτησης **pm4py.discover_petri_net_alpha()** της pm4py, θα μπορέσουμε να ανακαλύψουμε το petri net με βάση τον αλγόριθμο alpha. Στη συνέχεια οπτικοποιούμε το αποτέλεσμα και το αποθηκεύουμε σε ένα αρχείο.

```
def alpha_miner(file):
    count = 1
    net, initial_marking, final_marking = pm4py.discover_petri_net_alpha(file)
    gviz = pn_visualizer.apply(net, initial_marking, final_marking)
    pn_visualizer.view(gviz)
    pn_visualizer.save(gviz, '../Datasets/BPI/output_data/miners/export_alpha_miner_petri_net_'+str(count)+".png")
```

Εικόνα 28: Συνάρτηση alpha_miner



Εικόνα 29: Εξαγόμενο Petri net με χρήση του αλγόριθμου alpha miner(export_alpha_miner_petri_net_1)

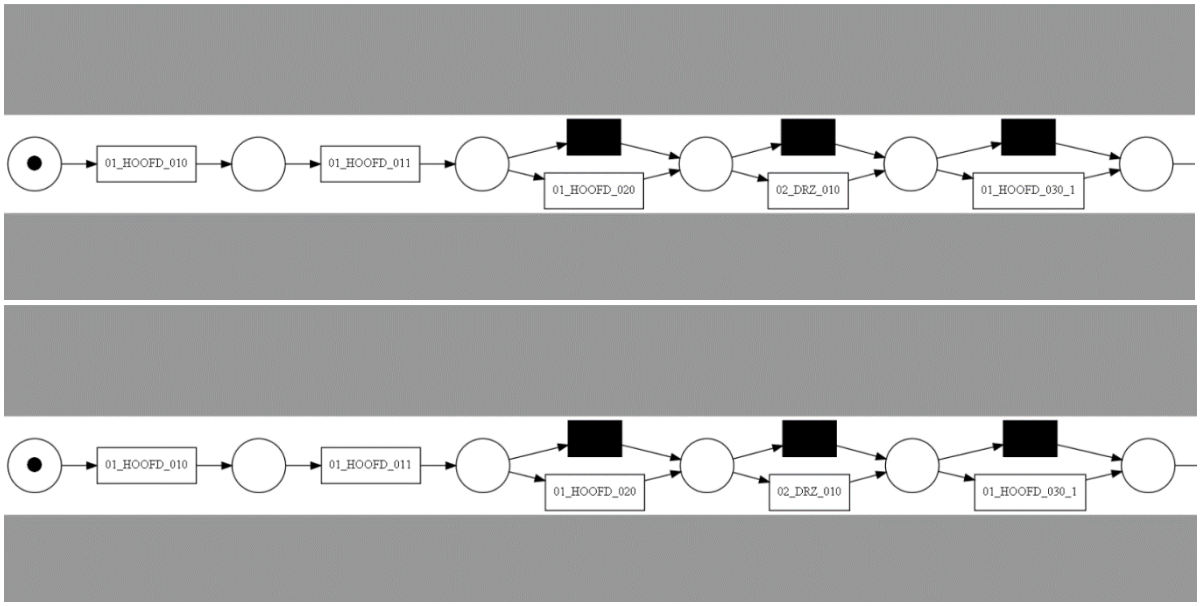
Στο πάνω μέρος βλέπουμε στα αριστερά τον κύκλο που δηλώνει το initial marking και ακολουθείται από τις διεργασίες, και στο κάτω μέρος βλέπουμε τον κύκλο που δηλώνει το final marking. Όπως αναφέραμε και σε προηγούμενη ενότητα, ο συγκεκριμένος αλγόριθμος έχει αρκετές ελλείψεις. Δε μπορεί να ανακαλύψει αόρατες δραστηριότητες με αποτέλεσμα να μας παρέχει ένα πολύ γενικό μοντέλο και έτσι δεν εγγυάται η ορθότητα του.

4.1.7 Inductive Miner

Παρομοίως, δημιουργούμε τη συνάρτηση inductive_miner και δίνουμε σαν είσοδο το αρχείο μας και με την συνάρτηση **pm4py.discover_petri_net_inductive()**, εξάγουμε το επιθυμητό petri net.

```
def inductive_miner(file):
    count = 1
    net, initial_marking, final_marking = pm4py.discover_petri_net_inductive(file)
    gviz = pn_visualizer.apply(net, initial_marking, final_marking)
    pn_visualizer.view(gviz)
    pn_visualizer.save(gviz, '../Datasets/BPI/output_data/miners/export_inductive_miner_petri_net_'+str(count)+".png")
```

Εικόνα 30: Συνάρτηση inductive_miner



Εικόνα 31: Εξαγόμενο Petri net με χρήση του αλγόριθμου inductive miner(export_inductive_miner_petri_net_1)

Παρατηρούμε ότι λαμβάνουμε ένα διαφορετικό διάγραμμα το οποίο περιέχει αόρατες μεταβάσεις οι οποίες απεικονίζονται με μαύρο ορθογώνιο. Ωστόσο έχουμε παρόμοια συμπεριφορά με το προηγούμενο μοντέλο, όσον αφορά τη σειρά εκτέλεσης των δραστηριοτήτων.

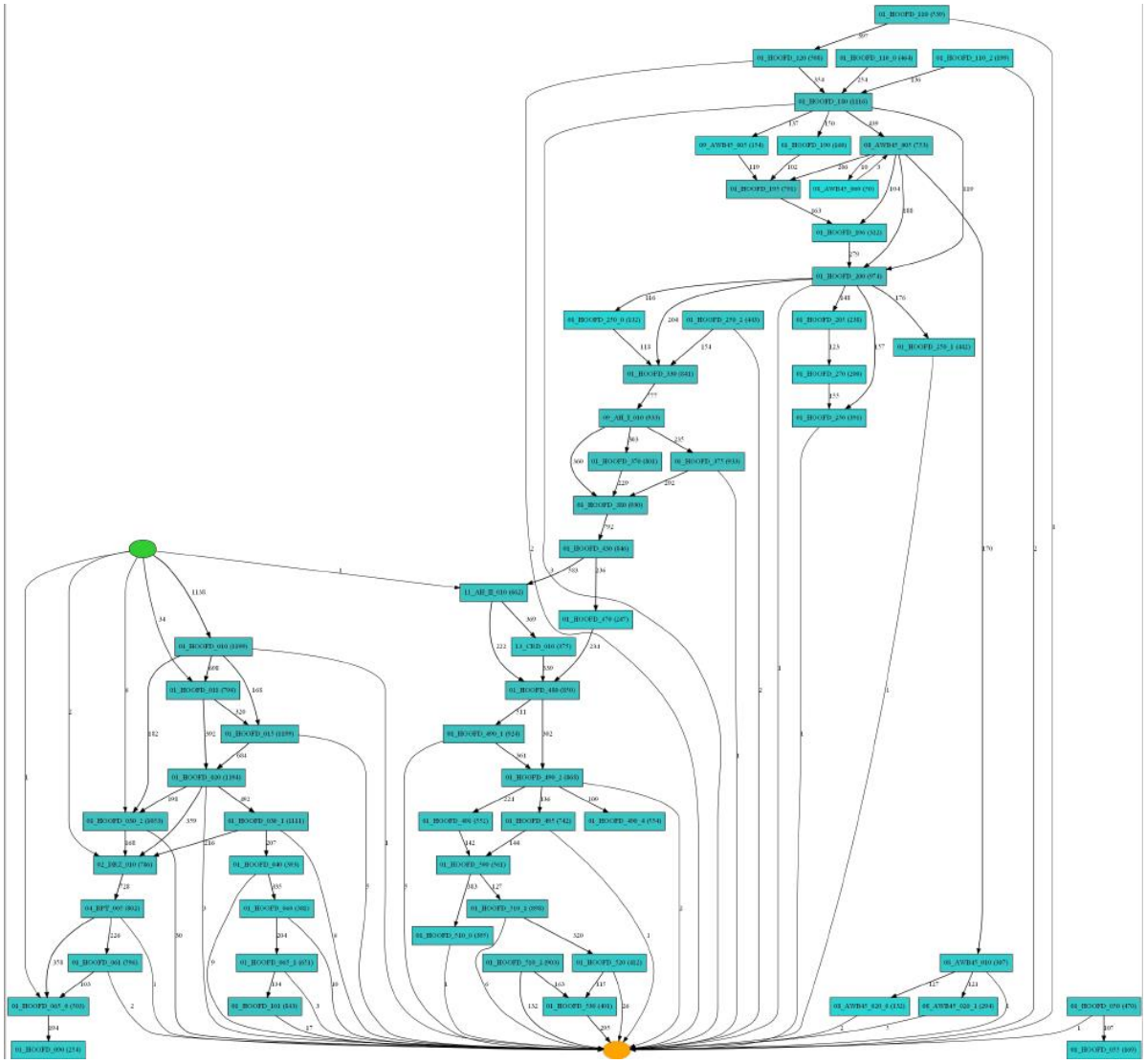
4.1.8 Heuristic Miner

Επόμενος αλγόριθμος που θα εφαρμόσουμε είναι ο Heuristic. Δημιουργούμε την αντίστοιχη συνάρτηση, χρησιμοποιούμε την **pm4py.discover_heuristics_net()** της pm4py, και παίρνουμε σαν αποτέλεσμα το heuristics net. Στη συνέχεια, εξάγουμε το petri net μέσω της συνάρτησης **pm4py.discover_petri_net_heuristics()**, το εμφανίζουμε στην οθόνη και έπειτα το αποθηκεύουμε σε μορφή .png.

```
def heuristic_miner(file):
    count = 1
    heu_net = pm4py.discover_heuristics_net(file, dependency_threshold=0.99)
    pm4py.view_heuristics_net(heu_net)
    # obtain a petri net based on the heuristics miner
    net, initial_marking, final_marking = pm4py.discover_petri_net_heuristics(log, dependency_threshold=0.99)
    gviz = pn_visualizer.apply(net, initial_marking)
    pn_visualizer.save(gviz, '../Datasets/BPI/output_data/miners/export_heuristic_miner_petri_net_'+str(count)+".png")
```

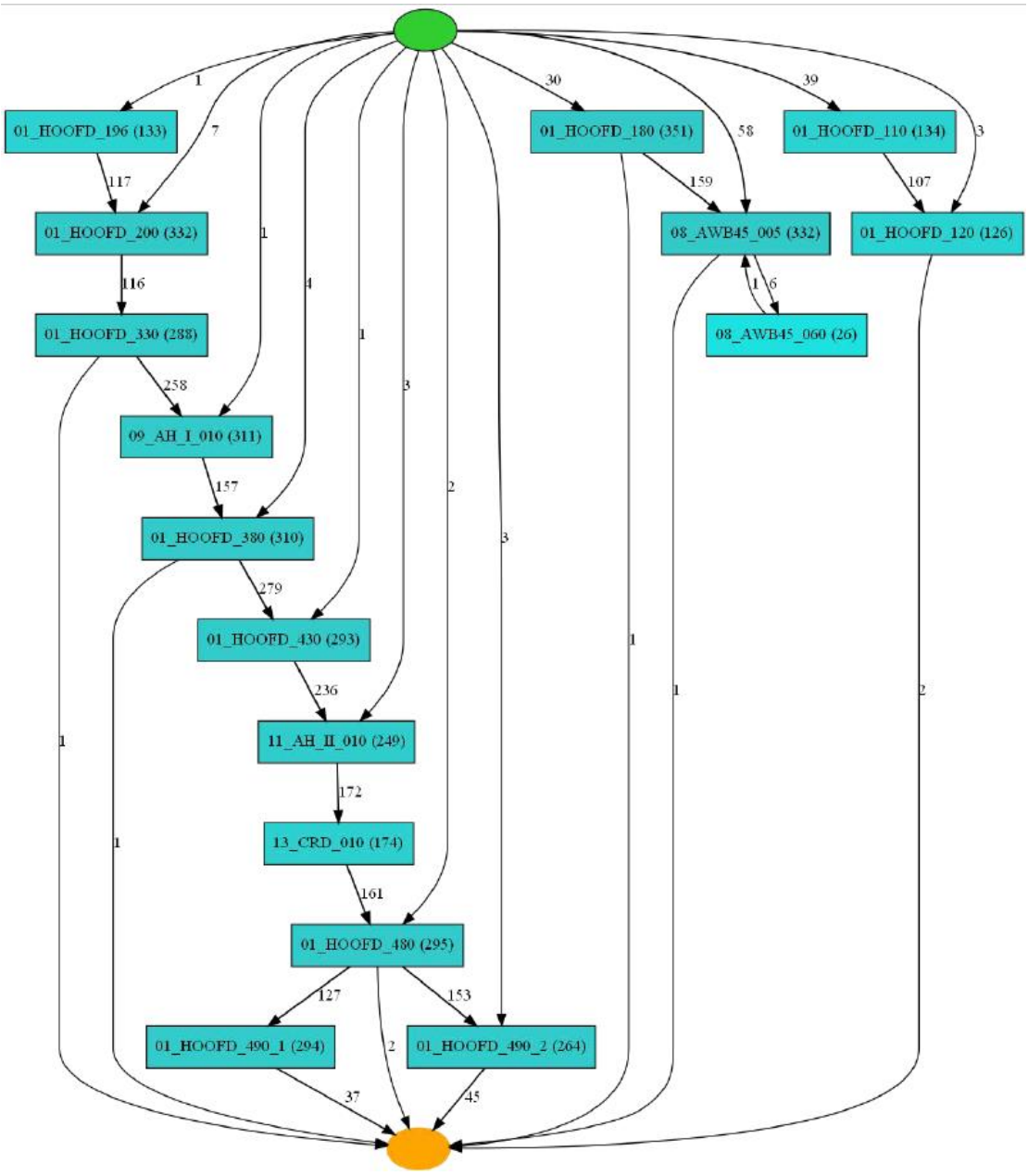
Εικόνα 32: Συνάρτηση heuristic_miner

Αρχικά, εφαρμόζουμε την συνάρτηση δίνοντας ως είσοδο ολόκληρο το event log, και βλέπουμε ότι ο αλγόριθμος λειτουργεί αρκετά καλά, εντοπίζοντας όλες τις διεργασίες, καθώς και αυτές που είναι κρυφές. Το πλεονέκτημα του heuristic miner είναι ότι υπάρχουν αρκετές παράμετροι που μπορούν να χρησιμοποιηθούν για την εξάλειψη ασήμαντων συστάδων, μια από αυτές είναι το **dependency threshold**. Στη συνέχεια, την εφαρμόζουμε σε ένα φιλτραρισμένο log, με βάση το resource ώστε να πάρουμε μια πιο λεπτομερής ανάλυση σε μικρότερη κλίμακα.



Εικόνα 33: Εξαγόμενο Petri net με χρήση του αλγόριθμου heuristic miner(export_heuristic_miner_petri_net_log)

Στην παραπάνω εικόνα έχουμε το petri net που δημιουργήθηκε δίνοντας ως είσοδο ολόκληρο το event log. Αν και πρόκειται για ένα μεγάλο αρχείο καταγραφής, έχουμε ξεκάθαρη εικόνα τόσο για τις διεργασίες και τη σειρά που εκτελούνται, όσο και για την γενικότερη συμπεριφορά του μοντέλου. Αυτό είναι το κύριο πλεονέκτημα του heuristic miner από τους υπόλοιπους αλγόριθμους ανακάλυψης διαδικασιών, οι οποίοι αδυνατούν να επεξεργαστούν ένα τόσο μεγάλο αρχείο καταγραφής συμβάντων.



Εικόνα 34: Εξαγόμενο Petri net με χρήση του αλγόριθμου heuristic miner(export_heuristic_miner_petri_net_filtered_log)

Το συγκεκριμένο μοντέλο περιέχει επίσης κρυφές μεταβάσεις όπως και το προηγούμενο. Το πλεονέκτημα του είναι ότι χρησιμοποιώντας το dependency threshold και το cleaning threshold, προκειμένου να αφαιρεθούν οι πιο αδύναμες άκρες, εξάγουμε ένα petri net το οποίο απεικονίζει μια λεπτομερή συμπεριφορά του μοντέλου και παρατηρούμε ότι εντοπίζει όλα τα μονοπάτια με ακρίβεια και μας δίνει μια καλύτερη οπτικοποίηση σε σχέση με τα προηγούμενα μοντέλα.

Σε επόμενη ενότητα θα εξετάσουμε ποιος αλγόριθμος είναι ο βέλτιστος για το συγκεκριμένο αρχείο καταγραφής συμβάντων, μέσω του ελέγχου συμμόρφωσης και της αξιολόγησης.

4.2 Πειραματικά αποτελέσματα

4.2.1 Έλεγχος συμμόρφωσης (Conformance checking)

4.2.1.1 Token-based replay

Θα εξετάσουμε τρεις περιπτώσεις. Αρχικά, εισάγουμε το αρχείο καταγραφής συμβάντων. Στη συνέχεια, για την πρώτη περίπτωση φιλτράρουμε το αρχείο με βάση το timestamp και εφαρμόζουμε τον αλγόριθμο Alpha miner. Στην δεύτερη περίπτωση το φιλτράρουμε με βάση το resource και εφαρμόζουμε τον αλγόριθμο Inductive miner και στην τρίτη περίπτωση τον αλγόριθμο Heuristic miner ώστε να εξάγουμε το Petri net και στις τρεις περιπτώσεις. Στη συνέχεια εφαρμόζουμε τον έλεγχο token-based replay, του οποίου τα αποτελέσματα τα αποθηκεύουμε στις μεταβλητές **replayed_traces, replayed_traces_resource και replayed_traces_resource_h** αντίστοιχα, οι οποίες περιέχουν τις παρακάτω πληροφορίες:

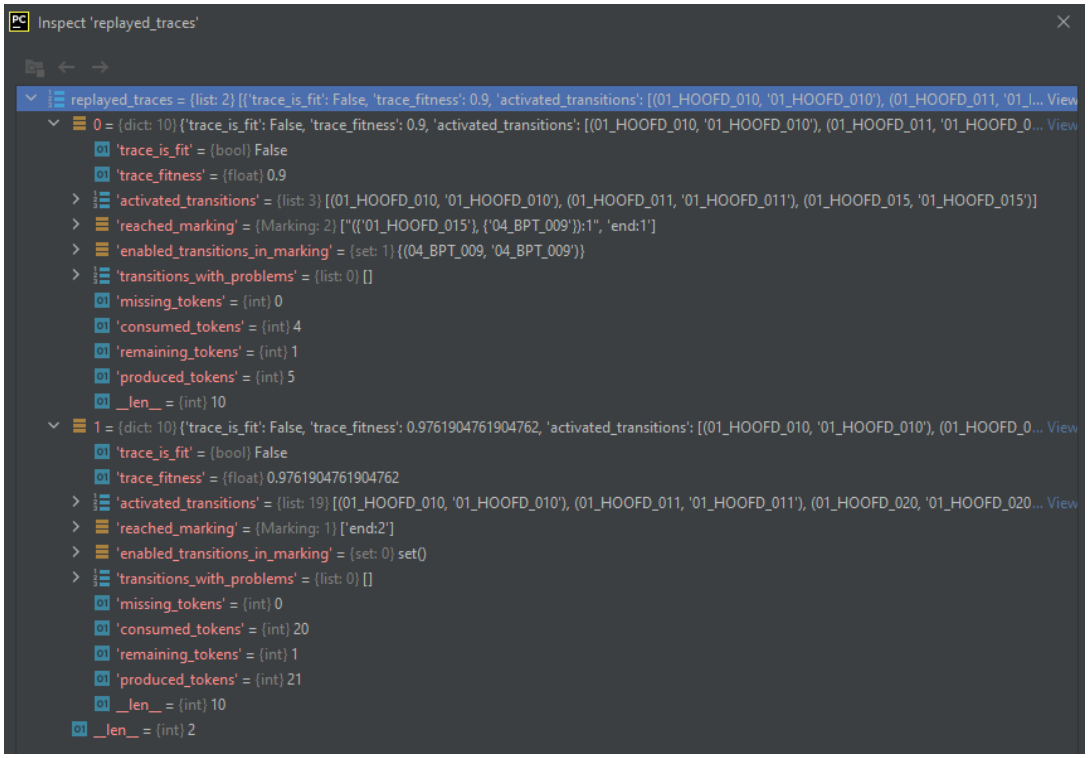
- **trace_is_fit**: Μια τιμή Boolean η οποία είναι True στην περίπτωση που το ίχνος ταιριάζει στο μοντέλο
- **trace_fitness**: Μια τιμή η οποία κυμαίνεται από 0-1 και αντιπροσωπεύει την καταλληλότητα του μοντέλου
- **activated_transitions**: Μια λίστα με τις μεταβάσεις που ενεργοποιούνται στο μοντέλο κατά τη διάρκεια του token-based replay
- **missing_tokens**: Τον αριθμό των missing tokens
- **consumed_tokens**: Τον αριθμό των consumed tokens
- **remaining_token**: Τον αριθμό των remaining tokens
- **produced_tokens**: Τον αριθμό των produced tokens

```
if __name__ == "__main__":

    # importing the dataset
    log = pm4py.read_xes('D:\Programs\PyCharm\Datasets\BPI\input_data\BPIC15_1.xes')

    filtered_log = timestamp_filter.filter_traces_contained(log, "2014-06-01 00:00:00", "2014-07-01 23:59:59")
    filtered_resource_log = pm4py.filter_event_attribute_values(log, "org:resource", ["2670601"], level="event", retain=True)

    net, initial_marking, final_marking = pm4py.discover_petri_net_alpha(filtered_log)
    net_r, initial_marking_r, final_marking_r = pm4py.discover_petri_net_inductive(filtered_resource_log)

    #we apply token-based replay
    replayed_traces = pm4py.conformance_diagnostics_token_based_replay(filtered_log, net, initial_marking, final_marking)
    replayed_traces_resource = pm4py.conformance_diagnostics_token_based_replay(filtered_resource_log, net_r, initial_marking_r, final_marking_r)
```

Εικόνα 35: Συνάρτηση εφαρμογής Token-based replay

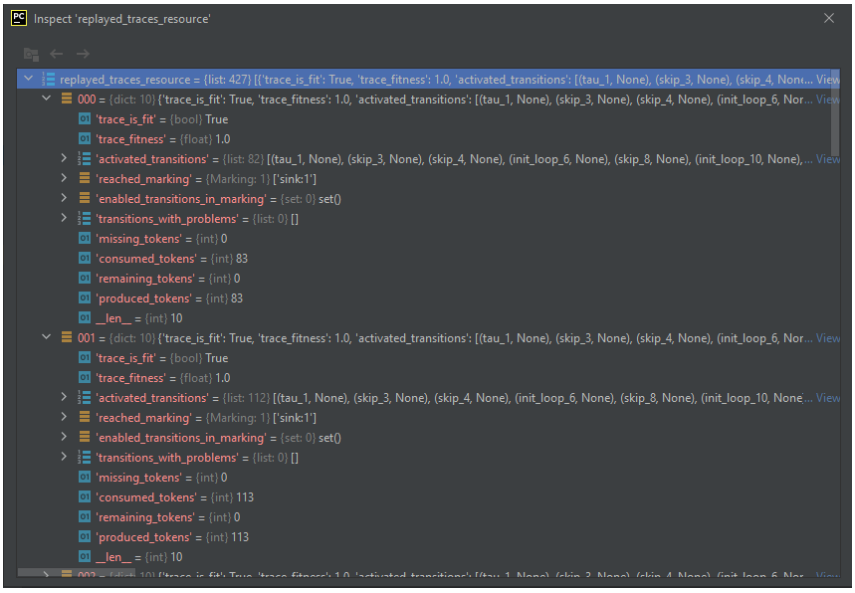
Στην πρώτη περίπτωση παρατηρούμε ότι η μεταβλητή trace_is_fit έχει τιμή false, αυτό σημαίνει ότι το ίχνος δεν ταιριάζει τέλεια με το μοντέλο. Ωστόσο, η τιμή trace_fitness είναι 0.9-0.97 και αυτό μας δείχνει ότι έχουμε μια αρκετά καλή αναγνώριση του ίχνους. Επίσης, μπορούμε να καταλάβουμε ότι το fitness είναι καλό, από τον αριθμό των missing, remaining tokens ο οποίος είναι 0 και 1 αντίστοιχα. Λαμβάνουμε πληροφορίες και για τα activated transitions τα οποία είναι 3 για το πρώτο ίχνος και 19 για το δεύτερο.

Επομένως, ο αλγόριθμος Alpha miner αποτυπώνει σχετικά καλά τη συμπεριφορά του μοντέλου αλλά δεν είναι ο βέλτιστος και αυτό οφείλεται στα μειονεκτήματα που αναφέραμε σε προηγούμενες ενότητες.



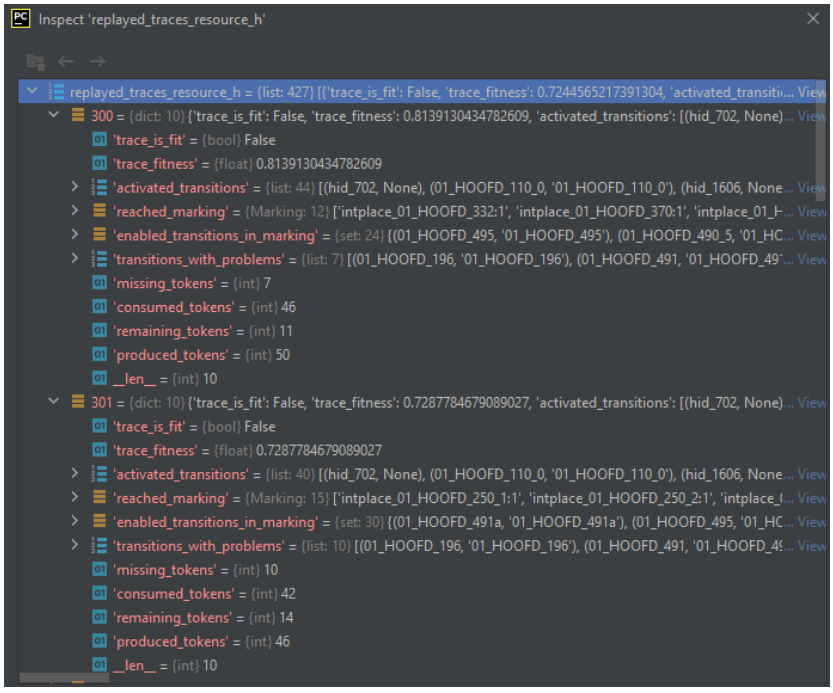
Εικόνα 36: Αποτελέσματα Token-based replay για alpha miner

Στην δεύτερη περίπτωση βλέπουμε ότι το trace_is_fit έχει τιμή true, το trace_fitness είναι 1.0 και οι μεταβλητές remaining και missing tokens είναι 0, οπότε συμπεραίνουμε ότι το ίχνος συμμορφώνεται άψογα με το μοντέλο και έχουμε το επιθυμητό αποτέλεσμα.



Εικόνα 37: Αποτελέσματα Token-based replay για inductive miner

Στην τελευταία περίπτωση, η μεταβλητή trace_is_fit έχει τιμή false, το trace_fitness είναι 0.72 , και οι μεταβλητές remaining και missing tokens είναι αρκετά υψηλές. Επομένως, ο συγκεκριμένος αλγόριθμος δε λειτουργεί καθόλου καλά για το αρχείο μας. Αυτό μπορεί να συμβαίνει διότι προκειμένου να επεξεργαστεί μεγαλύτερο όγκο δεδομένων, αφαιρεί ορισμένες άκρες, με αποτέλεσμα να γίνεται μια παραμορφοποίηση του μοντέλου και αυτό φαίνεται και από τα αποτελέσματα που μας έδωσε το token-based replay.



Εικόνα 38: Αποτελέσματα Token-based replay για heuristic miner

4.2.1.2 Alignments

Αρχικά, εισάγουμε το αρχείο καταγραφής. Στη συνέχεια εφαρμόζουμε Inductive miner, και υπολογίζουμε τα alignments. Η έξοδος, η οποία είναι μια λίστα, αναφέρει για κάθε ίχνος την αντίστοιχη στοίχιση μαζί με τα στατιστικά της. Με κάθε ίχνος, συσχετίζεται ένα λεξικό που περιέχει μεταξύ των άλλων τις ακόλουθες πληροφορίες:

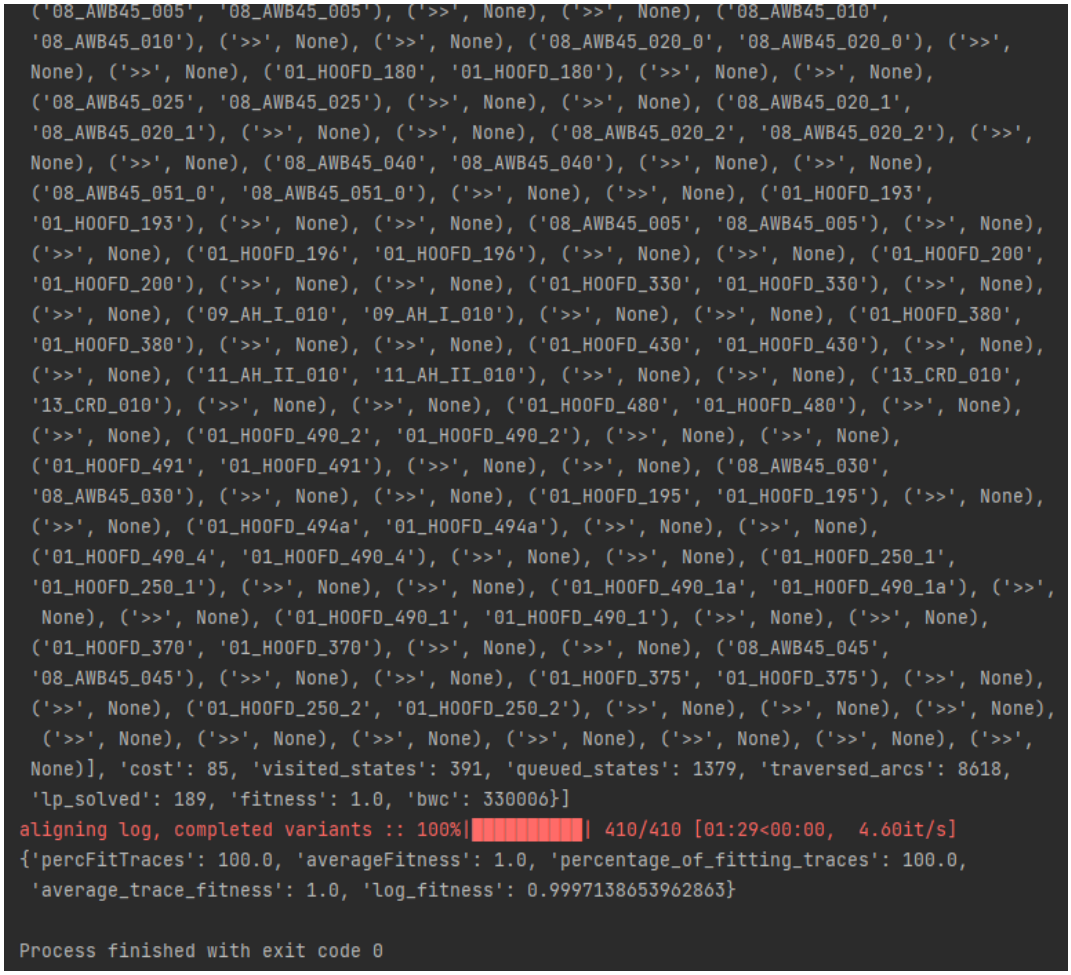
- **alignment**: Περιέχει τη στοίχιση (sync moves, moves on log, moves on model)
- **cost**: Περιέχει το κόστος της ευθυγράμμισης σύμφωνα με την παρεχόμενη συνάρτηση κόστους
- **fitness**: Ισούται με 1 εάν το ίχνος ταιριάζει απόλυτα

Στη συνέχεια, υπολογίζεται ένα μοντέλο διαδικασίας και υπολογίζονται τα alignments. Επιπλέον, υπολογίζεται η τιμή του fitness και εκτυπώνονται οι τιμές που προκύπτουν.

```
#alignments
aligned_traces = pm4py.conformance_diagnostics_alignments(filtered_resource_log, net_r, initial_marking_r, final_marking_r)
print(aligned_traces)

for trace in filtered_resource_log:
    for event in trace:
        event["customClassifier"] = event["concept:name"] + event["concept:name"]

parameters = {"pm4py:param:activity_key": "customClassifier"}
process_tree = inductive_miner.apply_tree(filtered_resource_log, parameters=parameters)
net_p, initial_marking_p, final_marking_p = process_tree_converter.apply(process_tree)
aligned_traces = alignments.apply_log(filtered_resource_log, net_p, initial_marking_p, final_marking_p, parameters=parameters)
log_fitness = replay_fitness.evaluate(aligned_traces, variant=replay_fitness.Variants.ALIGNMENT_BASED)
print(log_fitness)
```

Εικόνα 39: Κώδικας εφαρμογής alignment

Στην παρακάτω εικόνα έχουμε τα αποτελέσματα από το alignment. Παρατηρούμε ότι οι τιμές που αφορούν το fitness, όπως fitness, percFitTraces, averageFitness, percentage_of_fitting_traces, average_trace_fitness έχουν τιμή 1.0 ή 100.0. Επίσης το log_fitness είναι 0.99. Από αυτά τα αποτελέσματα συμπεραίνουμε ότι έχουμε ένα πολύ καλό alignment μεταξύ του ίχνους και του μοντέλου και ουσιαστικά επιβεβαιώνουμε και τις παρόμοιες τιμές που λάβαμε από την τεχνική Token-based replay για τον αλγόριθμο Inductive miner. Ακόμη, λαμβάνουμε πληροφορίες για το κόστος (cost) το οποίο είναι 85, τα visited_states που είναι 391 και τα traversed_arcs που είναι 8618.



Εικόνα 40: Αποτελέσματα alignment

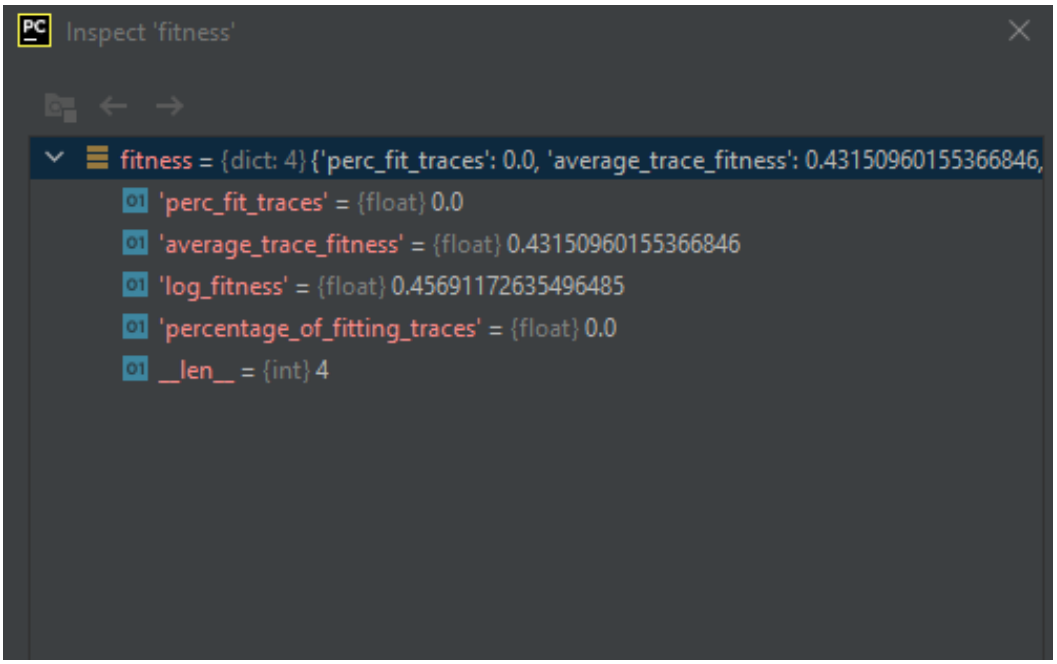
4.2.2 Αξιολόγηση (Evaluation)

4.2.2.1 Replay Fitness

Στην πρώτη περίπτωση, επιστρέφεται το ποσοστό των ιχνών που είναι πλήρως προσαρμοσμένα, μαζί με μια τιμή καταλληλότητας. Για να υπολογίσουμε την καταλληλότητα επανάληψης, μεταξύ ενός αρχείου καταγραφής συμβάντων και ενός μοντέλου Petri net, χρησιμοποιώντας τη μέθοδο token-based replay χρησιμοποιούμε τον παρακάτω κώδικα. Η τιμή που προκύπτει είναι ένας αριθμός μεταξύ 0 και 1.

Αρχικά, για τον αλγόριθμο Alpha miner.

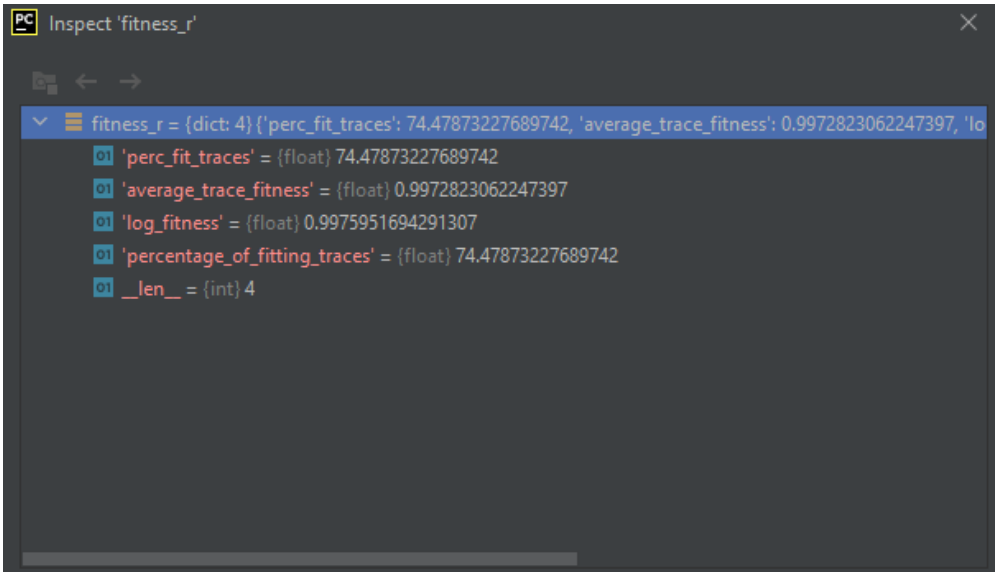
```
fitness = pm4py.fitness_token_based_replay(log, net, initial_marking,
final_marking)
```



Εικόνα 41: Replay fitness με τη μέθοδο token-based replay για Alpha miner

Επαναλαμβάνουμε την ίδια διαδικασία για τον αλγόριθμο Inductive miner.

```
fitness_r = pm4py.fitness_token_based_replay(log, net_r,
initial_marking_r, final_marking_r)
```



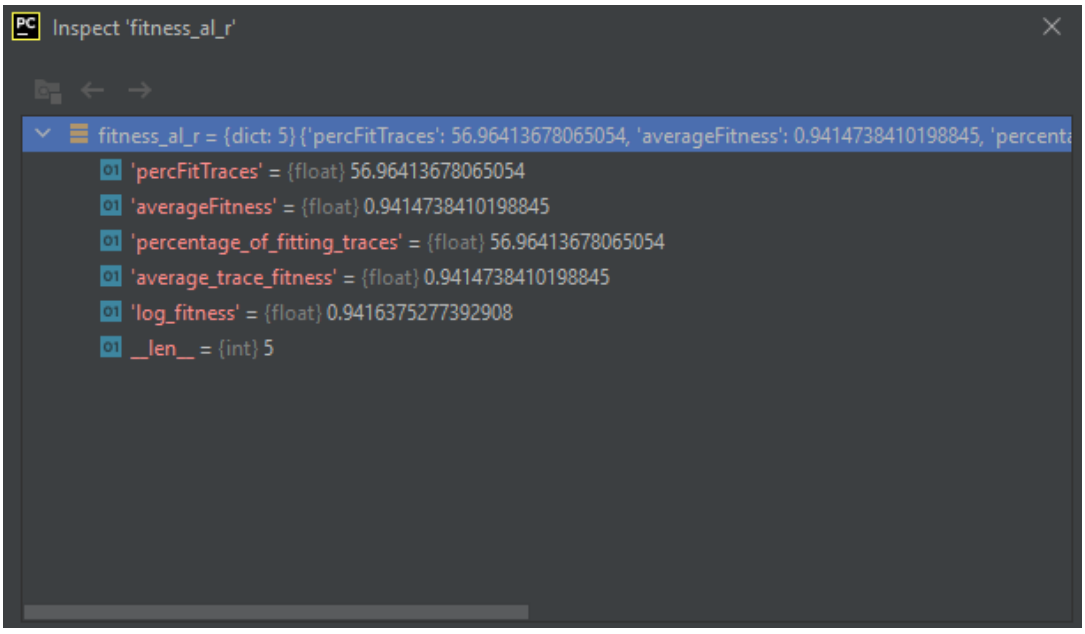
Εικόνα 42: Replay fitness με τη μέθοδο token-based replay για Inductive miner

Παρατηρούμε ότι λαμβάνουμε πολύ καλύτερα αποτελέσματα, η τιμή της καταλληλότητας ισούται με 0.99 και αυτό σημαίνει ότι η συμπεριφορά του αρχείου καταγραφής γίνεται αποδεκτή από το μοντέλο διαδικασίας.

Στην δεύτερη περίπτωση, επιστρέφεται το ποσοστό των ιχνών που είναι πλήρως προσαρμοσμένα, μαζί με μια τιμή καταλληλότητας που υπολογίζεται ως ο μέσος όρος των τιμών καταλληλότητας των μεμονωμένων ιχνών.

```
fitness_al = pm4py.fitness_alignments(log, net_r, initial_marking_r,
final_marking_r)
```

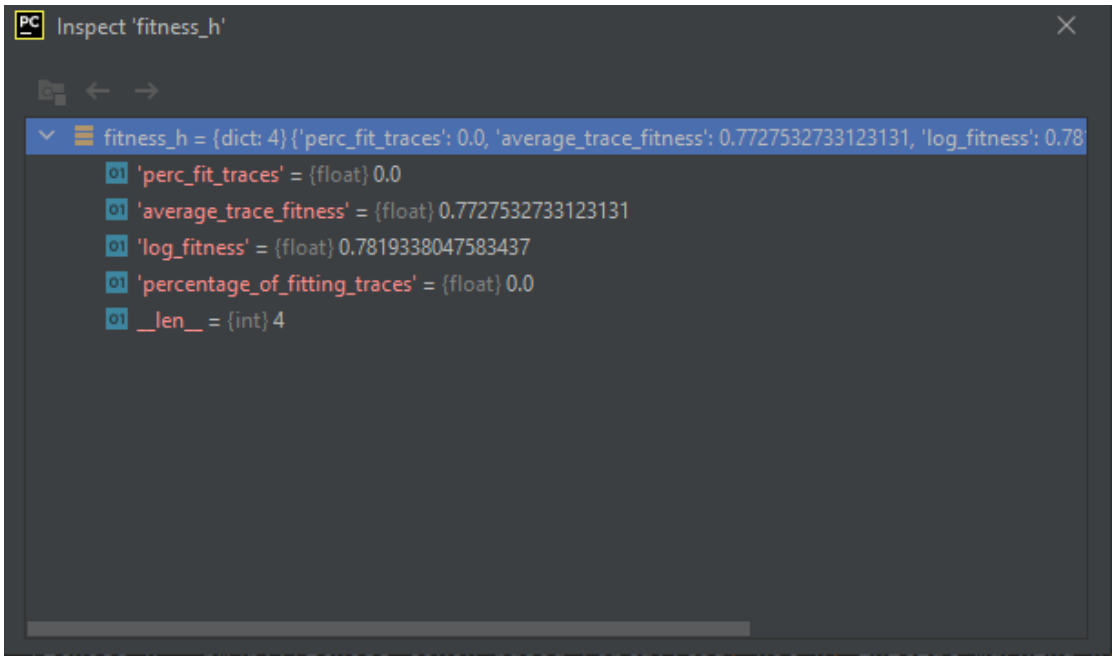
Και με αυτή τη μέθοδο λαμβάνουμε εξίσου καλά αποτελέσματα για τον συγκεκριμένο αλγόριθμο.



Εικόνα 43: Replay fitness με τη μέθοδο alignment για Inductive miner

Τέλος, για τον αλγόριθμο Heuristic miner.

```
fitness_h = pm4py.fitness_token_based_replay(log, net_h,
initial_marking_h, final_marking_h)
```



Εικόνα 44: Replay fitness με τη μέθοδο token-based replay για Heuristic miner

4.2.2.2 Ακρίβεια (Precision)

Και σε αυτή την περίπτωση υπάρχουν δύο μέθοδοι υπολογισμού της ακρίβειας:

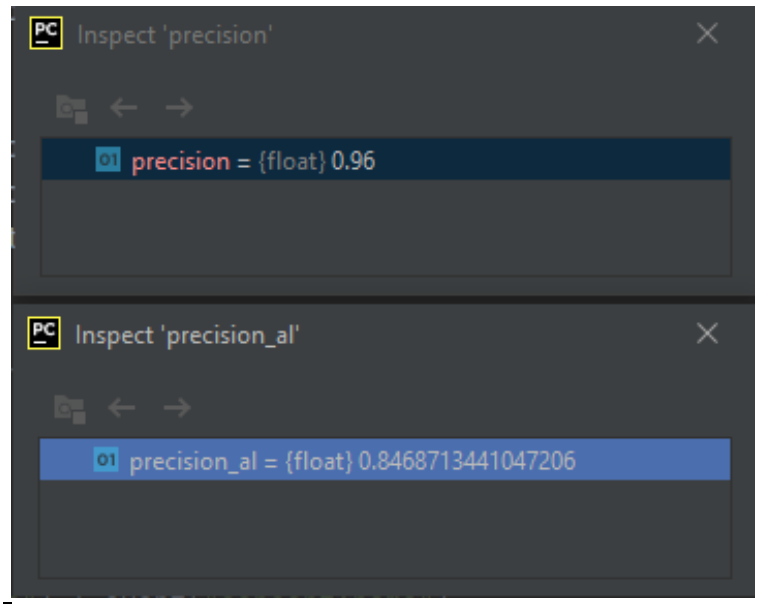
- **ETConformance (using token-based replay)**

```
precision = pm4py.precision_token_based_replay(filtered_log, net,
initial_marking, final_marking)
```

- **Align-ETConformance (using alignments):**

```
precision_al = pm4py.precision_alignments(filtered_resource_log,
net_h, initial_marking_h, final_marking_h)
```

Η κύρια διαφορά των δύο προσεγγίσεων είναι η μέθοδος επανάληψης. Η πρώτη είναι ταχύτερη αλλά βασίζεται σε αλγόριθμο Heuristic, επομένως το αποτέλεσμα της επανάληψης μπορεί να μην είναι ακριβές. Η δεύτερη προσέγγιση είναι ακριβή, αλλά μπορεί να είναι αργή εάν το state-space είναι μεγάλο.



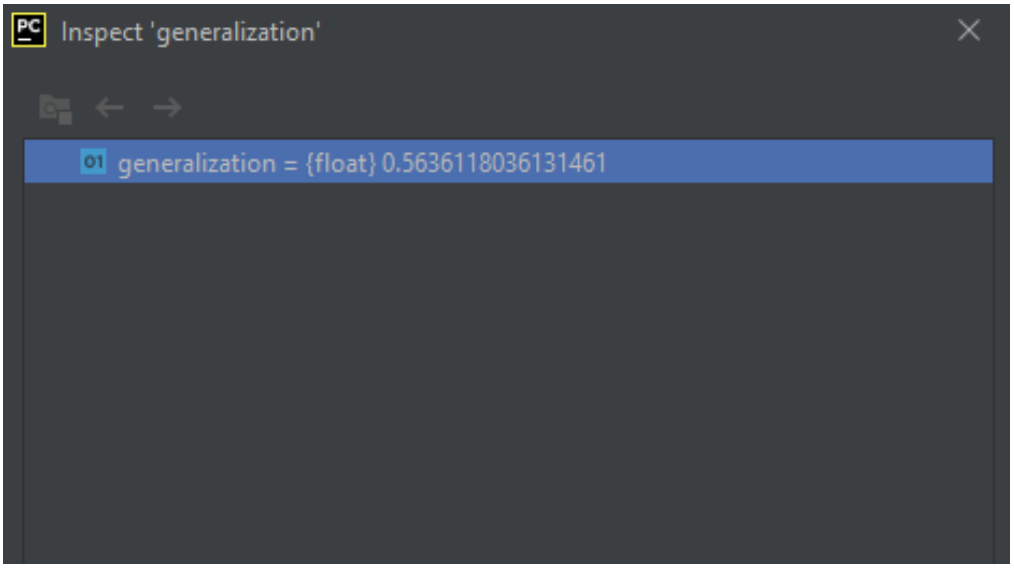
Εικόνα 45: Αποτελέσματα ακρίβειας (precision) για τις δύο μεθόδους

Και στις δυο περιπτώσεις η τιμή της ακρίβειας είναι υψηλή, οπότε το μοντέλο δεν επιτρέπει μεγαλύτερη συμπεριφορά από αυτή που καταγράφεται στο αρχείο.

4.2.2.3 Γενίκευση (Generalization)

Μια προσέγγιση για τη μέτρηση της γενίκευσης είναι η χρήση του alignment, ώστε να δούμε πόσο συχνά χρησιμοποιούνται τμήματα του μοντέλου. Εάν όλα τα μέρη του μοντέλου χρησιμοποιούνται συχνά, πιθανότατα αποτυπώνεται όλη η μελλοντική συμπεριφορά και επομένως η γενίκευση είναι υψηλή. Εάν υπάρχουν μέρη του μοντέλου που χρησιμοποιούνται σπάνια, είναι πιο πιθανό να υπάρχει συμπεριφορά που δεν φαίνεται ακόμα. Αυτό κάνει τη γενίκευση χαμηλή.

```
generalization =
generalization_evaluator.apply(filtered_resource_log, net_r,
initial_marking_r, final_marking_r)
```

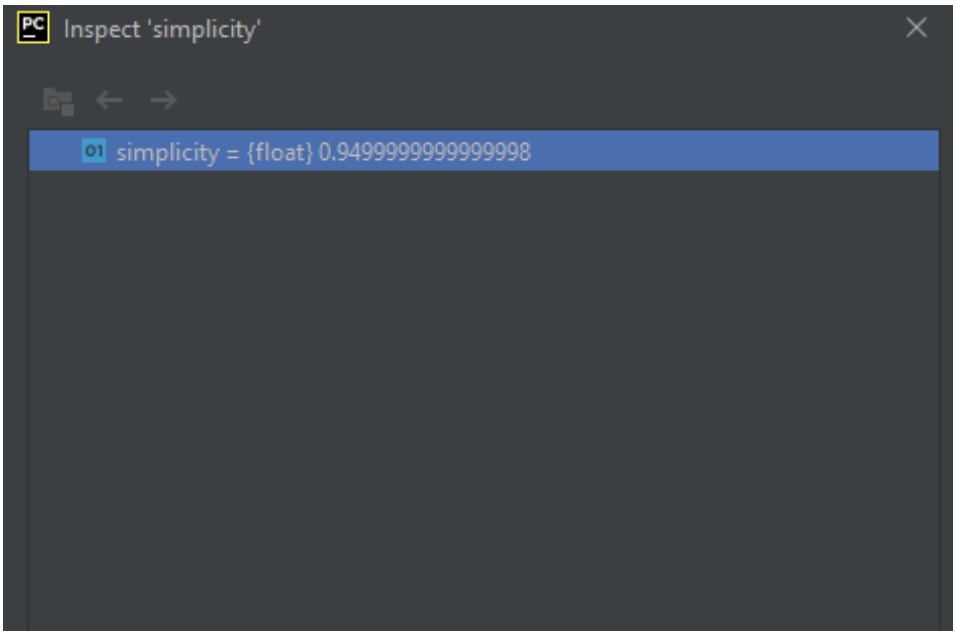


Εικόνα 46: Αποτελέσματα γενίκευσης (generalization)

4.2.2.4 Απλότητα (Simplicity)

Υπάρχουν πολλά μέτρα για να μετρηθεί η απλότητα ενός μοντέλου (π.χ. η διάμετρος του μοντέλου), ωστόσο έρευνες έχουν δείξει ότι το μέγεθος είναι ο κύριος δείκτης. Το μέγεθος μπορεί να αναφέρεται στον αριθμό των κόμβων στο μοντέλο και η διάμετρος στο μήκος της συντομότερης διαδρομής από έναν κόμβο έναρξης σε έναν κόμβο τερματισμού.

```
simplicity = simplicity_evaluator.apply(net)
```



Εικόνα 47: Αποτελέσματα απλότητας (simplicity)

Έχουμε πολύ υψηλή τιμή απλότητας, οπότε το μοντέλο είναι πολύ εύκολο να κατανοηθεί από έναν άνθρωπο.

4.2.3 Directly-Follows Graphs

4.2.3.1 Φιλτράρισμα με βάση τις δραστηριότητες/μονοπάτια (Filtering activities/paths)

Αρχικά, θα εφαρμόσουμε φιλτράρισμα στο ποσοστό των δραστηριοτήτων. Διατηρούνται οι πιο συχνές δραστηριότητες σύμφωνα με το ποσοστό, μαζί με τις δραστηριότητες που κρατούν το γράφημα συνδεδεμένο. Εάν καθοριστεί ποσοστό 0%, τότε ανακτάται η πιο συχνή δραστηριότητα. Στον κώδικα καθορίζουμε ποσοστό 0.2, ώστε να διατηρήσουμε το 20% των δραστηριοτήτων. Το φίλτρο εφαρμόζεται ταυτόχρονα στο DFG, στις δραστηριότητες έναρξης(sa), στις δραστηριότητες λήξης(ea) και στο λεξικό που περιέχει τις εμφανίσεις των δραστηριοτήτων. Με αυτό τον τρόπο διατηρείται η συνέπεια.

```
# importing the dataset
log =
pm4py.read_xes('D:\Programs\PyCharm\Datasets\BPI\input_data\BPIC15_1.
xes')

dfg, sa, ea = pm4py.discover_directly_follows_graph(log)
activities_count = pm4py.get_event_attribute_values(log,
"concept:name")

# filtering on activities percentage
dfg_act, sa_act, ea_act, activities_count =
dfg_filtering.filter_dfg_on_activities_percentage(dfg, sa, ea,
activities_count, 0.2)
```

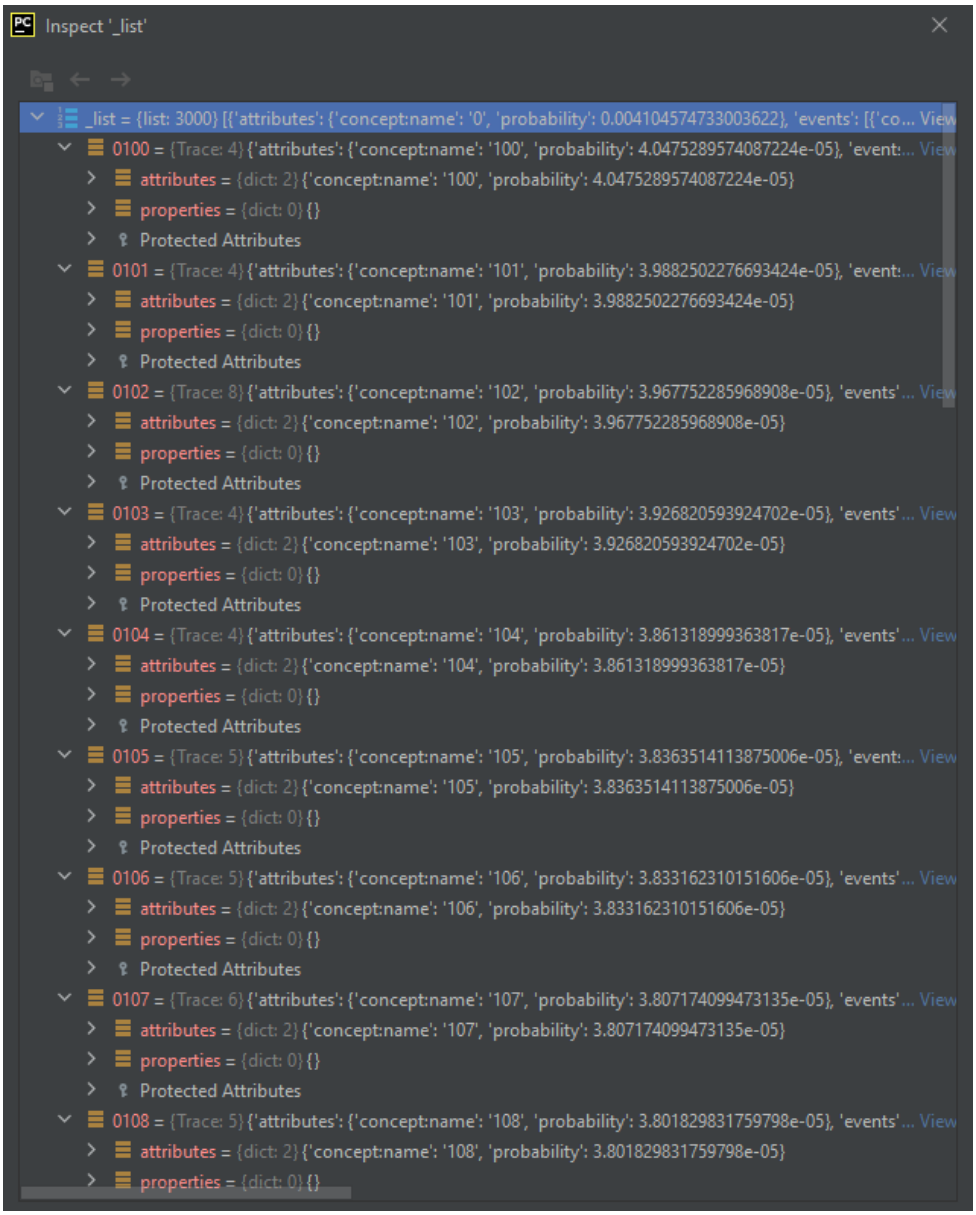
Στη συνέχεια, εφαρμόζουμε φιλτράρισμα στο ποσοστό των διαδρομών. Σε αυτή την περίπτωση, διατηρούνται οι πιο συχνές διαδρομές ανάλογα με το ποσοστό, μαζί με τις διαδρομές που είναι απαραίτητες για το γράφημα. Καθορίζουμε το ποσοστό σε 0.2, ώστε να διατηρήσουμε το 20% των μονοπατιών.

```
# filtering on paths percentage
dfg_p, sa_p, ea_p, activities_count =
dfg_filtering.filter_dfg_on_paths_percentage(dfg, sa, ea,
activities_count, 0.2)
```

4.2.3.2 Playout of a Directly-Follows Graph

Μια λειτουργία αναπαραγωγής(playout) σε ένα γράφημα Directly-Follows είναι χρήσιμη για την ανάκτηση των ιχνών που επιτρέπονται από το γράφημα. Σε αυτή την περίπτωση, ένα ίχνος είναι ένα σύνολο δραστηριοτήτων που επισκέπτονται στο γράφημα από τον κόμβο έναρξης(start node) έως τον κόμβο τέλους(end node). Μπορούμε να αντιστοιχίσουμε μια πιθανότητα σε κάθε ίχνος (υποθέτοντας ότι το γράφημα αντιπροσωπεύει μια αλυσίδα Markov). Συγκεκριμένα, μας ενδιαφέρει να πάρουμε τα πιο πιθανά ίχνη.

Στην παρακάτω εικόνα βλέπουμε για κάθε ίχνος την πιθανότητα(probability) εμφάνισης.



Εικόνα 48: Αποτελέσματα εφαρμογής playout of a DFG

4.2.3.3 Alignments on a Directly-Follows Graph

Μια δημοφιλής τεχνική ελέγχου συμμόρφωσης(conformance checking) είναι αυτή των ευθυγραμμίσεων(alignments). Οι ευθυγραμμίσεις εκτελούνται συνήθως σε Petri nets, ωστόσο αυτό μπορεί να πάρει χρόνο, καθώς τα δίκτυα αυτά ενδέχεται να είναι τεράστια σε μέγεθος. Μπορούμε να πραγματοποιήσουμε ευθυγράμμιση και σε γραφήματα directly-follows. Εφόσον, ο χώρος καταστάσεων σε τέτοια γραφήματα είναι μικρός, το αποτέλεσμα είναι ένας αρκετά αποτελεσματικός υπολογισμός των ευθυγραμμίσεων. Αυτό επιτρέπει τη λήψη γρήγορων διαγνωστικών για τις δραστηριότητες και τα μονοπάτια που εκτελούνται με λάθος τρόπο.

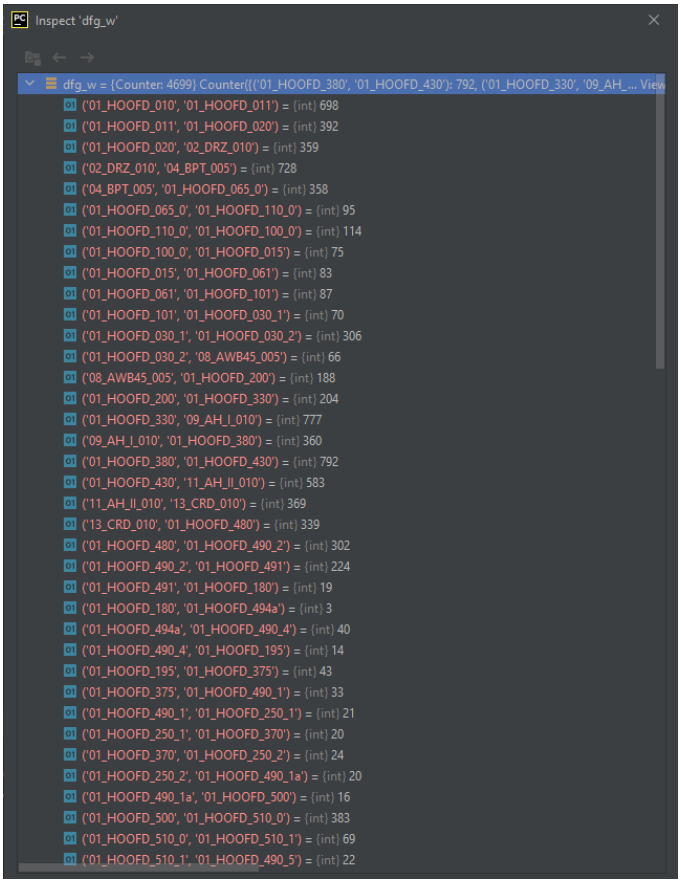
Εικόνα 49: Αποτελέσματα εφαρμογής alignments on a DFG

Η έξοδος των ευθυγραμμίσεων είναι ισοδύναμη με αυτή που λαμβάνεται έναντι των Petri net. Η έξοδος είναι μια λίστα που περιέχει για κάθε ίχνος το αποτέλεσμα της ευθυγράμμισης. Κάθε ευθυγράμμιση αποτελείται από ορισμένες κινήσεις από την αρχή μέχρι το τέλος τόσο του ίχνους όσο και του DFG. Μπορούμε να έχουμε κινήσεις συγχρονισμού(sync moves), κινήσεις στο αρχείο καταγραφής(moves on log) στην περίπτωση που μια κίνηση κατά την εκτέλεση της διεργασίας δεν μιμείται το DFG και κινήσεις στο μοντέλο(moves on model) αν μια κίνηση που δεν υποστηρίζεται από την εκτέλεση της διαδικασίας είναι αναγκαία στο μοντέλο.

4.2.3.4 Convert Directly-Follows Graph to a Workflow Net

Το γράφημα directly-follows είναι μια αναπαράσταση μιας διαδικασίας που παρέχεται από πολλά εμπορικά εργαλεία. Μια ιδέα που προήλθε από τον Sander Leemans είναι η μετατροπή του DFG σε ένα δίχτυ ροής εργασίας(workflow net), το οποίο μιμείται τέλεια το DFG. Αυτή η διαδικασία ονομάζεται εξόρυξη του DFG(mining). Με τον παρακάτω κώδικα εισάγουμε το αρχείο καταγραφής, υπολογίζουμε το DFG, το μετατρέπουμε σε workflow net και εκτελούμε ευθυγραμμίσεις.

```
dfg_w = dfg_discovery.apply(log)
net_w, im_w, fm_w = dfg_mining.apply(dfg)
```



Εικόνα 50: Αποτελέσματα εφαρμογής convert DFG to Workflow Net(dfg_w)

4.2.4 Converting a Petri net to a Process Tree

Μέσω της pm4py, είναι δυνατό να εξάγουμε ένα δέντρο διεργασίας μέσω ενός Petri net το οποίο έχει δομή block. Έχουμε ήδη το Petri net από προηγούμενη ενότητα:

```
net_r, initial_marking_r, final_marking_r =
pm4py.discover_petri_net_inductive(filtered_resource_log)
```

Στη συνέχεια, μέσω του παρακάτω κώδικα το μετατρέπουμε σε process tree και το εκτυπώνουμε:

```
tree = pm4py.convert_to_process_tree(net_r, initial_marking_r,
final_marking_r)
print(tree)
```

```
parsing log, completed traces :: 100%|██████████| 1199/1199 [00:04<00:00, 250.32it/s]
X( ->( '02_DRZ_020_2', '02_DRZ_010_2', '02_DRZ_030' ), ->( X( *tau*, '01_HOOFD_010' ), X( *tau*, '01_HOOFD_011' ), X( *tau*, *(
 ->( X( *tau*, '06_VD_020_2' ), X( *tau*, *( ->( X( *tau*, '06_VD_070' ), X( *tau*, *( *( X( '14_VRIJ_010', '01_HOOFD_020',
 '03_GBH_005', '01_HOOFD_099', '01_HOOFD_540', '01_HOOFD_130', '01_HOOFD_040', '02_DRZ_010', *( *tau*, X( '01_HOOFD_200',
 '09_AWB45_005', '11_AH_II_030', '08_AWB45_090_0', '01_HOOFD_120', '04_BPT_030', '08_AWB45_005', '06_VD_060_4',
 '01_HOOFD_490_2', '01_HOOFD_516', '01_HOOFD_809', '06_VD_030_1a', '07_OPS_055', '08_AWB45_090_2a', '08_AWB45_170',
 '11_AH_II_130', '11_AH_II_040_d', '01_HOOFD_250', '06_VD_020', '01_HOOFD_030_1', '08_AWB45_110', '11_AH_II_070_2',
 '01_HOOFD_110_1', '01_HOOFD_191', '11_AH_II_070_1', '11_AH_II_070_3a', '01_HOOFD_100', '11_AH_II_070_3', '01_HOOFD_470',
 '07_OPS_050', '08_AWB45_WAW_010', '01_HOOFD_811', '12_AP_030', '01_HOOFD_410_2', '01_HOOFD_440_2', '04_BPT_020', '12_AP_020_2',
  '01_HOOFD_100_0', '01_HOOFD_061', '06_VD_030_3a', '01_HOOFD_380', '04_BPT_005', '01_HOOFD_270', '12_AP_040', '01_HOOFD_197',
 '01_HOOFD_510_4', '08_AWB45_070_3a', '06_VD_030', '01_HOOFD_101', '08_AWB45_090_3', '01_HOOFD_532', '08_AWB45_020_2',
 '01_HOOFD_492_1', '11_AH_II_050', '01_HOOFD_510_3', '08_AWB45_020_1', '06_VD_030_5', '06_VD_030_3', '10_UOV_041b',
 '01_HOOFD_090', '01_HOOFD_110_0', '06_VD_030_5a', '01_HOOFD_250_2', '01_HOOFD_110_2', '11_AH_II_070', '08_AWB45_070_4',
 '07_OPS_070', '01_HOOFD_440_1', '01_HOOFD_420', '04_BPT_050', '08_AWB45_010', '11_AH_II_020_1', '11_AH_II_100', '13_CRD_030_0',
  '08_AWB45_080', '07_OPS_075', '11_AH_II_010', '01_HOOFD_350_2', '09_AH_I_020', '11_AH_II_037_3', '08_AWB45_045', '07_OPS_020',
  '05_EIND_010', '06_VD_060', '01_HOOFD_260', '01_HOOFD_495', '07_OPS_010', '01_HOOFD_809b', '12_AP_010', '01_HOOFD_055',
 '11_AH_II_040a', '01_HOOFD_410_0', '01_HOOFD_490_3a', '03_VD_060_0', '08_AWB45_070_1', '01_HOOFD_500', '06_VD_030_4',
 '06_VD_100', '01_HOOFD_491', '01_HOOFD_210_1', '11_AH_II_036', '08_AWB45_025', '11_AH_II_080', '06_VD_140', '04_BPT_060',
 '08_AWB45_020_0', '11_AH_II_080_2', '01_HOOFD_280_1', '01_HOOFD_445', '01_HOOFD_190', '01_HOOFD_190_2', '01_HOOFD_350_0',
 '01_HOOFD_459', '01_HOOFD_050', '01_HOOFD_210_0', '01_HOOFD_491a', '01_HOOFD_065_1', '01_HOOFD_510_1', '09_AH_I_010',
 '11_AH_II_140', '01_HOOFD_195_2', '01_HOOFD_430', '06_VD_010', '11_AH_II_040', '01_HOOFD_515', '01_HOOFD_492_0',
 '08_AWB45_070_1a', '01_HOOFD_250_1', '01_HOOFD_015', '01_HOOFD_350_1', '01_HOOFD_490_1', '06_VD_030_2', '01_HOOFD_446',
 '13_CRD_010', '06_VD_035', '12_AP_020_1', '01_HOOFD_250_0', '99_NOCODE_04', '01_HOOFD_490_5', '08_AWB45_060', '11_AH_II_020',
 '01_HOOFD_490_3', '01_HOOFD_510_0', '01_HOOFD_492_2', '01_HOOFD_195', '11_AH_II_055', '01_HOOFD_195_1', '13_CRD_020',
 '01_HOOFD_030_2', '11_AH_II_020_2', '01_HOOFD_330', '06_VD_050', '01_HOOFD_490_5a', '11_AH_II_080_1', '01_HOOFD_510_2a',
 '08_AWB45_030', '11_AH_II_037_1', '01_HOOFD_332', '08_AWB45_175', '11_AH_II_035', '01_HOOFD_493', '06_VD_040', '07_OPS_051',
 '15_NGV_010', '11_AH_II_037_2', '08_AWB45_070_5', '01_HOOFD_400', '01_HOOFD_440_1a', '04_BPT_010', '08_AWB45_090_2',
 '01_HOOFD_060', '01_HOOFD_192', '01_HOOFD_110', '06_VD_030_1', '01_HOOFD_375', '06_VD_060_3', '01_HOOFD_280_2',
 '01_HOOFD_190_1', '11_AH_II_110', '06_VD_060_2a', '01_HOOFD_480', '09_AH_I_040', '08_AWB45_070_3', '01_HOOFD_193',
 '01_HOOFD_516a', '08_AWB45_070_2', '08_AWB45_090_1', '06_VD_020_1', '01_HOOFD_210_2', '01_HOOFD_410_1', '01_HOOFD_519',
 '01_HOOFD_494a', '08_AWB45_070_5a', '01_HOOFD_181', '08_AWB45_040', '07_OPS_080_0', '08_AWB45_050', '12_AP_011',
 '01_HOOFD_510_2', '11_AH_II_145', '01_HOOFD_370', '08_AWB45_051_0', '01_HOOFD_180', '01_HOOFD_490_4', '10_UOV_060',
 '01_HOOFD_490_1a', '01_HOOFD_205', '08_AWB45_075', '01_HOOFD_065_0', '07_OPS_071', '08_AWB45_051_1', '11_AH_II_040c',
 '08_AWB45_052', '06_VD_060_2', '01_HOOFD_065_2', '11_AH_II_040b', '07_OPS_060_0', '06_VD_060_1', '01_HOOFD_196' ) ) ), *tau* ),
  *tau* ) ) ), *tau* ) ), X( *tau*, +( X( *tau*, '01_HOOFD_520' ), X( *tau*, '01_HOOFD_530' ) ), ->( X( *tau*, '01_HOOFD_446_0'
 ), X( *tau*, '01_HOOFD_446_1' ), X( *tau*, '01_HOOFD_446_2' ), X( *tau*, '01_HOOFD_447' ), X( *tau*, '01_HOOFD_450' ), X(
 *tau*, '01_HOOFD_455' ) ) ) ), *tau* ) ), X( *tau*, '08_AWB45_090_4', '04_BPT_040', '08_AWB45_150', ->( '01_BB_540',
 '01_BB_770', '01_HOOFD_790', '01_HOOFD_800' ) ) ) )
```

Εικόνα 51: Εξαγόμενο Process Tree

4.2.5 Προσομοίωση (Simulation)

Η pm4py προσφέρει αλγόριθμους προσομοίωσης, οι οποίοι ξεκινώντας από ένα μοντέλο, μπορούν να παράγουν μια έξοδο που ακολουθεί το μοντέλο και τους διαφορετικούς κανόνες που έχουν καθοριστεί από τον χρήστη.

4.2.5.1 Playout of a Petri Net

Εφόσον έχουμε το Petri net, μπορούμε να εφαρμόσουμε playout. Με τον παρακάτω κώδικα, θα πάρουμε δύο περιπτώσεις. Στην πρώτη περίπτωση θέλουμε να λάβουμε ένα αρχείο καταγραφής το οποίο θα περιέχει 50 ίχνη, ενώ στη δεύτερη λαμβάνουμε το αρχείο καταγραφής που περιέχει όλες τις εκτελέσεις μήκους μικρότερο ή ίσο με το 7.

```
# Playout of a Petri Net
# Log containing 50 traces
simulated_log_1 = simulator.apply(net_r, initial_marking_r,
variant=simulator.Variants.BASIC_PLAYOUT,
parameters={simulator.Variants.BASIC_PLAYOUT.value.Parameters.NO_TRAC
ES: 50})
print("Basic Playout: ", simulated_log_1)

# Log containing executions of length <= 7
simulated_log_2 = simulator.apply(net_r, initial_marking_r,
variant=simulator.Variants.EXTENSIVE,
parameters={simulator.Variants.EXTENSIVE.value.Parameters.MAX_TRACE_L
ENGTH: 7})
print("Extensive Playout: ", simulated_log_2)
```

```
D:\Programs\PyCharm\CvLab\venv\Scripts\python.exe D:/Programs/PyCharm/ProcessMining/bpi15_part4.py
parsing log, completed traces :: 100%|██████████| 1199/1199 [00:06<00:00, 181.49it/s]
Basic Playout:  [{'attributes': {'concept:name': '0'}, 'events': [{'concept:name': '01_HOOFD_010',
 'time:timestamp': datetime.datetime(1970, 4, 26, 20, 46, 40)}, '..', {'concept:name':
 '08_AWB45_090_4', 'time:timestamp': datetime.datetime(1970, 4, 26, 20, 46, 42)}]}, '....',
 {'attributes': {'concept:name': '49'}, 'events': [{'concept:name': '08_AWB45_090_4',
 'time:timestamp': datetime.datetime(1970, 4, 26, 20, 50, 46)}]}]
```

Εικόνα 52: Αποτέλεσμα Playout

4.2.5.2 Monte Carlo Simulation

Πρόκειται για μία προσομοίωση που σχετίζεται με το χρόνο και μας δίνει τη δυνατότητα να γνωρίζουμε πόσο πιθανό είναι να τερματιστεί μια εκτέλεση διεργασίας μετά από ένα δεδομένο χρονικό διάστημα. Αυτό οδηγεί σε καλύτερη εκτίμηση των Συμφωνιών Επιπέδου Υπηρεσιών (Service Level Agreements) ή σε καλύτερη αναγνώριση των περιπτώσεων διεργασίας που είναι πιο πιθανό να έχουν υψηλό χρόνο διεκπεραίωσης.

Προκειμένου να το πετύχουμε, θα χρειαστούμε το γράφημα απόδοσης directly-follows (performance DFG) που υπολογίσαμε σε προηγούμενη ενότητα, καθώς και την αναλογία άφιξης περιπτώσεων (case arrival ratio). Η αναλογία άφιξης περιπτώσεων είναι ο χρόνος που περνά, κατά μέσο όρο μεταξύ της άφιξης δύο διαδοχικών περιπτώσεων. Μπορεί να παρέχεται από το χρήστη ή να συναχθεί από το αρχείο καταγραφής συμβάντων. Το συμπέρασμα από το αρχείο καταγραφής συμβάντων γίνεται χρησιμοποιώντας την ακόλουθη εντολή:

```
# Monte Carlo Simulation
performance_dfg, start_activities, end_activities =
pm4py.discover_performance_dfg(filtered_log)

# Knowledge of the case arrival ratio
ratio = pm4py.get_rework_cases_per_activity(log)
print("Ratio: ", ratio)
```

```
Ratio:  {'08_AWB45_005': 146, '01_HOOFD_250_0': 2, '01_HOOFD_330': 2, '09_AH_I_010': 4,
 '01_HOOFD_380': 3, '01_HOOFD_375': 4, '01_HOOFD_190_2': 19, '01_HOOFD_100': 187, '01_HOOFD_130': 50,
  '01_HOOFD_110': 70, '01_HOOFD_120': 66, '01_HOOFD_180': 71, '01_HOOFD_190_1': 19, '01_HOOFD_200':
 38, '01_HOOFD_195_2': 1, '01_HOOFD_195_1': 2, '01_HOOFD_205': 89, '01_HOOFD_790': 13,
 '01_HOOFD_195': 1, '01_HOOFD_270': 2, '01_HOOFD_250': 1, '01_HOOFD_250_2': 2, '01_HOOFD_370': 2,
 '12_AP_010': 1, '01_HOOFD_400': 1, '01_HOOFD_420': 1, '01_HOOFD_440_1': 2, '01_HOOFD_445': 2,
 '01_HOOFD_440_2': 2, '01_HOOFD_450': 1, '01_HOOFD_451': 1, '10_UOV_065': 1, '08_AWB45_170': 1,
 '01_HOOFD_190': 18, '01_HOOFD_490_1': 3, '01_HOOFD_490_2': 2, '01_HOOFD_495': 3, '01_HOOFD_510_2':
 4, '01_HOOFD_510_1': 3, '01_HOOFD_520': 2, '01_HOOFD_505': 1, '01_HOOFD_530': 1, '01_HOOFD_065_2':
 3, '09_AWB45_005': 13, '11_AH_II_010': 4, '01_HOOFD_250_1': 1, '01_HOOFD_196': 1, '01_HOOFD_446': 1,
  '01_HOOFD_490_4': 3, '01_HOOFD_490_3': 1, '08_AWB45_020_1': 1, '06_VD_030_4': 1, '06_VD_060_1': 1,
 '06_VD_060_2': 1, '02_DRZ_030_2': 1, '02_DRZ_020_2': 1, '01_HOOFD_065_1': 1, '01_HOOFD_490_1a': 1,
 '01_HOOFD_510_2a': 1, '01_HOOFD_500': 1, '01_HOOFD_510_0': 1, '01_HOOFD_510_3': 1, '01_HOOFD_515':
 1, '01_HOOFD_519': 1, '01_HOOFD_532': 1}
```

Εικόνα 53: Αποτέλεσμα Ratio

Χρησιμοποιώντας την προσέγγιση εξόρυξης DFG, είναι δυνατή η ανάκτηση ενός μοντέλου Petri net. Αυτό το είδος μοντέλων είναι το προεπιλεγμένο για την προσομοίωση Monte Carlo, επειδή η σημασιολογία εκτέλεσης του είναι πολύ σαφής. Επιπλέον, το Petri net που εξάγεται από την προσέγγιση αυτή, είναι ένα υγιές workflow net που δίνει άλλες καλές ιδιότητες στο μοντέλο. Μπορεί να εφαρμοστεί με τον παρακάτω κώδικα:

```
net_mc, im_mc, fm_mc = pm4py.convert_to_petri_net(performance_dfg,
start_activities, end_activities)
```

Με τον παρακάτω κώδικα, μπορούμε να εφαρμόσουμε μια βασική προσομοίωση Monte Carlo. Στο συγκεκριμένο κομμάτι κώδικα, θα εκτελέσουμε μια προσομοίωση περιορισμένων πόρων, όπου ένα μέρος μπορεί να χωρέσει το πολύ 1 token ανά φορά.

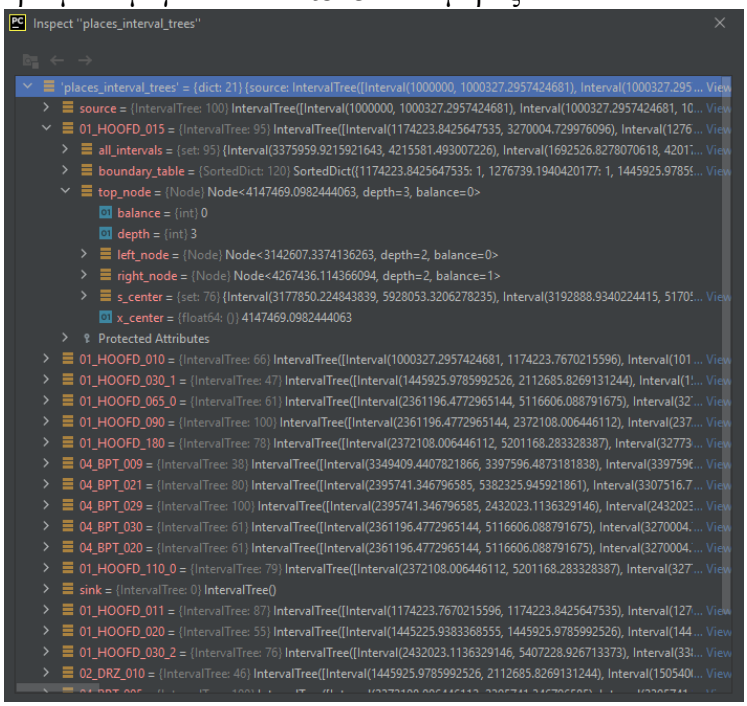
```
from pm4py.algo.simulation.montecarlo import algorithm as
montecarlo_simulation
from pm4py.algo.conformance.tokenreplay.algorithm import Variants

# DFG mining approach
net_mc, im_mc, fm_mc = pm4py.convert_to_petri_net(performance_dfg,
start_activities, end_activities)

parameters = {}
parameters[

montecarlo_simulation.Variants.PETRI_SEMAPH_FIFO.value.Parameters.TOK
EN_REPLAY_VARIANT] = Variants.BACKWARDS
parameters[montecarlo_simulation.Variants.PETRI_SEMAPH_FIFO.value.Par
ameters.PARAM_CASE_ARRIVAL_RATIO] = 10800

simulated_log_mc, res = montecarlo_simulation.apply(log, net_mc,
im_mc, fm_mc, parameters=parameters)
```

Τα κύρια αποτελέσματα της προσομοίωσης είναι:

- **simulated_log_mc:** Τα ίχνη που έχουν προσομοιωθεί κατά της διάρκειας της προσομοίωσης
- **res:** Το αποτέλεσμα της προσομοίωσης

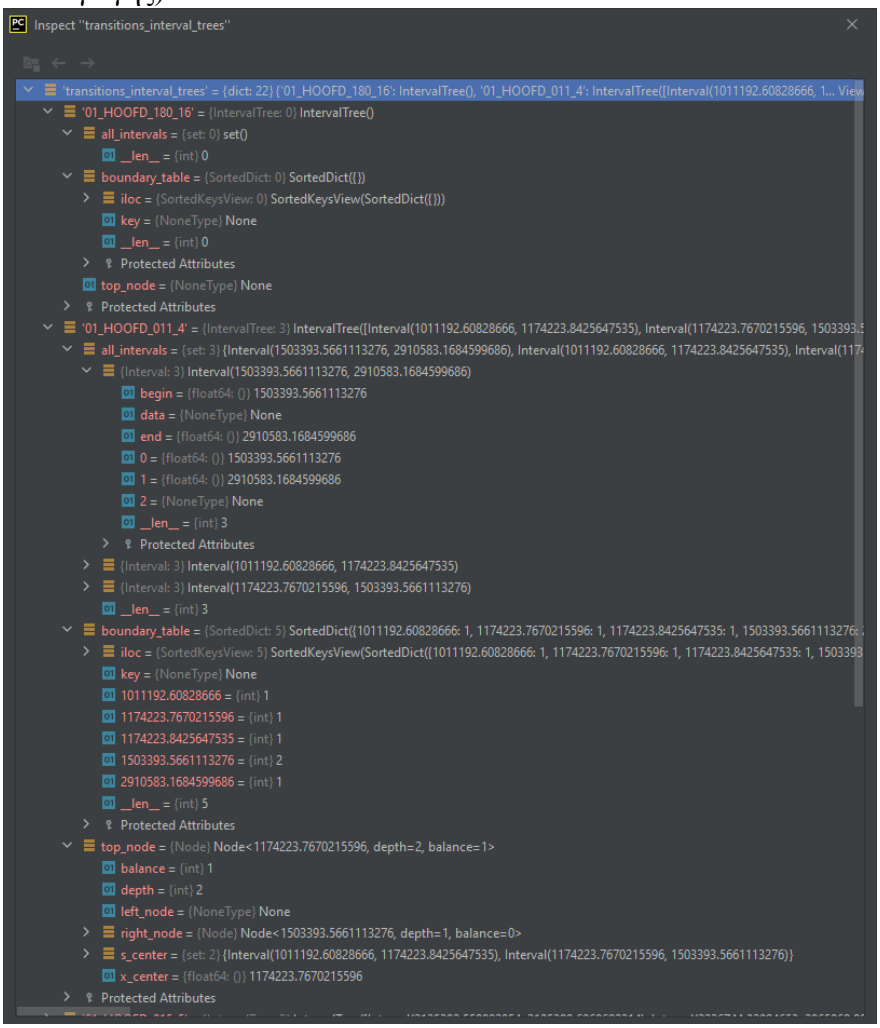
Μέσα στη μεταβλητή res, έχουμε τα ακόλουθα κλειδιά:

- **places_interval_trees:** Ένα δέντρο διαστήματος για κάθε μέρος, που φιλοξενεί ένα διάστημα για κάθε φορά που ήταν «γεμάτο» σύμφωνα με το καθορισμένο μέγιστο ποσό token ανά μέρος.



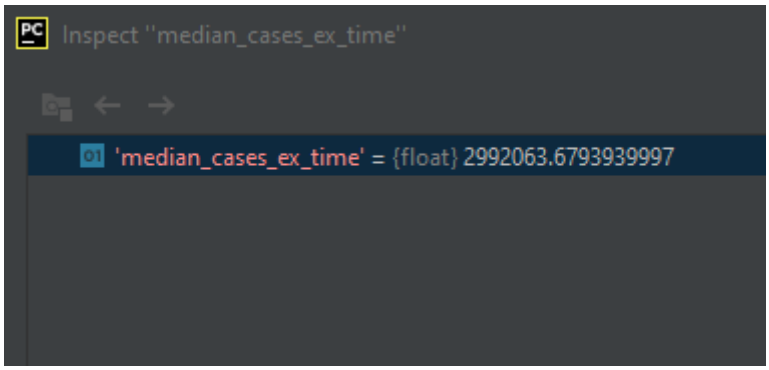
Εικόνα 54: Αποτελέσματα places_interval_trees

- **transitions_interval_trees:** Ένα δέντρο διαστήματος για κάθε μετάβαση, που περιέχει όλα τα χρονικά διαστήματα στα οποία η μετάβαση ενώ ήταν ενεργοποιημένη δεν ενεργοποιήθηκε (ο χρόνος μεταξύ μιας μετάβασης που ενεργοποιήθηκε πλήρως και της κατανάλωσης των token από τις θέσεις εισαγωγής).



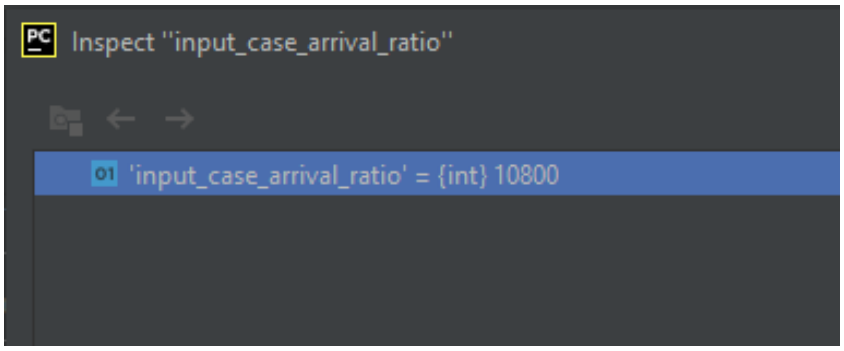
Εικόνα 55: Αποτελέσματα transition_interval_trees

- **median_cases_ex_time:** Ο μέσος χρόνος διεκπεραίωσης των περιπτώσεων στο προσομοιωμένο αρχείο καταγραφής.



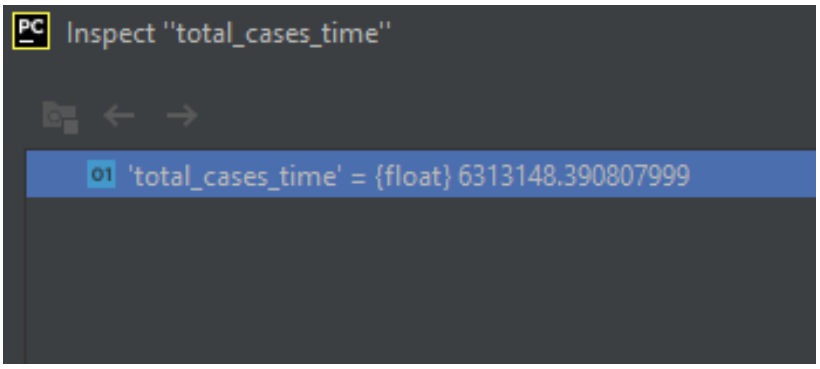
Εικόνα 56: Αποτελέσματα median_cases_ex_time

- **input_case_arrival_ratio:** Την αναλογία άφιξης περιπτώσεων που δόθηκε από τον χρήστη ή υπολογίστηκε αυτόματα από το αρχείο καταγραφής συμβάντων.



Εικόνα 57: Αποτελέσματα input_case_arrival_ratio

- **total_cases_time:** Τη διαφορά μεταξύ της τελευταίας χρονικής σφραγίδας του αρχείου καταγραφής και της πρώτης χρονικής σφραγίδας του προσομοιωμένου αρχείου καταγραφής.



Εικόνα 58: Αποτελέσματα total_cases_time

Τα τελευταία τέσσερα στοιχεία από τα παραπάνω κλειδιά είναι απλά αντικείμενα της python. Τα δύο πρώτα είναι αντικείμενα δέντρων διαστήματος που μπορούν να χρησιμοποιηθούν ώστε να εξάγουμε συγκεκριμένες πληροφορίες για το χρόνο.

Με το παρακάτω κομμάτι κώδικα, μπορούμε να εκτυπώσουμε για μια τυχαία μετάβαση στο μοντέλο, τον αριθμό των διαστημάτων που επικαλύπτονται για 11 διαφορετικά σημεία συμπεριλαμβανομένης της ελάχιστης και της μέγιστης χρονικής σήμανσης στο αρχείο καταγραφής, που κατανέμονται ομοιόμορφα σε όλο το χρονικό διάστημα του αρχείου καταγραφής.

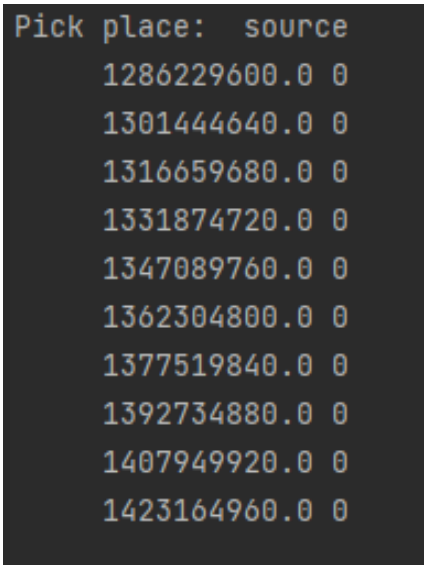
```
last_timestamp = max(event["time:timestamp"] for trace in log for
event in trace).timestamp()
first_timestamp = min(event["time:timestamp"] for trace in log for
event in trace).timestamp()
pick_trans = random.choice(list(res["transitions_interval_trees"]))
print("Pick trans: ", pick_trans)
n_div = 10
i = 0
while i < n_div:
    timestamp = first_timestamp + (last_timestamp - first_timestamp)
/ n_div * i
    print("\t", timestamp,
len(res["transitions_interval_trees"][pick_trans][timestamp]))
    i = i + 1
```

```
Pick trans:  01_HOOFD_030_2_3
     1286229600.0 0
     1301444640.0 0
     1316659680.0 0
     1331874720.0 0
     1347089760.0 0
     1362304800.0 0
     1377519840.0 0
     1392734880.0 0
     1407949920.0 0
     1423164960.0 0
```

Εικόνα 59: Αποτελέσματα για time specific information (transitions_interval_trees)

Θα επαναλάβουμε την ίδια διαδικασία για **places_interval_trees**, χρησιμοποιώντας τον παρακάτω κώδικα.

```
last_timestamp_places = max(event["time:timestamp"] for trace in log
for event in trace).timestamp()
first_timestamp_places = min(event["time:timestamp"] for trace in log
for event in trace).timestamp()
pick_place = random.choice(list(res["places_interval_trees"]))
print("Pick place: ", pick_place)
n_div = 10
i = 0
while i < n_div:
    timestamp_places = first_timestamp_places +
(last_timestamp_places - first_timestamp_places)/n_div * i
    print("\t", timestamp_places,
len(res["places_interval_trees"][pick_place][timestamp_places]))
    i = i + 1
```



Εικόνα 60: Αποτελέσματα για time specific information (places_interval_trees)

Χρησιμοποιώντας τις παραπάνω πληροφορίες μπορεί να γίνει δημιουργία γραφημάτων χρησιμοποιώντας προγράμματα όπως το Microsoft Excel. Η διαδικασία της προσομοίωσης μπορεί να συνεχιστεί με την ακόλουθη δομή:

- Λαμβάνεται υπόψη ένα αρχείο καταγραφής συμβάντων και ένα μοντέλο.
- Γίνεται μια λειτουργία επανάληψης μεταξύ του αρχείου καταγραφής και του μοντέλου.
- Η λειτουργία αυτή, έχει ως αποτέλεσμα την κατασκευή ενός στοχαστικού χάρτη, ο οποίος συσχετίζει σε κάθε μετάβαση μια κατανομή πιθανότητας και επιλέγεται εκείνη που μεγιστοποιεί την πιθανότητα των παρατηρούμενων τιμών κατά την επανάληψη.
- Στη συνέχεια, βρίσκεται η συχνότητα κάθε μετάβασης, η οποία βοηθά στην επιλογή μιας εκ των μεταβάσεων που ενεργοποιούνται σε μια σήμανση.
- Έπειτα ξεκινάει η διαδικασία της προσομοίωσης, στην οποία για κάθε ένα από τα ίχνη που δημιουργούνται, δημιουργείται ένα νήμα, ενώ γίνονται στοχαστικές επιλογές. Υπάρχει δυνατότητα χρήσης μια δεδομένης θέσης δίνοντας ένα αντικείμενο ως σηματοφόρο στην Python.
- Τέλος, καθορίζεται ένα μέγιστο χρονικό διάστημα για την προσομοίωση. Στην περίπτωση που ένα ή περισσότερα νήματα υπερβούν αυτό το χρονικό διάστημα, τότε «σκοτώνονται» και το αντίστοιχο ίχνος δεν προστίθεται στο αρχείο καταγραφής προσομοίωσης.

4.2.5.3 Extensive Playout of a Process Tree

Μια εκτεταμένη λειτουργία αναπαραγωγής (extensive playout operation) μας δίνει τη δυνατότητα να εξάγουμε ολόκληρη τη γλώσσα του μοντέλου διεργασίας. Τα δέντρα διεργασιών, με τη δομή τους από κάτω προς τα πάνω, επιτρέπουν την απόκτηση της γλώσσας ενός αρχείου καταγραφής με ευκολότερο τρόπο ξεκινώντας από τη γλώσσα των φύλλων και στη συνέχεια ακολουθώντας συγκεκριμένους κανόνες συγχώνευσης για τους τελεστές.

Επειδή η γλώσσα ενός δέντρου διεργασιών μπορεί να είναι τεράστια ή άπειρη, τα extensive playouts είναι δυνατά μέχρι κάποια όρια:

- Χρειάζεται να γίνει μια προδιαγραφή του μέγιστου αριθμού εμφανίσεων για έναν βρόχο. Αυτό διακόπτει μια εκτεταμένη λειτουργία αναπαραγωγής στον δεδομένο αριθμό εμφανίσεων.
- Εφόσον ο αριθμός των διαφορετικών εκτελέσεων εξακολουθεί να είναι απίστευτα μεγάλος, είναι δυνατό να καθοριστεί το μέγιστο μήκος ενός ίχνους που θα επιστραφεί. Έτσι, τα ίχνη που είναι πάνω από το μέγιστο μήκος θα απορρίπτονται.
- Επίσης, για να περιορίσουμε τον αριθμό των διαφορετικών εκτελέσεων, μπορούμε να ορίσουμε τον μέγιστο αριθμό ιχνών που θα επιστραφούν από τον αλγόριθμο.

Από τη δομή του δέντρου διεργασίας, μπορούμε να συμπεράνουμε το ελάχιστο μήκος ενός ίχνους το οποίο είναι επιτρεπτό από το μοντέλο διαδικασίας. Παρακάτω βλέπουμε ορισμένες λογικές ρυθμίσεις για το εκτεταμένο playout:

- Ο μέγιστος αριθμός ιχνών που επιστρέφονται από τον αλγόριθμο έχει οριστεί σε 100.000.
- Το μέγιστο μήκος ενός ίχνους που είναι έξοδος της αναπαραγωγής ορίζεται στο ελάχιστο μήκος ενός ίχνους που είναι αποδεκτό από ένα δέντρο διεργασίας.
- Ο μέγιστος αριθμός βρόχων ορίζεται ως το ελάχιστο μήκος ενός ίχνους διαιρούμενο με δύο.

4.3 Σχολιασμός Αποτελεσμάτων

Συνοψίζοντας, στην περίπτωση του A lpha miner έχουμε μια τιμή καταλληλότητας 0.9 που υποδεικνύει μια σχετικά καλή αντιστοίχιση μεταξύ του μοντέλου διαδικασίας που ανακαλύφθηκε και των δεδομένων αρχείου καταγραφής συμβάντων. Το μοντέλο που ανακαλύφθηκε είναι πιθανό να παρέχει σημαντικές πληροφορίες για την υποκείμενη διαδικασία. Όσον αφορά τον Inductive miner, έχουμε τιμή καταλληλότητας 1 που υποδηλώνει εξαιρετική αντιστοίχιση μεταξύ του ανακαλυφθέντος μοντέλου και των δεδομένων του αρχείου καταγραφής συμβάντων, γεγονός που υποδεικνύει ότι ο αλγόριθμος έχει ανακαλύψει μια ακριβή αναπαράσταση της διαδικασίας. Τέλος, για τον Heuristic miner έχουμε τιμή καταλληλότητας 0.7 που δηλώνει μια μέτρια αντιστοίχιση μεταξύ μοντέλου και αρχείου. Αυτό σημαίνει ότι το μοντέλο που ανακαλύφθηκε ενδέχεται να περιέχει ορισμένες ανακρίβειες. Ωστόσο, εξακολουθεί να παρέχει μια χρήσιμη αναπαράσταση της διαδικασίας που μπορεί να χρησιμοποιηθεί για ανάλυση και βελτίωση.

Ένα από τα κοινά προβλήματα στις διαδικασίες είναι ότι συχνά δεν είναι καλά κατανοητές ή δεν παρακολουθούνται σωστά. Αυτό μπορεί να οδηγήσει σε αναποτελεσματικότητα, συμφόρηση, ακόμη και λάθη. Αυτά τα προβλήματα μπορούν να επηρεάσουν αρνητικά τη συνολική απόδοση ενός οργανισμού και να περιορίσουν την ικανότητα του να επιτύχει τους στόχους του.

Για τη βελτίωση της ανακάλυψης της διαδικασίας, χρησιμοποιούμε διάφορες μετρήσεις. Μια μέτρηση είναι η καταλληλότητα του ίχνους, η οποία μετρά την ακρίβεια του μοντέλου διαδικασίας που ανακαλύφθηκε σε σύγκριση με τα δεδομένα καταγραφής συμβάντων. Μια άλλη μέτρηση είναι η ικανότητα επανάληψης, η οποία αξιολογεί πόσο καλά το μοντέλο που ανακαλύφθηκε μπορεί να αναπαράγει τα συμβάντα στο αρχείο καταγραφής. Ακόμη, σημαντικές μετρήσεις είναι η ακρίβεια και η γενίκευση, καθώς αξιολογούν το επίπεδο λεπτομέρειας και αφαιρετικότητας του μοντέλου. Τέλος, η απλότητα μετρά το επίπεδο πολυπλοκότητας και βοηθά στον εντοπισμό τυχόν περιττών λεπτομερειών που μπορούν να παραληφθούν για καλύτερη κατανόηση και ανάλυση.

Λαμβάνοντας υπόψη τις παραπάνω μετρήσεις, είναι δυνατό να βελτιωθεί η ποιότητα των μοντέλων διαδικασιών που ανακαλύφθηκαν και να αποκτηθούν καλύτερες γνώσεις για τις διαδικασίες που αναλύονται.

Κεφάλαιο 5ο: Συμπεράσματα και Μελλοντική Εργασία

Η παρούσα διπλωματική εργασία είχε ως στόχο την ανάλυση και την αξιολόγηση των επιχειρησιακών διαδικασιών στη Δημόσια Διοίκηση, με έμφαση σε Οργανισμούς Τοπικής Αυτοδιοίκησης με τη χρήση τεχνικών εξόρυξης διαδικασιών. Επίσης, όπως ανέφερα σε προηγούμενη ενότητα, προκειμένου να πετύχω εξόρυξη των διαδικασιών εφάρμοσα αρκετές τεχνικές που μας παρέχει η βιβλιοθήκη pm4py της Python. Τα δεδομένα είχαν τη μορφή event logs και αναλύθηκαν προκειμένου να εξαχθούν και να αξιολογηθούν τα μοντέλα των διαδικασιών, με σκοπό να εντοπιστούν αδύναμα σημεία και να βελτιωθούν οι διαδικασίες.

Στο θεωρητικό μέρος, προσέγγισα την Διαχείριση Επιχειρησιακών Διαδικασιών καθώς και την Εξόρυξη Διεργασιών. Στη συνέχεια αναφέρθηκα στην ερευνητική μεθοδολογία που ακολούθησα, παρέχοντας λεπτομέρειες σχετικά με την εισαγωγή και την επεξεργασία των δεδομένων προκειμένου να εφαρμοστούν σε αυτά οι αλγόριθμοι ανακάλυψης διαδικασιών, ο έλεγχος συμμόρφωσης και τέλος η ανάλυση απόδοσης.

Έπειτα, συνέχισα με την υλοποίηση η οποία εφαρμόζεται σε διαδικασίες χρηματοδότησης σε Οργανισμούς Τοπικής Αυτοδιοίκησης. Αρχικά, πέρασα τα δεδομένα από ορισμένα φίλτρα ώστε να είναι πιο εύκολη η επεξεργασία τους και στη συνέχεια έλαβα υπόψη τρεις περιπτώσεις χρήσης. Η πρώτη αφορούσε τον αλγόριθμο ανακάλυψης διαδικασιών Alpha miner, η δεύτερη τον Inductive miner και η τρίτη τον Heuristic miner. Για κάθε αλγόριθμο έτρεξα τις αντίστοιχες συναρτήσεις και ως έξοδο έλαβα τα αποτελέσματα τους τα οποία πέρασα από έλεγχο συμμόρφωσης τόσο με τη μέθοδο token-based replay όσο και με την μέθοδο alignments.

Τελευταίο βήμα ήταν η αξιολόγηση των αποτελεσμάτων βασισμένη σε τέσσερα κύρια χαρακτηριστικά. Το replay fitness, την ακρίβεια, τη γενίκευση και την απλότητα. Ξεκινώντας από τον αλγόριθμο Alpha miner, λάβαμε σχετικά καλά αποτελέσματα, καθώς αναγνώρισε ένα ικανοποιητικό μέρος από τη συμπεριφορά του μοντέλου. Ωστόσο, λόγω των μειονεκτημάτων του είναι λιγότερο αποδοτικός σε σχέση με τον Inductive miner ο οποίος μας έδωσε τα επιθυμητά αποτελέσματα. Αναλύοντας τα αποτελέσματα του ελέγχου συμμόρφωσης για τον Inductive miner φτάνουμε στο συμπέρασμα ότι πρόκειται για απόλυτη συμμόρφωση με το μοντέλο. Έχουμε τιμή trace fitness true και τα remaining, missing tokens είναι 0. Επίσης έχουμε μια πολύ καλή οπτικοποίηση με όλα τα πιθανά μονοπάτια. Αντιθέτως, για την περίπτωση του αλγόριθμου Heuristic miner, ο οποίος είναι κατασκευασμένος για να επεξεργάζεται μεγάλο όγκο δεδομένων, με την εφαρμογή του στο αρχείο καταγραφής συμβάντων, πραγματοποίησε αφαίρεση μερικών ακρών με αποτέλεσμα να παραμορφοποιηθούν τα μονοπάτια και να μην έχουμε σαφή εικόνα για τη συνολική συμμόρφωση του μοντέλου. Από τα αποτελέσματα που μας έδειξε το token-based replay συμπεραίνουμε ότι ο αλγόριθμος Heuristic miner είναι ο λιγότερος αξιόπιστος από τους τρεις για την περίπτωσή μας.

Συνοψίζοντας, μπορούμε να πούμε ότι στην περίπτωση των αλγόριθμων Alpha και Heuristic τα αποτελέσματα ενδέχεται να μην είναι αξιόπιστα. Ωστόσο για τον αλγόριθμο Inductive τα αποτελέσματα μπορούν να αξιολογηθούν ως επαρκή και είναι ένας κατάλληλος και εύχρηστος αλγόριθμος για το αρχείο καταγραφής συμβάντων που αναλύσαμε.

Τέλος, για μελλοντική μελέτη είναι αναγκαίο να φιλτράρουμε τα δεδομένα μας και να τα προετοιμάζουμε καταλλήλως για την ανακάλυψη διαδικασίας, καθώς και να εστιάζουμε σε ένα μικρό μέρος της συμπεριφοράς που θέλουμε να παρακολουθήσουμε.

Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές

Timo Pohl, (2019). An Inductive Miner Implementation for the PM4PY Framework. Bachelor's Thesis, i9 Process and Data Science (PADS) Chair, RWTH Aachen University

Eduard Šrol (2020). Process mining usage for potential insider threat identification utilizing PM4Py. Bachelor's Thesis, Masaryk University Faculty of Informatics

Ilias Merkoureas (2021). Development of a web-interface and web service API to support process mining techniques. Master Thesis, Department of Informatics and Computer Engineering, University of West Attica

Μαρία Παπαδοπούλου (2019). Ανασχεδιασμός Διαδικασιών στη Δημόσια Διοίκηση. Διπλωματική εργασία, Πρόγραμμα Μεταπτυχιακών Σπουδών στη Δημόσια Διοίκηση, Πανεπιστήμιο Μακεδονίας

Wil Van der Aalst (2020). Business Process Intelligence: Connecting Data and Processes. Retrieved from: https://www.researchgate.net/publication/283024393_EditorialBusiness_Process_Intelligence_Connecting_Data_and_Processes

Linda Tucci (2022). What is business process management? An in-depth BPM guide. Retrieved from: https://www.techtarget.com/searchcio/definition/business-process-management

Fraunhofer Institute for Applied Information Technology (2022). Pm4py – Documentation. Retrieved from: https://pm4py.fit.fraunhofer.de/docs

Process and Data Science Group – RWTH Aachen University. Process Discovery. Retrieved from: https://www.processmining.org/process-discovery.html

Process and Data Science Group – RWTH Aachen University. Conformance Checking. Retrieved from: https://www.processmining.org/conformance.html

Process and Data Science Group – RWTH Aachen University. Performance Analysis. Retrieved from: https://www.processmining.org/performance.html

Process and Data Science Group – RWTH Aachen University. The input for Process Mining. Retrieved from: https://www.processmining.org/extract-filter-clean-event-data.html

IBM. Process Mining. Retrieved from: https://www.ibm.com/topics/process-mining

c3d3 (2020). Process Mining with Python tutorial: A healthcare application. Retrieved from: https://medium.com/@c3_62722/process-mining-with-python-tutorial-a-healthcare-application-part-1-ae02027a050

Thomas H. Davenport, Andrew Spanyi (2019). What Process Mining Is, and Why Companies Should Do It. Retrieved from: https://hbr.org/2019/04/what-process-mining-is-and-why-companies-should-do-it

Γενική Διεύθυνση Γεωργίας και Αγροτικής Ανάπτυξης. Η κοινή γεωργική πολιτική με μια ματιά. Ανακτήθηκε από: https://agriculture.ec.europa.eu/common-agricultural-policy/cap-overview/cap-glance_el