



DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA

DIMES

Corso di Laurea Magistrale in Ingegneria Informatica

Progetto di Metodi informatici per
l'analisi dei processi

Sherlock VS Moriarty

Docenti

Prof.ssa Antonella Guzzo
Prof. Eugenio Vocaturo

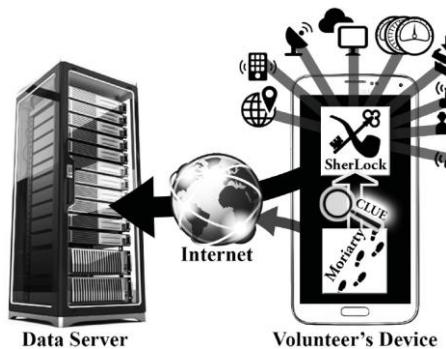
Studenti

Lorenzo Morelli (207038)
Antonio Piluso (204865)
Antonio Rosanò (204967)

Introduzione	3
Preprocessing	4
Analisi del dataset	11
Numero di case, tipo di attività, risorse ed eventi.....	11
Mappa generale del processo	13
Analisi con Dotted chart.....	17
Process discovery.....	20
Conformance checking.....	30
Analisi deviazioni con Celonis.....	33
Analisi deviazioni con ProM.....	38
Resource analysis.....	40
Social Network Analysis	44
Analisi con Dotted Chart.....	46
Performance analysis.....	49
Risorsa: 0a50e0962	49
Risorsa: ec3b3592f1	56
Anomaly detection.....	59
Risorsa: ec3b3592f1	59
Risorsa: 0a50e09262.....	66
Conclusioni.....	70

INTRODUZIONE

Lo sviluppo del progetto si basa su un dataset per smartphone ottenuto da un esperimento di raccolta dati a lungo termine. Il dataset contiene 10 miliardi di record di dati raccolti da 30 utenti su un periodo di 1,6 anni e altri 20 utenti per 6 mesi (per un totale di 50 utenti attivi). L'esperimento coinvolge due agenti per smartphone: **SherLock** e **Moriarty**. **SherLock** raccoglie un'ampia varietà di dati di software e sensori ad alta frequenza di campionamento. **Moriarty** perpetua vari attacchi all'utente e registra le sue attività, fornendo così le etichette per il dataset **SherLock**.



Lo scopo principale del dataset è quello di aiutare i professionisti della sicurezza e i ricercatori accademici a sviluppare metodi innovativi per l'individuazione implicita di comportamenti dannosi negli smartphone, in particolare da dati ottenibili senza privilegi di superuser (root), grazie al diverso approccio nella registrazione dei dati rispetto agli altri dataset dello stesso ambito, come si può vedere in figura:

Dataset	Objective	# of users	Devices	Launch	Duration	Feature	Sample Rates
SherLock	Implicit cybersecurity & general mobile phone data collection	50 (national)	Samsung Galaxy S5	2014	ongoing	Device settings/mode, running applications, audio features, BT/WiFi probe, network stats, telephony, location, power, screen on/off, motion, call/SMS log, Moriarty activity, broadcast intents, IO interrupt counts, others.	Fastest: every 5 seconds Most data: every minute
Device Analyzer	General mobile phone data collection	30,443 (international)	Heterogeneous: Android	2011	ongoing	Device settings/mode, running applications, audio features, BT/WiFi probe, network stats, telephony, location, power, screen on/off, motion, call/SMS log, others.	Every 5 minutes
LiveLab Project	Measuring wireless networks and smartphone users	34 (students)	iPhone 3GS	2010	1 year	App, WiFi, cellular, device activity, call/SMS logs, battery, network traffic, others.	Every 15 minutes
LDCC	Social sensing	170 (locals)	Nokia N95	2009	2 years	Location, call/SMS log, BT/WiFi probe, telephony, motion, audio, app events, calendar entries, phonebook entries	Every 60-300 seconds
Social Evolution Reality Mining	Social sensing	80 (students)	unknown	2008	1.75 years	BT and WiFi signal strength, location labels, call/SMS logs.	Every 6 minutes
	Social sensing	100 (faculty and students)	Nokia 6600	2004	9 months	Mobile and BT signal strengths, location, call/SMS logs, running Nokia applications.	Every 6 minutes

Il passo in avanti fatto con il dataset **Sherlock** è quindi caratterizzato dall'utilizzo di un rate maggiore di raccolta dei dati, che permette di avere una granularità migliore per effettuare l'analisi.

L'obiettivo del nostro progetto, a dispetto degli altri, è quello di andare a sfruttare e apprendere al meglio le tecniche di Process Mining sia per fare un'analisi classica del dataset, sia per effettuare un'analisi di Anomaly Detection.

PREPROCESSING

A differenza degli altri progetti, in questo caso è stata necessario dedicare molto tempo per il processamento del dataset. Questo perché il dataset di riferimento non registra processi ma eventi, quindi il team ha avuto la libertà di scegliere quale tipo di processo iniettare nel dataset. Questa libertà si è tradotta in un “problema di ottimizzazione” in quanto, avendo la possibilità di scegliere cosa sia un processo, c’è stata un’analisi approfondita del dataset che ha portato a determinate scelte e conseguentemente a diverse versioni di quest’ultimo. Di conseguenza si può dire che la fase di preprocessing è stata forse quella più importante e cruciale per il progetto in quanto la scelta del CaselD è di fondamentale importanza per tutte le analisi successive. Di seguito viene illustrato il lavoro fatto durante questa fase, che già indicativamente può essere sintetizzato andando a vedere l’indice del file utilizzato per mantenere queste informazioni:

Dataset 1.0.....	1
Process mining su dataset 1.0	7
Dataset 2.0.....	17
Analisi dataset 2.0	18
Process discovery dataset 2.0.....	21
Dataset 3.0.....	27
Process discovery dataset 3.0.....	33
Conformance checking dataset 3.0	47
Dotted chart dataset 3.0	60
Deviazioni Dataset 3.0 e miglioramento del modello.....	64

Figure 1: Indice del file relativo al preprocessing

Come si può notare dall’indice, il lavoro dedicato alla modifica del dataset è stato importante ed è da questo che ne sono uscite diverse versioni che verranno trattate in questa sezione. Il grosso problema riscontrato è che in ogni versione del dataset è stato necessario effettuare buona parte delle analisi prima di capire che non era adatto alle esigenze imposte dal progetto. Questo continuo aggiornamento di versione è stato un punto cruciale all’interno del progetto perché per avere dei riscontri positivi (o negativi) di una determinata scelta, come quella del CaselD. Il team era obbligato ad effettuare dei test di analisi (process discovery, resource analysis ecc) e solo dopo aver fatto abbastanza test si riusciva a cogliere i pro e i contro della scelta precedente. Questo aspetto ha portato ad un miglioramento continuo del dataset ma anche ad un notevole impiego di tempo per registrare i problemi relativi alle precedenti scelte. La prima parte dello sviluppo del progetto è stata caratterizzata proprio da questo ciclo che vedeva interscambiarsi la fase di **preprocessing** con quella di **testing** per quantificare la qualità del dataset. Una volta che questo ciclo ha dato le risposte volute, si è partiti con l’analisi vera e propria.

Come si può vedere dall’immagine in figura 1, per valutare la bontà di alcune scelte è stato impiegato molto tempo e molto studio anche per via dei vari problemi che ci sono stati nel maneggiare questi dataset giganteschi. A tal proposito sono stati utilizzati diversi strumenti come **PowerBi**, **Dax** e **Python** che hanno permesso di rendere possibili modifiche su file così grandi.



Figure 2: Stack software utilizzato nella fase di preprocessing

Il primo problema in assoluto che si è dovuto affrontare è stato la scelta di quale dataset utilizzare per l’analisi, in quanto i dati catturati superano di gran lunga il Terabyte e sono suddivisi in diverse porzioni.

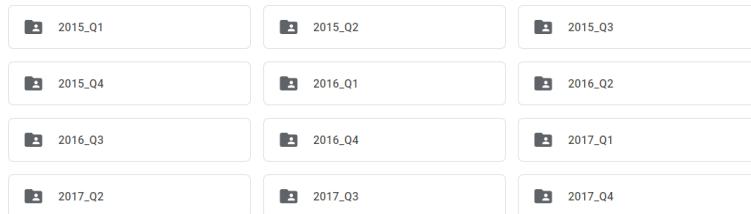


Figure 3: Suddivisione dataset per trimestre

a_anon_T1.tsv	Yisroel Mirsky	25 dic 2017	Yisroel Mirsky	280 MB
Allbroadcastprobe.tsv	Yisroel Mirsky	22 dic 2017	Yisroel Mirsky	972 MB
T2.tsv	Yisroel Mirsky	22 dic 2017	Yisroel Mirsky	12 GB
T3.tsv	Yisroel Mirsky	22 dic 2017	Yisroel Mirsky	3 GB
anon_T4.tsv	Yisroel Mirsky	22 dic 2017	Yisroel Mirsky	10 GB
anon_Wifi.tsv	Yisroel Mirsky	22 dic 2017	Yisroel Mirsky	439 MB
Application.7z	Yisroel Mirsky	22 dic 2017	Yisroel Mirsky	6 GB
Application.tsv	Yisroel Mirsky	22 dic 2017	Yisroel Mirsky	127 GB

Figure 4: Suddivisione in tipologie di dataset

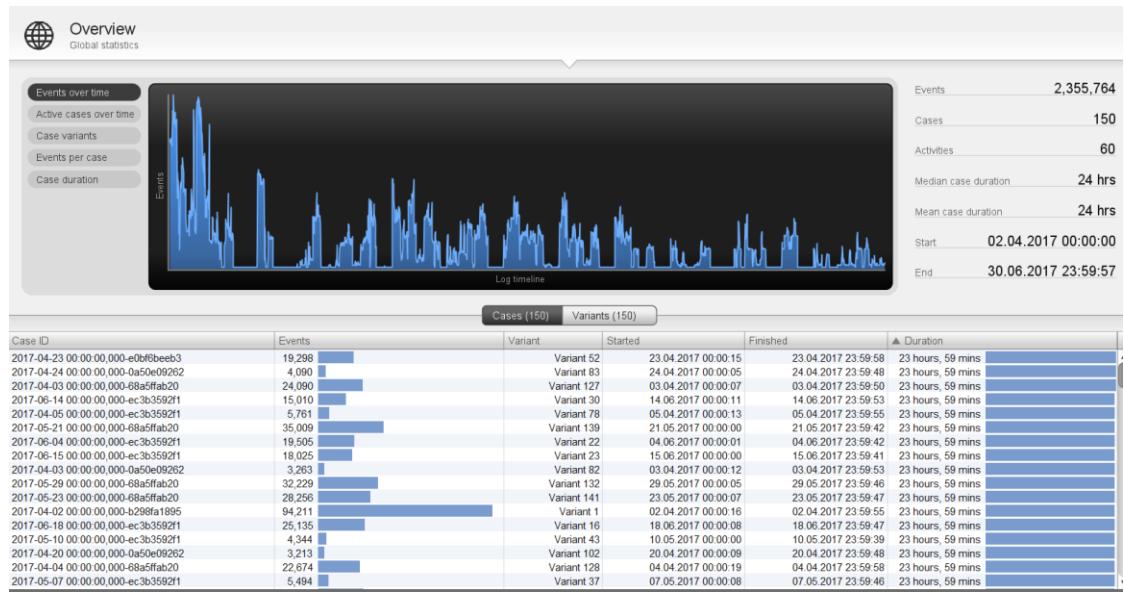
Si è deciso di scegliere in maniera più intelligente possibile uno dei tanti dataset disponibili. La scelta è ricaduta in **ALLBROADCAST** che sostanzialmente mantiene al suo interno tutti gli eventi relativi a connessioni, appunto broadcast, come il Bluetooth o il wifi ecc. Questo dataset è quello che tra tutti riesce meglio a modellare un “processo” in quanto possiede al suo interno diverse informazioni utili che possono essere mappate e analizzate come tale.

“A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. You can deliver a broadcast to other apps by passing an Intent to sendBroadcast() or sendOrderedBroadcast(). ”

Figure 5: Definizione eventi broadcast da documentazione Android

Come precedentemente accennato, il dataset scelto è risultato però difficile da manipolare con gli strumenti standard in quanto non permettono agevolmente di gestire una mole di dati così grande (per esempio con excel si riesce a gestire al massimo un milione di tuple, e quasi tutti gli editor di testo non sono in grado di aprire questi file di questa grandezza). Proprio per questo motivo si è scelto l’utilizzo dei software **PowerBI** e **Dax Studio**, che hanno consentito una manipolazione più semplice e agevole del dataset. Più nello specifico, si è riusciti ad estrarre la data dal **timestamp** e suddividere le diverse componenti per creare la colonna **CaseID** relativa al giorno, successivamente ordinata in maniera crescente. Il dataset ottenuto corrisponde alla **versione 1.0**.

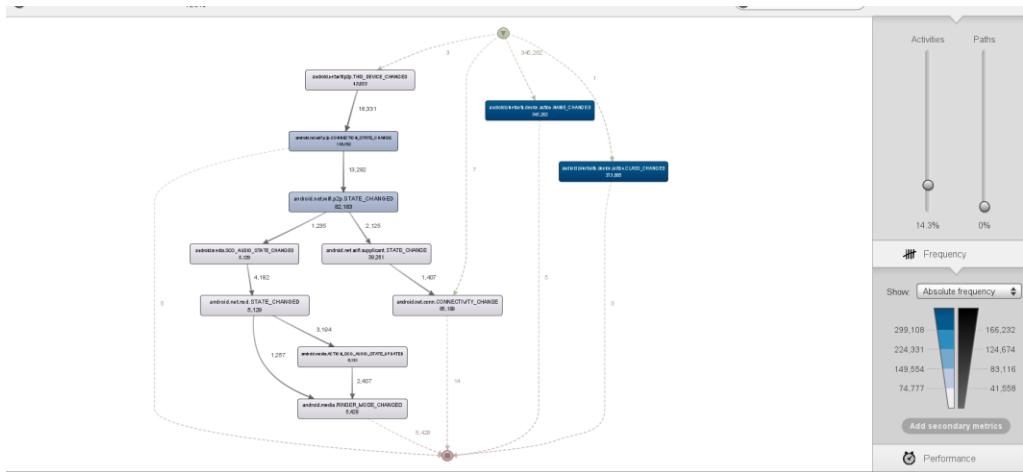
Un miglioramento fatto fin da subito è stato quello di caratterizzare il **CaseID** non solo per giorno ma anche per risorsa, in maniera da caratterizzare i processi come le giornate per ogni singolo utente. Questa miglioria corrisponde alla **versione 1.1** del dataset, di seguito analizzata:



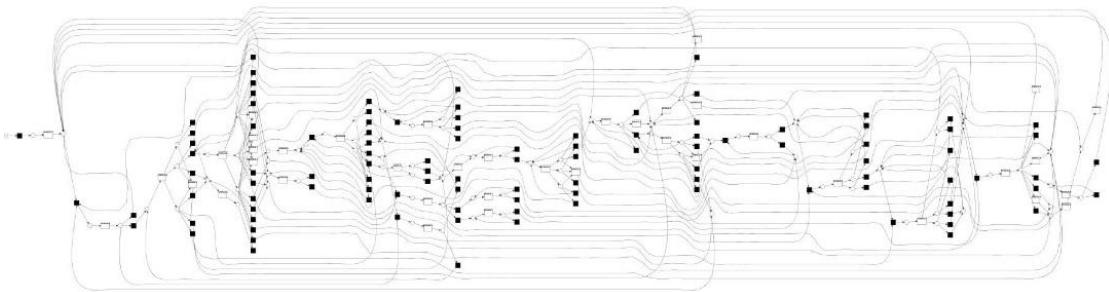
Nelle foto sopra riportata è stato già applicato un filtro per eliminare eventuali tracce spurie che non duravano 24h. Questa versione del dataset risulta avere delle caratteristiche molto diverse rispetto ai dataset forniti per la BPIChallenge.

Prima di tutto il dataset era composto da tantissimi eventi e pochissimi **CaseID** e questo porta direttamente ad un problema nel “leggere” i processi. Tale problema è generato dalla natura del dataset che non è fatto per codificare processi, tant’è vero che la maggior parte del lavoro fatto dal team è stato quello di andare ad analizzare e codificare il miglior concetto di processo possibile da poter iniettare artificialmente all’interno degli eventi. Questo sbilanciamento tra eventi e case ha portato ad una grossa problematica: andare a codificare dei processi che sostanzialmente risultavano univoci all’interno del dataset. Catturando infatti eventi generati da un dispositivo elettronico (con un potenziale di migliaia di eventi al minuto) risulta impensabile anche solo notare qualche vicinanza all’interno di due case casuali, perché probabilmente definire il processo come un’intera giornata potrebbe risultare incompatibile con la logica che vi è dietro un dispositivo del genere. I dispositivi quali gli smartphone sono infatti dei dispositivi che implementano al loro interno dei micro-servizi (scansioni Bluetooth, pairing wifi, aggiornamento componenti grafiche ecc) che cooperano tra di loro formano dei servizi più grandi (la gestione del Bluetooth, la gestione del wifi ecc). Proprio per questo motivo si è pensato di catturare questi micro-servizi e vederli come processo, in modo da avere un legame più stretto tra questi due aspetti, che può essere molto utile nel corso delle varie analisi. A supporto di quanto detto, nelle versioni 1.0 e 1.1 del dataset possiamo vedere che ogni case genera un variante diversa e quindi non risultano esserci due case uguali, com’è possibile vedere nell’immagine seguente:

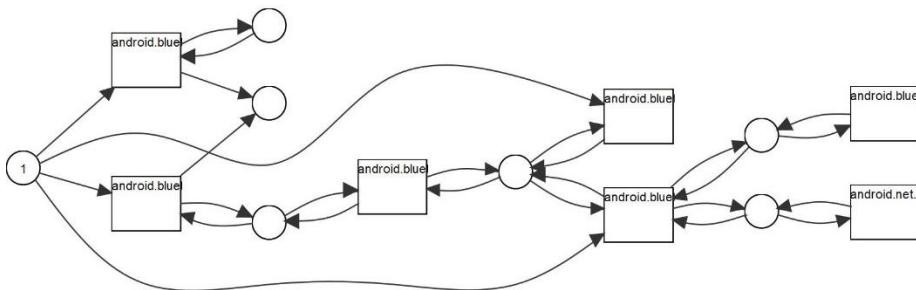
Case ID	Events	Variant	Started	Finished	Duration
2017-04-23 00:00:00,000-e0bf6beeb3	19,298	Variant 52	23.04.2017 00:00:15	23.04.2017 23:59:58	23 hours, 59 mins
2017-04-24 00:00:00,000-a50e09262	4,090	Variant 83	24.04.2017 00:00:05	24.04.2017 23:59:48	23 hours, 59 mins
2017-04-03 00:00:00,000-68a5ffab20	24,090	Variant 127	03.04.2017 00:00:07	03.04.2017 23:59:50	23 hours, 59 mins
2017-06-14 00:00:00,000-ec3b3592f1	15,010	Variant 30	14.06.2017 00:00:11	14.06.2017 23:59:53	23 hours, 59 mins
2017-04-05 00:00:00,000-ec3b3592f1	5,761	Variant 78	05.04.2017 00:00:13	05.04.2017 23:59:55	23 hours, 59 mins
2017-05-21 00:00:00,000-68a5ffab20	35,009	Variant 139	21.05.2017 00:00:00	21.05.2017 23:59:42	23 hours, 59 mins
2017-06-04 00:00:00,000-ec3b3592f1	19,505	Variant 22	04.06.2017 00:00:01	04.06.2017 23:59:42	23 hours, 59 mins
2017-06-15 00:00:00,000-ec3b3592f1	18,025	Variant 23	15.06.2017 00:00:00	15.06.2017 23:59:41	23 hours, 59 mins
2017-04-03 00:00:00,000-a50e09262	3,263	Variant 82	03.04.2017 00:00:12	03.04.2017 23:59:53	23 hours, 59 mins
2017-05-29 00:00:00,000-68a5ffab20	32,229	Variant 132	29.05.2017 00:00:05	29.05.2017 23:59:46	23 hours, 59 mins
2017-05-23 00:00:00,000-68a5ffab20	28,256	Variant 141	23.05.2017 00:00:07	23.05.2017 23:59:47	23 hours, 59 mins
2017-04-02 00:00:00,000-b298fa1895	94,211	Variant 1	02.04.2017 00:00:16	02.04.2017 23:59:55	23 hours, 59 mins
2017-06-18 00:00:00,000-ec3b3592f1	25,135	Variant 16	18.06.2017 00:00:08	18.06.2017 23:59:47	23 hours, 59 mins
2017-05-10 00:00:00,000-ec3b3592f1	4,344	Variant 43	10.05.2017 00:00:00	10.05.2017 23:59:39	23 hours, 59 mins
2017-04-20 00:00:00,000-a50e09262	3,213	Variant 102	20.04.2017 00:00:09	20.04.2017 23:59:48	23 hours, 59 mins
2017-04-04 00:00:00,000-68a5ffab20	22,674	Variant 128	04.04.2017 00:00:19	04.04.2017 23:59:58	23 hours, 59 mins
2017-05-07 00:00:00,000-ec3b3592f1	5,494	Variant 37	07.05.2017 00:00:08	07.05.2017 23:59:46	23 hours, 59 mins



Disco in maniera oggettiva ritornava anche uno pseudo-processo che a primo sguardo potrebbe sembrare descrittivo del dataset, ma in realtà risulta praticamente inutile in quanto, dalle considerazioni precedentemente fatte, non vi è nessun collegamento che può mettere in correlazione i **CaseID** (essendo univoci). In definitiva ci siamo accorti che non ha senso provare ad estrapolare un major-behavior da un dataset con queste caratteristiche, tant'è che l'applicazione di alcune tecniche di Process Discovery portavano sostanzialmente a delle gigantesche petri-net spaghetti style che rappresentavano un'enumerazione (anche poco accurata) delle tracce presenti nel dataset senza codificare nulla a livello concettuale:



Non avendo alcun tipo di correlazione tra i case che compongono il dataset, il risultato di qualsiasi filtro utilizzato su ProM risulta sostanzialmente troppo brusco, tagliando fuori quasi tutte le varianti e restituendo dei modelli di processo inutilizzabili in quanto troppo semplici:

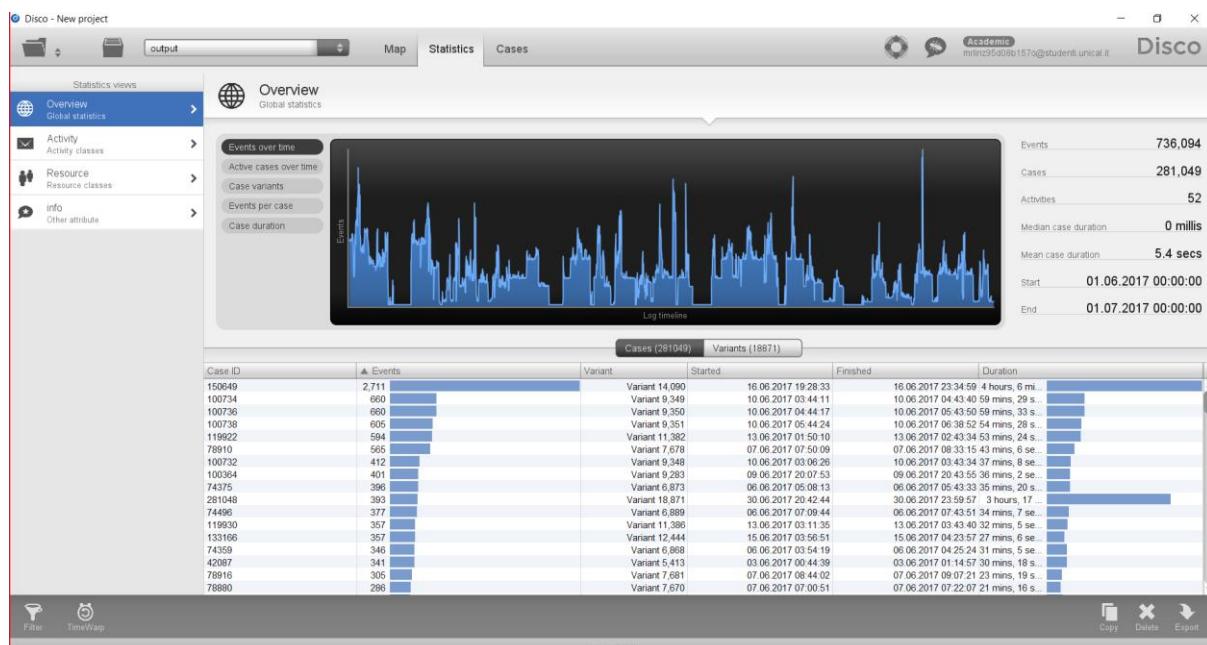


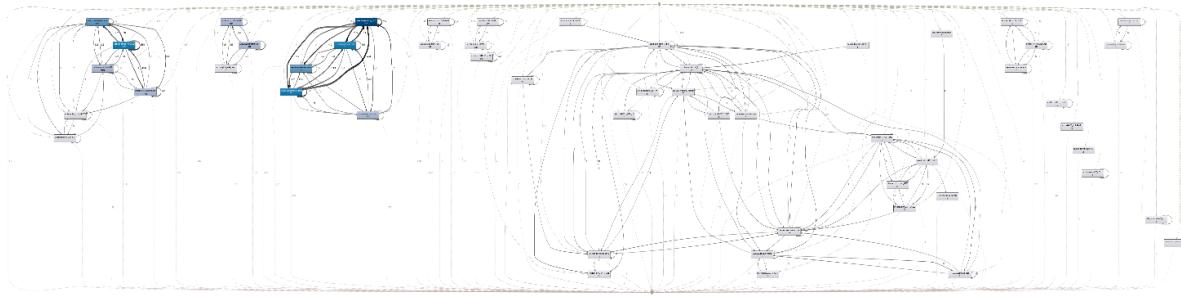
Grazie allo studio più approfondito di alcune caratteristiche del dataset, si è quindi pensato di cambiare **CaseID** in quanto si è visto che era possibile inglobare in un case le attività che venivano generate da determinate primitive. Si è perciò scelto come nuovo **CaseID** le componenti che generano gli eventi. Questa modifica è stata notevolmente di aiuto in quanto andando a cambiare la visione dei processi è stato possibile ottenere una migliore rappresentazione. La logica dietro questo ragionamento è la seguente: trovare un modo per migliorare la qualità dei case rendendoli effettivamente più significativi ma senza scomodare tecniche complesse quali deep learning.

Più nel dettaglio, si è deciso di accomunare nello stesso caso tutte le attività contigue presenti nel dataset che sono generati dalla stessa componente.

NewCaseId	Case ID	Activity	Complete Timestamp	Variant
1	0a50e09262	android.media.ACTION_SCO_AUDIO_STATE_UPDATED	01/06/2017 02:17:15	Variant 1
1	0a50e09262	android.media.RINGER_MODE_CHANGED	01/06/2017 02:17:15	Variant 1
1	0a50e09262	android.media.SCO_AUDIO_STATE_CHANGED	01/06/2017 02:17:15	Variant 1
2	0a50e09262	android.net.conn.CONNECTIVITY_CHANGE	01/06/2017 02:17:15	Variant 1
3	0a50e09262	android.net.nsd.STATE_CHANGED	01/06/2017 02:17:15	Variant 1
4	0a50e09262	android.net.wifi.p2p.CONNECTION_STATE_CHANGE	01/06/2017 02:17:15	Variant 1
4	0a50e09262	android.net.wifi.p2p.STATE_CHANGED	01/06/2017 02:17:15	Variant 1
4	0a50e09262	android.net.wifi.p2p.THIS_DEVICE_CHANGED	01/06/2017 02:17:15	Variant 1
5	0a50e09262	android.net.wifi.RSSI_CHANGED	01/06/2017 02:17:15	Variant 1
5	0a50e09262	android.net.wifi.STATE_CHANGE	01/06/2017 02:17:15	Variant 1
6	0a50e09262	android.net.wifi.suplicant.STATE_CHANGE	01/06/2017 02:17:15	Variant 1
7	0a50e09262	android.media.ACTION_SCO_AUDIO_STATE_UPDATED	01/06/2017 02:17:20	Variant 1
7	0a50e09262	android.media.RINGER_MODE_CHANGED	01/06/2017 02:17:20	Variant 1
7	0a50e09262	android.media.SCO_AUDIO_STATE_CHANGED	01/06/2017 02:17:20	Variant 1
8	0a50e09262	android.net.conn.CONNECTIVITY_CHANGE	01/06/2017 02:17:20	Variant 1
9	0a50e09262	android.net.nsd.STATE_CHANGED	01/06/2017 02:17:20	Variant 1
10	0a50e09262	android.net.wifi.p2p.CONNECTION_STATE_CHANGE	01/06/2017 02:17:20	Variant 1
10	0a50e09262	android.net.wifi.p2p.STATE_CHANGED	01/06/2017 02:17:20	Variant 1
10	0a50e09262	android.net.wifi.p2p.THIS_DEVICE_CHANGED	01/06/2017 02:17:20	Variant 1

Questo perché effettivamente, studiando il paper del progetto **Sherlock** e la documentazione android, abbiamo notato che la maggior parte dei micro-servizi (che si vuole trattare come processo) gestiti dai dispositivi vengono appunto incapsulati in una sequenza di primitive. Pertanto, vedere il processo come una successione contigua di registrazioni di tali eventi ha portato effettivamente il dataset ad un salto di qualità importate per quanto riguarda la distribuzione ed il rapporto del numero di eventi, case e varianti che diventano accettabili per poter andare a effettuare delle analisi di questo tipo. Dai primi test sul software disco è inoltre emerso che la visione del processo risulta molto più fedele al comportamento, simil automa a stati finiti, che può essere codificato all'interno di un dispositivo di questo tipo. Quello che ci si aspettava era infatti di trovare una "rete" molto ampia in cui vi erano codificati diversi servizi rappresentati da sotto-grafi responsabili della gestione di quel servizio. Effettuando tale modifica, si è infatti ottenuto il seguente risultato:





Questo upgrade è stato possibile sempre grazie all'utilizzo congiunto di **PowerBi**, **Dax** e **Python**.

Di seguito vengono riportati frammenti del codice per la modifica del dataset:

```
previous=''
sys.stdout.write('[')
sys.stdout.flush()
for row in rows:
    if(count%100000 == 0):
        sys.stdout.write('*')
        sys.stdout.flush()
    tmp=re.split("[A-Z]", row[1])[0]
    if (tmp == previous):
        row.insert(0,CaseIdcount)
    else:
        previous = tmp
        CaseIdcount+=1
        row.insert(0,CaseIdcount)
    output.append(row)
    count+=1
sys.stdout.write(']')
sys.stdout.flush()

##### to write csv####
```

Da questa evoluzione in poi ci sono state altre due versioni che sostanzialmente sono un miglioramento di quella sopra citata (v2.1 e la v2.2) in cui, a causa di due piccoli problemi riscontrati, si è andati a modificare leggermente l'ordinamento del dataset e il codice python per ottenere un dataset più pulito.

La prima miglioria è stata quello di ordinare, a parità di timestamp, le tracce per evento in modo da non avere situazioni spiacevoli in cui tracce che potevano essere accomunate in uno stesso case risultavano divise in quanto nello stesso istante veniva fatta anche un'altra rilevazione spuria (essendo spesso eventi paralleli). Questo fenomeno è dovuto anche alla granularità delle rilevazioni che risulta essere nell'ordine dei secondi, mentre gli eventi generati dalle componenti sono nell'ordine dei millisecondi, e proprio per questo capitava spesso che nello stesso secondo venivano registrati tantissimi eventi.

```
1,00:00:00,000",android.bluetooth.adapter.action.DISCOVERY_STARTED,{  
1,00:00:00,000",android.bluetooth.adapter.action.LOCAL_NAME_CHANGED,  
1,00:00:00,000",android.bluetooth.adapter.action.SCAN_MODE_CHANGED,{  
1,00:00:00,000",android.bluetooth.adapter.action.SCAN_MODE_CHANGED,{  
1,00:00:00,000",android.bluetooth.adapter.action.STATE_CHANGED,68a5f1  
2,00:00:00,000",android.bluetooth.device.action.CLASS_CHANGED,68a5f1  
2,00:00:00,000",android.bluetooth.device.action.CLASS_CHANGED,68a5ff  
2,00:00:00,000",android.bluetooth.device.action.NAME_CHANGED,68a5ffa  
2,00:00:00,000",android.bluetooth.device.action.NAME_CHANGED,68a5ffa  
2,00:00:00,000",android.bluetooth.device.action.UUID,68a5ffb20,Vari  
2,00:00:00,000",android.bluetooth.device.action.UUID,68a5ffb20,Vari  
3,00:00:00,000",android.net.wifi.p2p.CONNECTION_STATE_CHANGE,68a5ff  
3,00:00:00,000",android.net.wifi.p2p.CONNECTION_STATE_CHANGE,68a5ffe  
3,00:00:00,000",android.net.wifi.p2p.CONNECTION_STATE_CHANGE,68a5ffe  
3,00:00:00,000",android.net.wifi.p2p.STATE_CHANGED,68a5ffb20,Varia  
3,00:00:00,000",android.net.wifi.p2p.STATE_CHANGED,68a5ffb20,Varia  
3,00:00:00,000",android.net.wifi.p2p.THIS_DEVICE_CHANGED,68a5ffb20,
```

```
1,00:00:00,000",android.bluetooth.device.action.NAME_CHANGED,68a5ff  
2,00:00:00,000",android.net.wifi.p2p.STATE_CHANGED,68a5ffb20,Varia  
3,00:00:00,000",android.bluetooth.adapter.action.STATE_CHANGED,68a5f  
4,00:00:00,000",android.net.wifi.p2p.CONNECTION_STATE_CHANGE,68a5ff  
4,00:00:00,000",android.net.wifi.p2p.CONNECTION_STATE_CHANGE,68a5ff  
4,00:00:00,000",android.net.wifi.p2p.STATE_CHANGED,68a5ffb20,Varia  
5,00:00:00,000",android.bluetooth.adapter.action.SCAN_MODE_CHANGED,  
6,00:00:00,000",android.bluetooth.device.action.NAME_CHANGED,68a5ff  
7,00:00:00,000",android.bluetooth.adapter.action.SCAN_MODE_CHANGED,  
7,00:00:00,000",android.bluetooth.adapter.action.DISCOVERY_STARTED,  
8,00:00:00,000",android.net.wifi.p2p.THIS_DEVICE_CHANGED,68a5ffb20  
9,00:00:00,000",android.bluetooth.device.action.CLASS_CHANGED,68a5f  
10,00:00:00,000",android.net.wifi.p2p.CONNECTION_STATE_CHANGE,68a5f  
11,00:00:00,000",android.bluetooth.device.action.CLASS_CHANGED,68a5f  
11,00:00:00,000",android.bluetooth.device.action.UUID,68a5ffb20,Va  
11,00:00:00,000",android.bluetooth.device.action.UUID,68a5ffb20,Va  
12,00:00:00,000",android.bluetooth.adapter.action.LOCAL_NAME_CHANGE
```

La seconda miglioria invece è stata dettata dal fatto che, non avendo ordinato le risorse (nel nostro caso gli utenti), all'interno del dataset vi erano alcuni case che riportavano una durata anomala rispetto alle altre in quanto gli eventi di risorse diverse apparivano intersecati. Questo problema è stato risolto, sempre con la combo **PowerBI/Dax**, ordinando prima di tutto le risorse, a parità di risorse per timestamp e a parità di timestamp per evento.

	Activity	Resource	Date	Time
1	android.bluetooth.adapter.action.DISCOVERY_FINISHED	ec3b3592f1	01.06.2017	09:14:34
2	android.bluetooth.adapter.action.DISCOVERY_FINISHED	ec3b3592f1	01.06.2017	09:14:35
3	android.bluetooth.adapter.action.SCAN_MODE_CHANGED	ec3b3592f1	01.06.2017	09:14:35
4	android.bluetooth.adapter.action.STATE_CHANGED	ec3b3592f1	01.06.2017	09:14:35
5	android.bluetooth.adapter.action.STATE_CHANGED	ec3b3592f1	01.06.2017	09:14:35
6	android.bluetooth.adapter.action.DISCOVERY_STARTED	0a50e09262	01.06.2017	09:14:50

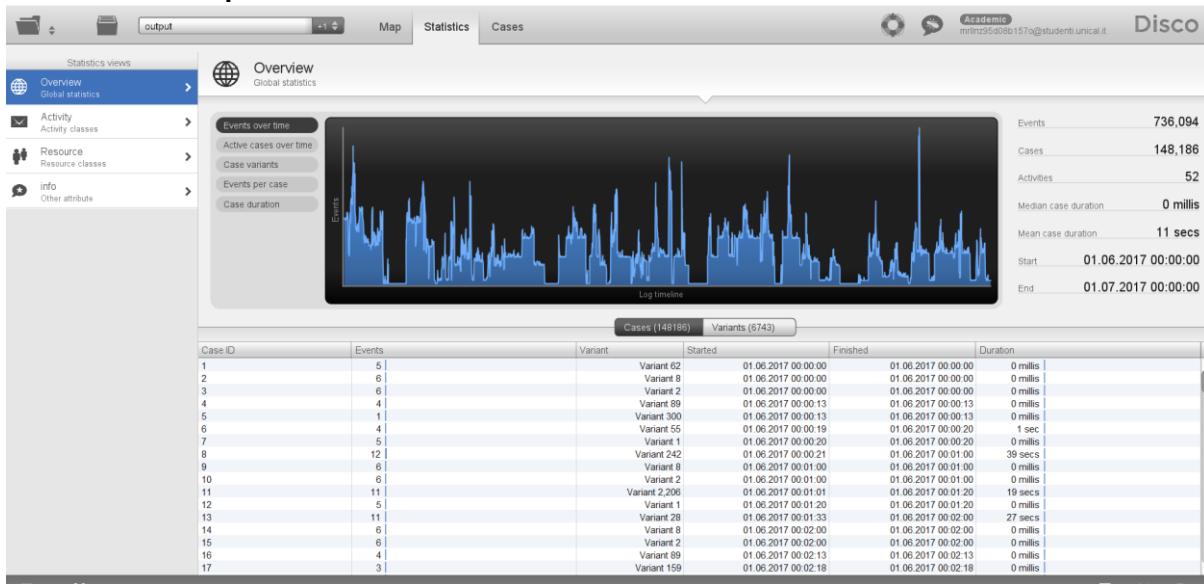
	Activity	Resource	Date	Time
1	android.bluetooth.adapter.action.DISCOVERY_FINISHED	ec3b3592f1	01.06.2017	09:14:34
2	android.bluetooth.adapter.action.DISCOVERY_FINISHED	ec3b3592f1	01.06.2017	09:14:35
3	android.bluetooth.adapter.action.SCAN_MODE_CHANGED	ec3b3592f1	01.06.2017	09:14:35
4	android.bluetooth.adapter.action.STATE_CHANGED	ec3b3592f1	01.06.2017	09:14:35
5	android.bluetooth.adapter.action.STATE_CHANGED	ec3b3592f1	01.06.2017	09:14:35

Queste modifiche progressive hanno portato ad un dataset abbastanza rivoluzionato ma che su diversi aspetti ha dato le risposte necessarie. Non ci si aspettavano dei risultati ed una compatibilità perfetta ma rispetto alle prime versioni il miglioramento è netto e le risposte avute dalle diverse analisi sono state soddisfacenti.

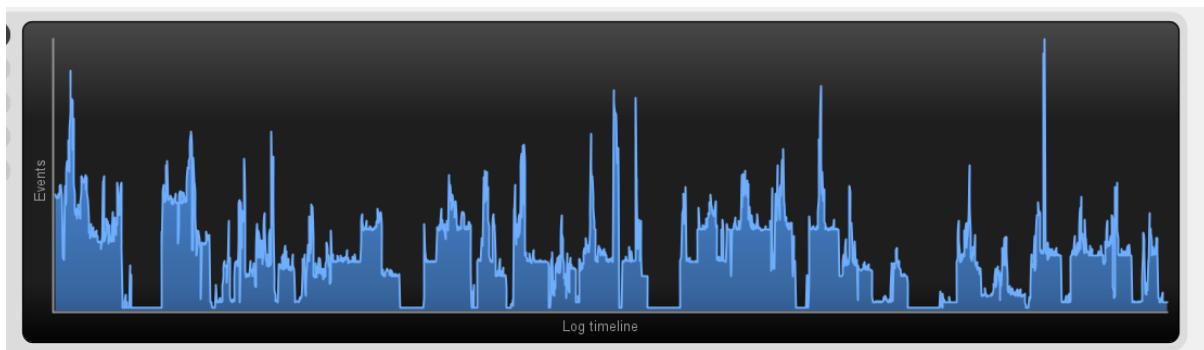
ANALISI DEL DATASET

Dopo le varie modifiche discusse nel capitolo precedente, Allbroadcast risulta essere abbastanza adatto per iniziare l'analisi andando ad utilizzare gli strumenti studiati a lezione. Come già accennato in precedenza, il dataset preso in considerazione contiene al suo interno tutti gli eventi registrati nell'arco di un mese di attività del progetto **Sherlock**. Tali dati riguardano i **broadcast intent**, che sostanzialmente sono degli eventi che il dispositivo manda a tutte le sue componenti per notificare l'intento di voler eseguire una determinata operazione.

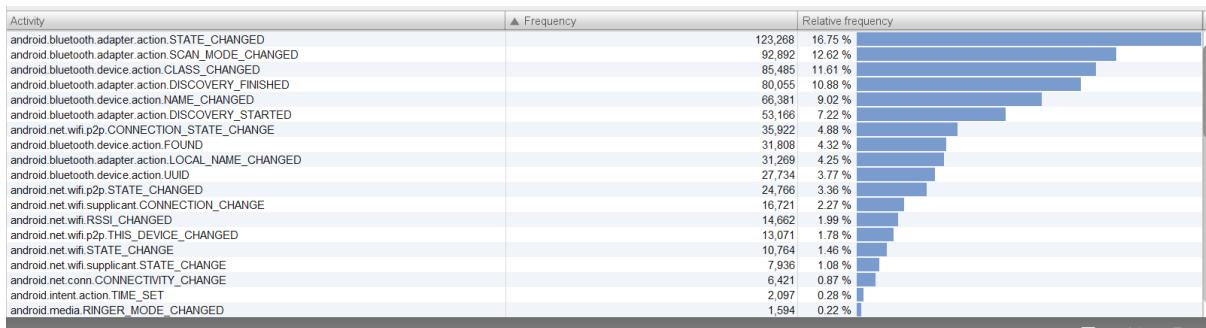
Numero di case, tipo di attività, risorse ed eventi



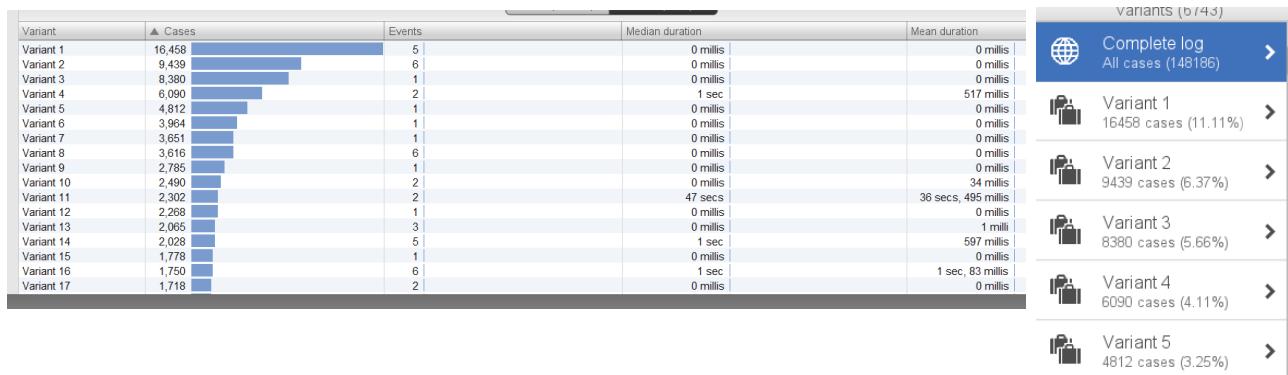
Il dataset in questione contiene 736094 eventi che vanno a formare 148186 case con un numero di varianti pari a 6743. Le tipologie di attività registrate risultano essere 52 ed il lasso di tempo preso in considerazione riguarda il mese di Giugno 2017.



Analizzando meglio la distribuzione degli eventi, si può notare come non sia per niente uniforme in quanto si hanno molti picchi e soprattutto molte zone senza alcuna rilevazione. Questo è dovuto al fatto che, non essendo un dataset di processi, esso cattura l'utilizzo dei dispositivi riguardanti le diverse risorse, e quindi probabilmente le zone in cui le registrazioni sono nulle, sono dovute al fatto che spesso i soggetti spegnevano il dispositivo impedendo la cattura degli eventi. Questa caratteristica sarà confermata dalle successive analisi ed è perfettamente in linea con il progetto in quanto i volontari avevano la piena possibilità di utilizzare il dispositivo a loro piacimento.



Come si può notare dalla foto precedente, vi è un forte squilibrio tra le attività in quanto già le top 10 attività caratterizzano il 91,13% degli eventi. Le più frequenti sono ovviamente quelle relative al Bluetooth ed al wifi, che sono rispettivamente il 80,44% e 16,82%. Dalle analisi successive si è scoperto che nei soggetti interessati, l'utilizzo del dispositivo Bluetooth veniva fatto in maniera massiva, anche perché il protocollo per la gestione del Bluetooth risulta essere molto dispendioso.



Si nota che grazie al raggiungimento della versione 2.2 del dataset, finalmente si ottiene un significativo miglioramento del numero delle varianti. Come si può notare dalle immagini, vi è un numero molto ristretto di varianti che vanno a caratterizzare quasi tutto il dataset. Tali varianti presentano una lunghezza che varia da 1 a 6 eventi e la loro durata può variare dai 0 millisecondi a 47 secondi. Si prende in esempio le due varianti più comuni:

Activity	Resource	Date	Time	info
1 android.bluetooth.device.action.CLASS_CHANGED	ec3b3592f11	01.06.2017	00:01:20	android.bluetooth.device.extra.CLASS: 340408android.bluetooth.device.extra.DEVICE: A0:14:3D:14:D0:18
2 android.bluetooth.device.action.CLASS_CHANGED	ec3b3592f11	01.06.2017	00:01:20	android.bluetooth.device.extra.CLASS: 1f0000android.bluetooth.device.extra.DEVICE: 04:C2:3E
3 android.bluetooth.device.action.NAME_CHANGED	ec3b3592f11	01.06.2017	00:01:20	android.bluetooth.device.extra.DEVICE: 04:C2:3E:BD:37:03android.bluetooth.device.extra.NA
4 android.bluetooth.device.action.NAME_CHANGED	ec3b3592f11	01.06.2017	00:01:20	android.bluetooth.device.extra.DEVICE: A0:14:3D:14:D0:18android.bluetooth.device.extra.NA
5 android.bluetooth.device.action.UUID	ec3b3592f11	01.06.2017	00:01:20	android.bluetooth.device.extra.DEVICE: A0:14:3D:14:D0:18android.bluetooth.device.extra.UU

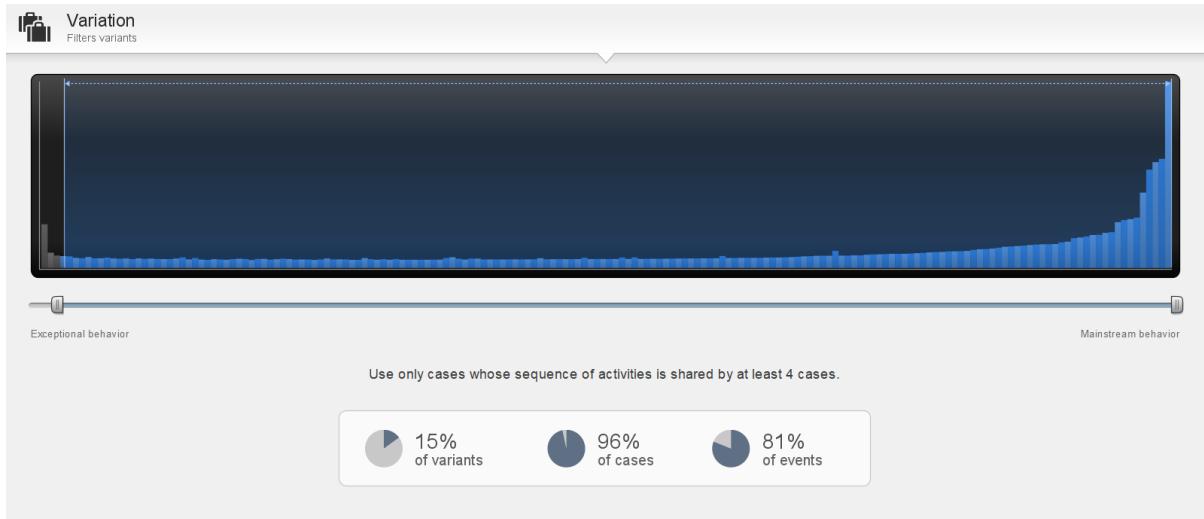
Activity	Resource	Date	Time	info
1 android.net.wifi.p2p.CONNECTION_STATE_CHANGE	68a5ffab20	01.06.2017	00:00:00	connectedDevIntAddress:NullnetworkInfo: NetworkInfo: type: WIFI_P2P[], state: DISCONNECTED
2 android.net.wifi.p2p.CONNECTION_STATE_CHANGE	68a5ffab20	01.06.2017	00:00:00	connectedDevIntAddress:NullnetworkInfo: NetworkInfo: type: WIFI_P2P[], state: DISCONNECTED
3 android.net.wifi.p2p.CONNECTION_STATE_CHANGE	68a5ffab20	01.06.2017	00:00:00	connectedDevIntAddress:NullnetworkInfo: NetworkInfo: type: WIFI_P2P[], state: DISCONNECTED
4 android.net.wifi.p2p.STATE_CHANGED	68a5ffab20	01.06.2017	00:00:00	wifi_p2p_state: 2n
5 android.net.wifi.p2p.STATE_CHANGED	68a5ffab20	01.06.2017	00:00:00	wifi_p2p_state: 1n
6 android.net.wifi.p2p.THIS_DEVICE_CHANGED	68a5ffab20	01.06.2017	00:00:00	wifiP2pDevice: Device: Galaxy S5n deviceAddress: c2:bd:d1:cc:8c:c2n primary type: 10-0050

Per approfondire il significato delle azioni: <https://developer.android.com/reference/kotlin/classes>

La prima variante gestisce il processo di aggiornamento dei dispositivi Bluetooth in quanto viene registrato il cambiamento sia della classe (CLASS_CHANGED) che del nome (NAME_CHANGED) del dispositivo remoto, con successiva generazione di un broadcast intent per la propagazione dello UUID (Universally Unique Identifier).

La seconda variante gestisce l'aggiornamento dello stato del wifi (CONNECTION_STATE_CHANGE), notifica al dispositivo l'abilitazione o meno del wifi (STATE_CHANGED) e successivamente si registra un aggiornamento delle proprietà della componente che gestisce il wifi (THIS_DEVICE_CHANGED).

Una caratteristica di questa nuova versione è di avere infatti tantissime varianti, la stragrande maggioranza delle quali non risulta essere significativa a livello di frequenze, in quanto spesso possono essere registrati dei comportamenti spuri che in questi ambiti vengono visti come varianti. Questa caratteristica è ben visibile se si applica il filtro delle varianti su Disco:



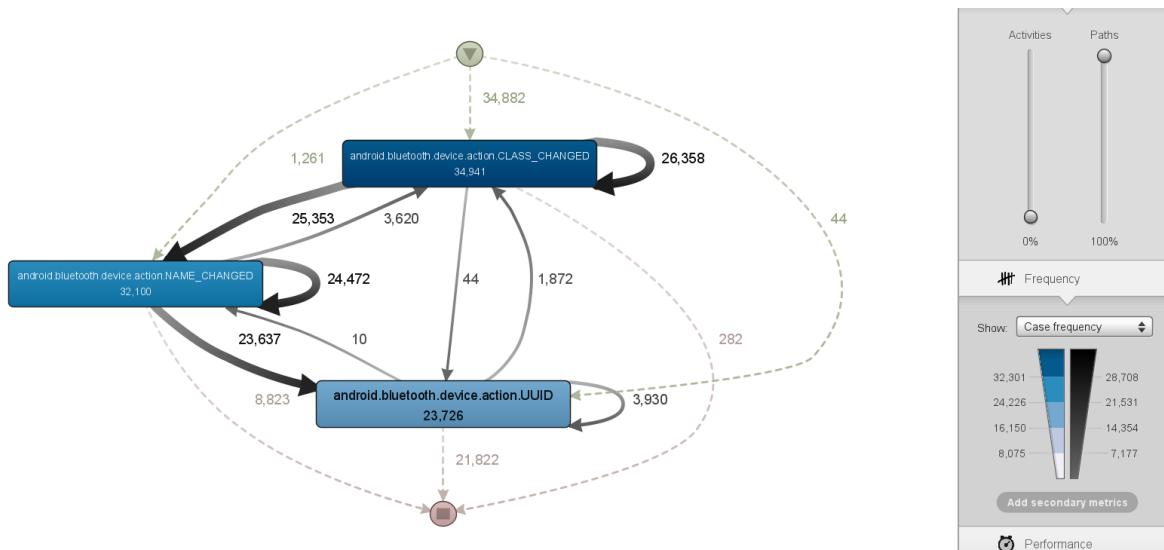
Andando infatti ad eliminare il 4% dei case, otteniamo una riduzione dell'85% delle varianti. Nell'analisi è stato applicato questo filtro proprio per andare ad eliminare le tracce spurie che potevano andare a compromettere la fase di Process Discovery.

All resources (4)		
Resource	Frequency	Relative frequency
ec3b3592f1	420,335	57.1 %
0a50e09262	221,531	30.1 %
68a5ffab20	89,479	12.16 %
e0bf6beeb3	4,749	0.65 %

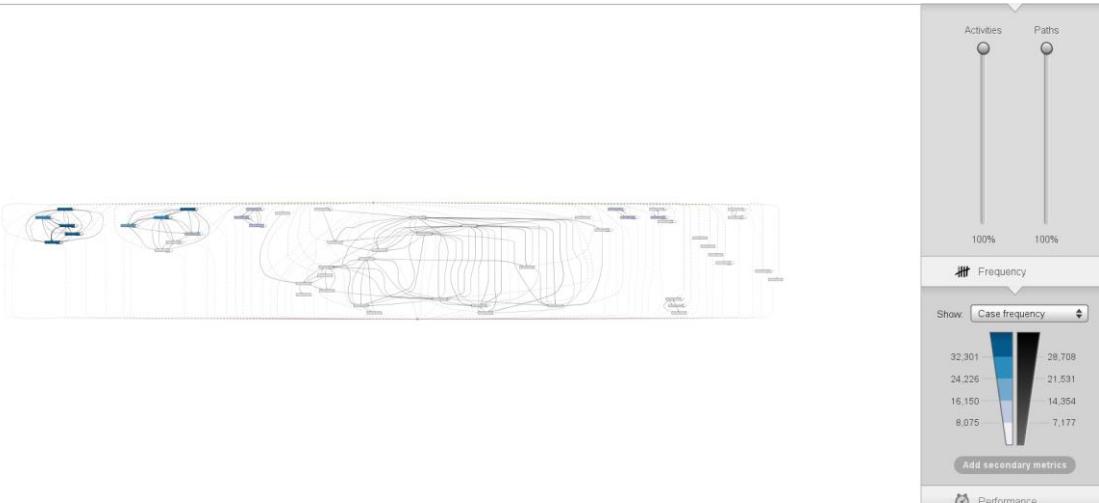
Le risorse presenti nel dataset sono le seguenti. È possibile notare come le prime due risorse sono di gran lunga più significative rispetto alle ultime due, come viene spiegato ed analizzato nel capitolo relativo all'analisi delle risorse.

Mappa generale del processo

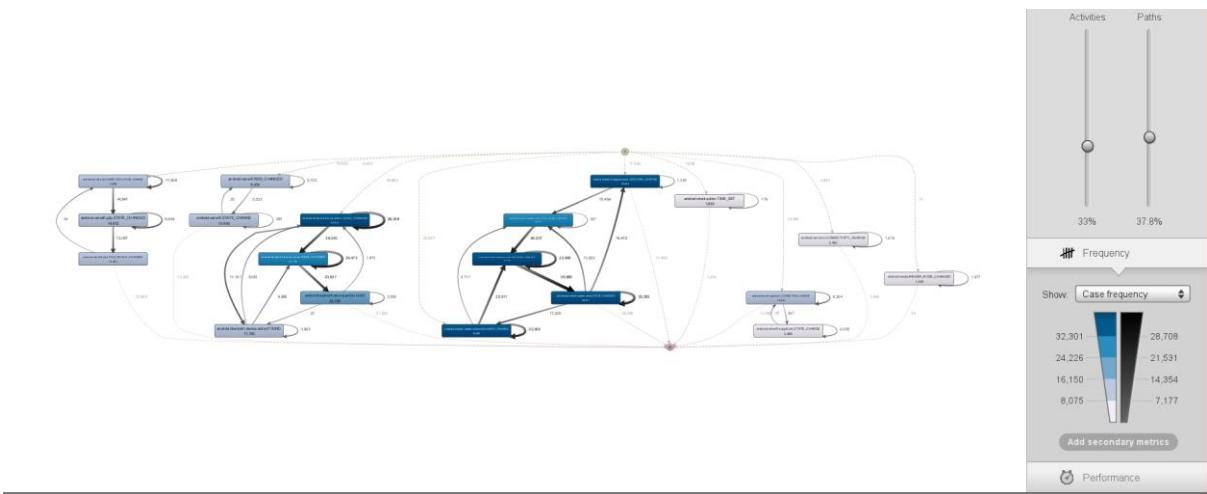
Utilizzando il software Disco e visualizzando il minor numero di activity con tutti i path possibili, il risultato ottenuto è il seguente:



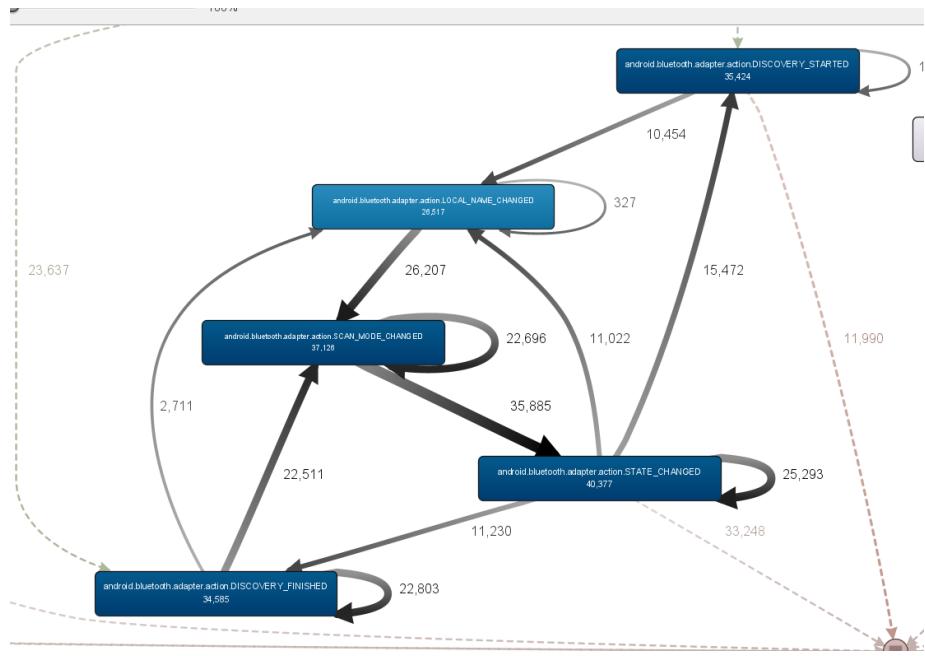
Da qui si può notare come il path più frequente è proprio quello relativo alla prima variante analizzata e come ci si aspettava (vista la natura del dataset). Esso modella dei micro-servizi che probabilmente sono codificati all'interno dei dispositivi secondo una logica a grafo. Questa caratteristica la si può notare dal fatto che i path rappresentati nella precedente figura danno la possibilità (almeno in questo caso) di generare una qualsiasi tipo di sequenza di questi tre eventi, in quanto vi sono collegamenti da ogni evento verso ogni altro evento, e anche su sé stesso. Il fatto che ci sia un numero più elevato di occorrenze di un path specifico sta a significare che probabilmente per determinate gestioni ci sia una sequenzialità intrinseca degli eventi. La presenza di tutte le altre tipologie di path ci dà comunque conferma che ogni evento può essere fatto indipendentemente dagli altri, probabilmente per gestire altre tipologie di servizi. Come si è visto nella prima variante, la propagazione dello UUID viene fatta a valle dell'aggiornamento dello stato e nome del dispositivo remoto, ma nulla vieta che venga fatta in altre situazioni.



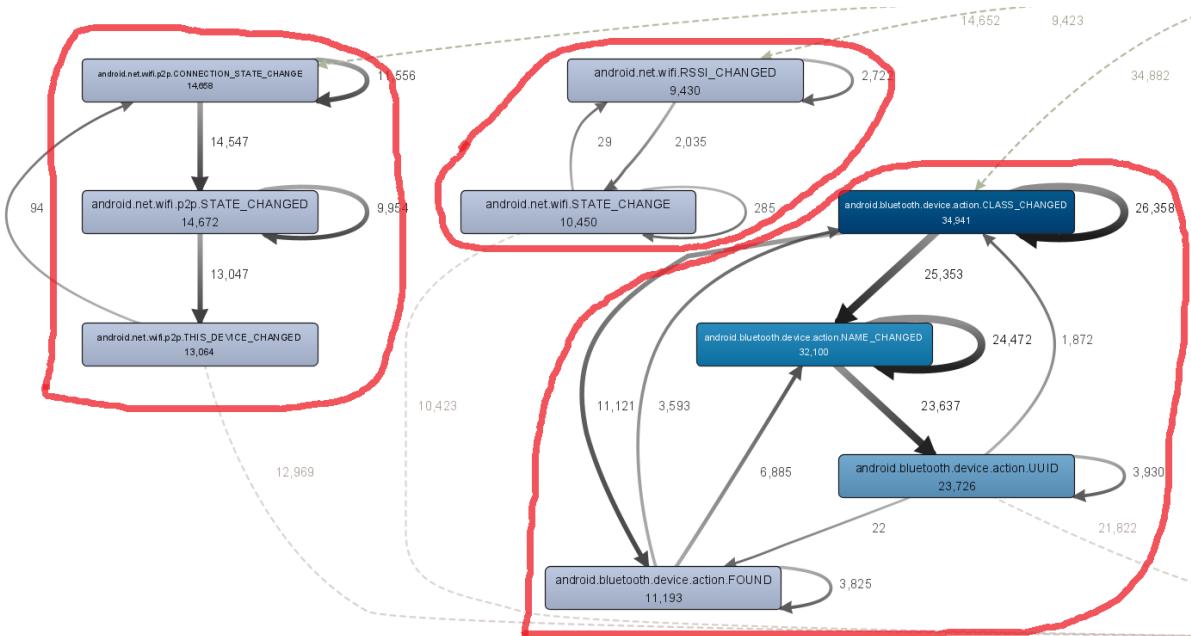
Come si può notare in questo caso, andando a settare entrambi i parametri al 100%, si hanno ulteriori conferme su quanto detto precedentemente. Il modello di processo che ne deriva, infatti, è molto ampio e sembrerebbe essere diviso in blocchi che vanno a codificare micro-servizi più o meno complessi e più o meno frequenti.



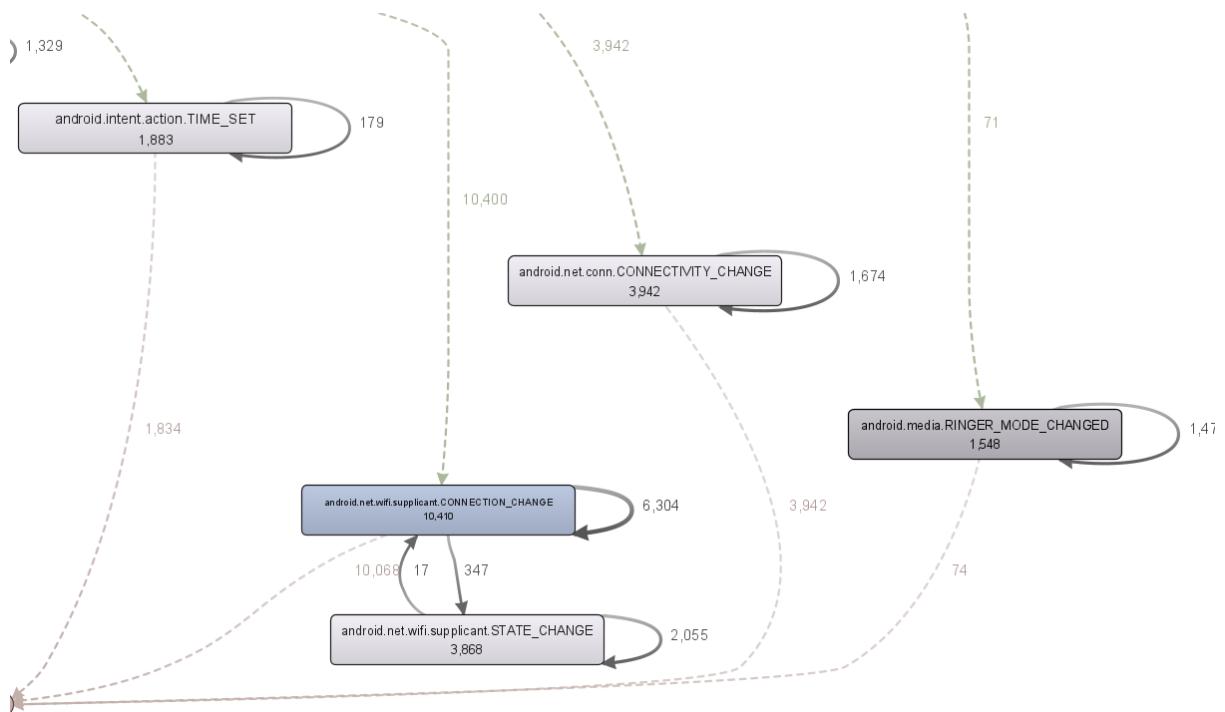
Andando a variare un po' i parametri, si riesce ad ottenere una visione un po' più chiara e più di alto livello dei servizi che vengono codificati all'interno del dispositivo. In seguito si analizzano più da vicino i risultati ottenuti:



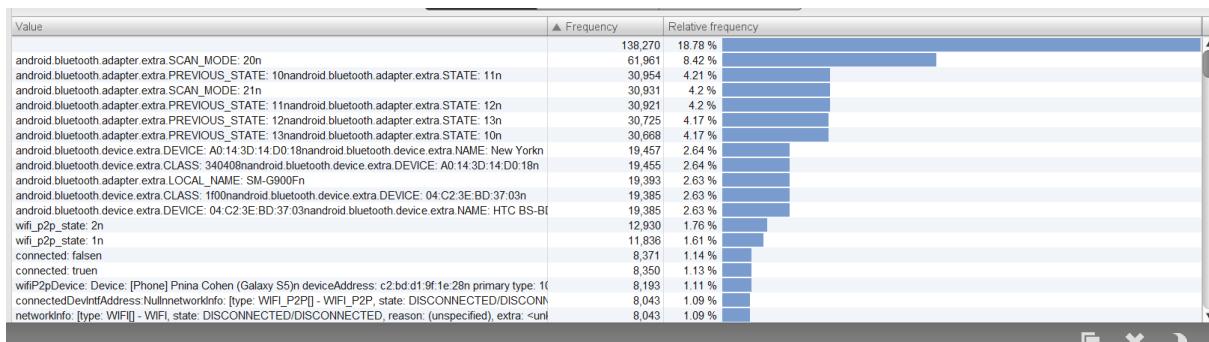
Questo sotto grafo codifica il processo più frequente, che in questo caso gestisce una scansione Bluetooth con relativo aggiornamento dello stato e del nome dei dispositivi remoti. Come si può vedere dai path, si ha una sequenza frequente, ma anche in questo caso ci sono istanze che ci fanno capire come queste azioni possano essere fatte spesso in ciclo o in ordine diverso.



In questa immagine sono stati evidenziati altri tre casi interessanti, tra cui vi è anche una versione molto simile alla prima variante analizzata, che si conclude però con evento FOUND che notifica al dispositivo il fatto di aver scovato altri dispositivi compatibili.



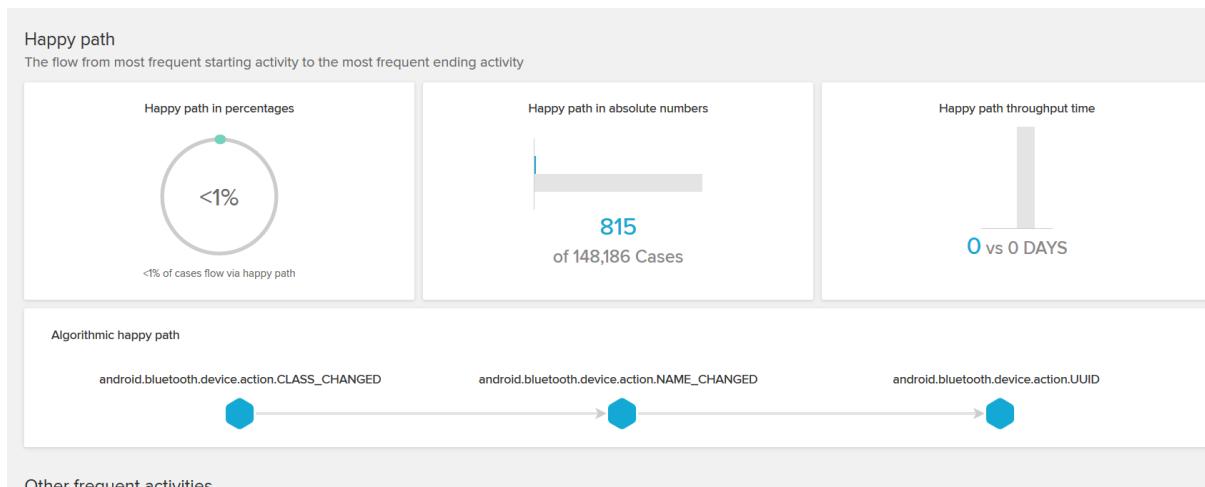
È possibile notare altre tipologie di gestioni molto più piccole e meno frequenti, spesso caratterizzate da un ciclo su se stesse, che di solito codificano cambi di stato e/o modalità interne al dispositivo.



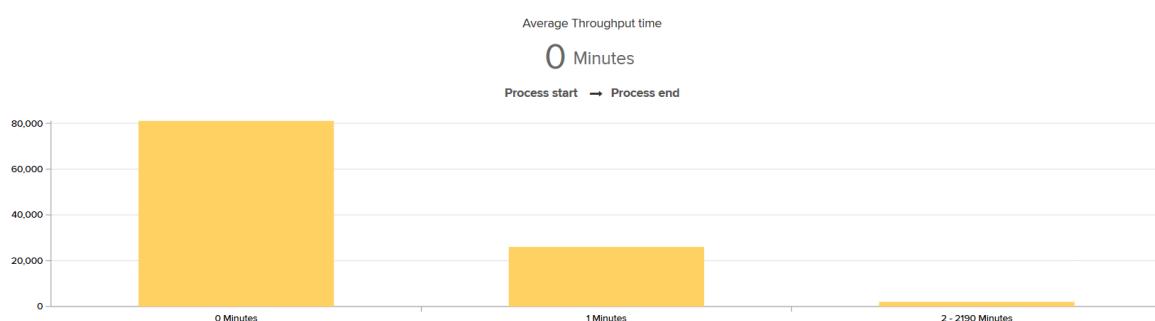
All'interno del dataset preso in considerazione, si è riusciti ad estrapolare informazioni aggiuntive dei processi che sono stati materializzati nella colonna "info". Com'è possibile vedere dall'immagine precedente, purtroppo la maggior parte degli eventi non possiedono questo campo, mentre gli altri danno delle informazioni aggiuntive che si possono racchiudere in:

- Potenza del segnale wifi
- Potenza del segnale Bluetooth
- Informazioni relative ad identificatori Bluetooth (friendly name e/o MAC)
- Informazioni relative ad identificatori wifi (SSID o device address)
- Informazioni di stato
- Errori

È stato utilizzato anche il software Celonis in diverse parti dello sviluppo del progetto, cercando di sfruttare le sue immense potenzialità:



Come ci conferma anche Celonis, l'happy path coincide proprio con la variante più frequente precedentemente analizzata.



Come ci si aspettava, anche Celonis dà conferma che la maggior parte degli eventi sono gestiti in meno di un minuto. Purtroppo, questo software non permette di specificare il tempo con una granularità al di sotto del minuto. Bisogna comunque notare che ci sono attività che raggiungono l'intero minuto, e altre che lo superano. A fronte di questo risultato particolare sulle ultime due categorie, si è scoperto che quelle relative alla prima tipologia sono relative alle scansioni, che spesso raggiungono il minuto di attività (soprattutto se non vano a buon fine), mentre l'ultima categoria può essere attribuita a dei cicli di scansione e tentativi di connessione, non andati a buon fine, che vengono collegati nello stesso caso.

Analisi con Dotted chart

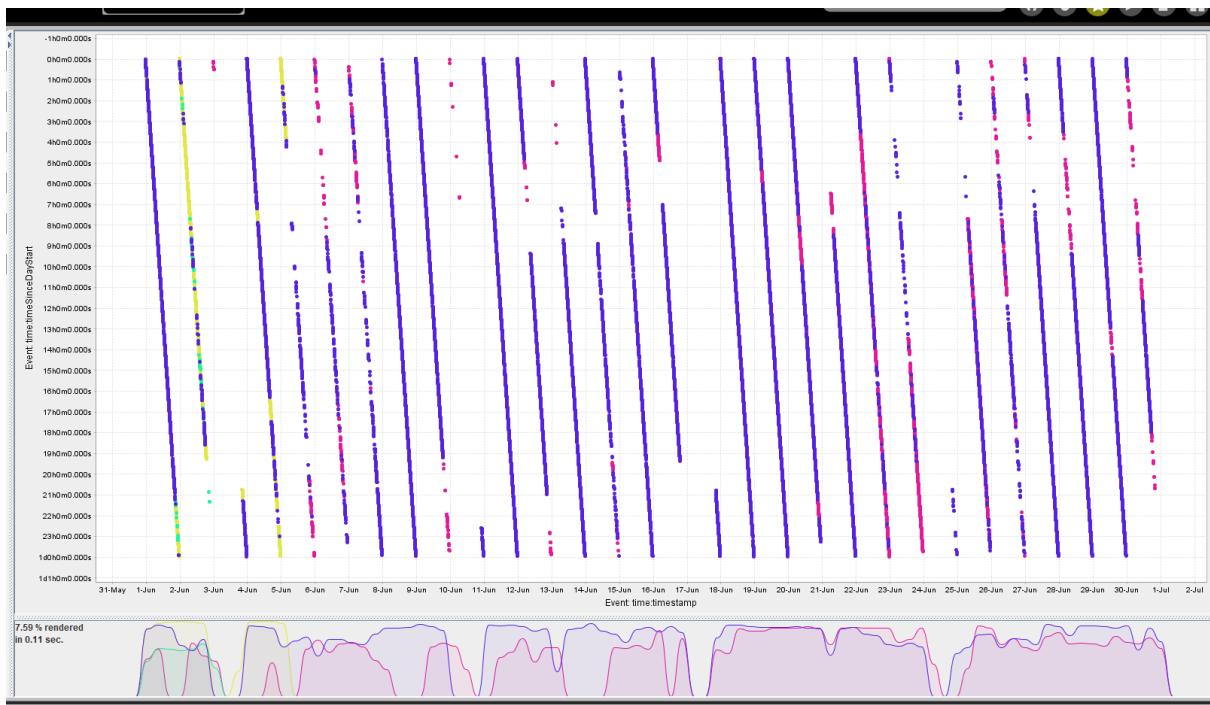
Si va ora ad analizzare il dataset sotto un'altra ottica, ovvero quella fornita dai dotted chart. I dotted chart sono uno strumento molto versatile e semplice per analizzare il dataset sotto una veste geometrica andando a visualizzare le informazioni come un insieme di punti sul piano cartesiano. Inoltre, vi è una notevole libertà sia per quanto riguarda le informazioni da visualizzare sugli assi, sia per quanto riguarda il significato da dare alla grandezza ed al colore dei punti. In seguito verranno riportate diverse analisi dalle quali si otterranno informazioni per lo più comportamentali per quanto concerne la sezione di dataset scelto.



Qui è possibile visualizzare la distribuzione dei vari eventi lungo tutto l'asse temporale interessato. Si nota che, come previsto, alcuni eventi sono molto più frequenti rispetto ad altri e risultano delle assenze di registrazione che spesso coincidono con tutte le attività durante alcune giornate, dove probabilmente i volontari hanno spento il dispositivo. Questa cadenza è abbastanza precisa, e questo ci può far pensare anche ad un'organizzazione o a delle direttive date dai responsabili del progetto Sherlock.



Da questa analisi è possibile andare a vedere qual è la generazione dei case (e quindi dei processi) lungo l'arco temporale, con il colore che caratterizza le varie risorse. Dal grafico si può notare che due risorse (giallo e verde) su quattro hanno partecipato solo all'inizio della rilevazione, mentre quella fucsia ha contribuito in maniera discontinua, a differenza della risorsa blu che ha contribuito in maniera decisiva durante l'intero arco del mese.



In quest'altro grafico è riportata la distribuzione degli eventi nell'arco di ogni giornata e da quale risorsa vengono svolti (grazie all'utilizzo dei colori). È possibile notare che ci sono alcune giornate con pochi eventi, come per il 4 giugno, e altre giornate molto ricche, come per il 5 giugno. Dal grafico sembrerebbe inoltre che alcune giornate siano occupate interamente da un'unica risorsa (per esempio la risorsa blu), ma analizzando più accuratamente il grafico presente nella parte inferiore dell'immagine è possibile notare che in questi casi sono comunque presenti anche altre risorse (come la fucsia) ma in maniera meno rilevante.

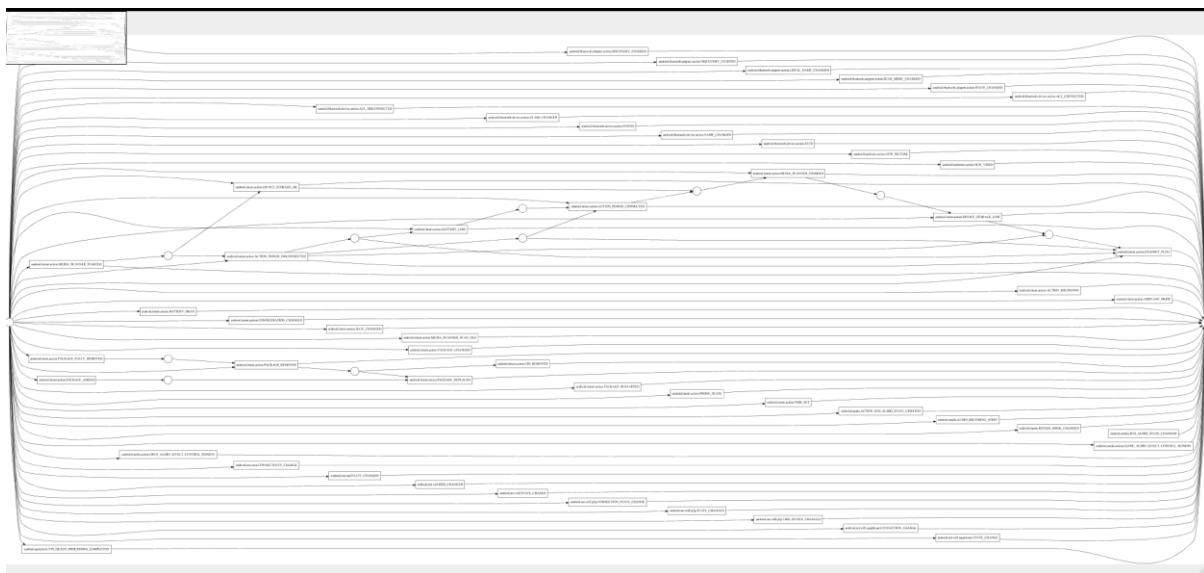
PROCESS DISCOVERY

L'attività di Process Discovery è uno dei task principali del Process Mining. L'obiettivo di questo task è modellare il comportamento del processo. In base all'algoritmo che si sceglie di applicare si ottengono modelli con qualità diverse.

A differenza delle analisi di Process Discovery viste da altri progetti, in questo caso l'attenzione è stata posta nello scovare il miglior algoritmo di mining per cercare di estrapolare il comportamento corretto e più frequente all'interno del log. Essendo il dataset molto particolare, spesso le attività vengono riportate in un preciso punto all'interno di un processo, ma non è scorretto trovarle con altri ordini, o trovare frequenti ripetizioni. A dispetto dell'approccio classico visto durante il corso, in questa parte verranno quindi fatte molte meno considerazioni riguardanti la congruenza degli eventi (per esempio accettare un ordine quando non si è ancora fatto il check sulla disponibilità della merce in magazzino) in quanto quest'accortezza probabilmente non ha senso nel dataset di riferimento. L'obiettivo è infatti scovare delle anomalie, ma non è detto che tali anomalie corrispondano necessariamente ad una deviazione del processo, in quanto potrebbero anche essere rappresentate da un processo codificato in maniera corretta, che però fatto in un determinato contesto risulta essere un'anomalia. Questo perché vi sono due concetti di anomalia:

- La prima anomalia che si vuole catturare è l'utilizzo abusivo di alcune componenti. Un possibile attacco per esempio sono scansioni della rete malevole. Questa anomalia è individuabile andando ad effettuare uno studio delle frequenze di alcuni processi che se tramite l'analisi risultano avere dei rate e dei tempi eccessivi, potrebbero essere segno di attacchi.
 - La seconda tipologia corrisponde alle varie deviazioni dei comportamenti frequenti. Queste sono individuabili con un approccio simile alla Conformance Checking e l'analisi delle deviazioni.

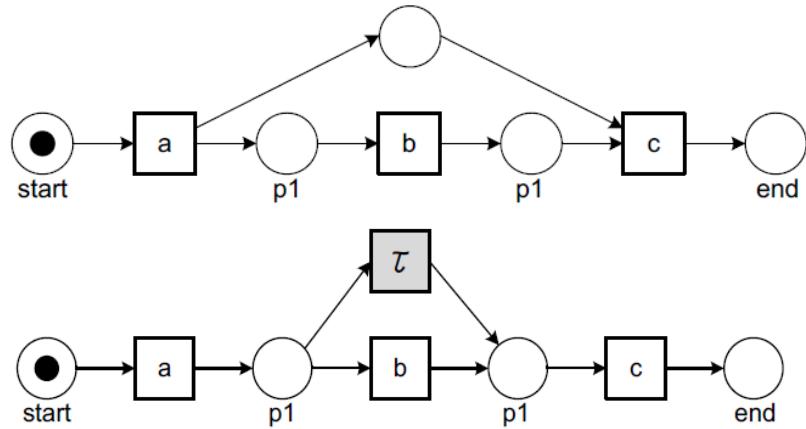
Queste due caratteristiche portano alle proprietà che si vogliono ottenere nel modello di processo. Proprio per questo si è deciso di generare il miglior modello di processo possibile che riuscisse a codificare in maniera corretta i micro-servizi più frequenti e che non abbia una precisione troppo elevata, in maniera tale da non catturare tutte le possibili varianti. Per prima cosa si è deciso di utilizzare sul dataset non filtrato l'algoritmo dell'**alpha miner** per dare un'idea della complessità:



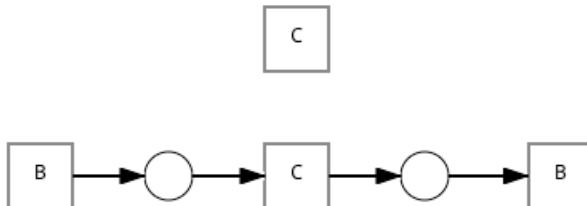
Come prevedibile il modello risulta essere enumerativo ovvero un modello che prova a codificare tutte le istanze di processo che trova all'intero del dataset, creando un path per quasi ogni tipologia di processo.

L'alpha miner non sarà l'algoritmo scelto per modellare il processo in quanto i suoi problemi intrinseci, per l'obiettivo del progetto, sono troppo rilevanti. Di seguito si sintetizzano i principali problemi relativi all'algoritmo alpha miner:

- 1) **Representational Bias** relativa ai task duplicati: problematica dovuta alle caratteristiche imposte dal linguaggio di rappresentazione scelto (in questo caso petri nets). In alcuni casi, tale problema si può risolvere tramite l'utilizzo delle silent transitions che permettono di saltare alcuni task.

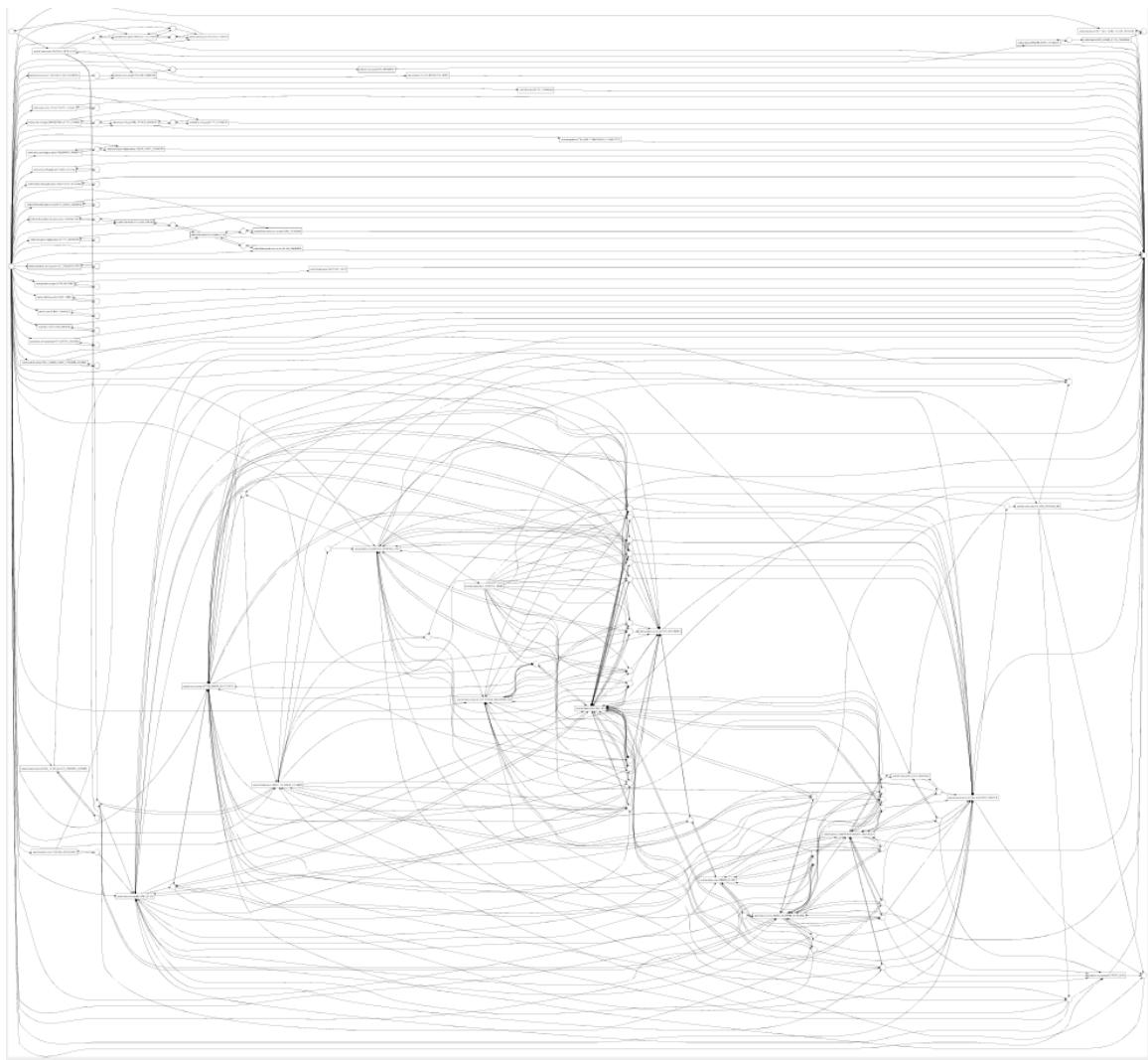


- 2) **Loop:** è noto che l'Alpha Miner non riesce a catturare i cicli.



- 3) **Robustezza:** è noto che l'Alpha Miner lavora considerando esclusivamente le relazioni tra le varie tracce senza valutarne la frequenza all'interno del log. Di conseguenza qualsiasi rumore o outlier influenza la struttura della rete.

Di seguito viene riportata la stessa analisi utilizzando un'evoluzione dell'alpha miner, ovvero l'alpha miner++. Anche in questo caso non sarà questo l'algoritmo scelto per estrapolare il modello di processo in quanto, pur riuscendo a risolvere alcuni dei limiti sopra citati, non può risolvere, per sua natura, l'assenza di significato delle frequenze che in questo ambito risulta invece un aspetto cardine:



Com'è possibile vedere dal grafico incomprensibile, sicuramente la variante ++ riesce a catturare "meglio" le varie relazioni e lo si può notare dalla complessità del grafico risultante, che probabilmente prova a codificare letteralmente ogni possibile path.

Successivamente ci si è spostati verso degli algoritmi di mining più compatibili con dataset reali, come può essere l'**Heuristic Miner** e l'**Inductive Miner**. Verranno di seguito riportati i risultati dei due algoritmi, sempre ricordando che il tutto è applicato sul dataset non filtrato:

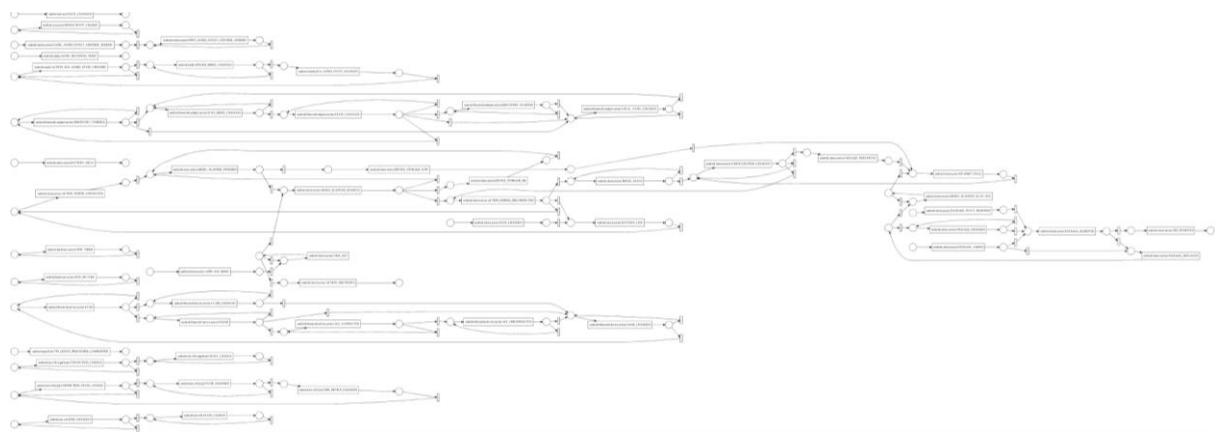


Figure 6: Modello generato da heuristic miner

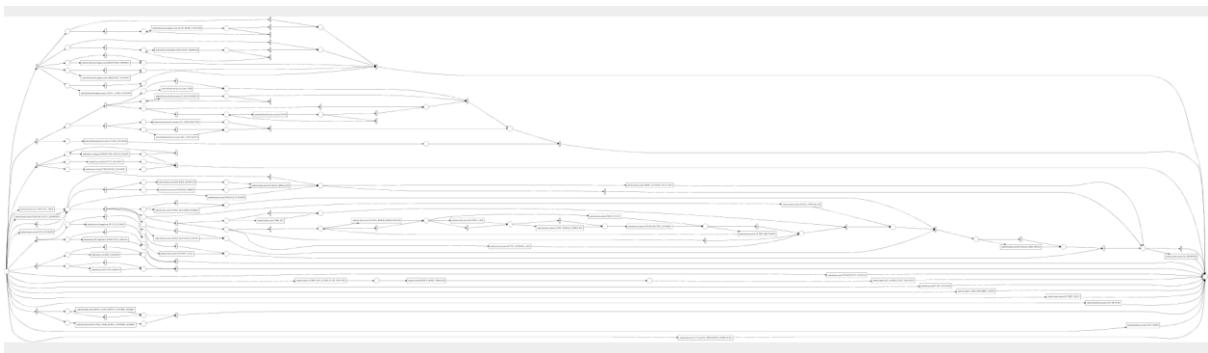
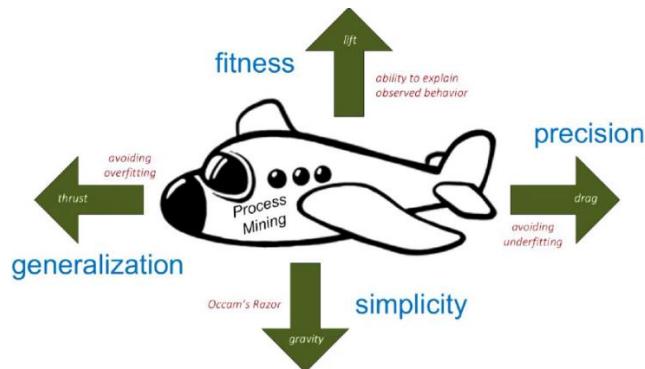


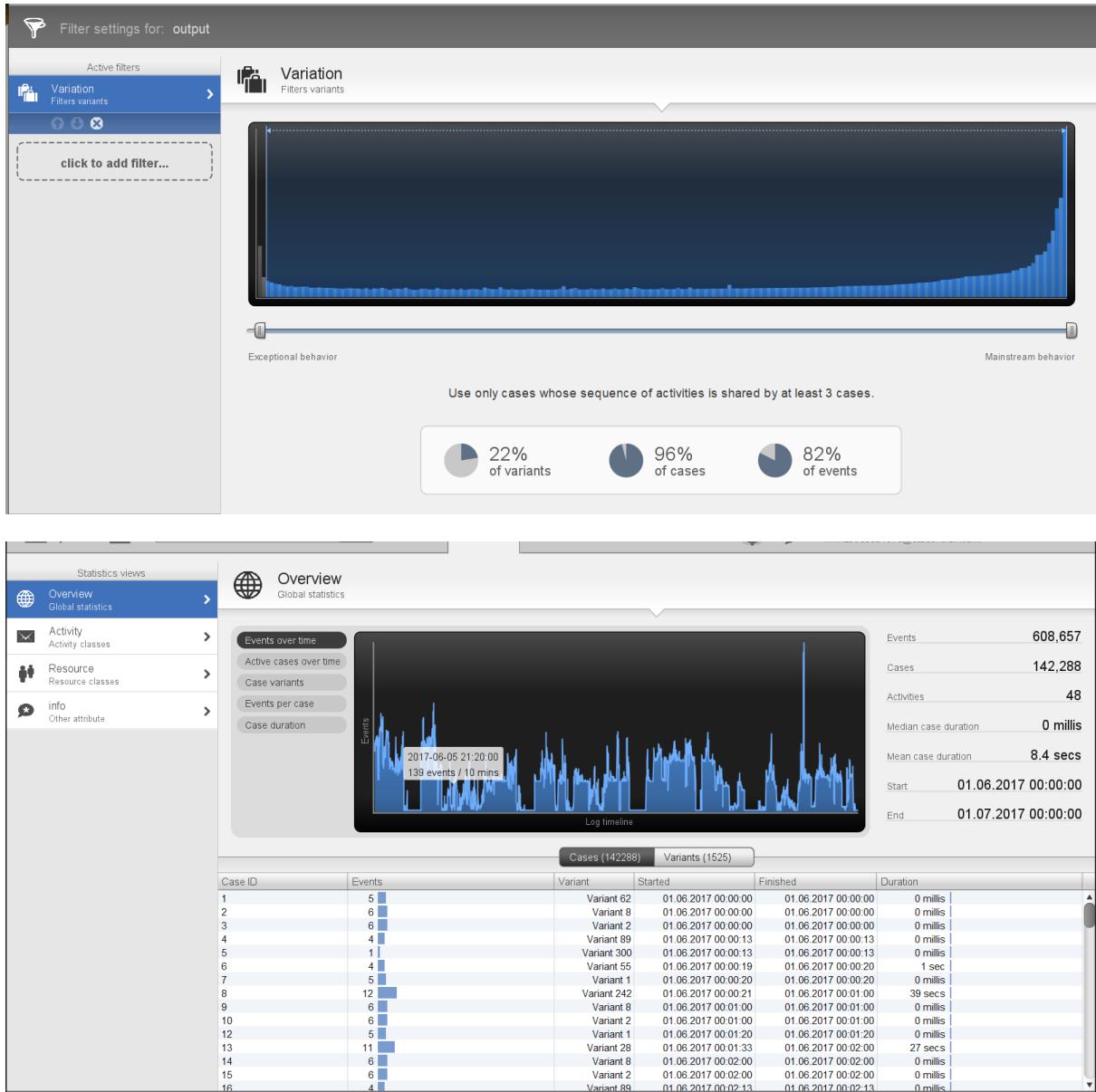
Figure 7: Modello generato da inductive miner

È possibile notare come da entrambi i risultati vi è sicuramente un miglioramento nella codifica dei processi, in quanto grazie all'introduzione ed al peso che viene dato anche alle frequenze, i due algoritmi (anche se in maniera diversa) riescono a catturare i micro-servizi che regolano il comportamento del dispositivo. Si è visto all'interno del corso come l'**Heuristic miner** sia un ottimo algoritmo per estrapolare modelli di processi da dataset reali, ma in questo caso il suo output risulta essere leggermente meno qualitativo rispetto all'**Inductive miner**. L'Heuristic miner infatti restituisce diverse sottoreti (sempre petri nets) che codificano più o meno bene i vari servizi, mentre l'Inductive miner riesce anche a correlare questi servizi fornendo una singola petri nets che risulta essere una work flow net secondo il tool **analyze Woflan**. Queste prime analisi sono state fatte per andare a vedere le varie caratteristiche degli algoritmi e il loro comportamento su un dataset non propriamente canonico.

Successivamente si è deciso di andare a pulire il dataset applicando dei filtri per cercare di ottenere un modello di processo che sia molto più generale e semplice. Si è visto come sia molto importante avere un equilibrio tra le quattro proprietà di un modello di processo:



Analizzando il dataset, una delle caratteristiche da smussare corrisponde all'elevato numero di varianti, pochissime delle quali risultano essere significative (a livello di frequenza). Si nota come nelle immagini di seguito, grazie all'utilizzo di Disco, si è potuto applicare un filtro sulle varianti che ha evidenziato lo squilibrio che c'è tra il numero delle varianti e il numero di eventi. Questo perché andando ad eliminare una parte relativamente piccola di eventi (18%) le varianti risultano diminuire dell'78% e questa caratteristica ha permesso di ripulire il dataset perdendo pochissime informazioni.



Come si può vedere dai risultati, il numero di varianti diminuisce nettamente e da qui si è ripartiti utilizzando ProM per andare ad estrarre il modello di processo, questa volta sul dataset ripulito. Di seguito vengono riportati i risultati, ancora una volta, della coppia di algoritmi Heuristic miner ed Inductive miner:

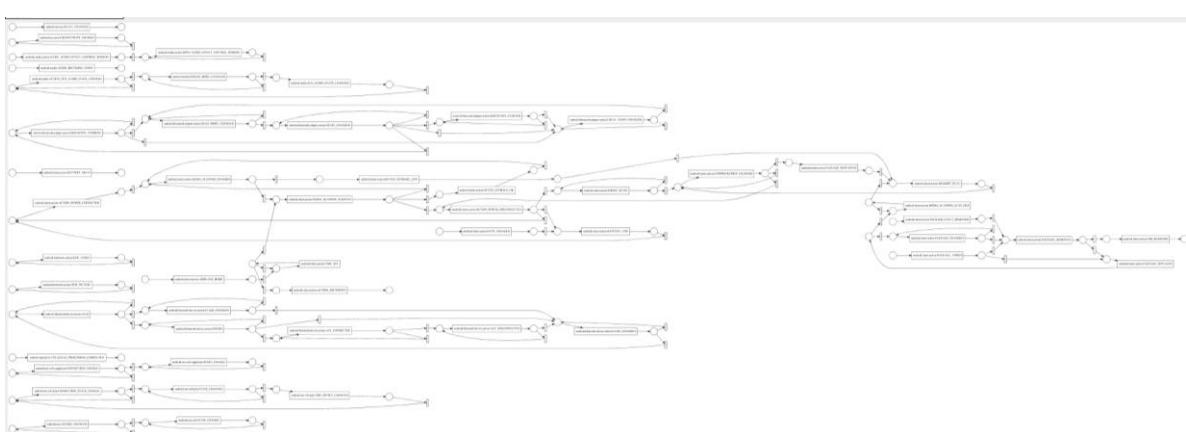


Figure 8: Modello generato da Heuristic miner

Grazie al filtro applicato al dataset, il modello risulta essere molto più comprensibile, ma mantiene le problematiche precedentemente trattate:

- Il modello generato in realtà consiste in un insieme di modelli di processo (petri nets), alcune delle quali sono anche collegate, e questa segmentazione dei processi risulta essere un problema per le analisi in quanto si dovrebbero effettuare delle modifiche per rendere questo modello una workflow net. Tali modifiche sono state effettuate, utilizzando il tool di ProM “**add artificial event**”, ma il risultato ottenuto ha indebolito di molto la qualità del modello.
- Anche riuscendo a codificare i micro-servizi, non riesce a cogliere alcuni eventi che almeno per la loro frequenza dovrebbero essere disposti in un determinato ordine (bluetooth.DISCOVERY_STARTED seguito da bluetooth.DISCOVERY_FINISHED) che invece risultano essere codificati in maniera non opportuna.
- Non riesce a collegare in sequenza eventi che invece dovrebbero essere collegati, quali la successione tra action.UUID e action.NAME_CHANGED

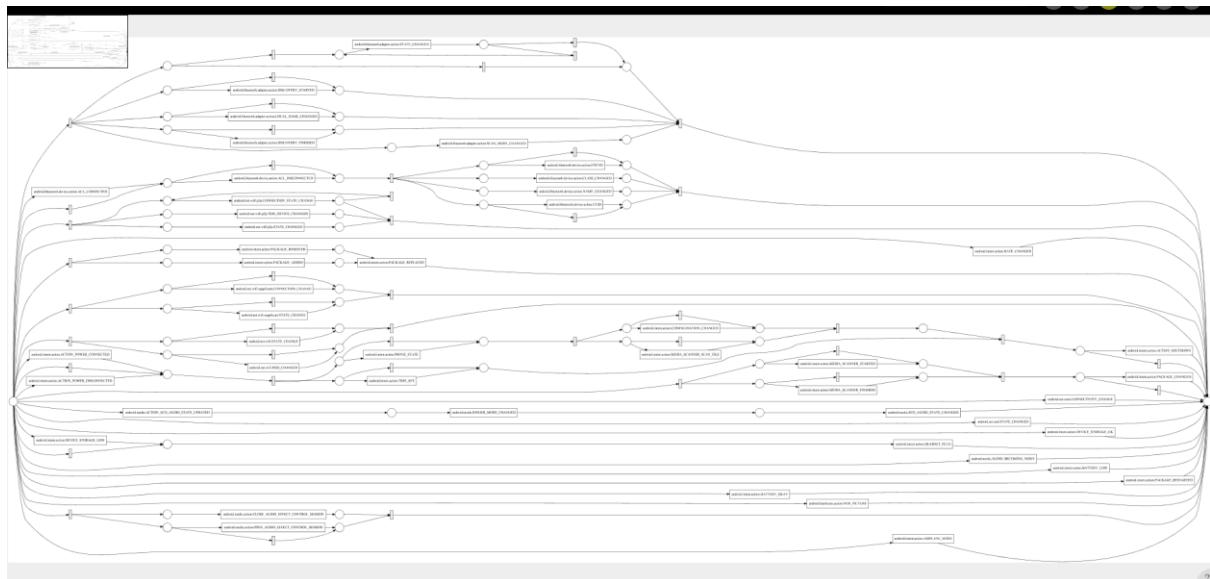


Figure 9: Modello generato da Inductive miner

Anche l'Inductive miner in questo caso non dà risultati sbalorditivi, anche se riporta meno problemi rispetto al precedente:

- Le sequenzialità di alcuni processi viene codificata in maniera non del tutto errata in quanto viene gestita come se fosse un parallelo o comunque con una scelta non deterministica. Questo perché, come si è visto nel dataset, una determinata azione può comparire in qualsiasi ordine, anche se data la maggior frequenza di alcuni processi rispetto ad altri, ci si sarebbe aspettati una maggiore precisione da questo punto di vista

Woflan Diagnosis on Net "40a61184-cfd9-4a35-ade5-0eb434a27d7a"

The net is a sound workflow net.

Un aspetto positivo del modello restituito dall'Inductive miner è la caratteristica di avere una workflow net sound, come ci viene confermato dall'applicazione del plugin di Woflan. Ne segue che la rete:

- Non contiene Dead Transition
- È safe (rete 1-boundend)
- Raggiungibilità dell'end place
- Option to complete (è sempre possibile raggiungere un marking dove solo l'end place contiene un token)

In questa parte però, la complessità della rete risulta essere ancora eccessiva ed è per questo che si è provato ad utilizzare le tecniche standard viste a lezione per andare a semplificare ulteriormente il modello di processo. Dopo un'analisi dei vari filtri presenti su ProM, la scelta finale è ricaduta sul “**filter using simple heuristic**”, usato spesso a lezione.

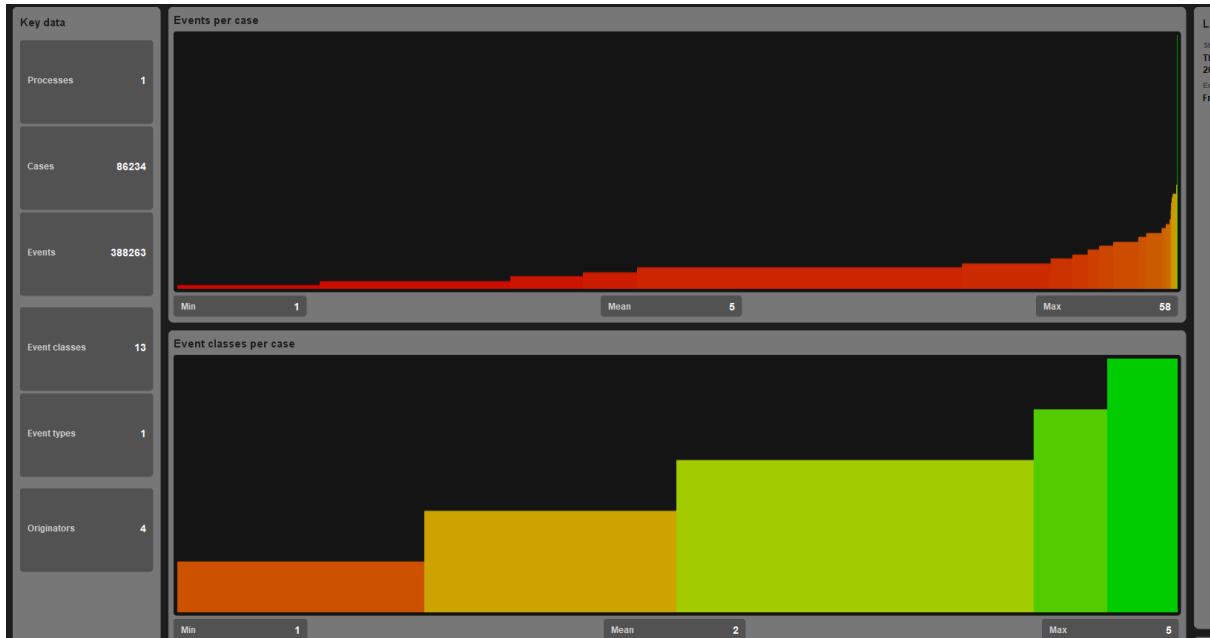


Figure 10: Dataset ottenuto dall'applicazione di “filter using simple heuristic” con parametri 95, 95, 90

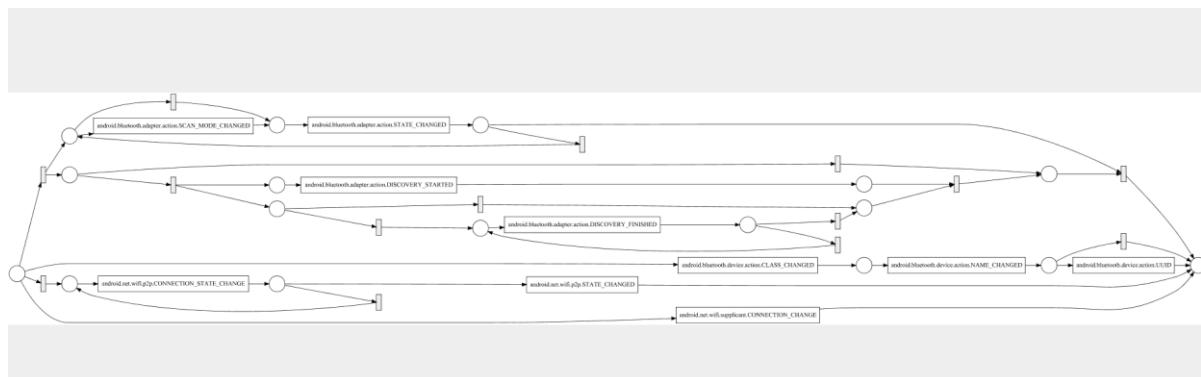
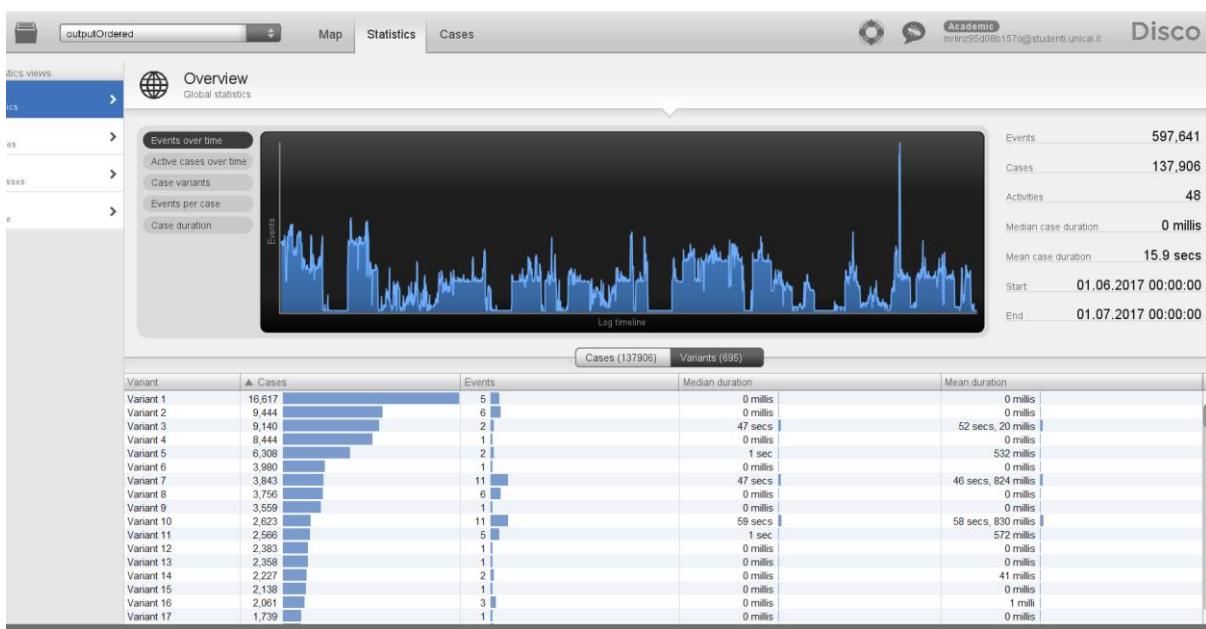
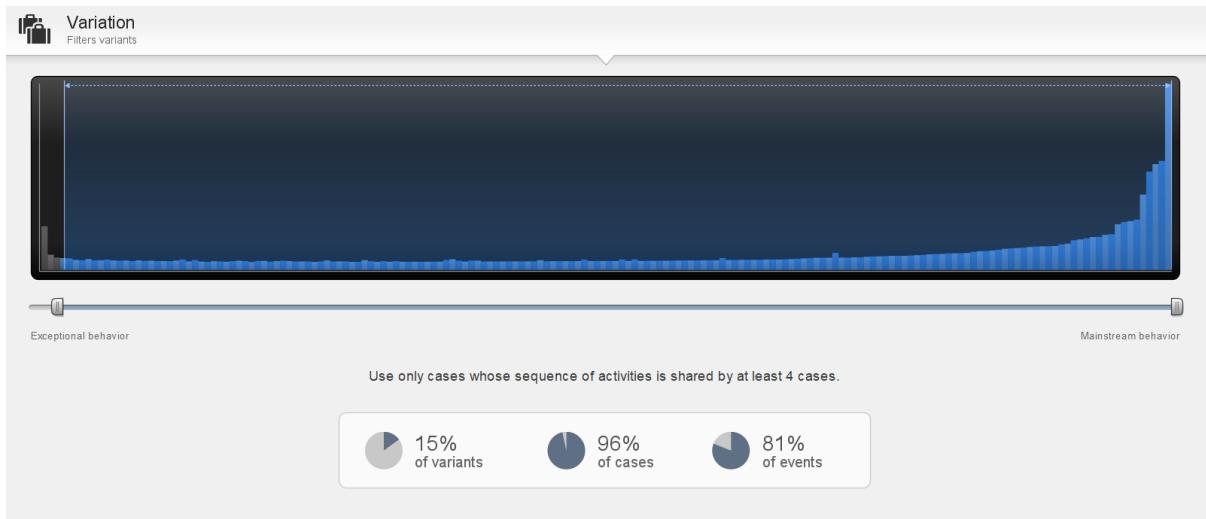


Figure 11: Modello generato da Inductive miner

Da questo filtro il risultato che si ottiene dall'inductive miner è troppo basilare in quanto probabilmente il doppio filtro utilizzato sia su disco che su ProM ha reso il dataset meno ricco di informazioni. Da qui si è provato a filtrare in maniera diversa, applicando in sequenza un filtro sulle varianti utilizzando Disco, e il “**filter with simple heuristic**” di ProM, andando però questa volta ad applicare un filtro leggermente più forte su Disco e molto più leggero su ProM. Si è infatti notato che il “**filter with simple heuristic**” plugin, se non settato in maniera opportuna, restituisce un dataset troppo povero dal punto di vista delle attività contenute in esso.



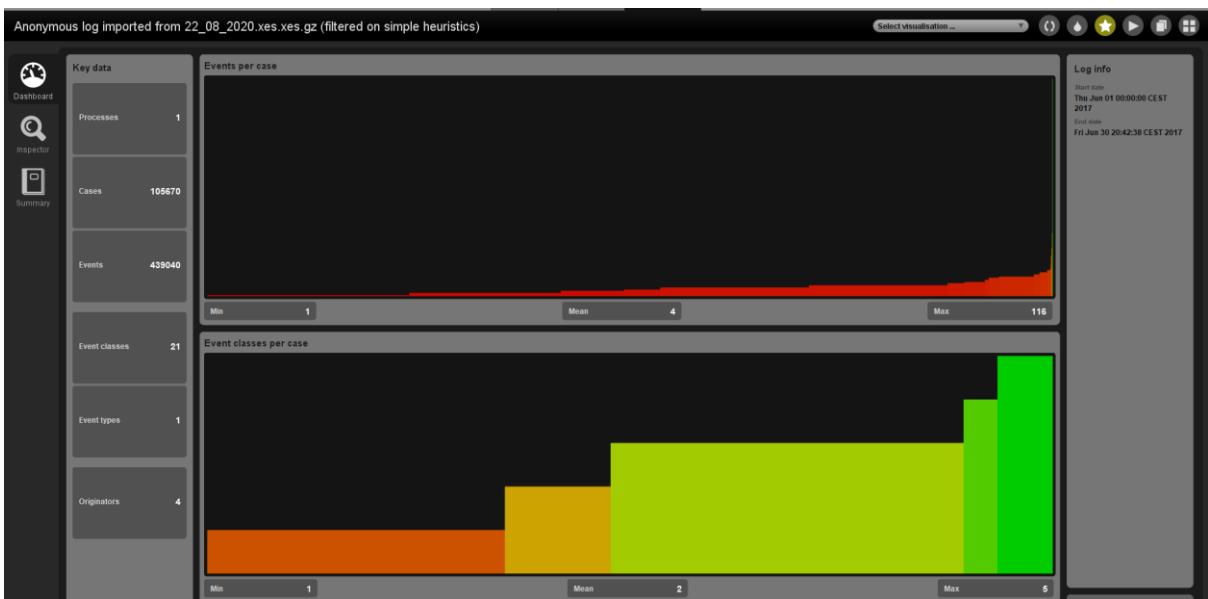
In maniera molto simile al filtro precedente, si è deciso questa volta di spingersi fino all'85% in meno di varianti, che vanno a causare una perdita del 19% degli eventi. Questo perché nell'analisi precedente ci si è accorti che con il 22% di varianti venivano codificati tantissimi processi con frequenza pari a 1. In questo caso è invece possibile vedere come il numero di varianti scenda ulteriormente fino a raggiungere le 695 varianti, che a fronte di tutti i test fatti, risulta essere un ottimo compromesso per le analisi successive.

Successivamente il dataset è stato filtrato utilizzando il solito plugin “**filter with simple heuristic**”. In questa fase c’è stato un impiego di tempo notevole che ha caratterizzato il tuning dei parametri forniti dal plugin. Dopo numerosi test la miglior configurazione è stata quella riportata nelle immagini seguenti:

The image consists of three side-by-side screenshots of a software application titled "Log Filter".

- Left Screenshot:** Titled "Start events", it shows a list of Android log entries starting with a green event. The list includes: android.bluetooth.adapter.action.DISCOVERY_FINISHED+complete, android.bluetooth.adapter.action.DISCOVERY_STARTED+complete, android.bluetooth.adapter.action.LOCAL_NAME_CHANGED+complete, android.bluetooth.adapter.action.SCAN_MODE_CHANGED+complete, android.bluetooth.adapter.action.STATE_CHANGED+complete, android.bluetooth.device.action.ACL_CONNECTED+complete, android.bluetooth.device.action.CLASS_CHANGED+complete, android.bluetooth.device.action.FOUND+complete, android.bluetooth.device.action.NAME_CHANGED+complete, android.bluetooth.device.action.UUID+complete, android.hardware.action.NEW_PICTURE+complete, android.intent.action.ACTION_POWER_CONNECTED+complete, android.intent.action.ACTION_POWER_DISCONNECTED+complete, android.intent.action.ACTION_SHUTDOWN+complete, android.intent.action.AIRPLANE_MODE+complete, and android.intent.action.BATTERY_LOW+complete.
- Middle Screenshot:** Titled "End events", it shows a list of Android log entries ending with a green event. The list includes: android.bluetooth.adapter.action.DISCOVERY_FINISHED+complete, android.bluetooth.adapter.action.DISCOVERY_STARTED+complete, android.bluetooth.adapter.action.LOCAL_NAME_CHANGED+complete, android.bluetooth.adapter.action.SCAN_MODE_CHANGED+complete, android.bluetooth.adapter.action.STATE_CHANGED+complete, android.bluetooth.device.action.FOUND+complete, android.bluetooth.device.action.NAME_CHANGED+complete, android.bluetooth.device.action.UUID+complete, android.intent.action.CONFIGURATION_CHANGED+complete, android.intent.action.HEADSET_PLUG+complete, android.intent.action.MEDIA_SCANNER_SCAN_FILE+complete, android.intent.action.PHONE_STATE+complete, android.intent.action.TIME_SET+complete, android.media.ACTION_SCO_AUDIO_STATE_UPDATED+complete, android.media.ACTION_SCO_AUDIO_STATE_CHANGED+complete, and android.net.conn.CONNECTIVITY_CHANGE+complete.
- Right Screenshot:** Titled "Event filter", it shows the same lists as the first two but with a different background color. It includes a "Select top percentage:" slider set to 99, a "Log name:" field containing "Anonymous log imported from 22_08_2020.xes.xes.gz (filtered on simple heuristics)", and buttons for "Cancel", "Previous", "Next", and "Finish".

In tutti e tre, la percentuale scelta è stata del 99% in quanto in questa fase si voleva applicare un filtro molto lieve, solo per eliminare alcune attività spuri che potevano essere rimaste da varianti poco frequenti. C'è inoltre stata una modifica nel primo e nel secondo step in quanto, dopo uno studio sulle primitive Android, sono stati scelti e selezionati solo i case che effettivamente potevano iniziare o finire un'azione (per esempio deselezionando DISCOVERY_FINISHED come start event, e deselezionando DISCOVERY_STARTED come finish event) per aumentare la precisione dei processi descritti.



Questo è il risultato ottenuto dal doppio filtraggio, che comunque rimane molto significativo a livello numerico. Ottenuto questo risultato c'è stata un ulteriore fase di tuning dei parametri per quanto riguarda l'Inductive miner che in questo caso ha visto interessato come parametro il **Noise Threshold**, con un range che varia da 0.10 al 0.30. Tali modelli sono stati presi in considerazione per la successiva parte di **Conformance Checking** in cui si sono andati ad analizzare le qualità sotto vari punti di vista.

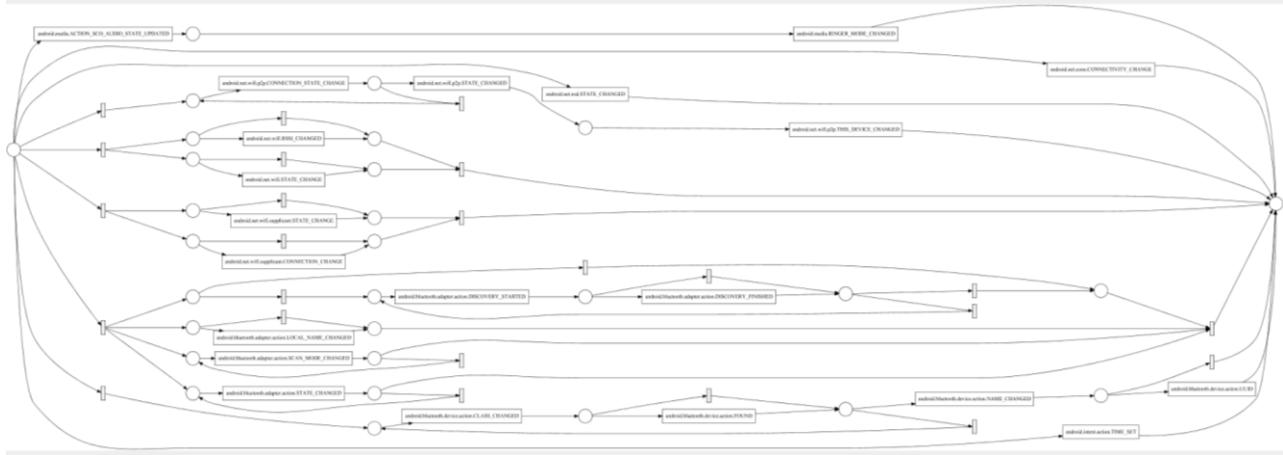


Figure 12: Modello generato da Inductive miner con N.T. = 0.20

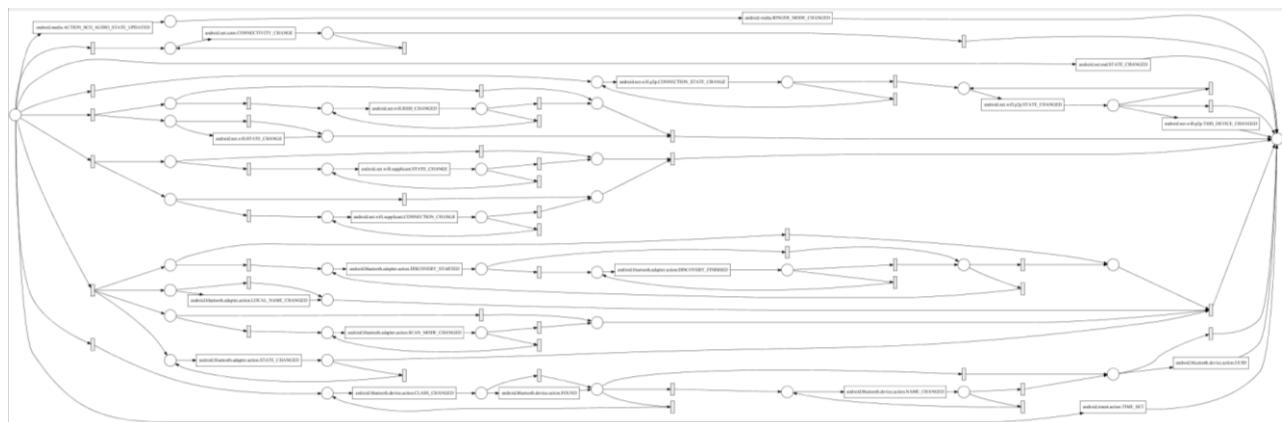


Figure 13: Modello generato da Inductive miner con N.T. = 0.10

I modelli ottenuti dall'Inductive miner da ora in poi sono da considerare sound workflow net in quanto sono stati tutti testati con il relativo plugin. L'unica caratteristica che cambia variando il parametro di Noise Threshold è che all'aumentare di tale parametro, gli archi che vengono codificati nel modello diminuiscono, e quindi si ottiene un modello più pulito e più semplice, mentre al diminuire di questo parametro il modello risulta essere più preciso ma più complesso, in quanto racchiude al suo interno più path. Questa correlazione verrà trattata più approfonditamente nella parte di Conformance Checking in cui grazie ai plugin forniti da ProM si riusciranno a dare delle metriche (generalization, fitness e precision) per valutare qualitativamente i modelli.

CONFORMANCE CHECKING

L'attività di conformance è uno dei tre task principali del Process Mining. Viene eseguito confrontando l'event log con il modello al fine di rilevare, individuare e spiegare le deviazioni dal modello e misurarne, mediante opportuni parametri, la gravità delle deviazioni. Per valutare la qualità del modello ottenuto esistono delle misure qualitative e quantitative spesso in opposizione tra loro. Si è scelto di utilizzare il plug-in di Conformance “**Replay a log on Petri Nets for Conformance Analysis**” ottenendo il seguente risultato:

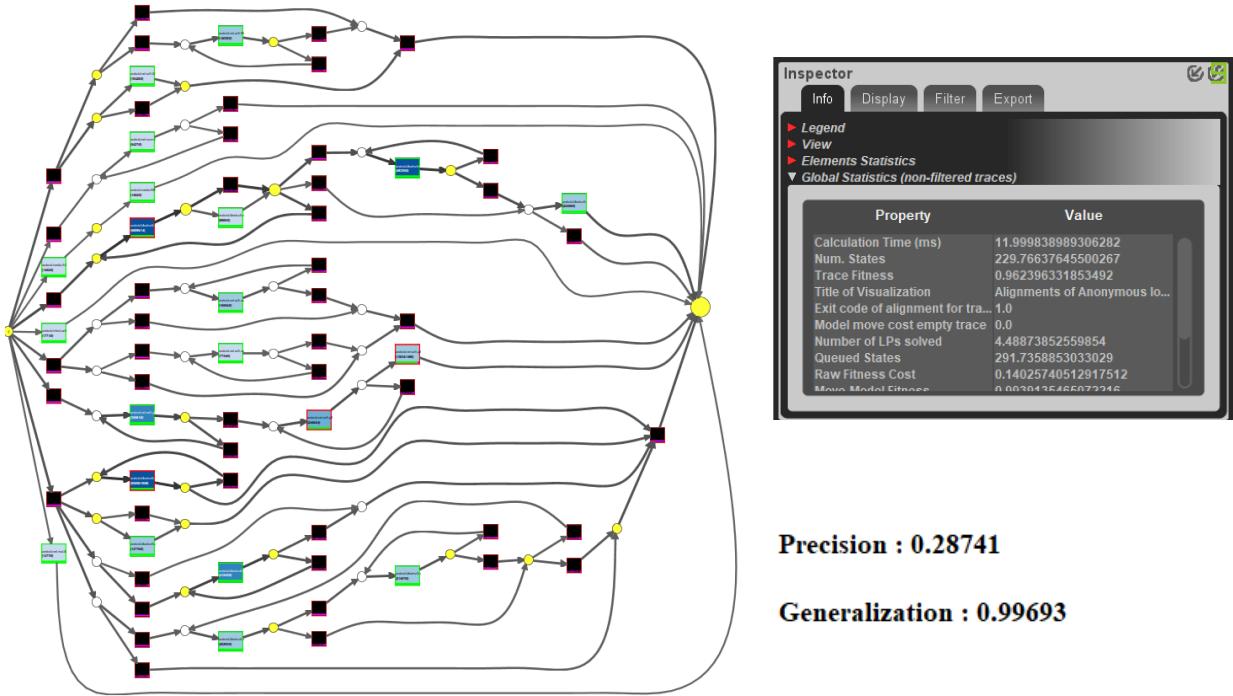
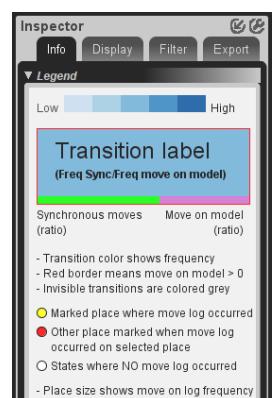


Figure 14: Risultato ottenuto utilizzando il modello restituito da Inductive miner con N.T. = 0.10

In questa parte dell'analisi l'attenzione è stata volta a trovare il giusto compromesso tra le varie metriche a disposizione. Di seguito verrà riportata una breve descrizione di quelle che sono le metriche e le procedure che sono state utilizzate per valutare la bontà delle soluzioni:

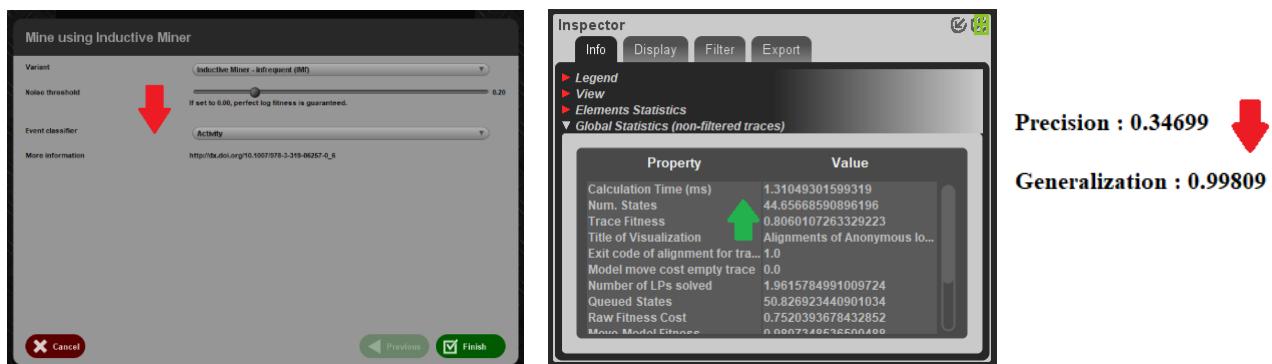
- **Alignment:** è una procedura del Process Mining che può essere utilizzata in varie fasi di analisi (operazioni di pre-elaborazione o nelle fasi successive per rispondere a domande legate alla conformance). Nell'ambito della conformance consente di misurare di quanto le tracce dell'event log corrispondono alle tracce generate dal modello. Dalla rete ricavata in precedenza, si può valutare l'alignment tenendo conto di quali siano le transizioni della rete bordate di rosso. Come è riportato anche nella legenda, il rosso identifica transizioni con una move maggiore di zero e quindi una riduzione dell'alignment su quella transizione. Gli scostamenti che si presentano sono una conseguenza dei filtri precedentemente applicati all'event log. Per estrarre il modello è stato infatti necessario rimuovere dai log eventi con bassa frequenza e quindi non notevolmente rilevanti.
- **Fitness:** consente di valutare la correttezza del modello ricavato mettendolo in relazione con il log di partenza. Viene calcolata come il rapporto tra il numero delle tracce del log che vengono rappresentate nel modello e il numero totale di tracce dell'event log. La Fitness è stata calcolata utilizzando il plug in “**Replay a Log on Petri Nets for Conformance Analysis**”. Questa misura dà un'idea sulla qualità del modello ottenuto, tuttavia non basta in quanto potrebbero comunque esserci fenomeni di Overfitting. Quest'ultimo indicherebbe che il modello è troppo specifico per il log e quindi non è abbastanza generale.



- **Precision:** nel processo di mining vengono utilizzate misure di precisione per quantificare quanto il modello di processo sovrasta il comportamento osservato nell'event log. La precisione ideale è pari a 1 ed implica che tutte le possibili tracce generate dal modello corrispondono a quelle del log, viceversa, una precisione più bassa indica che il modello codifica tracce aggiuntive rispetto a quelle del log effettivo. Quest'ultima condizione può portare al fenomeno di Underfitting, situazione in cui il modello è troppo generale per rappresentare il log correttamente. La Precisione è stata calcolata utilizzando il plug-in "**Measure Precision/ Generalization**".
- **Generalization:** indica di quanto il modello riesce a generalizzare il log. Questo concetto è strettamente correlato alla simplicity (non si vuole ottenere solo un modello generale ma anche il più semplice possibile). Per il calcolo di questa misura è stato utilizzato "**Measure Precision/ Generalization**".

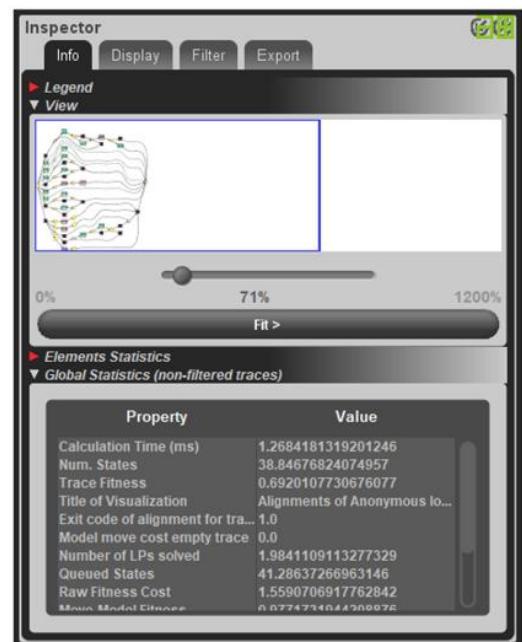
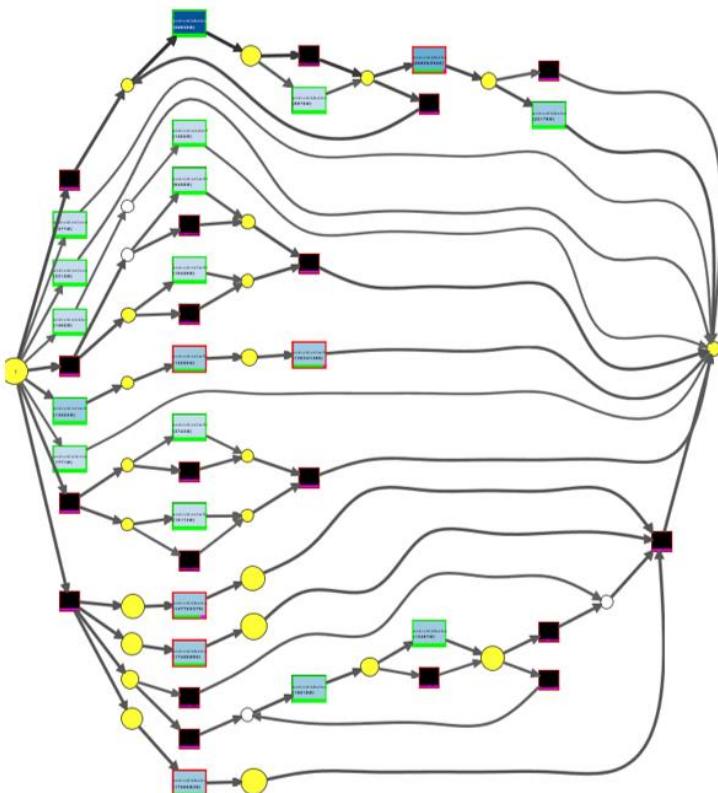
Dopo aver definito tutte le metriche interessate, si andrà a discutere i risultati ottenuti dal primo modello (figura 14). Il modello risulta avere una fitness pari a 0.96, una precisione pari a 0.287 e la generalization pari a 0.99. Da questi valori è possibile trarre alcune conclusioni, ovvero che il modello estratto utilizzando il Noise Threshold a 0.10 è un modello estremamente rappresentativo, cioè codifica in maniera molto precisa la maggior parte dei path, i comportamenti descritti dal dataset e cattura in maniera molto soddisfacente anche tutti i cicli caratteristici di dataset di questo tipo. Queste considerazioni trovano conferma dai due valori di fitness e generalization che risultano essere quasi al massimo. Tuttavia, vi è sempre un trade-off da rispettare che si materializza nell'analisi della precisione. Questo perché riuscendo a catturare al meglio anche i loop di lunghezza 1, lunghezza 2, e così via, il modello riesce generare sicuramente molte più tipologie di tracce rispetto a quelle di partenza. Questa grande espressività trasmessa dal modello si ripercote quindi sulla precisione, com'è possibile vedere dal valore estremamente basso ottenuto.

È da questa considerazione che si basa tutta la parte di conformance checking in quanto molto dell'impegno utilizzato per produrre questa parte è stato dedicato a risolvere anche in questo caso un "problema di ottimizzazione" in quanto come si vedrà successivamente c'è uno stretto legame tra il parametro dell'Inductive miner e la semplicità del modello, che a sua volta è strettamente collegata con la fitness e la precisione del modello stesso. Ci sono quindi questi tre parametri che entrano in gioco e l'obiettivo è trovare il giusto compromesso:



Il fenomeno dal quale nasce tutta l'analisi è quello che si evidenzia nella sequenza di foto sopra riportate. Il problema è cercare il giusto compromesso tra questi tre parametri in quanto per ottenere una precisione alta si è obbligati a rendere meno complesso il modello, e di conseguenza si ottiene una fitness minore. Per ottenere una fitness adeguata si è costretti a rendere il modello più complesso, ma di conseguenza la precisione si abbassa, e quindi l'obiettivo è trovare il giusto compromesso che permetta di ottenere un modello abbastanza rappresentativo del dataset e abbastanza preciso.

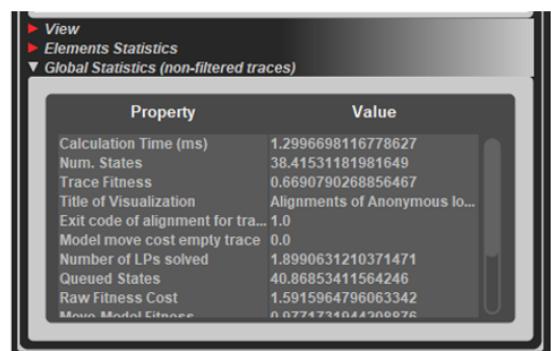
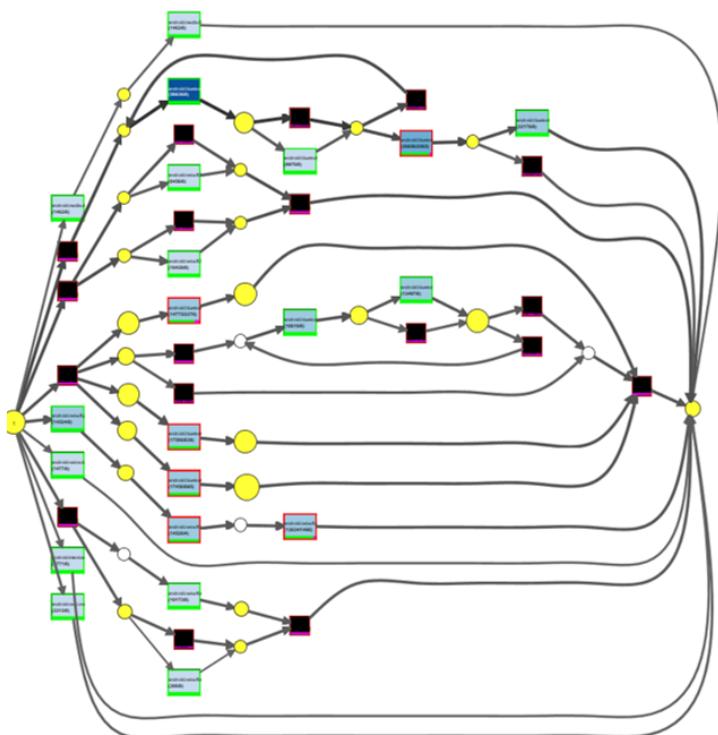
Di seguito vengono riportati alcuni test che confermano il fenomeno sopra riportato, dove è stato utilizzato lo stesso valore di Noise Threshold, ovvero 0.25, ma preso da bound diversi. Questo perché probabilmente ProM ha una granularità superiore alle due cifre visualizzate dal plugin, in quanto da vari test ci si è accorti che dallo stesso valore di Noise Threshold venivano generati modelli abbastanza differenti:



Precision : 0.48931

Generalization : 0.99681

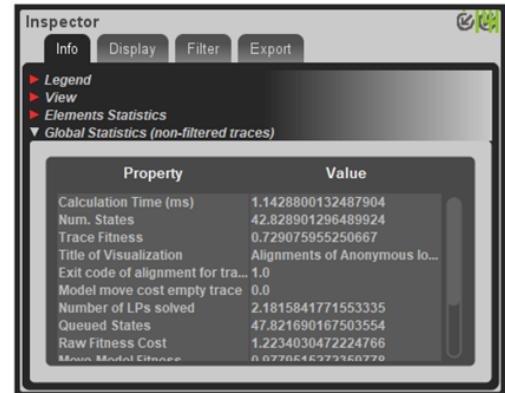
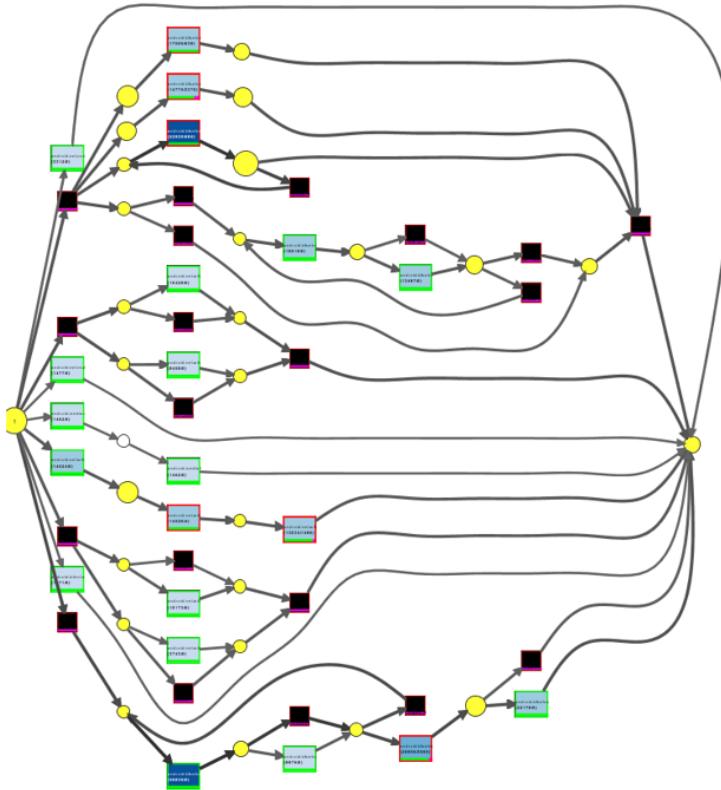
Figure 15: Risultato ottenuto utilizzando il modello restituito da Inductive miner con N.T. ≈ 0.255



Precision : 0.50050

Generalization : 0.99589

Figure 16: Risultato ottenuto utilizzando il modello restituito da Inductive miner con N.T. ≈ 0.259



Precision : 0.42425

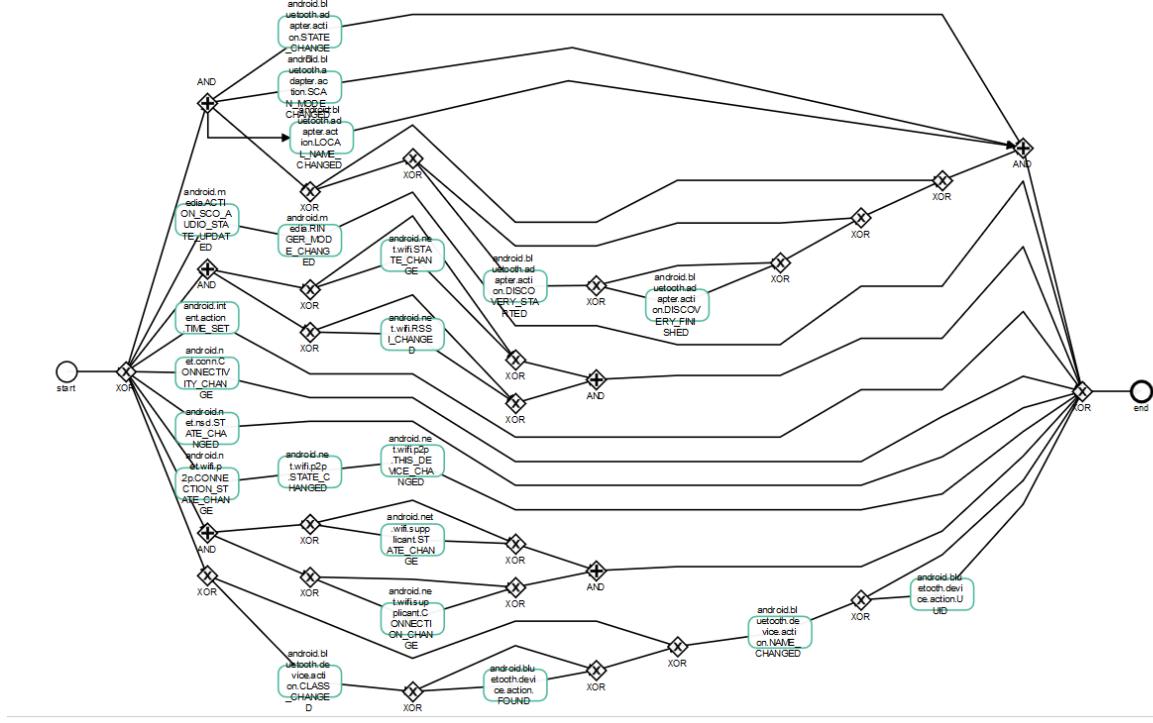
Generalization : 0.99747

Figure 17: Risultato ottenuto utilizzando il modello restituito da Inductive miner con N.T. ≈ 0.251

Analisi deviazioni con Celonis

Il modello generato su ProM (ultima figura, con parametro 0.251) è stato usato come modello di riferimento per le analisi di conformità sul software Celonis. Tale scelta è stata presa in quanto si è deciso di puntare più sulla fitness a discapito della precisione. Questo perché a differenza di altri contesti incentrati sui processi aziendali, in questo caso (con una descrizione dei micro-servizi) è sembrato più opportuno puntare su un modello che sia più rappresentativo dei dati a disposizione piuttosto che ottenere un modello che riesca meglio a generalizzare i comportamenti. Trattandosi di servizi già cablati all'interno del dispositivo, non avrebbe avuto molto senso spingere sulla generalizzazione più marcata di tali servizi in quanto la maggior parte viene già catturata dal modello.

Poiché il software prende in input un modello BPMN è stato necessario convertire attraverso il plug-in “**Convert Petri net to BPMN diagram**” di ProM la petri net in rete BPMN. Il modello generato ed importato su Celonis è il seguente:



Grazie all'utilizzo del software Celonis e dei tools di Conformance Checking si è riusciti ad analizzare il modello e il dataset andando ad estrapolare delle informazioni utili che hanno portato ad una miglioria nel modello stesso. Celonis permette in maniera molto intuitiva di andare ad individuare quali sono i comportamenti che non cattura il modello rispetto al dataset. Di seguito vengono riportate le più frequenti:

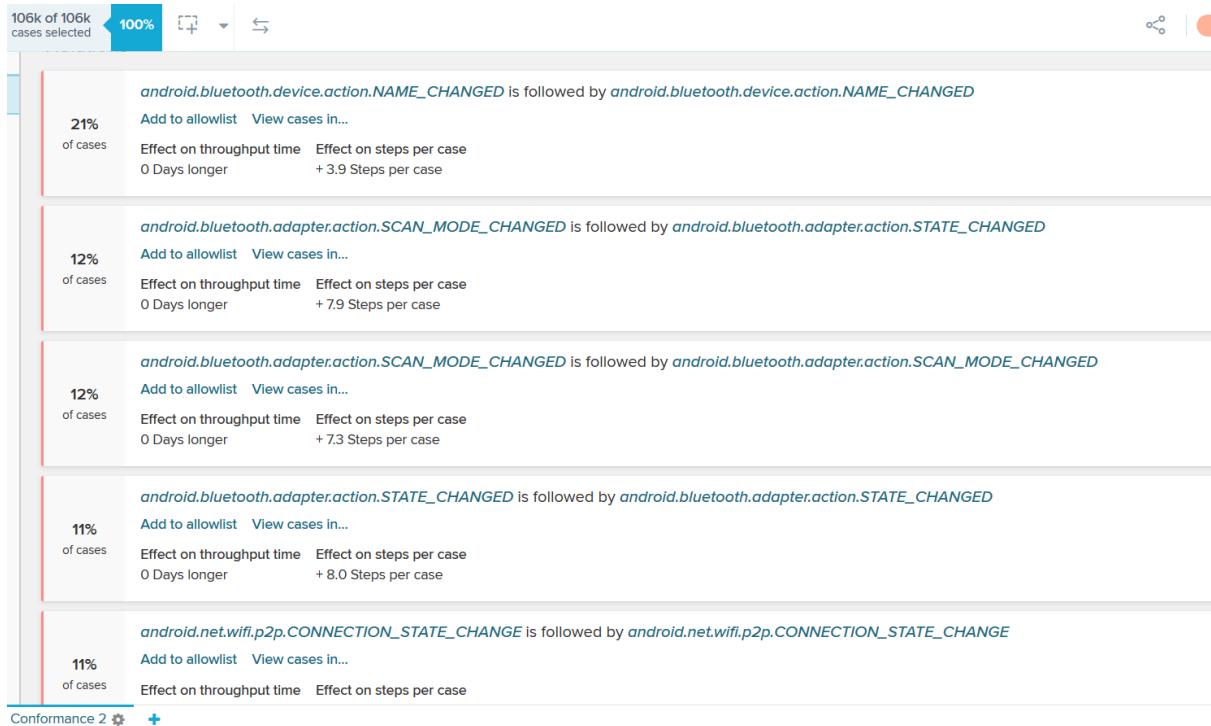


Figure 18: Top violazioni

Come si può vedere dalla precedente immagine, ci sono delle percentuali importanti di dataset che non vengono codificate all'interno del modello, ma se si va ad analizzarle meglio la grande maggioranza di queste codifica un ciclo su sé stesso (loop di lunghezza 1). Come detto più volte, questa tipologia di eventi ha la caratteristica di poter

ciclare su sé stessa un numero indefinito di volte. In questo caso, per semplicità di utilizzo, si è considerato questa tipologia di processi come processi differenti, ma il tutto può esser visto come facenti parte di un solo processo più complesso oppure di un sotto-processo. Proprio per questo motivo, sono state trattate come “corrette” tutte quelle deviazioni che codificano un loop di lunghezza 1 grazie al software Celonis che permette di scegliere quali violazioni mettere in un'apposita lista, chiamata Allowlist:

Percentage of cases	Event Followed by Allowed Event	Action
21%	<code>android.bluetooth.device.action.NAME_CHANGED</code> is followed by <code>android.bluetooth.device.action.NAME_CHANGED</code>	Remove from allowlist View cases in...
12%	<code>android.bluetooth.adapter.action.SCAN_MODE_CHANGED</code> is followed by <code>android.bluetooth.adapter.action.SCAN_MODE_CHANGED</code>	Remove from allowlist View cases in...
11%	<code>android.bluetooth.adapter.action.STATE_CHANGED</code> is followed by <code>android.bluetooth.adapter.action.STATE_CHANGED</code>	Remove from allowlist View cases in...
11%	<code>android.net.wifi.p2p.CONNECTION_STATE_CHANGE</code> is followed by <code>android.net.wifi.p2p.CONNECTION_STATE_CHANGE</code>	Remove from allowlist View cases in...
11%	<code>android.net.wifi.p2p.STATE_CHANGED</code> is followed by <code>android.net.wifi.p2p.STATE_CHANGED</code>	Remove from allowlist View cases in...

Conformance 2

Dopo una prima scrematura delle violazioni, il risultato ottenuto dal modello è il seguente:



Questo risultato è abbastanza soddisfacente in quanto si riesce a codificare il 76% degli eventi, anche se si ottengono 32 tipologie diverse di violazioni, che potenzialmente potrebbero essere delle anomalie. Tali violazioni fanno parte della prima delle due tipologie di anomalie precedentemente definite, ovvero quella individuabile mediante una violazione del path codificato nel modello. Di seguito vengono riportate le violazioni più frequenti:

Percentage of cases	Event Followed by Allowed Event	Action
12%	<code>android.bluetooth.adapter.action.SCAN_MODE_CHANGED</code> is followed by <code>android.bluetooth.adapter.action.STATE_CHANGED</code>	Add to allowlist View cases in...
10%	<code>android.bluetooth.adapter.action.LOCAL_NAME_CHANGED</code> is followed by <code>android.bluetooth.adapter.action.SCAN_MODE_CHANGED</code>	Add to allowlist View cases in...
8%	Incomplete case	Add to allowlist View cases in...
1%	<code>android.bluetooth.device.action.NAME_CHANGED</code> is followed by <code>android.bluetooth.device.action.CLASS_CHANGED</code>	Add to allowlist View cases in...
1%	<code>android.bluetooth.device.action.NAME_CHANGED</code> executed as <code>START</code> activity	Add to allowlist View cases in...

Un'ulteriore scrematura che è stata fatta sul modello è basata proprio sulle violazioni. Questo perché effettivamente, dopo alcuni approfondimenti fatti sul comportamento dei micro-servizi inerenti a quelle violazioni, ci si è accorti che le prime due violazioni più frequenti codificano dei path che sono logicamente accettabili, ma che erano stati tagliati fuori dal modello durante i vari filtri. Si è quindi deciso di andare ad apportare delle modifiche al modello per riuscire a codificare in maniera corretta anche quelle due tipologie di path. Tali modifiche sono state fatte in maniera manuale grazie all'utilizzo di Celonis, che una volta caricato il modello di base permette di andare a modificarlo a piacimento tramite l'apposita interfaccia. Tali modifiche consistono nell'aggiunta di due attività con relativi collegamenti in **xor**, che hanno codificato la possibilità di eseguire in maniera opzionale le prime due pseudo violazioni. Si riporta successivamente il modello modificato:

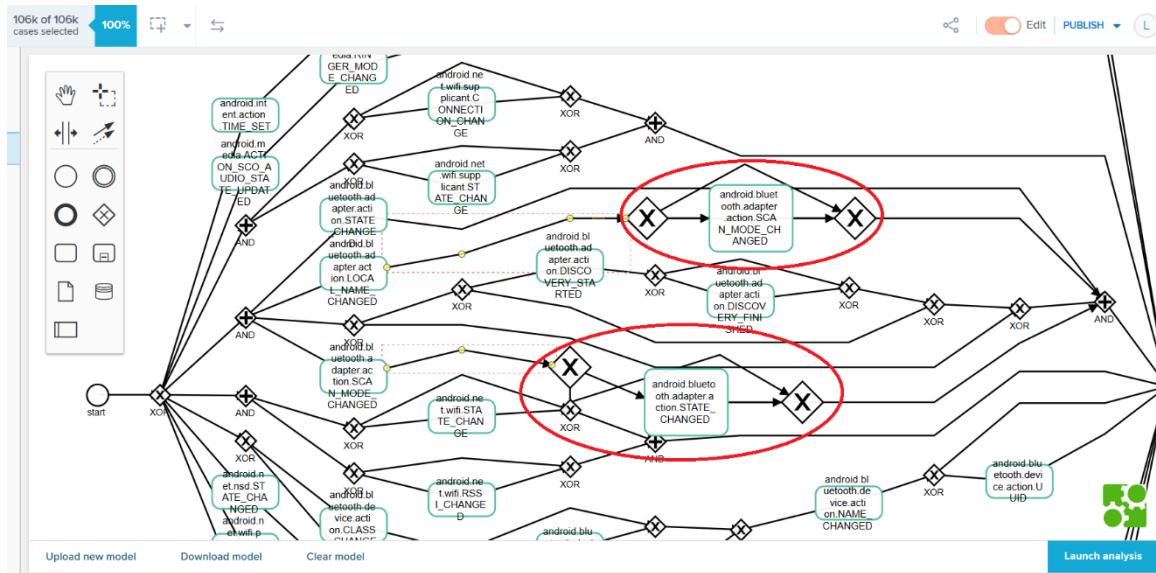


Figure 19: Modifiche manuali apportate al modello

Si riportano in seguito i risultati ottenuti dalla modifica del modello, che risultano essere molto soddisfacenti:



È possibile vedere come la percentuale di tracce che fitta sale al 89%, le violazioni diminuiscono nettamente ottenendone 22 ed, a seguito di alcune modifiche, anche l'Allowlist scende a 10. Per completezza verranno successivamente riportate le violazioni e i path consentiti:

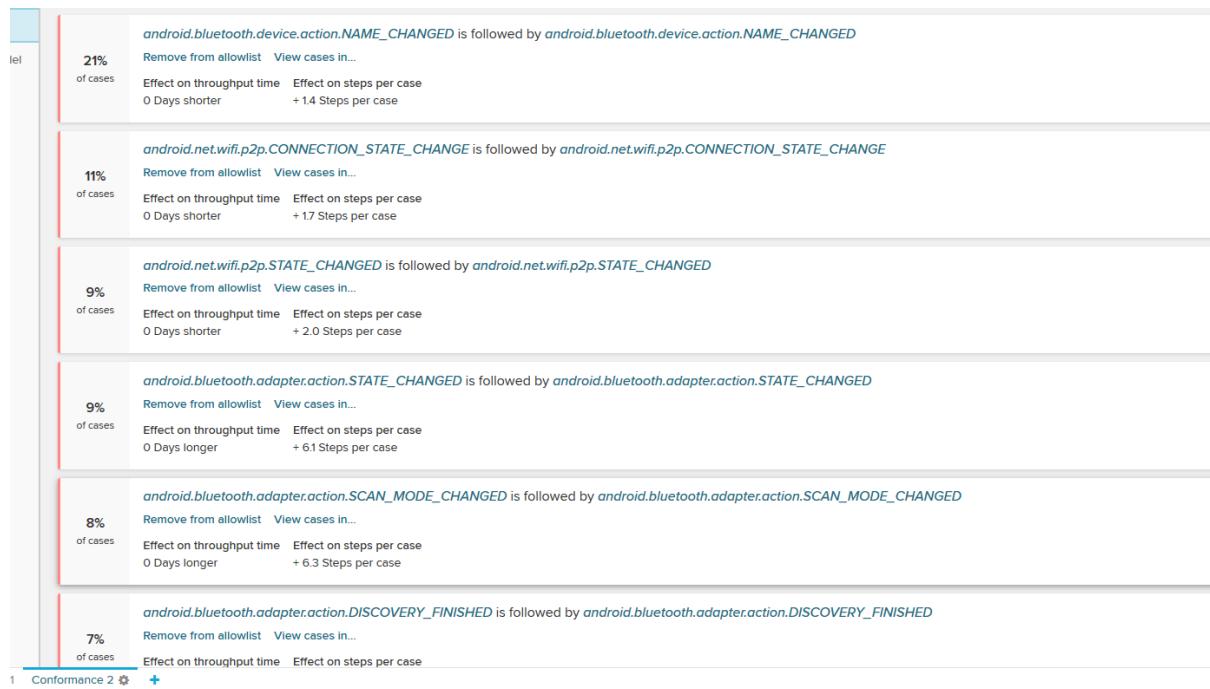


Figure 20: Allowlist

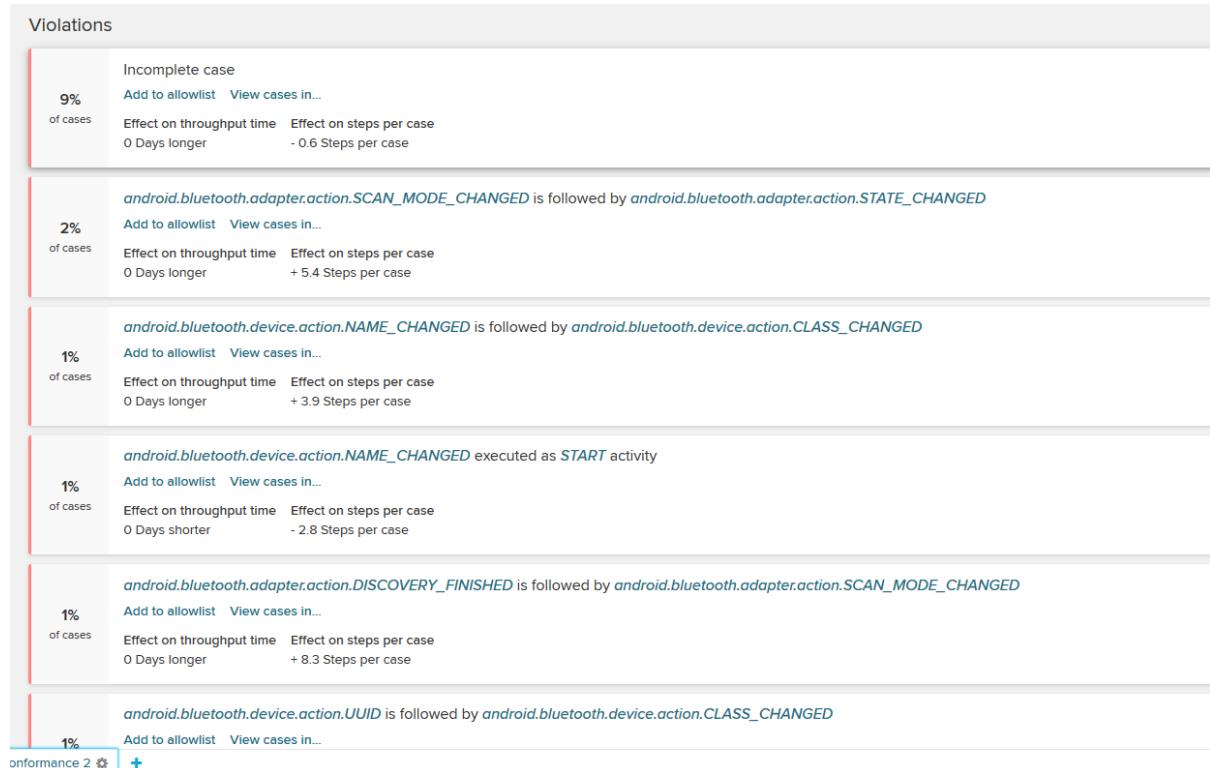
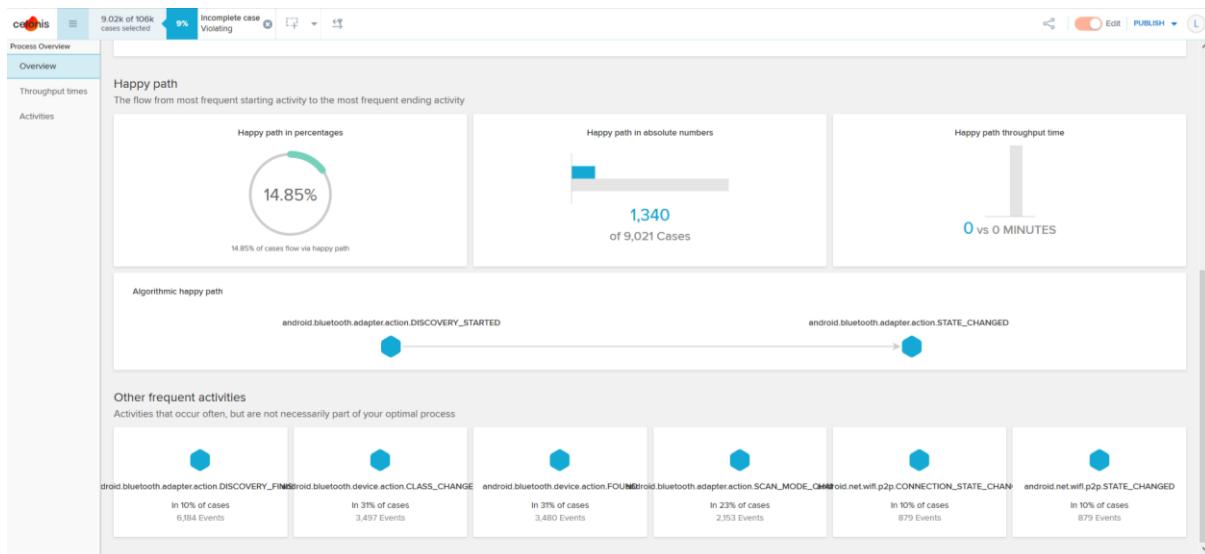


Figure 21: Top violazioni

Analizzando le violazioni rimanenti, la cosa che salta all'occhio è che la prima violazione (in termini di frequenza) risulta dall'insieme delle tracce incomplete presenti nel dataset, confrontate con il modello. Andando ad analizzarle più a fondo grazie ai filtri di Celonis ed utilizzando il tool di process overview, si ottengono i seguenti risultati:

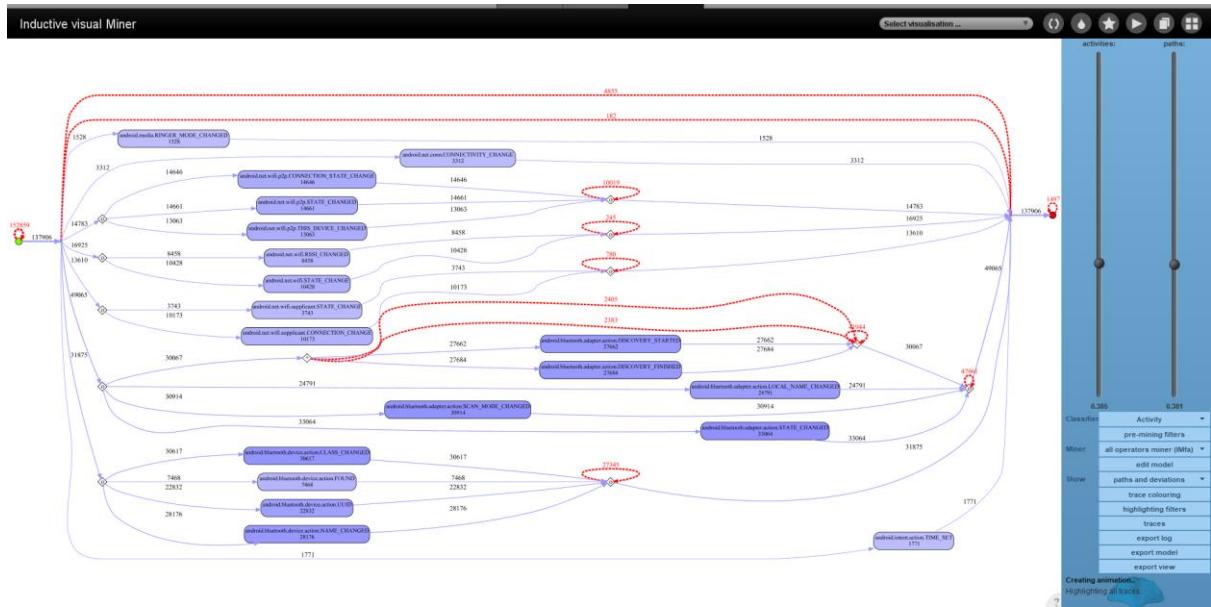


Si nota come non vengono codificate dal modello diverse attività presenti nel tab “**other frequent activities**” e l’happy path in questo caso risulta essere la sequenza di eventi DISCOVERY_STARTED e STATE_CHANGED che rappresenta il 14.85% dei cases presenti in questa porzione di dataset. L’anomalia di tali eventi verrà trattata successivamente nel capitolo di Anomaly detection in cui, grazie ai tool di ProM, si riuscirà ad analizzare al meglio le singole tracce per classificare eventuali trace come anomali. Purtroppo, il tool di Conformance Checking non permette di affrontare tale problematica più a fondo, ed è per questo che si è deciso di ricorrere ad altri tool presenti in ProM per analizzare singolarmente i vari trace.

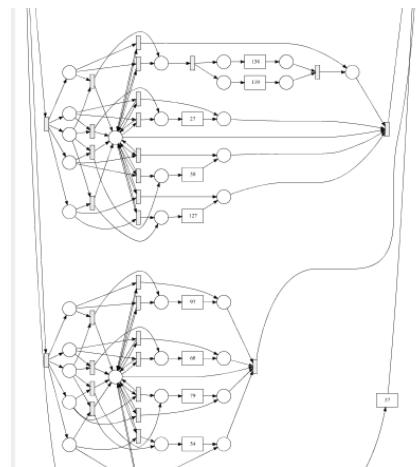
Analisi deviazioni con ProM

Altro plugin utilizzato per approfondire il tema delle deviazioni, è stato “**Inductive visual miner**” che permette in stile Disco di estrapolare un modello di processo variando i due parametri “activities” e “path”, permettendo così di andare a visualizzare in maniera più intuitiva le deviazioni (rappresentate con linee tratteggiate rosse, con specificata la frequenza). Purtroppo, questo tool non ha fornito dei risultati abbastanza soddisfacenti in quanto probabilmente utilizzando una versione diversa di Inductive miner, il modello ottenuto per rappresentare i processi risulta essere molto particolare e non adatto alle esigenze del progetto (nonostante siano stati effettuati diversi test sui parametri in gioco), soprattutto per due aspetti:

- Non riesce minimamente a codificare alcune sequenze note di attività, quali DISCOVERY_STARTED con DISCOVERY_FINISHED e così via.
- La sequenzialità di alcune attività viene codificata in maniera molto particolare tramite l’utilizzo dell’**or** che quindi non ne garantisce l’ordine corretto dovuto al non determinismo.



Come si può vedere dall'interfaccia, le principali deviazioni risultano essere dei cicli che non vengono codificati all'interno di questo modello (meno interessanti da un punto di vista di Anomaly Detection), mentre altre deviazioni rappresentano dei veri e propri skip di attività che invece potrebbero essere più interessanti (sempre in ottica di Anomaly Detection). Purtroppo, questa analisi non viene ulteriormente approfondita in quanto, pur estrapolando il modello per analizzarlo con altri plugin di ProM, tale modello risulta essere inutilizzabile per la presenza dei troppi **or** che vengono codificati in maniera troppo complessa in petri-net.



RESOURCE ANALYSIS

Rispetto agli altri progetti, questo dataset ha obbligato a scegliere la risorsa da utilizzare. Tra tutte le colonne disponibili si è scelto come risorsa le persone, in quanto l'analisi iniziale era finalizzata ad individuare le possibili correlazioni tra le persone che generavano gli eventi registrati da Sherlock e dare conferma di alcuni aspetti trattati all'interno del paper relativo al dataset. La Resource Analysis è finalizzata a scovare le possibili relazioni tra le risorse e analizzarle andando a descriverne anche alcune proprietà standard studiate a lezione, come **handover of work**, che possono dare anche delle indicazioni a livello di performance.

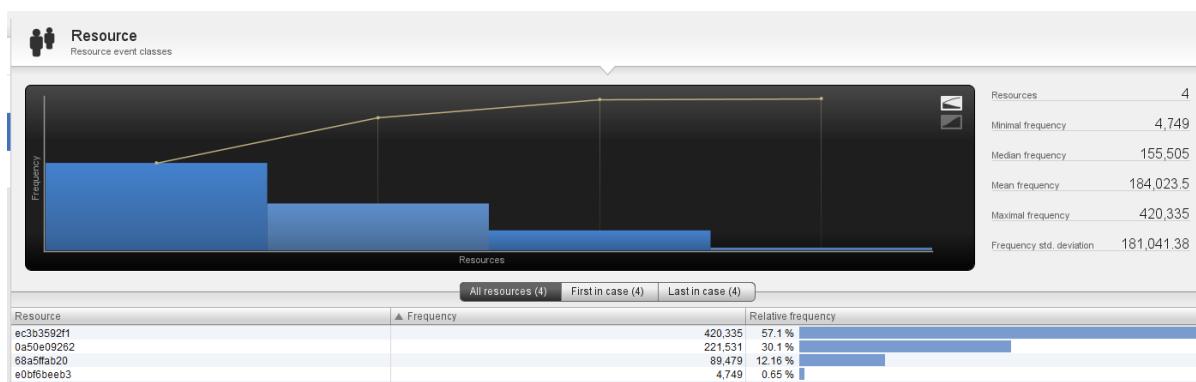
Il dataset è formato da volontari che si sono prestati a questo progetto per una raccolta di dati sfruttabili in ambito di sicurezza. Proprio per questo, ci si aspetta che non vi sia alcun collegamento tra le risorse in quanto gli utenti sono indipendenti tra di loro. Questo ovviamente è una forzatura dettata dalla natura del dataset e dal fatto che per effettuare tale analisi si è dovuto creare artificialmente il concetto di processo, e quindi in questa analisi più che mai ha poco senso analizzare a fondo le relazioni tra le risorse in gioco, almeno finché non si prende come risorsa la persona. Di seguito si riporta l'analisi che conferma le considerazioni precedentemente riportate.

Nel periodo preso in considerazione entrano in gioco solo 4 dei 50 volontari, come abbiamo già visto nella parte relativa ai Dotted Chart:

Timestamp	userid	AGE	had the same phone	phone has changed, approximat	phone has changed, what model	you got the new ZeroLem	usually turn off your	usually turn off your	What is your gender?	the highest degree or	your primary language
2016/05/01 10:12:18 PM GMT+2	ec3b3592f1	22	no	08/12/2015	Samsung Galaxy S5	4	No	Yes	Female	High school graduate	English
2016/05/02 8:28:38 PM GMT+2	0a50e09262	25	yes		Samsung Galaxy S5	7	No	battery runs out	Female	Bachelors degree	English
2016/04/26 8:15:49 PM GMT+2	68a5ffab20	33	no	4/26/2015	Samsung Galaxy S5	7	No	Yes	Male	Bachelors degree	English
2016/04/26 7:30:58 PM GMT+2	e0bf6beeb3	29	yes		Samsung Galaxy S5	5	No	No	Male	High school graduate	English

Figure 23: Informazioni aggiuntive sui volontari

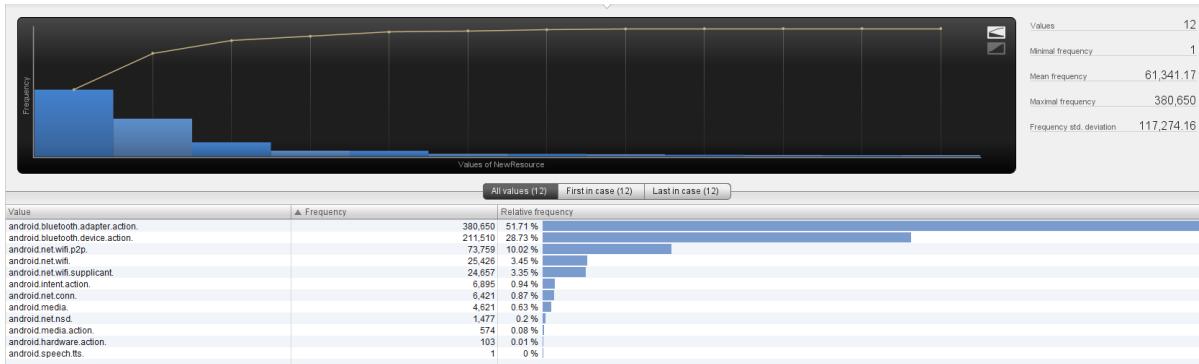
Come si può notare, i volontari sono due ragazzi e due ragazze molto giovani, come si è già notato dalle attività presenti nel dataset, in cui vi è un utilizzo massiccio di Bluetooth (con casse e cuffie) e della rete.



Com'è possibile vedere dall'immagine, la maggior parte degli eventi (oltre il 57%) sono svolti dall'utente ec3b3592f1, mentre il resto sono svolti dagli utenti 0a50e09262 (il 30%) e 68a5ffab20 (il 12,16%). L'utente e0bf6beeb3 svolge, in questo periodo, meno dell'1% delle attività. Questa grande differenza che vi è tra il numero di eventi svolti dai primi due utenti, e quelli svolti degli ultimi due, è dovuto al fatto che il periodo di tempo da noi

reso in considerazione va dal 01-06-2017 al 01-07-2017. Gli ultimi due utenti (in ordine di frequenza) terminano invece la loro attività rispettivamente nei giorni 05-06-2017 e 02-06-2017 (date confermate dalla precedente analisi dei Dotted Chart), e i loro dati sono quindi relativi ad un periodo di tempo molto breve (5 giorni per il primo e 2 giorni per il secondo).

Prendendo invece in considerazione le componenti degli smartphone come risorse, si ottiene:



In questo caso, la risorsa più utilizzata è quella del Bluetooth (suddivisa in bluetooth.adapter e bluetooth.device), seguita dal wifi. Già solo queste due risorse svolgono più del 95% degli eventi.

Riprendendo ora gli utenti come risorse, si analizzano quelle che sono le durate medie dei vari case.



La durata media dei case, riportata su disco, corrisponde a 0 millisecondi. Questo valore è ovviamente sbagliato, ed è dovuto al fatto che la maggior parte degli eventi vengono svolti in pochi millisecondi ma nel dataset vi è una precisione fino ai secondi, e di conseguenza tutti i millisecondi sono 0. Si nota comunque che alcuni case sono gestiti in tempi molto lunghi, che superano anche un intera giornata:

Case ID	Events	Variant	Started	Finished	Duration
878	342	Variant 1,327	03.06.2017 00:44:39	04.06.2017 13:13:29	1 day, 12 hours
4168	45	Variant 1,770	10.06.2017 06:44:24	11.06.2017 07:12:08	1 day, 27 mins
39651	2	Variant 3	23.06.2017 01:01:58	23.06.2017 13:25:33	12 hours, 23 mins
77309	91	Variant 2,807	06.06.2017 00:55:35	06.06.2017 08:36:46	7 hours, 41 mins
96604	7	Variant 117	13.06.2017 01:47:41	13.06.2017 06:37:59	4 hours, 50 mins
8282	2,711	Variant 2,281	16.06.2017 19:28:33	16.06.2017 23:34:59	4 hours, 6 mins
49086	393	Variant 2,486	30.06.2017 20:42:44	30.06.2017 23:59:57	3 hours, 17 mins
5605	82	Variant 1,963	15.06.2017 00:47:25	15.06.2017 03:25:21	2 hours, 37 mins

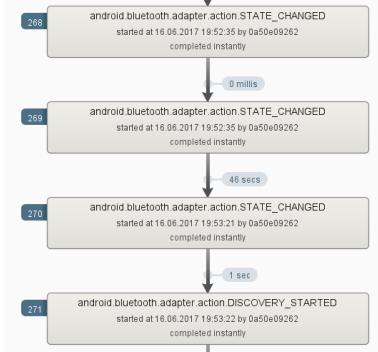
Particolarmente strani sono i primi tre case, in cui vi è una durata molto elevata. Si analizzano più nel dettaglio:



Si può notare che in tutti e 3 i casi, il tempo che intercorre tra il penultimo evento e l'ultimo è molto grande, e questa può essere un'anomalia. In realtà, da un'analisi più approfondita, è emerso che tutti gli utenti non facevano uso del

telefono durante il sabato (probabilmente il dispositivo veniva spento), e di conseguenza non si hanno misurazioni in quelle giornate.

Caso differente dai precedenti si ha con il case 8282, che supera le 4 ore. Analizzandolo più nel dettaglio, non si trova nessun tipo di anomalia:



la maggior parte degli eventi in questo caso vengono infatti gestiti in 0 millisecondi, e tutti gli altri in una manciata di secondi (solitamente non più di 46 secondi). La durata di 4 ore dunque non è attribuibile ad una durata eccessiva di alcuni eventi, come nel caso precedente, ma al numero eccessivo di eventi presenti in questo caso (2711). Da un'analisi più approfondita si è vista la presenza di un pattern ripetuto, dovuto al fatto di aver lasciato il Bluetooth acceso senza averlo connesso ad alcun dispositivo e portandolo così a ripetute scansioni che ne vanno a caratterizzare la durata eccessiva:

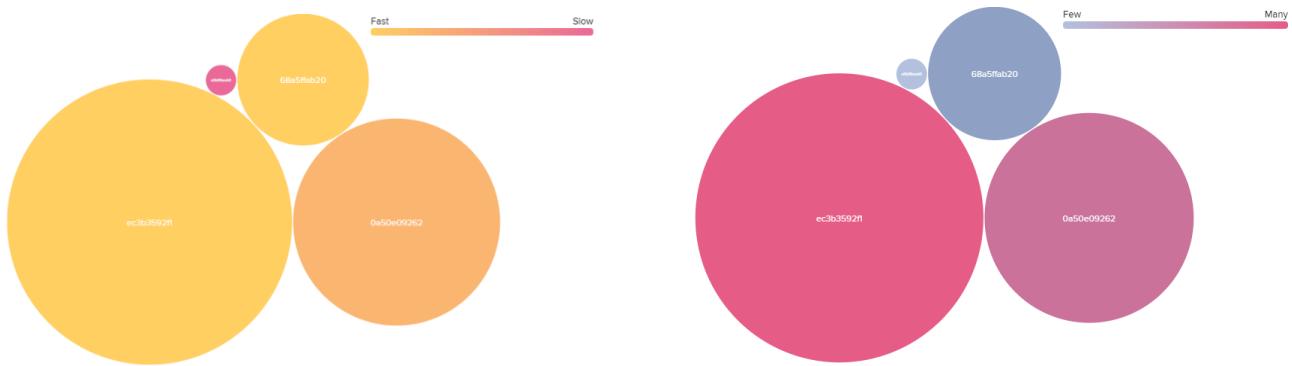
.adapter.action.STATE_CHANGED	0a50e09262	16.06.2017	19:28:33	android.bluetooth.adapter.extra.PREVIOUS_STATE: 12
.adapter.action.STATE_CHANGED	0a50e09262	16.06.2017	19:28:34	android.bluetooth.adapter.extra.PREVIOUS_STATE: 13
.adapter.action.LOCAL_NAME_CHANGED	0a50e09262	16.06.2017	19:29:19	android.bluetooth.adapter.extra.LOCAL_NAME: Pnina Cohen (Galaxy S5)
.adapter.action.SCAN_MODE_CHANGED	0a50e09262	16.06.2017	19:29:19	android.bluetooth.adapter.extra.SCAN_MODE: 21
.adapter.action.SCAN_MODE_CHANGED	0a50e09262	16.06.2017	19:29:19	android.bluetooth.adapter.extra.SCAN_MODE: 20
.adapter.action.STATE_CHANGED	0a50e09262	16.06.2017	19:29:19	android.bluetooth.adapter.extra.PREVIOUS_STATE: 10
.adapter.action.DISCOVERY_STARTED	0a50e09262	16.06.2017	19:29:20	android.bluetooth.adapter.extra.PREVIOUS_STATE: 11
.adapter.action.STATE_CHANGED	0a50e09262	16.06.2017	19:29:20	android.bluetooth.adapter.extra.PREVIOUS_STATE: 11
.adapter.action.DISCOVERY_FINISHED	0a50e09262	16.06.2017	19:29:33	android.bluetooth.adapter.extra.PREVIOUS_STATE: 12
.adapter.action.DISCOVERY_FINISHED	0a50e09262	16.06.2017	19:29:33	android.bluetooth.adapter.extra.PREVIOUS_STATE: 12

Situazione analoga si ha con il caso 49086, con una durata superiore alle 3 ore. Il numero di eventi in questo caso è pari a 393, numero decisamente minore al caso precedente ma in questo caso gli eventi gestiti in 0 millisecondi sono pochissimi, e la maggior parte hanno un tempo medio di 30 secondi (essendo relativi ad una scansione Bluetooth dovuta probabilmente al collegamento di un dispositivo Bluetooth sempre connesso, come smartwatch)



Si può dire che l'elevata durata di questi ultimi due casi non sembra dovuta a qualche tipo di anomalia, ma all'attitudine dell'utente nell'utilizzo massivo di risorse Bluetooth.

Effettuando una Social Analysis con l'utilizzo del software Celonis, quello che si ottiene sono questi due grafici:



Notiamo che la risorsa più lenta (a livello di durata media dei cases) è la risorsa e0bf6beeb3, mentre la più veloce è ec3b3592f1. Questo risultato è dovuto al fatto che per la risorsa ec3b3592f1 vi sono numerosi eventi con durata "0" millisecondi, al contrario della risorsa e0bf6beeb3 per la quale, avendo solo dati relativi a due giornate, il numero di eventi con 0 millisecondi è estremamente basso. Si può inoltre notare che la risorsa ec3b3592f1 ha un numero di eventi maggiori rispetto a 0a50e09262, nonostante abbia un giorno in meno di misurazioni. A supporto di questo si può vedere, grazie all'utilizzo di Celonis, il profilo di utilizzo del dispositivo relativo alle due risorse più frequenti

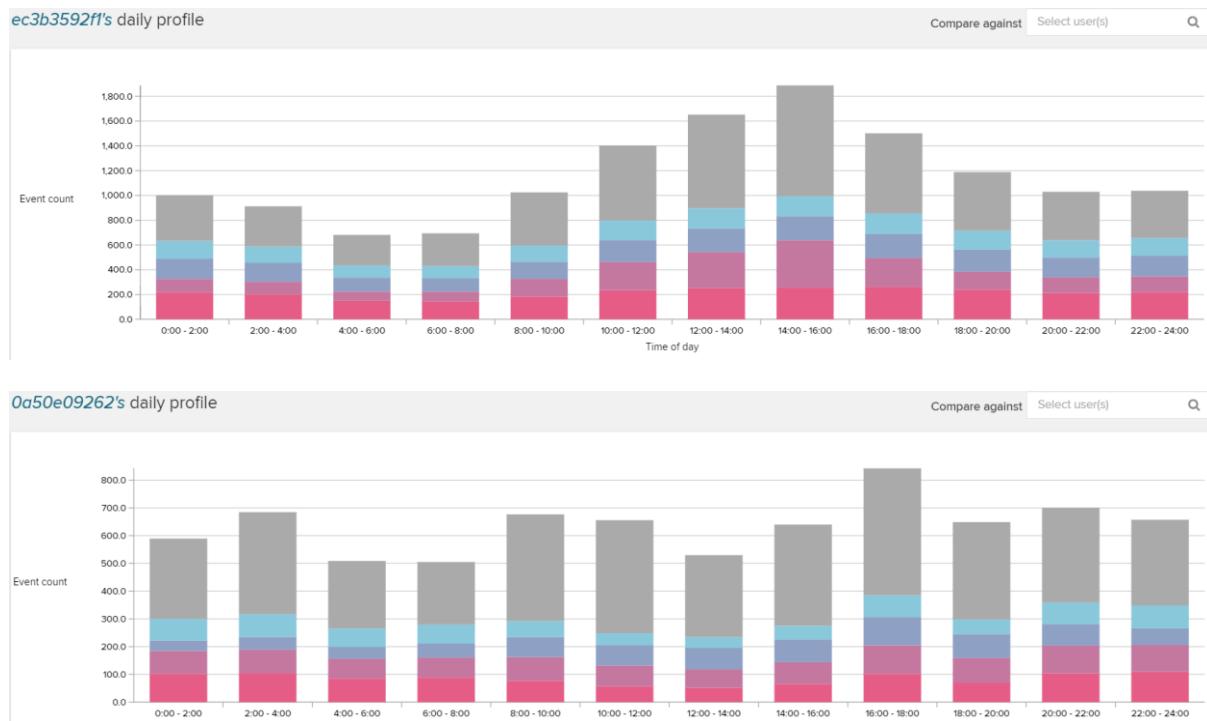


Figure 24: Profilo di utilizzo, in media, del dispositivo durante l'arco della giornata

Anche sotto questa visualizzazione viene confermato il fatto che la prima risorsa genera molte più attività rispetto alla seconda. Tale informazione la si può vedere dal picco di attività registrate durante la giornata che risulta essere in media 1800 per la prima risorsa e 800 per la seconda.

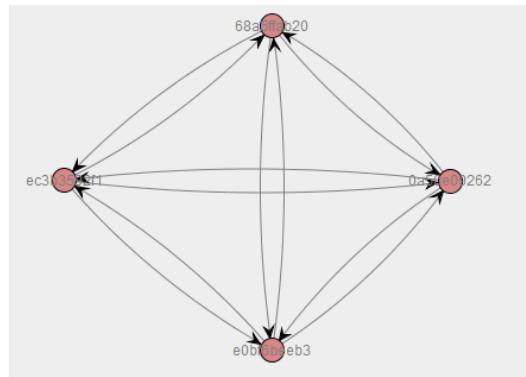
È possibile notare un comportamento diverso tra i due utenti dovuto ad una diversa tipologia di utilizzo del terminale. Nel primo caso è netta la distinzione tra l'elevato utilizzo giornaliero e il basso utilizzo notturno mentre nel secondo caso si può notare una distribuzione di utilizzo più uniforme.

Social Network Analysis

Si analizza ora il dataset tramite l'utilizzo del software ProM. Anche in questo caso, vista la natura del dataset, non ci si aspetta la correlazione tra risorse dato che gli utenti di questo dataset sono completamente indipendenti l'uno dall'altro. Verranno utilizzati **plugin di social network**, il cui obiettivo è quello di analizzare le risorse rispetto ad una relazione specifica. I plugin che verranno utilizzati sono i seguenti:

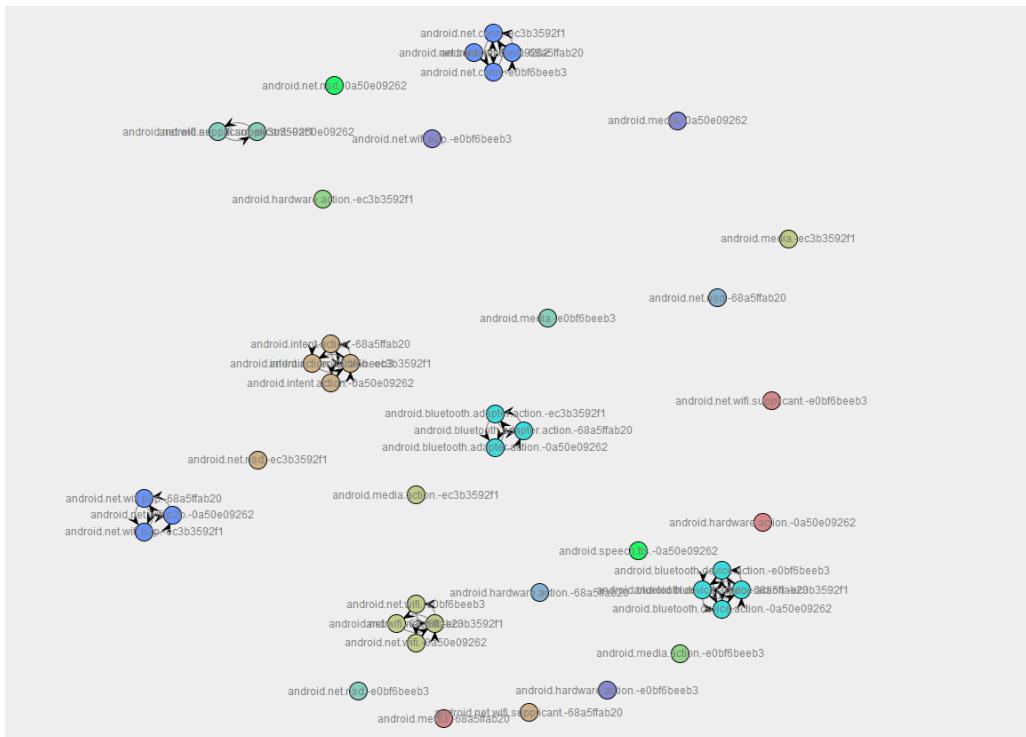
1. **Handover of work**: evidenzia quelli che sono i passaggi di lavoro tra una risorsa e un'altra. Nel dettaglio, vi è un handover of work se ci sono due attività sequenziali dove la prima è completata da una risorsa e la seconda è completata da un'altra risorsa.
2. **Subcontracting**: una risorsa esegue un lavoro per conto di qualcun altro. In questo caso quindi non solo si riceve del lavoro, ma questo lavoro viene poi restituito. In pratica vi è una risorsa A che esegue un'attività tra due attività eseguite dalla risorsa B.
3. **Reassignment**: rappresenta un'attività che viene riassegnata ad altri. La risorsa A assegna un'attività alla risorsa B, e B la riassegna a C.
4. **Working together**: sta ad identificare le risorse che usualmente lavorano insieme. In questo caso non si pensa più alla successione diretta, ma si va solo a vedere se hanno lavorato insieme in un processo.
5. **Similar Task**: plugin che si concentra sulle attività che ogni originator svolge; vi è una clusterizzazione rispetto ad un concetto di similitudine.

Primo plugin, quello di **Handover of work**, è stato utilizzato su una prima versione del dataset ottenendo questo risultato:

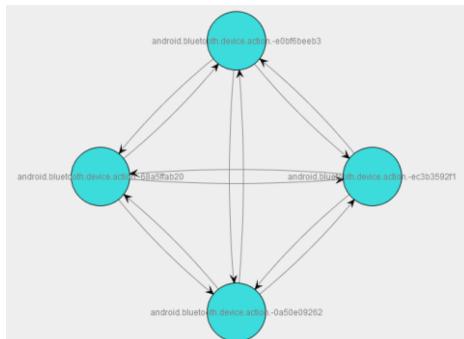


Questo risultato va contro le idee iniziali in quanto sembrerebbe che vi siano dei passaggi di lavoro tra le quattro risorse. Tale considerazione ha portato ad un'analisi più profonda del dataset, scovando alcune imperfezioni al suo interno. Queste hanno portato a migliorarne la qualità con un ulteriore fase di preprocessing, ottenendo la versione finale del dataset (versione 2.2) che è stata poi utilizzata per il resto dell'analisi.

Dall'utilizzo del plugin ci si aspetterebbe quindi un risultato completamente diverso, con le quattro risorse distaccate e senza alcun collegamento. Per scovare il problema si è scelto di selezionare come risorse sia gli utenti che le componenti dello smartphone, per visualizzare eventuali collegamenti tra attività di utenti differenti, ottenendo il seguente risultato:



Più nello specifico, si trova questo passaggio di lavoro tra i Bluetooth dei quattro dispositivi:



Da questo risultato si può notare che ci sono correlazioni tra lo stesso evento di utenti differenti e questo è impensabile data la natura del dataset. È possibile concludere che questo risultato inatteso è dovuto ad uno scorretto ordinamento del dataset: essendo la prima versione ordinata solo per tempo, si ha un'unione tra eventi svolti da risorse differenti, e questo non va bene in quanto le risorse in questo caso sono completamente distaccate tra di loro. Si è quindi passati ad un'ulteriore versione del dataset, in cui si ha un ordinamento prima per risorsa e poi per timestamp. Applicando ora il l'Handover of work sul nuovo dataset si ottiene il seguente risultato:



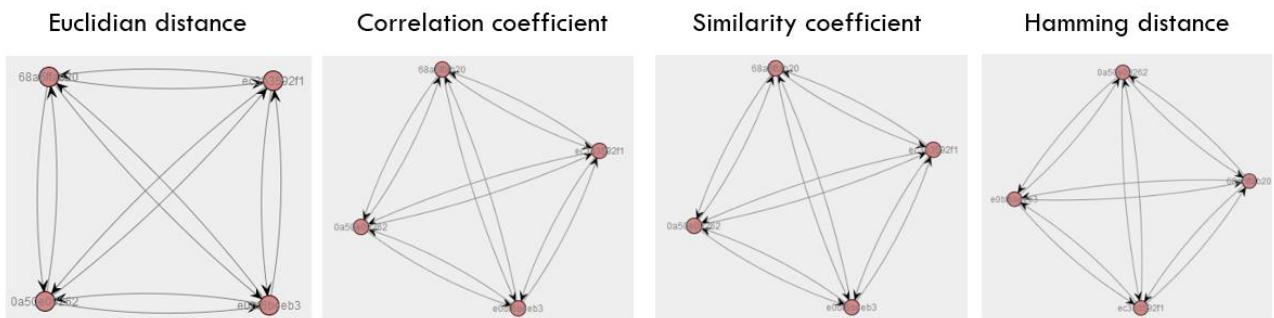
Non risultano dunque passaggi di lavoro tra una risorsa e l'altra, com'è giusto che sia. Lo stesso risultato, con le quattro risorse non collegate, si ha con i plugin di subcontracting, reassignment e working together. Tutta questa

analisi conferma l'idea iniziale dell'indipendenza tra risorse dettata dalle indicazioni date dalla documentazione del progetto Sherlock.

Unico plugin di Social Network che restituisce un risultato differente, è il plugin di **Similar Tasks**. Il Similar Tasks Analyzer si concentra sulle attività che ogni originator svolge. Il presupposto in questa analisi è che le risorse che fanno cose simili hanno relazioni più forti di quelle che fanno cose completamente diverse. Ogni risorsa ha quindi un "profilo" basato sulla frequenza con cui svolge attività specifiche. Esistono quattro tipi di metriche di distanza tra due profili:

1. **Distanza Euclidea**: è la distanza "ordinaria" tra due punti basata sul concetto di distanza sul piano cartesiano. (Fornisce buoni risultati solo se le risorse eseguono volumi di lavoro comparabili).
2. **Coefficiente di correlazione di Pearson**: viene spesso utilizzato per trovare la relazione tra variabili come età e pressione sanguigna, variabili molto diverse tra di loro.
3. **Coefficiente di similarità**: è un indice statistico utilizzato per confrontare la somiglianza e la diversità dei punti.
4. **Distanza di Hamming**: date due tracce, vede quanti sono gli errori, le posizioni che non corrispondono.

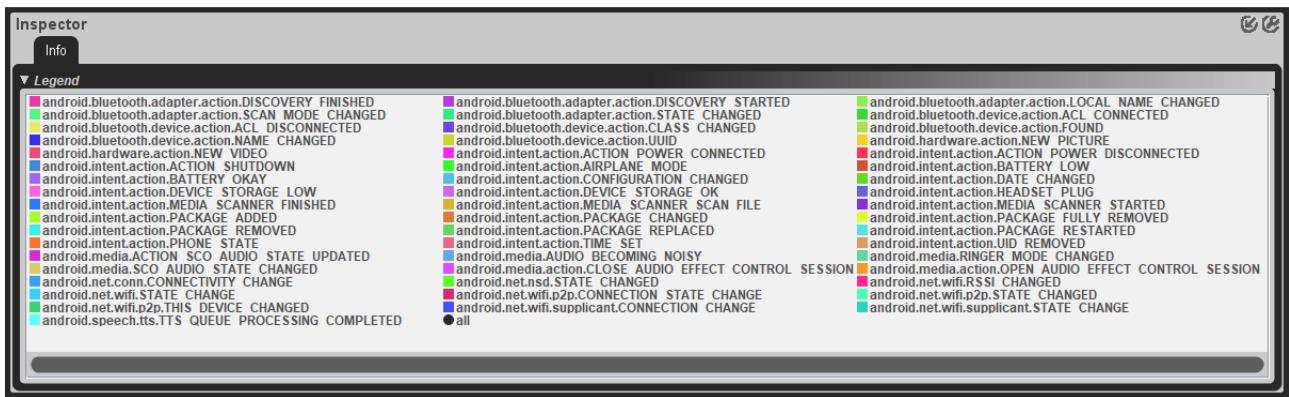
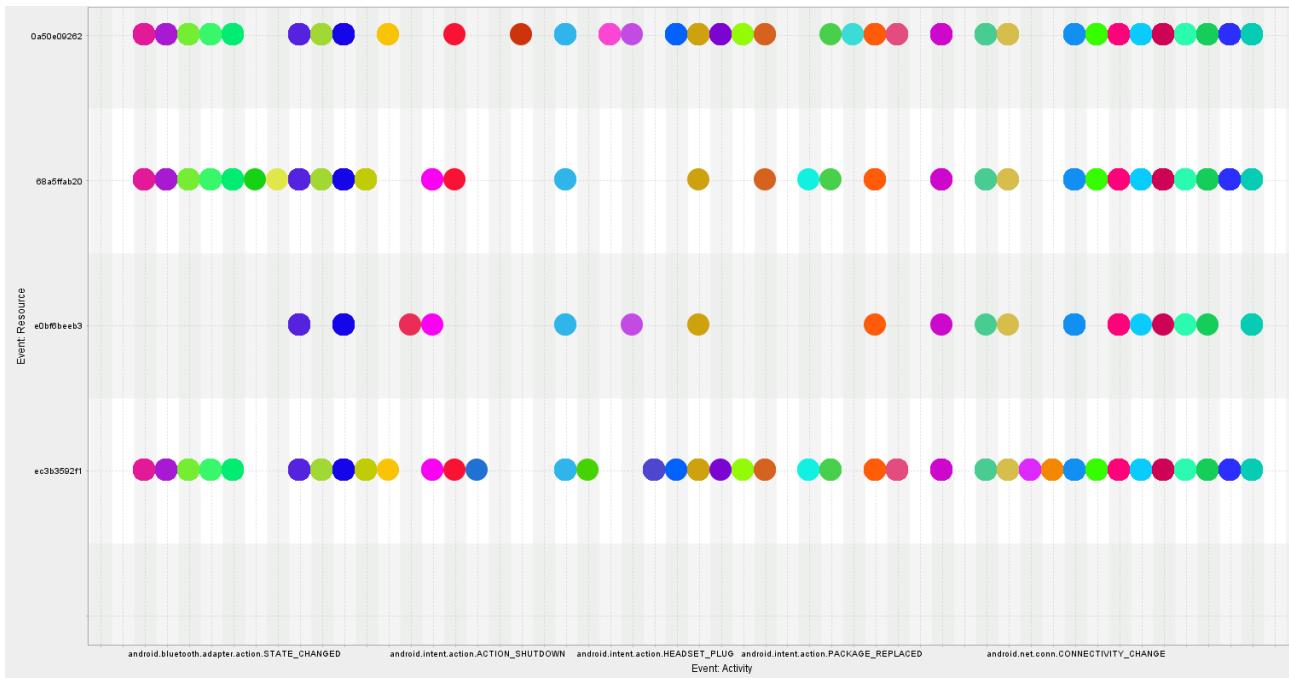
Si applica quindi il plugin con le quattro metriche, ottenendo i seguenti risultati:



Da queste risposte è possibile affermare che i task tra le varie risorse sono simili.

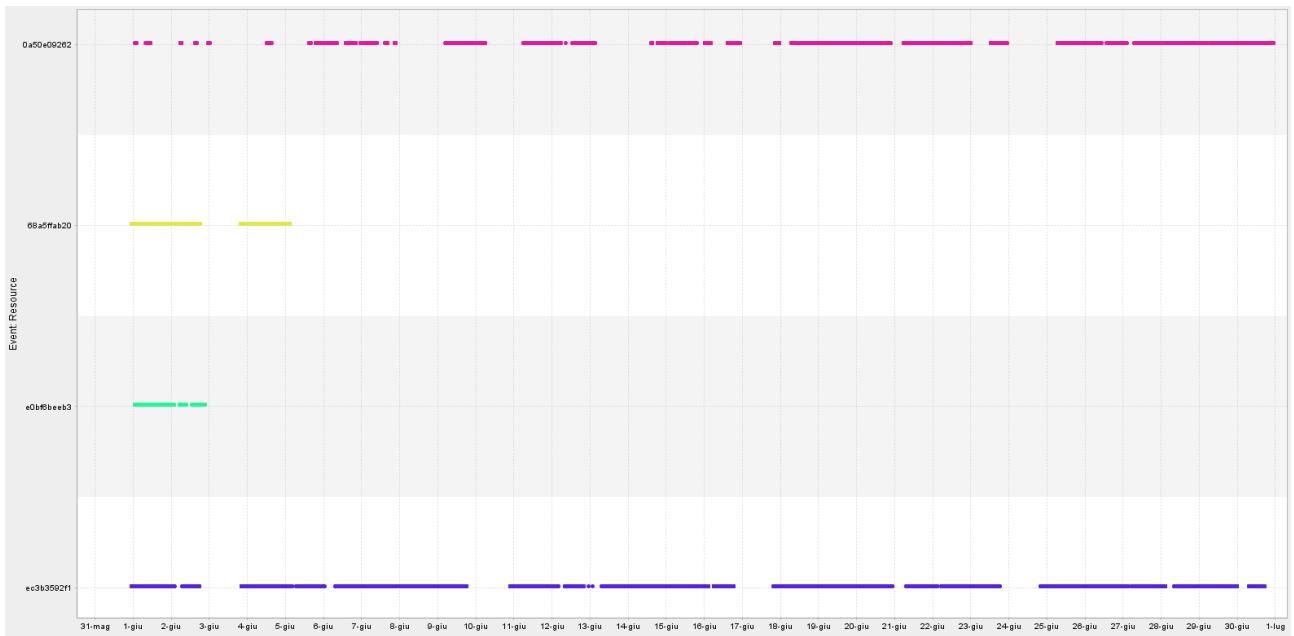
Analisi con Dotted Chart

Verrà di seguito effettuata l'analisi delle risorse tramite l'utilizzo delle Dotted Chart plugins del software ProM. Si cerca prima di tutto di capire se ci sono attività svolte unicamente da una risorsa, e attività svolte da tutte le risorse. Viene inserito quindi sull'ascissa le attività e sull'ordinata le risorse. Essendo però il numero delle attività elevato (e con nomi molto lunghi), non vi è una corretta visualizzazione sull'asse delle ascisse e si è quindi scelto di settare il colore in base alle attività, che si può vedere nel dettaglio nella leggenda sotto riportata.



È possibile notare che ci sono alcune attività svolte solo da determinati utenti. In particolare, l'attività "android.bluetooth.device.action.ACL DISCONNECTED" è svolta solo dalla risorsa 68a5ffab20, come pure l'attività "android.intent.action.DATE CHANGED" è svolta solo dalla risorsa ec3b3592f1. Molto evidente è la differenza di attività svolte tra la risorsa ec3b3592f1, che svolge quasi tutte le attività, e la risorsa e0bf6beeb3, che ne svolge un numero nettamente inferiore. Questa profonda differenza è sicuramente dovuta al fatto che per la risorsa e0bf6beeb3 si hanno dati relativi solo a 2 giorni. Comunque sia numerose sono le attività svolte da tutte le risorse, come le attività "android.intent.action.CONFIGURATION CHANGED" e "android.net.wifi.STATE CHANGE", ma non c'è nessuna risorsa che svolte tutte le attività nel periodo di tempo preso in considerazione.

Si va ora a modificare i parametri del Dotted Chart inserendo sulle ascisse il tempo e sulle ordinate le risorse:

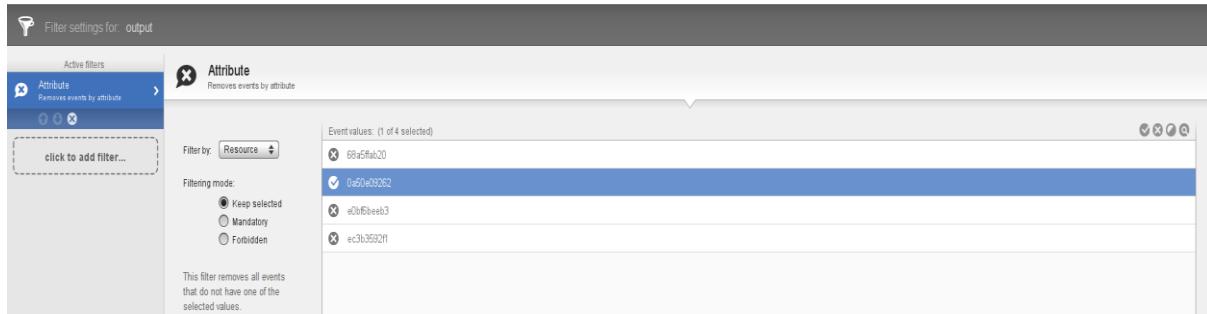


Da questo grafico è possibile prima di tutto confermare quanto detto in precedenza, ovvero che le risorse 68a5ffab20 e e0bf6beeb3 hanno pochissimi giorni di attività. Oltre a questo, si notano numerosi “buchi” di attività, molto evidenti soprattutto per l’utente 0a50e09262 nei primi giorni. Si è infatti scoperto che ogni venerdì e/o sabato del mese non vi sono raccolte di dati per quasi 24 ore, probabilmente a causa di uno spegnimento dello smartphone delle risorse.

PERFORMANCE ANALYSIS

L'analisi delle performance è un task fondamentale nel processo di mining, perché permette di studiare il sistema dal punto di vista delle sue prestazioni reali. Attraverso questa analisi sarà possibile individuare le attività critiche e i colli di bottiglia che influiscono negativamente sulla durata dell'intero processo. Al contrario, prestazioni elevate saranno indicatrici di un processo ben imposto e ben implementato.

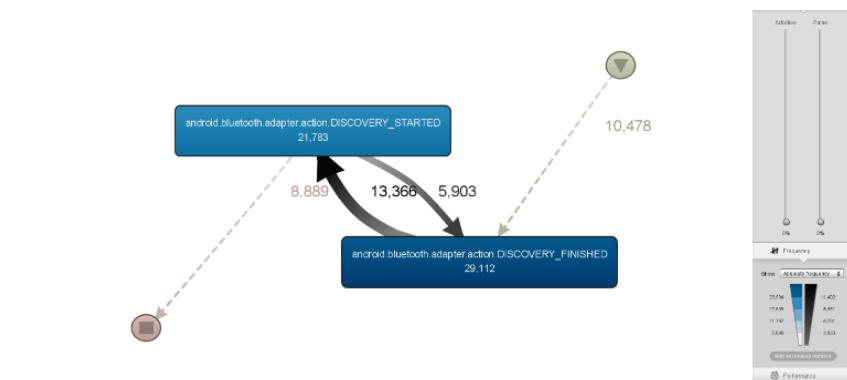
Nel dataset le risorse corrispondono agli utenti che hanno partecipato al progetto Sherlock. Ogni risorsa dunque effettuerà un utilizzo differente del proprio telefono, pertanto il primo passo per una migliore analisi delle risorse è suddividere l'intero dataset per ogni utente. Questo filtraggio è stato eseguito usando i tool di disco:



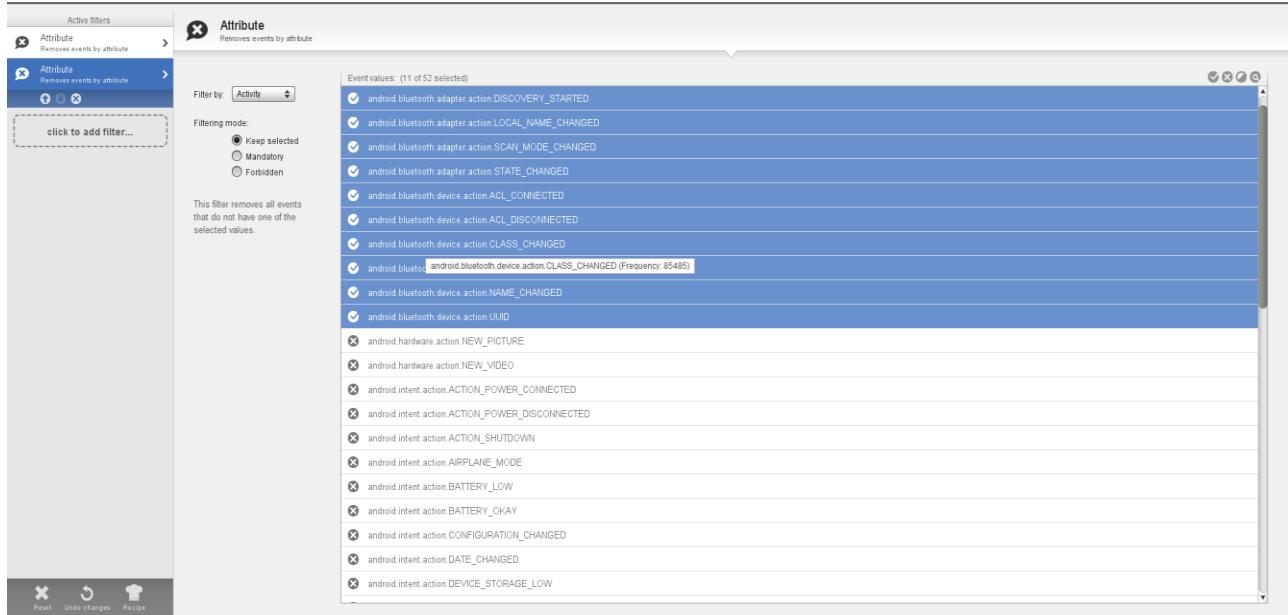
Quindi da un dataset si è passati ad avere 4 dataset che vanno a codificare come il dispositivo è stato utilizzato durante l'esperimento di raccolta dati da quella determinata risorsa presa in considerazione. Di questi 4 se ne andranno ad utilizzare solamente 2 in quanto sono 2 le risorse i cui dati vengono presi per tutta la finestra di tempo che si è presa in considerazione mentre gli utenti rimanenti cesseranno di generare eventi già dalla prima settimana. Le analisi sulle performance che si andranno ad effettuare saranno molto differenti rispetto a quelle che si effettuano di solito nel Process Mining, questo perché non si sta lavorando con un vero e proprio processo ma con un processo fittizio creato ad-hoc per tali analisi e quindi un processo che non si presta bene per l'utilizzo di tali tool. Si andranno a considerare le performance di utilizzo del dispositivo, quindi si analizzeranno quali sono le attività che saranno maggiormente eseguite in media dal dispositivo durante la giornata. Inoltre, un'ulteriore caratteristica che bisogna prendere in considerazione sono i tempi che intercorrono tra le varie attività. Essi non corrispondono alle durate delle varie attività bensì al tempo che intercorre tra l'inizio di un'attività e l'altra pertanto è possibile prendere in considerazione tali tempi per comprendere con che frequenza avvengano le attività durante la giornata.

Risorsa: 0a50e0962

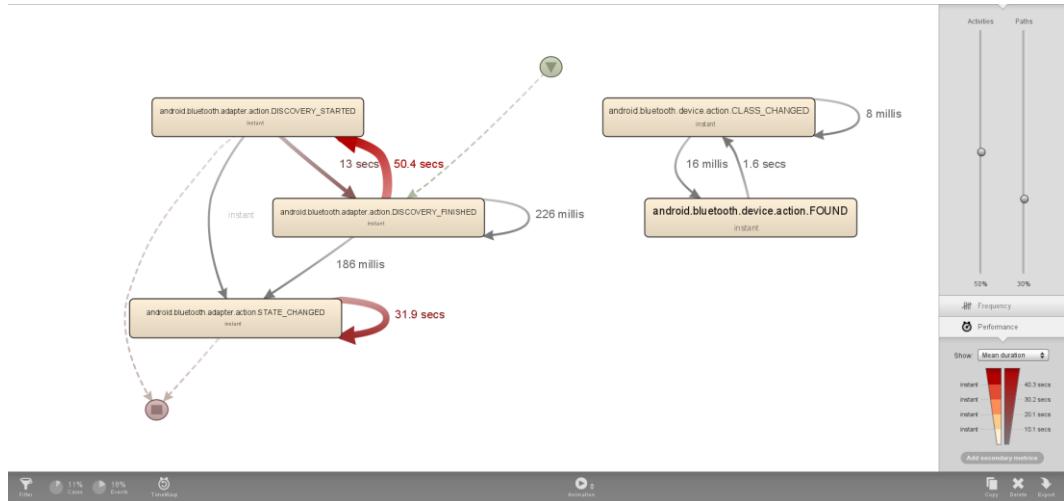
Per effettuare l'analisi delle performance si andrà ad utilizzare il tool di performance di Disco che permette di andare a lavorare in modo molto semplice con le varie durate presenti all'interno del dataset.



Si nota fin da subito che le attività più frequenti nel periodo preso in considerazione risultano essere le attività legate all'utilizzo del Bluetooth. È possibile verificare se sono distribuite durante la giornata oppure se sono concentrate in particolari momenti. Si va quindi a filtrare il dataset tenendo in considerazione solo le attività relative al Bluetooth.



Si va ora a verificare il tutto considerando inizialmente il parametro "Mean duration":



Inizialmente ci si concentra sulla scansione Bluetooth che è formata da due attività:

- DISCOVERY_FINISHED
- DISCOVERY_STARTED

Il tempo che intercorre tra DISCOVERY_STARTED e DISCOVERY_FINISHED lo si può intendere come il tempo che impiega il telefono ad effettuare una scansione ed è di circa 12 secondi che corrisponde ad un tempo non anomalo, così come viene confermato dalla relativa documentazione Android.

ACTION_DISCOVERY_STARTED

Added in API level 5

```
public static final String ACTION_DISCOVERY_STARTED
```

Broadcast Action: The local Bluetooth adapter has started the remote device discovery process.

This usually involves an inquiry scan of about 12 seconds, followed by a page scan of each new device to retrieve its Bluetooth name.

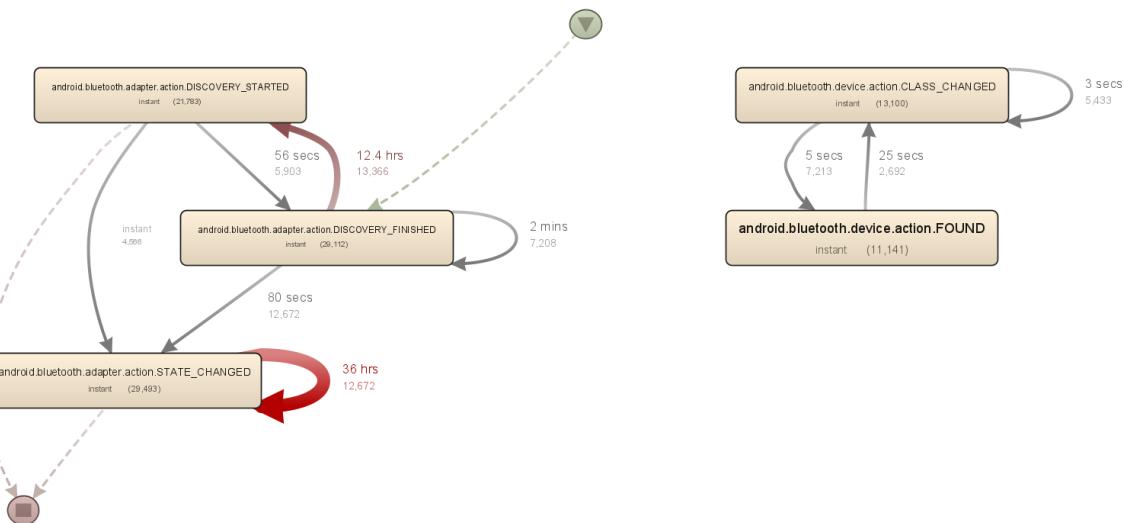
Register for `BluetoothDevice#ACTION_FOUND` to be notified as remote Bluetooth devices are found.

Device discovery is a heavyweight procedure. New connections to remote Bluetooth devices should not be attempted while discovery is in progress, and existing connections will experience limited bandwidth and high latency. Use `cancelDiscovery()` to cancel an ongoing discovery.

Requires `Manifest.permission.BLUETOOTH` to receive.

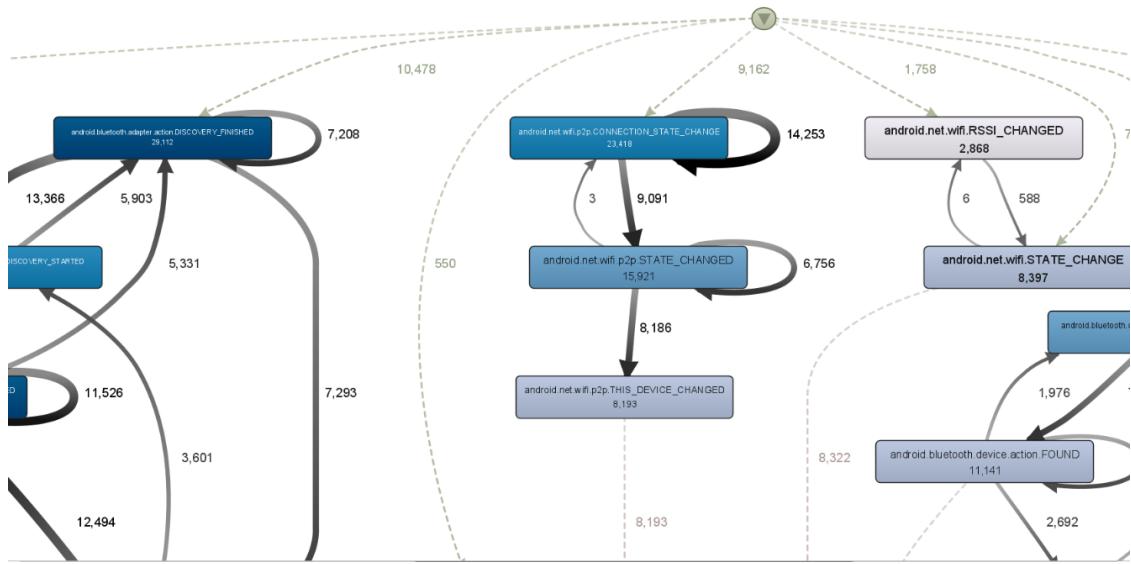
Constant Value: "android.bluetooth.adapter.action.DISCOVERY_STARTED"

Il tempo che intercorre, invece, tra DISCOVERY_FINISHED e DISCOVERY_STARTED può essere visto come il tempo che intercorre tra una scansione e l'altra, e questo fa notare che durante la giornata avvengono numerose scansioni Bluetooth. Prendendo infatti in considerazione il parametro max duration, si nota che nel caso più "lungo" passano 50.4 secondi. Questo può essere giustificato andando ad ipotizzare che durante la giornata l'utente tenga sempre acceso il Bluetooth effettuando continue scansioni. Altra attività molto frequente risulta essere STATE_CHANGED che si presenta quando lo stato dell'adattatore Bluetooth locale è stato modificato, ad esempio quando il Bluetooth viene spento o viene acceso. Quest'ultimo aspetto però va in contrasto con quanto detto in precedenza, ovvero il fatto che il Bluetooth dell'utente rimanga acceso per tutta la durata dell'esperimento, questo significa quindi che un ulteriore ipotesi che si può attuare è che, indipendentemente dall'esperimento svolto, l'utente fa un massiccio utilizzo dei dispositivi Bluetooth nella sua quotidianità. Si va ora a vedere le stesse attività ma stavolta prendendo in considerazione il massimo tempo che intercorre tra di esse.

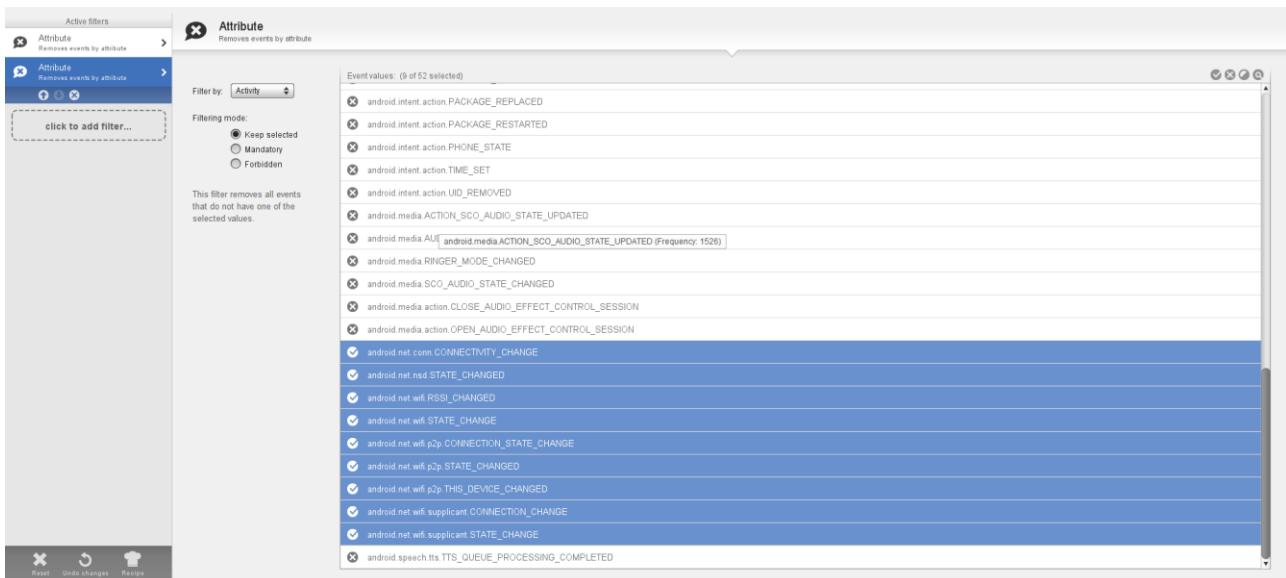


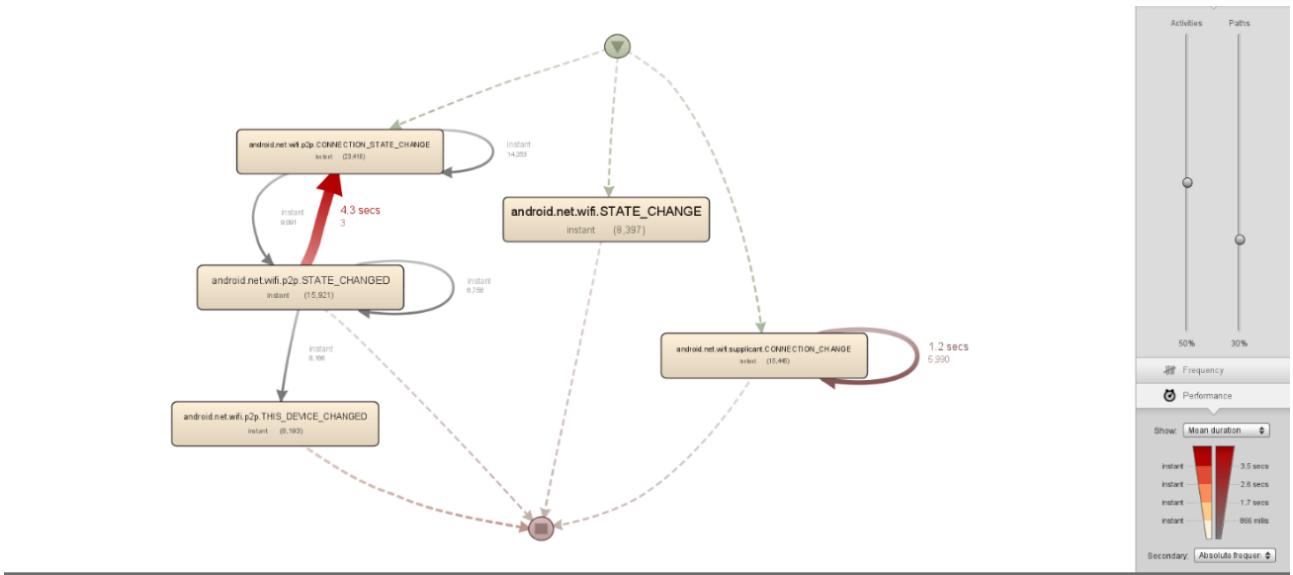
Si può vedere che il massimo tempo che intercorre tra lo svolgimento di un'azione e l'altra è stato di 12,4 ore per quanto riguarda una scansione e l'altra e 36 ore per quanto riguarda un cambio di stato e l'altro. Questo fa intuire che non tutti i giorni del mese preso in considerazione l'utente fa un utilizzo massiccio del Bluetooth del dispositivo ma sono presenti anche giornate in cui ciò non avviene e quindi si può anche confermare il fatto che i Bluetooth non sono rimasti sempre accesi per l'esperimento effettuato. Da tutto ciò è possibile concludere che, poiché si prende in considerazione un mese del dataset dove non avviene nessuna tipologia di attacco, l'utilizzo eccessivo dei Bluetooth del dispositivo di questa risorsa non è qualcosa di anomalo quindi non è dovuto a nessun malware presente nello smartphone. Questo però può essere una cosa negativa perché se ci dovessero essere degli attacchi che sfruttano questo tipo di connessioni, questi non saranno facilmente individuabili.

Si va ora a considerare tutte le possibili attività, passando alla visuale relativa alle frequenze:

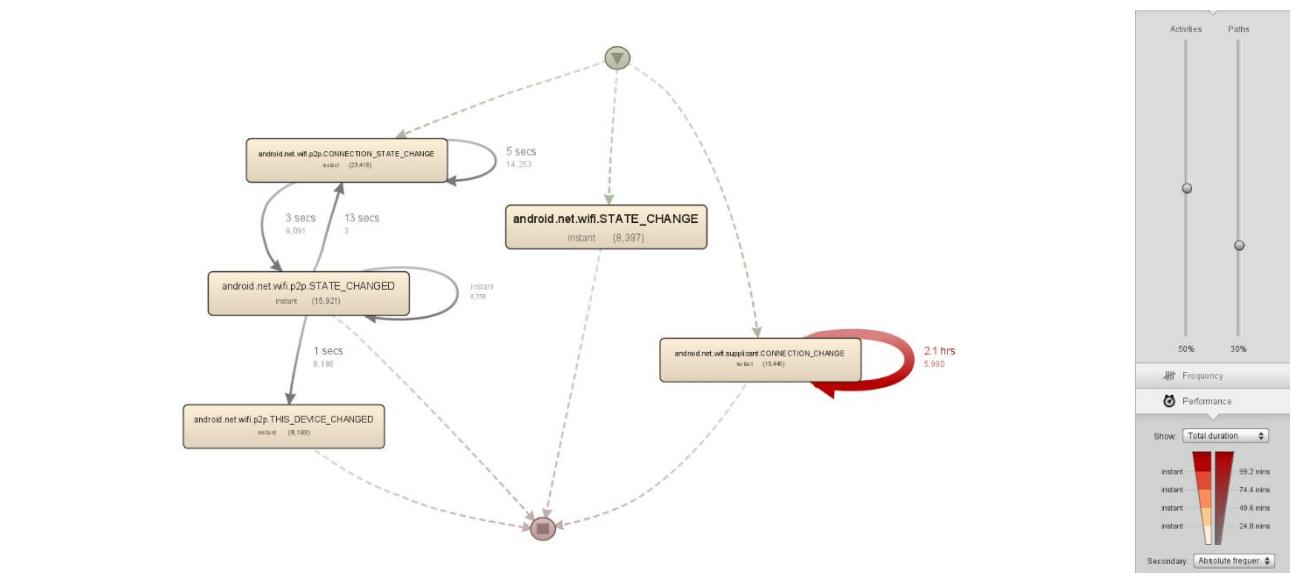
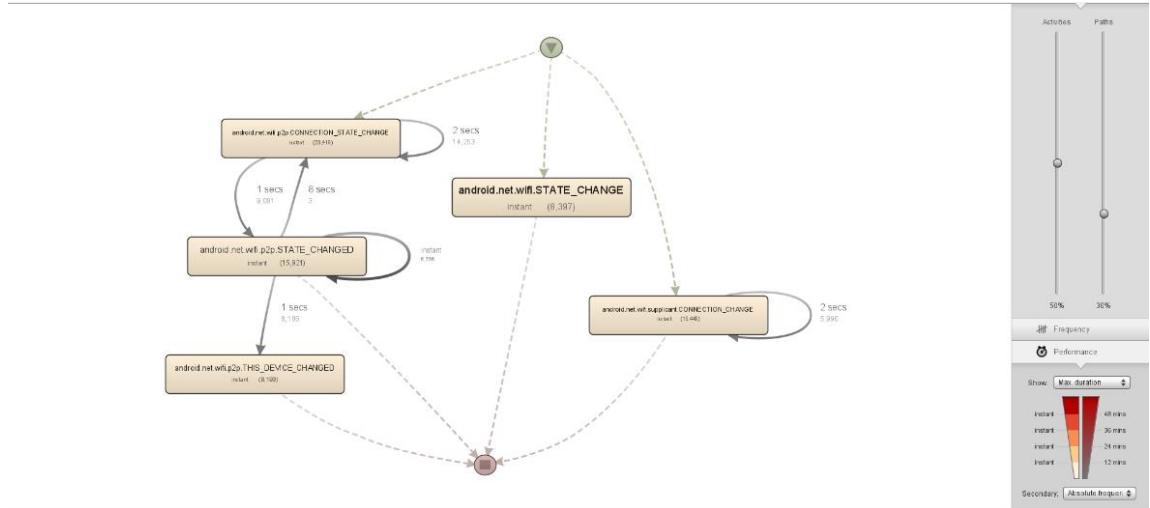


Si può notare che vi sono altre attività molto frequenti e che corrispondono ad attività legate all'utilizzo del wifi. Si va pertanto ad analizzarle meglio cercando così di capire quanto inficiano nelle performance (intese come descritto all'inizio del capitolo) del dispositivo. Si filtra il log e si utilizza il tool di performance di disco per comprendere il loro comportamento:

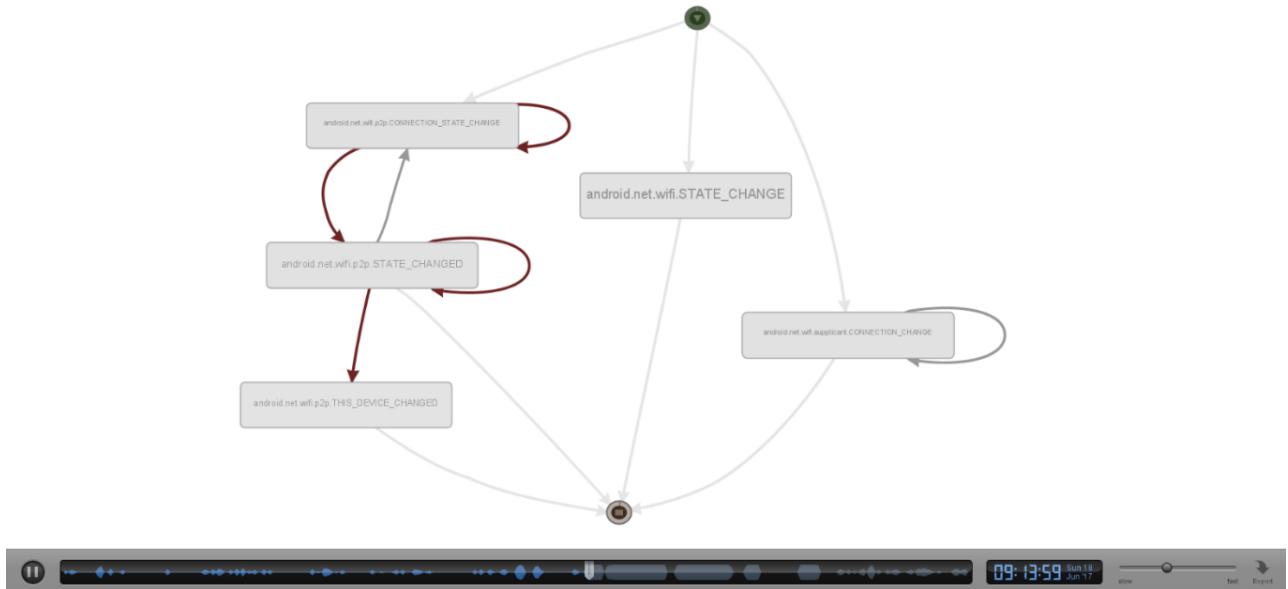




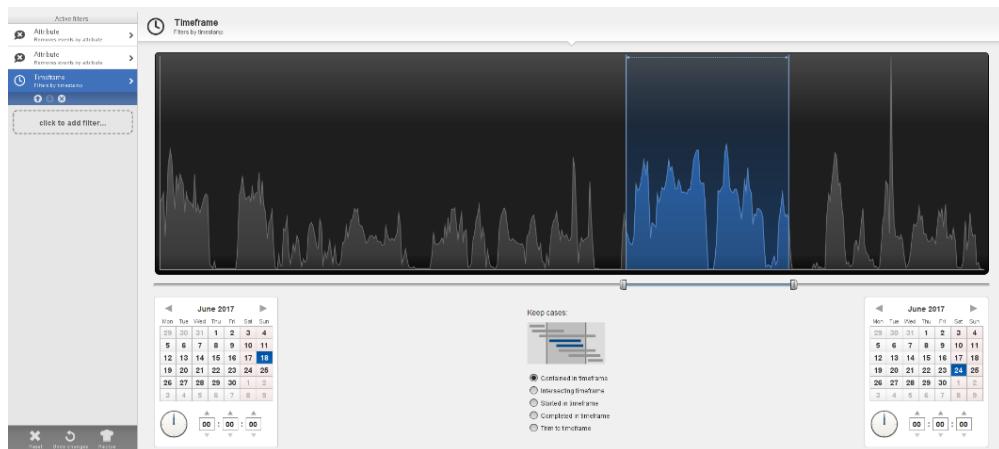
Si nota da subito che i tempi che intercorrono tra un'attività e l'altra sono molto brevi (alcuni avvengono all'istante) quindi si potrebbe intuire che durante la giornata ci sia un uso massiccio del wifi. Si prova quindi a vedere il tempo massimo e anche il tempo totale per comprendere meglio il tutto:



Come si può notare, sono presenti tempi anomali. Questo è dovuto a ciò che è stato detto ad inizio capitolo, ovvero che per come sono registrati i tempi di esecuzione delle attività, i tool di performance di disco, così come disco stesso, non riescono sempre ad attuare delle analisi congruenti. Si prova quindi a eseguire una via differente tramite l'utilizzo del tool di animazione presente in disco:



Attraverso di esso è possibile comprendere che, in realtà, il wifi non è molto utilizzato durante il giorno dall'utente e che il picco di utilizzo va dal 18 al 23 giugno. Si va pertanto ad analizzare meglio questa settimana filtrando il log:

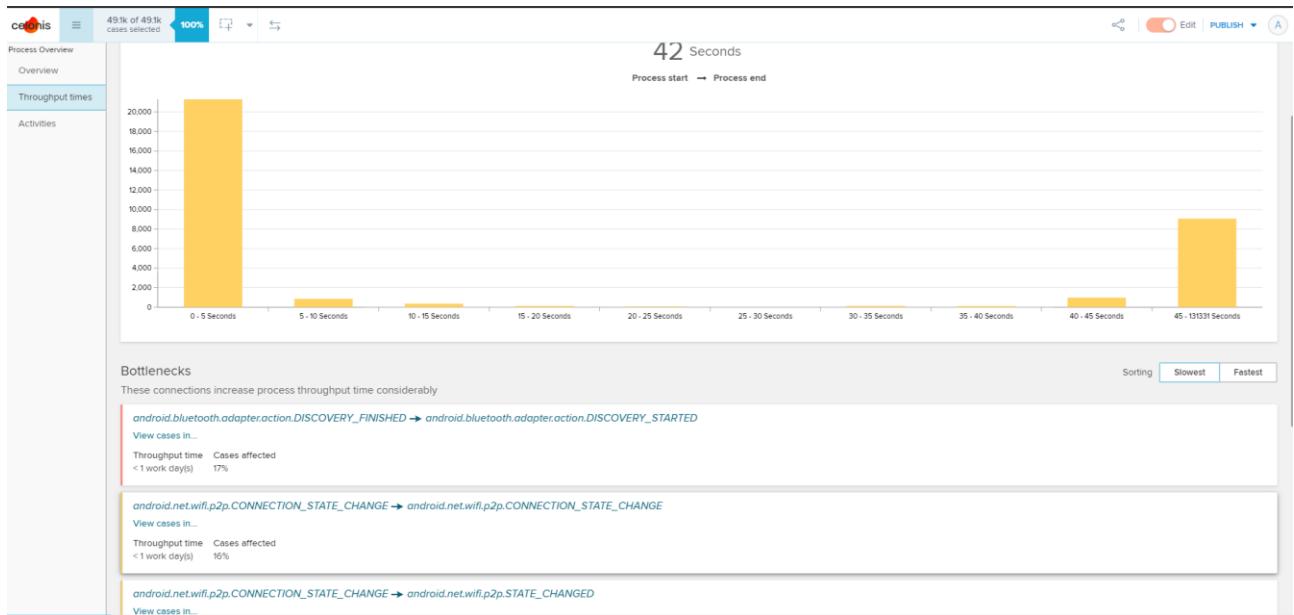


Attraverso questa analisi si ha la conferma di ciò che si è detto in precedenza. Durante tutto il mese ci sono circa 23,418 eventi di android.net.wifi.p2p.CONNECTION_STATE_CHANGE mentre nella settimana presa in considerazione si hanno 19,240 eventi, che corrispondono a circa l'82% delle attività totali, e questo accade anche per le altre attività legate al wifi.

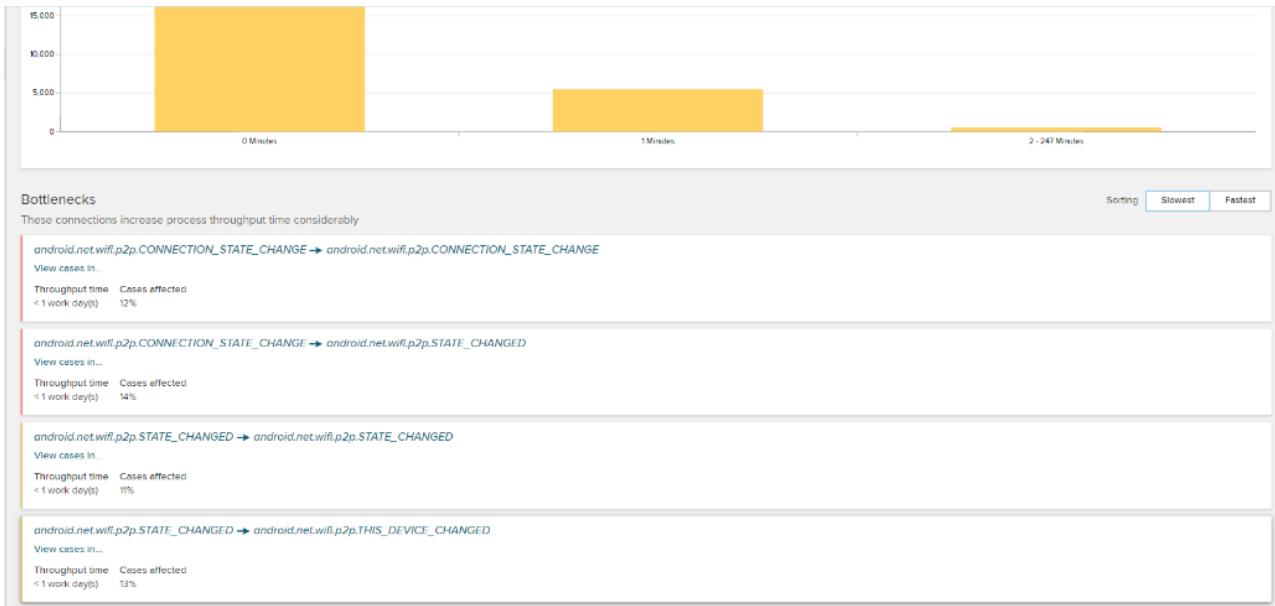


Activity	Frequency	Relative frequency
android.net.wifi.p2p.CONNECTION_STATE_CHANGE	19,240	32.66 %
android.net.wifi.p2p.STATE_CHANGED	12,951	22.01 %
android.net.wifi.p2p.RESPONSE_CONNECTION_CHANGE	12,289	21.14 %
android.net.wifi.STATE_CHANGE	6,549	11.12 %
android.net.wifi.p2p.THIS_DEVICE_CHANGED	6,508	11.05 %
android.net.wifi.RSSI_CHANGED	257	0.44 %
android.net.wifi.suplicant.STATE_CHANGE	243	0.41 %
android.net.conn.CONNECTIVITY_CHANGE	171	0.29 %
android.net.mds.STATE_CHANGED	59	0.1 %

Detto questo si può supporre che l'uso dei wifi per questo utente non è un attività svolta quotidianamente e quindi sarà più evidente trovare delle anomalie legate ad esso. Un ulteriore analisi che si può attuare consiste nell'uso della piattaforma celonis che permette di comprendere in modo molto semplice i colli di bottiglia presenti nel processo. È possibile intendere il throughput alto come un qualcosa di positivo in quanto implicherebbe che il dispositivo non è occupato tutto il giorno da quella attività (ovviamente questo lo si può dire per come sono presi i tempi nel dataset ma soprattutto per come sono stati presi in considerazione i tempi ad inizio capitolo)



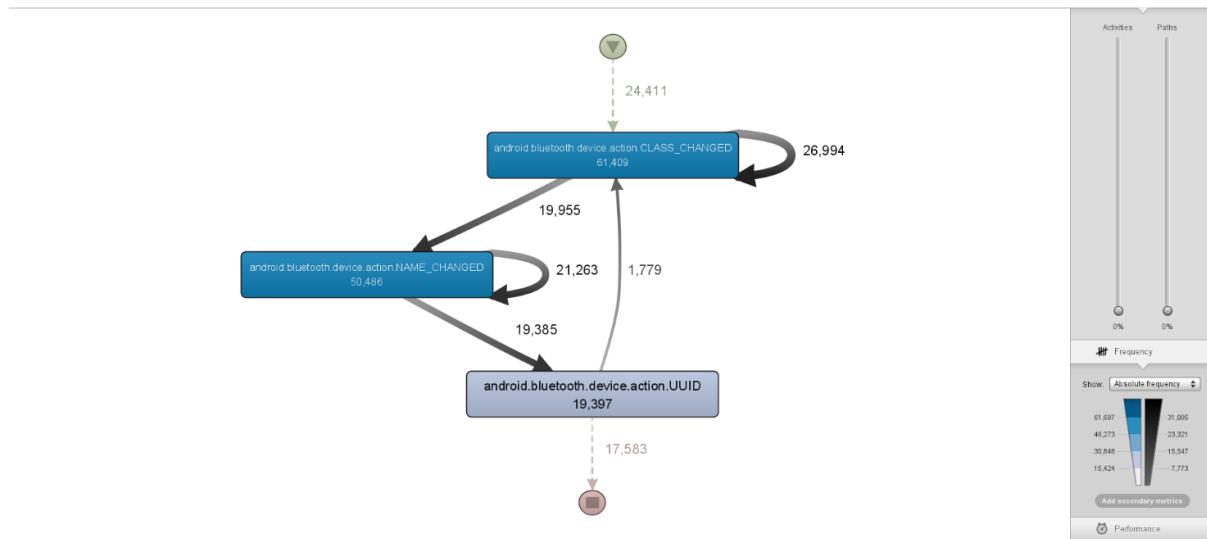
Si nota però che i risultati ottenuti non soddisfino molto l'analisi effettuata in precedenza in quanto il bottleneck lo si aspettava per le attività legate al wifi. Questo sarà sicuramente legato a come è stato registrato il tempo delle attività. Si può procedere pertanto per un'altra via, ovvero andando a modificare il modo con cui vengono visualizzati i dati prendendo come riferimento un nuovo case id. Tale case sarà formato dalla risorsa e dal timestamp. In questo modo si avranno tracce formate da tutte le attività che vengono effettuate quel giorno dal dispositivo. Una volta scelto questo case id e caricati i dati, si andranno a filtrare ulteriormente (scegliendo solo le attività che concernono l'utilizzo del wifi). Di seguito viene caricato il contenuto su Celonis:



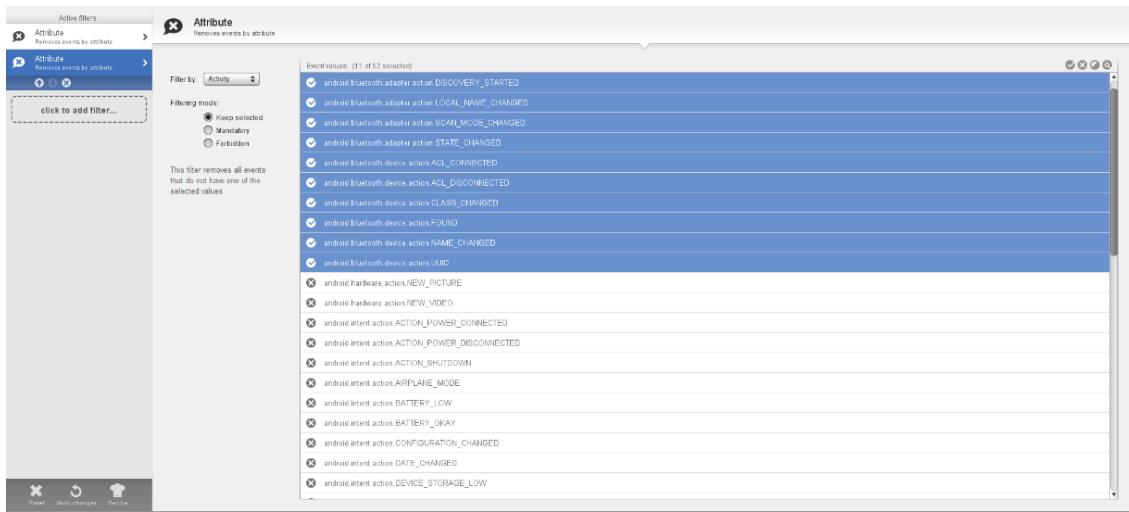
È possibile notare che il collo di bottiglia è dato dalle attività wifi come ci si aspettava, riuscendo a confermare le analisi fatte tramite il tool di performance di disco.

Risorsa: ec3b3592f1

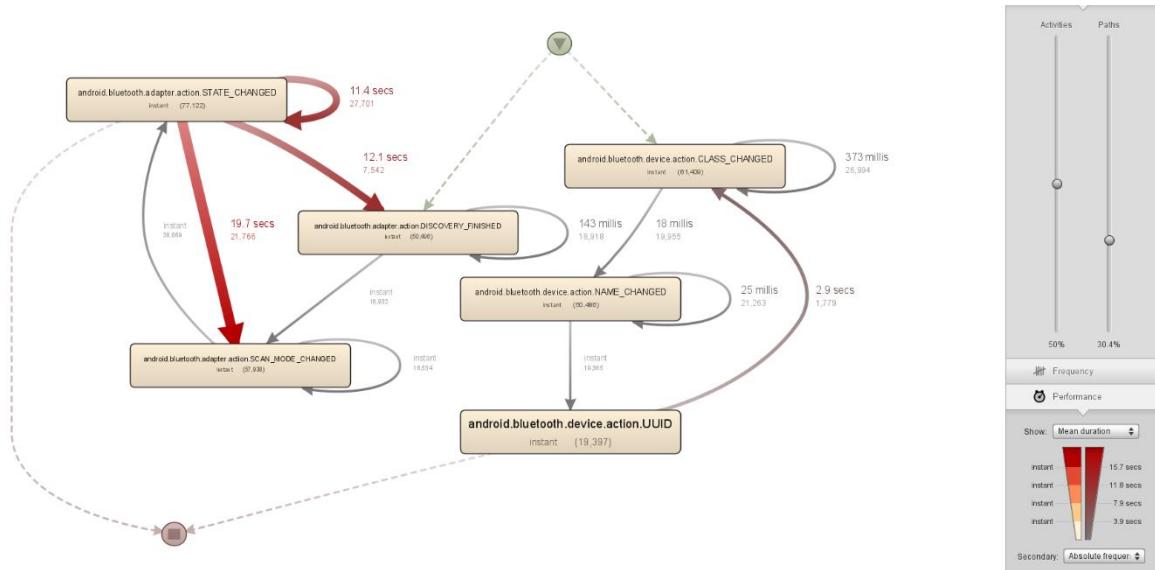
Si va ora ad effettuare le analisi per questa nuova risorsa così come fatto per la risorsa precedente. Si può notare che anche in questo caso le attività più frequenti risultano essere quelle legate al Bluetooth ma questa volta non sono relative alla scansione ma alla connessione con ulteriori dispositivi:



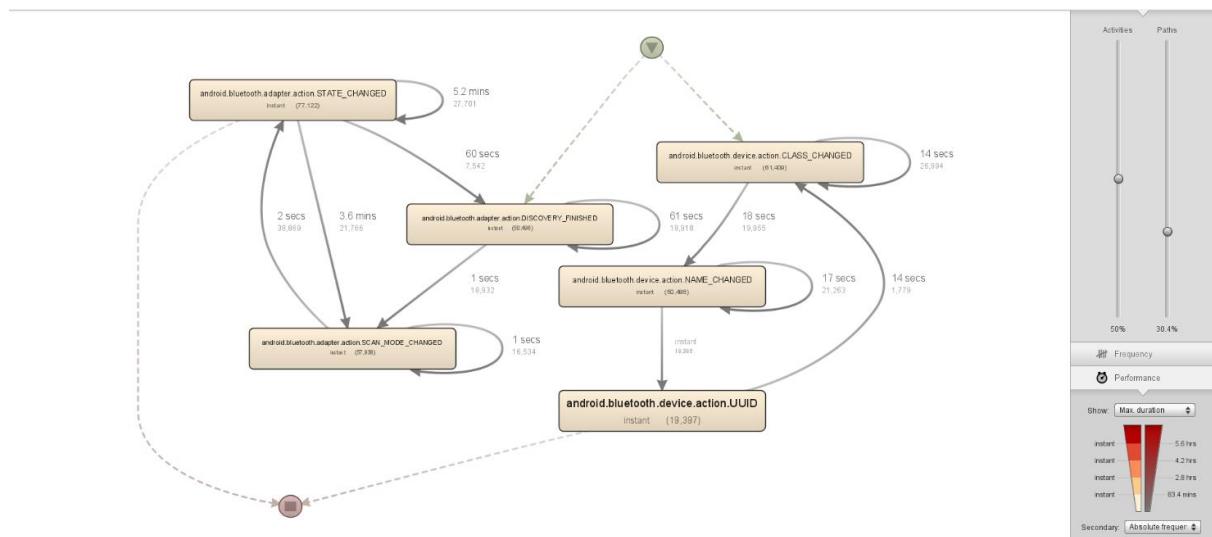
Si va quindi a filtrare il dataset considerando solo le attività legate al Bluetooth:



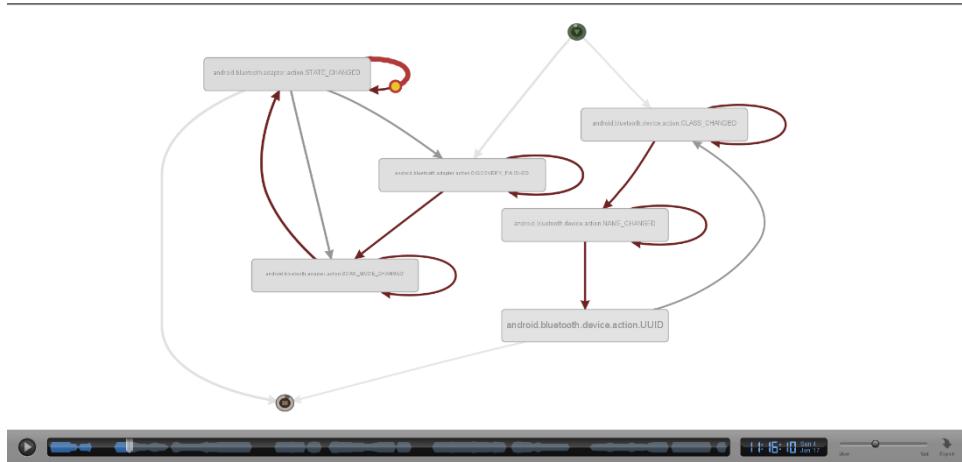
Si analizza il risultato ottenuto tramite il parametro mean duration:



I tempi che intercorrono tra le varie attività sono molto piccoli quindi ora si andrà ad utilizzare il parametro Max duration per vedere se ogni giorno viene fatto un uso elevato del Bluetooth:

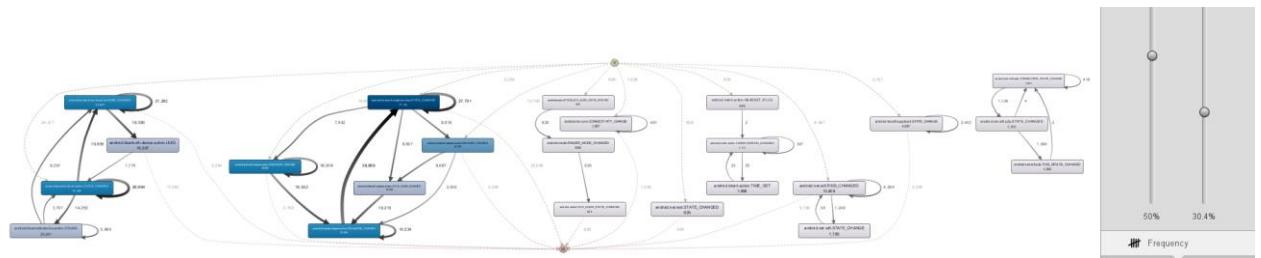


Si nota che anche in questo caso i tempi non sono molto elevati quindi si rileva un uso maggiore rispetto all'utente precedente. Per comprendere meglio la distribuzione è possibile andare ad utilizzare il tool animazione di disco per comprendere l'andamento del flusso delle attività:



Si nota che quanto detto in precedenza risulta essere corretto ed inoltre che ogni fine settimana l'utilizzo del Bluetooth scende a zero. Quindi è molto probabile che durante questo periodo l'utente spenga il telefono oppure i Bluetooth.

Si vanno ora a considerare anche le altre attività:



Si nota che in questo caso il dispositivo sia per lo più occupato da attività legate al Bluetooth e non vi sono ulteriori attività con una frequenza elevata.

L'analisi delle performance effettuata considerando i tempi di utilizzo delle risorse del dispositivo, quindi, ha fatto comprendere che l'utilizzo del Bluetooth per i due utenti è comune nella giornata quindi se saranno presenti delle anomalie legate ad esso risulterà essere più complesso riuscire a scovarle. Viceversa, tutte le altre attività non sono molto presenti pertanto possono essere monitorate più facilmente in caso di comportamenti più anomali del solito. Ovviamente queste analisi non sono affidabili al 100% in quanto il metodo con cui sono state registrate le attività e i tempi all'interno del dataset vanno ad influenzare negativamente l'utilizzo di tool come ad esempio il tool di performance di disco.

ANOMALY DETECTION

Detto nella maniera più semplice possibile, il processo di Anomaly Detection consiste nel riconoscimento di condizioni o circostanze inattese all'interno di un dataset. Per contenuto o evento inatteso si intende qualcosa che differisce dalla norma. Si tratta di uno degli ambiti più fertili in ambito di sicurezza informatica.

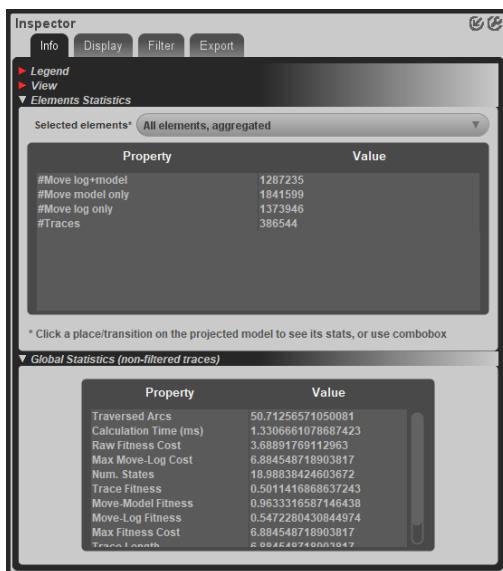
In questo capitolo si andrà ad analizzare il dataset con le tecniche di Process Mining apprese durante il corso, con il fine di scovare l'eventuale presenza di anomalie nei dispositivi degli utenti che hanno partecipato al progetto e capire quindi quali sono le risorse intaccate da essi. Il dataset che verrà preso in considerazione per lo studio delle anomalie corrisponde a quello la cui finestra temporale va dal 1 aprile al 30 aprile in quanto, come letto nella descrizione del progetto Sherlock, è un periodo in cui sono stati registrati degli attacchi da parte del software malevolo **moriarty**.

Quello che si andrà a fare sarà utilizzare il task di conformance per comprendere se sono presenti dei case con un path che devia dal major behavior. Attraverso lo studio effettuato nei capitoli precedenti infatti, si è riusciti ad ottenere un modello che rispecchia il normale comportamento che gli utenti hanno avuto durante il periodo in cui non sono stati presenti gli attacchi, quindi, va a rappresentare ciò che gli utenti svolgono con il proprio smartphone durante la giornata. Tale modello sarà utilizzato nella conformance checking insieme al log descritto in precedenza. Tramite l'utilizzo del plugin di proM “**replay a log petri net for a conformance analysis**” si otterrà la misura di quanto il modello fitta con il log e pertanto si potrà comprendere se sono avvenuti degli attacchi o comunque se sono presenti dei comportamenti anomali che si discostano da quello comune. Tale procedura di Anomaly Detection è una delle procedure più usate quando si cerca di combinare aspetti riguardanti la sicurezza con le tecniche di process mining. Sono presenti infatti paper che ne descrivono il funzionamento con annessi risultati, tra questi vi è il paper **Anomaly Detection using Process Mining** (Fabio Bezerra¹, Jacques Wainer¹, and W.M.P. van der Aalst²) al quale si è fatto riferimento per questo capitolo del progetto.

L'analisi verrà effettuata su gli utenti separatamente, in particolare si andranno a visionare i log dei due utenti **0a50e09262**, **ec3b3592f1** che corrispondono alle risorse i cui dati sono stati raccolti per tutta la finestra temporale di giugno, quindi sono le risorse su cui noi abbiamo costruito il modello di processo.

Risorsa: **ec3b3592f1**

Per prima cosa si andrà ad avviare il plugin di proM come detto in precedenza:



Si nota subito che il valore di fitness ottenuto è minore rispetto a quello che si è raggiunto nei capitoli precedenti (0.7-0.8). Questo è probabilmente dovuto al fatto che il log utilizzato non è stato modificato in alcun modo, ovvero non sono stati applicati filtri o manipolazioni, pertanto saranno presenti alcune tracce il cui comportamento non è stato codificato nel modello principale. L'abbassamento di questo valore può essere inoltre dovuto alla presenza di qualche anomalia, pertanto sarà necessario effettuare altre analisi per comprendere meglio il tutto. Il prossimo passo, pertanto sarà quello di verificare il numero di tracce che non fittano con il nostro modello tramite il tool “**Check Conformance using ETConformance**”:

Metric

- ETC Precision Metric (ETCp): **0.6465**
- Gamma: **0.0**
- Heuristics: **Lazy Invisible**

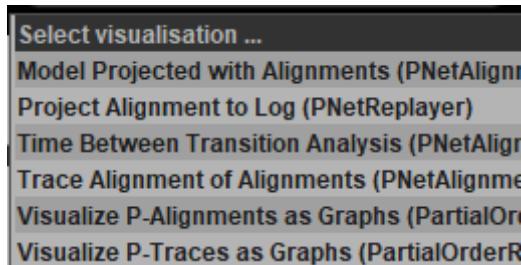
Log and Model

- Model: **Petri net**
- Log: **Anonymous log imported from outputAnomaly.xes.gz**
- |Traces|: **386544**
- |Non Fit Traces|: **277434 of 386544**
- Average Trace Size: **7.0**

Prefix Automaton

- IN: **1273**
- 0-ESCAPING: **3814**
- Gamma-ESCAPING: **0**
- OUTER: **0**
- NON FIT: **83100**
- NON DET: **0**
- TOTAL: **88187**

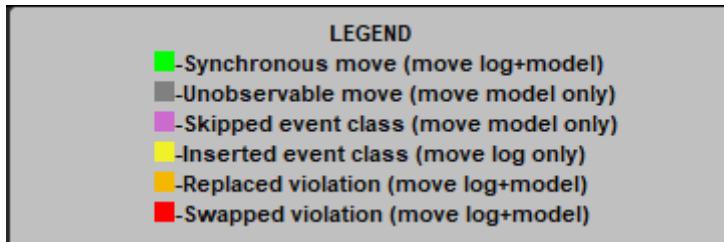
Si nota fin da subito che su 386544 tracce si hanno 277434 tracce che non fittano e che corrispondono ad un numero molto elevato. Una prima ipotesi per giustificare tali risultati è che all'interno del log possono essere presenti dei cicli che non si riescono a rappresentare correttamente però, per avere una maggiore certezza su ciò che si è ottenuto, si vanno ad esaminare le varie tracce che non fittano con il plugin “**replay a petri net for a conformance analysis**” e selezionando la seconda tipologia di view (project alignment to log) si ottiene una visualizzazione dell'alignment effettuato su di esse:



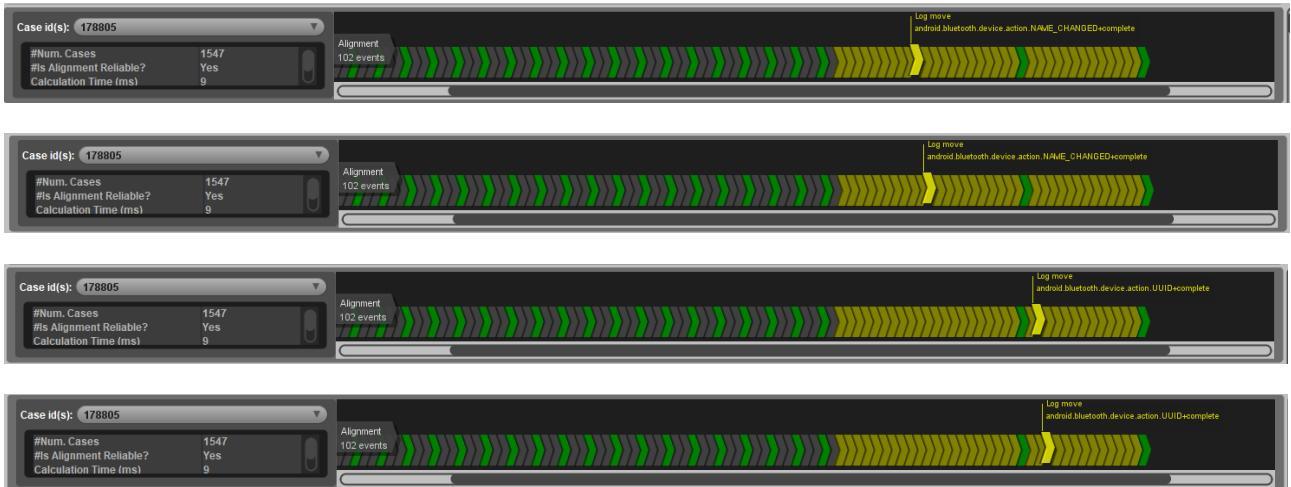
Si ottiene pertanto una view così formata:



Tale view fa riferimento alla legenda:



Attuato questo passo si vanno a verificare le tracce più anomale, ovvero quelle tracce che presentano più move effettuate al loro interno. Si prende in considerazione pertanto una tra le tracce con più move effettuate che corrisponde alla traccia con id pari a 178805:



Da ciò che è possibile vedere nell'immagine precedente, uno dei problemi che ha portato ad un valore di fitness basso è la presenza di cicli nel log che non vengono codificati correttamente nel modello. Si sta lavorando con un dataset che non rappresenta un vero processo ma si è cercato di realizzare un processo artificiale per essere analizzato attraverso le tecniche di process mining, quindi la presenza di cicli all'interno del log è una cosa molto comune in quanto rappresentano le attività svolte dal telefono che non sono sempre legate tra di loro e spesso si ripetono più volte nell'arco della giornata con frequenze differenti, quindi difficili da codificare in un modello di processo. Si va ora ad analizzare ulteriori tracce per comprendere se le numerose move siano dovute soltanto alla presenza di cicli non correttamente rappresentati.

CaselD 170174:

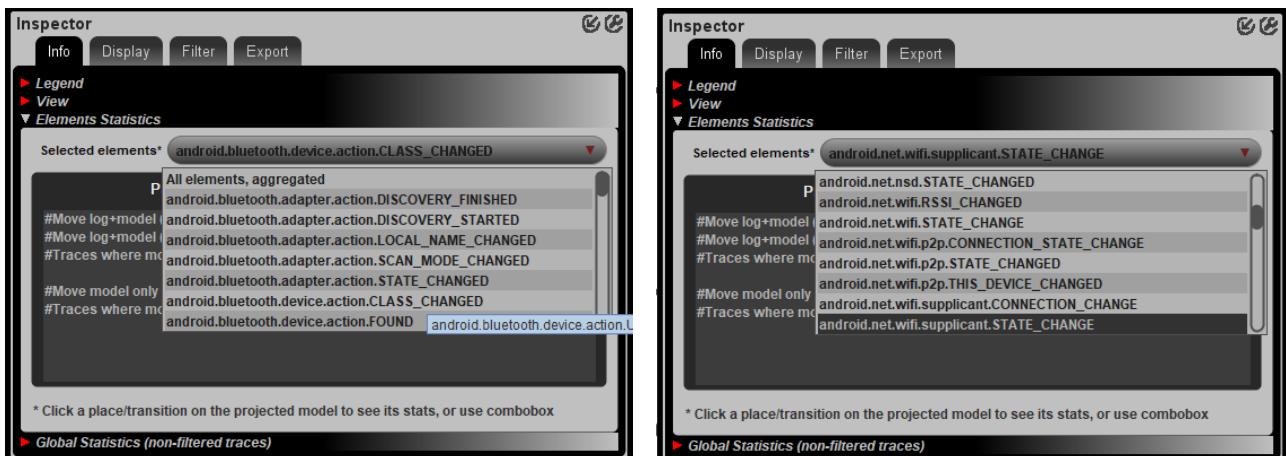


Anche in questo caso il problema è dovuto ai cicli presenti nel case. Si analizza un ulteriore caso che risulta essere particolare, ovvero quello identificato dal case 158270:

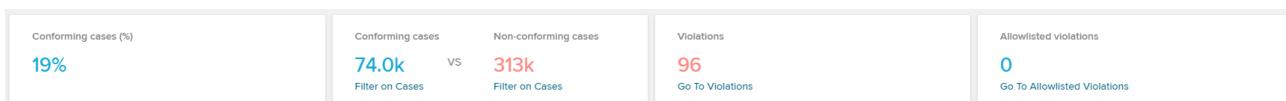


In questo caso vi sono due move del log che non hanno a che fare con un ciclo ma corrispondono all'avvio di una scansione e al cambiamento di stato del Bluetooth. Queste azioni essendo delle move log (come riportato nella legenda) corrispondono a delle attività inserite all'interno della traccia del log affinché fittasse con il modello usato. Questo potrebbe indicare che il numero di scansioni o comunque di utilizzo del Bluetooth sia minore rispetto a quello effettuato nel periodo di analisi senza la presenza di attacchi. Questo potrebbe indicare la presenza di qualche malware che vada ad intaccare il funzionamento del Bluetooth.

Una volta notato ciò, ci si sofferma sulle analisi delle deviazioni analizzando le move effettuate ma stavolta utilizzando le info presenti nella view principale del plugin “**Replay a petri nets for a conformance analysis**”. Si nota che avvengono molte move legate al Bluetooth ed al wifi:



Si va ad utilizzare ora il tool di conformità di Celonis:



Si può notare che anche in questo caso la conformità è molto bassa.

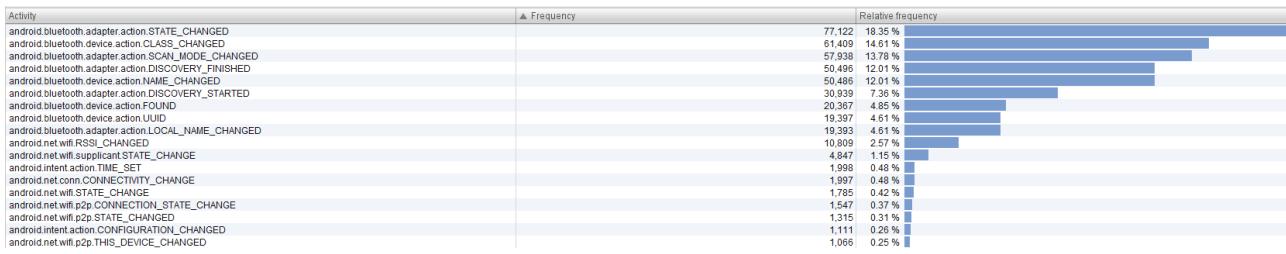
19% of cases	<i>android.bluetooth.device.action.NAME_CHANGED</i> is followed by <i>android.bluetooth.device.action.NAME_CHANGED</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days shorter + 15.4 Steps per case
16% of cases	<i>android.bluetooth.device.action.UUID</i> is followed by <i>android.bluetooth.device.action.UUID</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days shorter + 171 Steps per case
14% of cases	<i>android.bluetooth.adapter.action.DISCOVERY_FINISHED</i> executed as <i>START</i> activity Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days longer + 4.1 Steps per case
12% of cases	Incomplete case Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days longer + 0.7 Steps per case
11% of cases	<i>android.bluetooth.adapter.action.SCAN_MODE_CHANGED</i> is followed by <i>android.bluetooth.adapter.action.SCAN_MODE_CHANGED</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days longer + 8.2 Steps per case
10% of cases	<i>android.bluetooth.adapter.action.STATE_CHANGED</i> is followed by <i>android.bluetooth.adapter.action.STATE_CHANGED</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days longer + 7.9 Steps per case

Con Celonis però si può avere la conferma che la maggior parte dei problemi sono dovuti alla presenza di cicli. Nonostante questo, sono presenti anche alcune tracce che non rappresentano dei cicli ma che comunque non fittano con il modello. Tali tracce sono ovviamente più interessanti da analizzare anche se non si ha comunque la certezza che ciò sia dovuto alla presenza di un malware.

3% of cases	<i>android.bluetooth.adapter.action.SCAN_MODE_CHANGED</i> is followed by <i>android.bluetooth.adapter.action.STATE_CHANGED</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days longer + 6.7 Steps per case
2% of cases	<i>android.bluetooth.adapter.action.LOCAL_NAME_CHANGED</i> is followed by <i>android.bluetooth.adapter.action.STATE_CHANGED</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days longer + 8.1 Steps per case
2% of cases	<i>android.net.wifi.suplicant.CONNECTION_CHANGE</i> is followed by <i>android.net.wifi.suplicant.CONNECTION_CHANGE</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days shorter + 3.5 Steps per case
2% of cases	<i>android.bluetooth.device.action.NAME_CHANGED</i> is followed by <i>android.bluetooth.device.action.CLASS_CHANGED</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days shorter + 15.2 Steps per case

L'analisi effettuata tramite la conformance checking ha portato a soffermarsi su alcune attività chiave ma, nonostante ciò, non sono state rilevate deviazioni evidenti o pericolose in cui sono presenti delle primitive che vadano ad intaccare il corretto funzionamento del dispositivo.

Un ulteriore step dell'Anomaly detection consiste nell'andare a visualizzare le frequenze delle attività. Un'anomalia infatti potrebbe consistere nell'eccessivo utilizzo di una parte dell'hardware rispetto a ciò che si era registrato nel mese di giugno. Questo tipo di tecnica è stata utilizzata all'interno del paper **"Applying Process Mining Techniques to DNS Traces Analysis"** dove si analizzavano le anomalie presenti nei flussi di pacchetti DNS eseguendo delle analisi che venivano fatte utilizzando i tool di frequenza di disco. Il primo passo consiste nell'andare a verificare le frequenze delle attività del mese in cui non erano presenti degli attacchi:



successivamente si andranno a considerare le frequenze delle attività presenti nel mese di aprile:



Si può notare che vi è un numero minore di attività del Bluetooth, conforme a ciò che si è detto nell'analisi precedente, ma si nota anche che c'è una crescita eccessiva legata all'uso del wifi. Infatti, si nota che **android.net.wifi.suplicant.STATE_CHANGE** passa da 1.15% a 5.4% (più del doppio), **android.net.wifi.p2p.CONNECTION_STATE_CHANGE** passa da 0,37% a 2,89% e così via. Questa crescita può essere pertanto dovuta alla presenza di qualche malware che sfrutta molto il wifi.

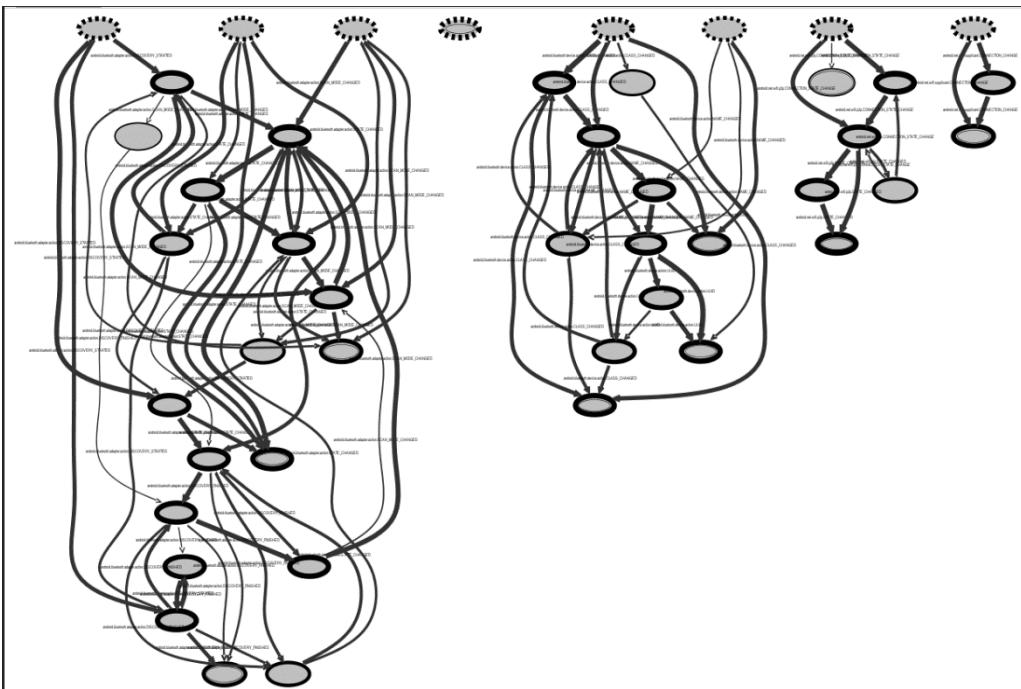
Come ultimo passo dell'analisi di questa risorsa si va ad analizzare le informazioni extra presenti nel dataset di riferimento. Per i Bluetooth è possibile notare che ci sono semplici connessioni con dispositivi audio (HTC BS).

Activity	Resource	Date	Time	Extra
1 android.bluetooth.device.action.CLASS_CHANGED	ec3b3592f1	19.04.2017	13:35:36	android.bluetooth.device.extra.CLASS: 1100android.bluetooth.device.extra.DEVICE: 04:C2:3E:BD:37:03n
2 android.bluetooth.device.action.CLASS_CHANGED	ec3b3592f1	19.04.2017	13:35:36	android.bluetooth.device.extra.CLASS: 340408android.bluetooth.device.extra.DEVICE: A0:14:3D:14:D0:18n
3 android.bluetooth.device.action.NAME_CHANGED	ec3b3592f1	19.04.2017	13:35:36	android.bluetooth.device.extra.DEVICE: 04:C2:3E:BD:37:03nandroid.bluetooth.device.extra.NAME: HTC BS-BD3703n
4 android.bluetooth.device.action.NAME_CHANGED	ec3b3592f1	19.04.2017	13:35:36	android.bluetooth.device.extra.DEVICE: A0:14:3D:14:D0:18nandroid.bluetooth.device.extra.NAME: New Yorkn
5 android.bluetooth.device.action.UUID	ec3b3592f1	19.04.2017	13:35:36	android.bluetooth.device.extra.DEVICE: A0:14:3D:14:D0:18nandroid.bluetooth.device.extra.UUID: [Landroid.os.Parcelable;@a3ddaa0n

Per il wifi invece si nota che vengono effettuate molte più scansioni rispetto a ciò che accadeva nel mese di giugno:

Activity	Resource	Date	Time	Extra
1 android.net.wifi.suplicant.STATE_CHANGE	ec3b3592f1	03.04.2017	17:08:30	newState: SCANNINGn

Per concludere l'analisi, di seguito verranno riportati i risultati ottenuti tramite l'utilizzo in catena di tool "**Mine Transition System**" e "**Animate transition System**". Per poter far ciò si è ripartiti dal log di giugno per poter estrapolare il major behavior sotto questo nuovo formalismo. Dopo vari test sui parametri presenti nel tool il risultato ottenuto è questo:



Come si può notare dall'immagine, il major behavior va a codificare le più frequenti attività che il dispositivo gestisce. Partendo da sinistra a destra i grafi che vengono generati:

- Gestione ricerca e connessioni Bluetooth
- Gestione pairing e sincronizzazione Bluetooth
- Gestione wifi direct
- Gestione wifi

Il modello ottenuto verrà dato in input al secondo plugin ottenendo una visione specializzata alla visualizzazione delle performance:



In seguito, viene riportato e discusso il risultato:



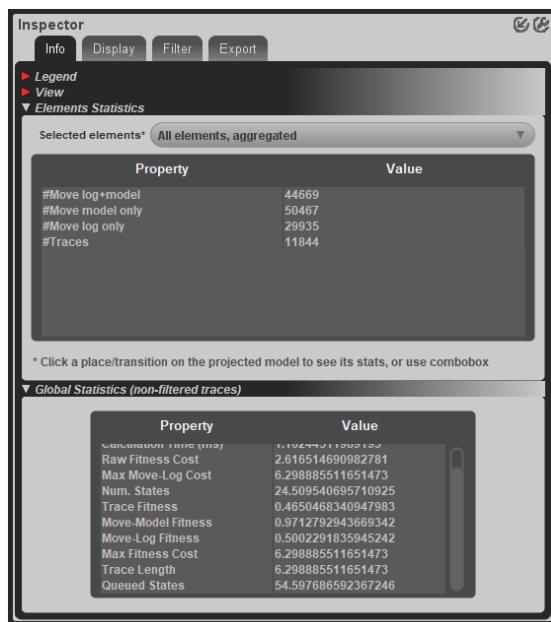
Questo è il comportamento di riferimento che si ottiene dal dataset di giugno. Successivamente si andrà a comparare tale fenomeno con il dataset di aprile, analizzando la differenza della frequenza delle attività:



Come si può notare dai risultati vi è un uso massiccio di tutte e tre le componenti, soprattutto della componente relativa al wifi (grafo all'estrema destra delle immagini) che conferma alcune delle considerazioni precedentemente fatte sulle frequenze e sulle potenziali anomalie. Un aspetto da sottolineare è che dalle analisi effettuate in precedenza si evince come l'uso delle primitive legate al Bluetooth sia in realtà inferiore, anche se in questo caso il grafico risulta essere fuorviante in quanto la complessità del grafo che viene estrapolato da un log non filtrato (per catturare le eventuali anomalie) è maggiore e per tale motivo vengono visualizzati più path simulati nello stesso istante. Altra caratteristica è che rispetto all'analisi sul dataset di giugno, sono più numerosi i path in rosso (relativi alla gestione del Bluetooth) che si vanno a spegnere durante la simulazione. Tale considerazione conferma quanto è stato precedentemente scritto.

Risorsa: 0a50e09262

Anche per questa risorsa si andranno a svolgere le stesse analisi viste in precedenza:



Si nota che la fitness ottenuta è minore rispetto alla risorsa precedente, si passa infatti da uno 0.5 a uno 0.46. Si attua il plugin **“Check Conformance using ETConformance”** per verificare il numero di tracce che non fittano:

Metric

ETC Precision Metric (ETCp): **0.4739**
 Gamma: **0.0**
 Heuristics: **Lazy Invisible**

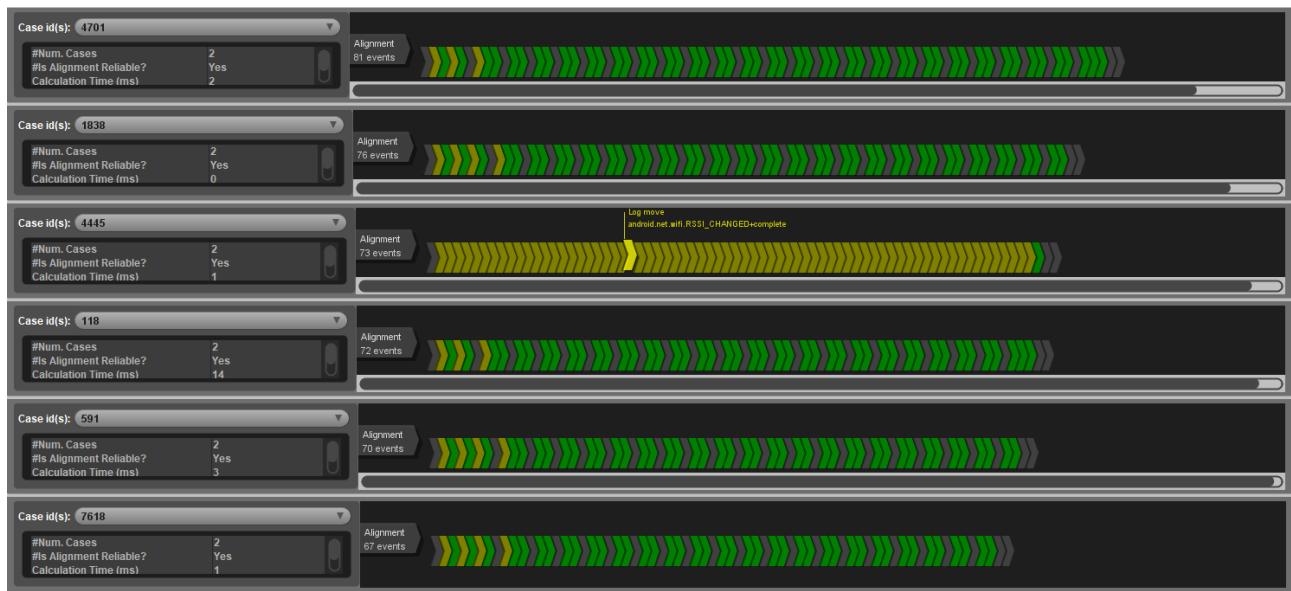
Log and Model

Model: **Petri net**
 Log: **Anonymous log imported from allBOrdered3Aprile.xes.gz**
 |Traces|: **11844**
 |Non Fit Traces|: **7632** of **11844**
 Average Trace Size: **6.0**

Prefix Automaton

IN: **58**
 0-ESCAPING: **56**
 Gamma-ESCAPING: **0**
 OUTER: **0**
 NON FIT: **3038**
 NON DET: **0**
 TOTAL: **3152**

Si nota che anche in questo caso il numero di tracce che non fittano è eccessivo rispetto al numero di tracce totali. Una cosa molto particolare è che il numero di move effettuato è minore rispetto a quelle effettuate nella risorsa precedente (prendendo sempre in considerazione il numero differente di attività totali delle due risorse):



Le poche move presenti sono dovute anche in questo caso ai cicli. Si va pertanto ad eseguire il tool di Celonis per avere delle conferme sulle analisi effettuate:



Cosa molto particolare è che il valore della conformance ritornato da Celonis è maggiore rispetto all'utente di prima nonostante quest'ultimo avesse una fitness maggiore. Questo può essere dovuto al fatto che le uniche problematiche che si hanno per questa risorsa sono legate ai cicli e non ad anomalie particolari. Ciò può essere confermato tramite la stessa view di Celonis:

16% of cases	<i>android.net.wifi.p2p.CONNECTION_STATE_CHANGE</i> is followed by <i>android.net.wifi.p2p.CONNECTION_STATE_CHANGE</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days shorter + 4.4 Steps per case
14% of cases	<i>android.net.wifi.p2p.STATE_CHANGED</i> is followed by <i>android.net.wifi.p2p.STATE_CHANGED</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days shorter + 4.6 Steps per case
12% of cases	<i>android.net.wifi.suplicant.CONNECTION_CHANGE</i> is followed by <i>android.net.wifi.suplicant.CONNECTION_CHANGE</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days longer + 0.7 Steps per case
8% of cases	Incomplete case Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days longer + 2.4 Steps per case

4% of cases	<i>android.bluetooth.adapter.action.DISCOVERY_FINISHED</i> is followed by <i>android.bluetooth.adapter.action.DISCOVERY_FINISHED</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days longer + 36.5 Steps per case
3% of cases	<i>android.bluetooth.adapter.action.SCAN_MODE_CHANGED</i> is followed by <i>android.bluetooth.adapter.action.SCAN_MODE_CHANGED</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days longer + 49.0 Steps per case
3% of cases	<i>android.bluetooth.adapter.action.STATE_CHANGED</i> is followed by <i>android.bluetooth.adapter.action.STATE_CHANGED</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days longer + 56.1 Steps per case
2% of cases	<i>android.bluetooth.device.action.FOUND</i> is followed by <i>android.bluetooth.device.action.FOUND</i> Add to allowlist View cases in... Effect on throughput time Effect on steps per case 0 Days longer + 15.5 Steps per case

Si passa ora all'analisi delle frequenze degli eventi. Il primo passo consiste nell'andare a verificare le frequenze delle attività del mese in cui non erano presenti degli attacchi:



successivamente si andranno a considerare le frequenze delle attività presenti nel mese di aprile:

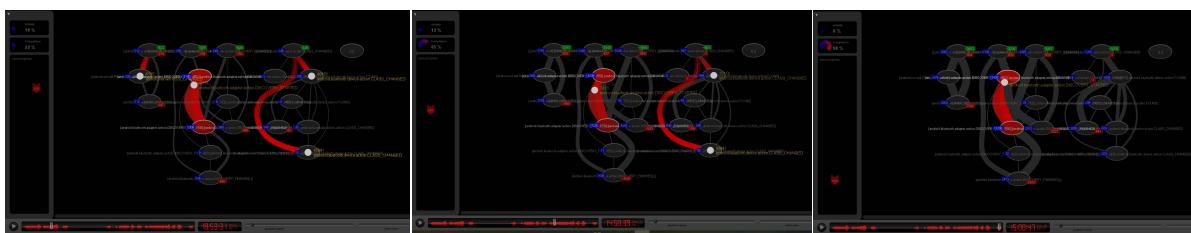


Si nota che a differenza dell'utente precedente qui si ha un aumento delle scansioni Bluetooth. Ovviamente questo aumento non è relativo al numero di casi in generale perché in realtà si passa da 29 mila a 22 mila (quindi una riduzione), ma viene preso in considerazione in base al numero di attività totali nella finestra temporale di appartenenza.

All'interno delle informazioni extra del dataset legate alle scansioni non si trova nulla di anomalo:

	Activity	Resource	Date	Time	Extra
1	android.bluetooth.adapter.action.DISCOVERY_FINISHED	0a50e09262	02.04.2017	08:44:06	
2	android.bluetooth.adapter.action.DISCOVERY_STARTED	0a50e09262	02.04.2017	08:44:53	

Come fatto anche nella risorsa precedente viene effettuata un'ulteriore analisi conclusiva per cercare conferme di quanto precedentemente detto su disco:



Come si può vedere dal risultato, il modello che si ottiene è estremamente più semplice in quanto vi è un numero minore di attività. Per quanto riguarda invece le frequenze, si può notare come ci sia un utilizzo preponderante delle componenti del Bluetooth che risulta essere abbastanza continuo in tutto l'arco temporale analizzato. L'analisi ha quindi confermato quanto detto in precedenza. Successivamente verranno riportate le conclusioni dell'intera analisi effettuata.

CONCLUSIONI

Attraverso le analisi effettuate si è giunti alla conclusione che per quanto riguarda il primo utente le anomalie possono essere identificate nel maggiore uso delle componenti del wifi mentre per quanto riguarda il secondo utente l'anomalia viene rappresentata in un maggiore uso delle componenti del Bluetooth. Si andranno ora a confrontare le analisi effettuate con la descrizione del dataset **moriarty** in modo tale da visionare la tipologia di attacchi effettuata da quest'ultimo e gli utenti che sono stati attaccati.

Aprendo il dataset di **moriarty** si vedono gli utenti coinvolti negli attacchi:

	A
1	1a1a12314b
2	5b76bedcac
3	55153967c4
4	5c1f751a99
5	68a5ffab20
6	72edf5f08f
7	860550d355
8	8ef449026a
9	b63c849327
10	e0bf6beeb3
11	e22b9f3772
12	ec3b3592f1

Si nota che tra di essi è presente solo l'utente **ec3b3592f1** tra gli utenti analizzati quindi si può dire già da ora di aver ottenuto un **falso positivo** che corrisponde all'eccessivo uso del Bluetooth del secondo utente.

Dopo aver analizzato le risorse è possibile vedere la tipologia di attacco effettuato. Esso si basa sulla continua scansione delle reti wifi con il dispositivo attaccato.

In this version Moriarty we present a web media player (of YouTube). The app has two types of sessions, benign and malicious.

1. Benign – normal web media player.
2. Malicious – Every constant period of time, the malicious service check's for Wifi connection and when it finds one, scans the network.

Malicious

- MalService – a service that waits for a wifi connection and search for all the hosts at the same network. Once it finishes, It tries to connect to each one as HTTP server and as FTP server and scans the files and folders inside.

Le azioni svolte da **moriarty** sono:

Action	Details	action type	behavior
View Changed	Entered Video Grid View	Network Scanner	benign
	Entered YouTube Player View	Network Scanner	benign
	Entered Preferences View	Network Scanner	benign
	Entered Search Video View	Network Scanner	benign
Scanning Network	Scan Started	Network Scanner	malicious
	Found ## hosts, ## FTP servers with ## files and folders And ## HTTP servers with ## files and folders	Network Scanner	malicious
	Scan Interrupted	Network Scanner	malicious
	Deleting temporary files	Network Scanner	malicious
Session Type Changed	Session is now benign	Network Scanner	benign
	Session is now malicious	Network Scanner	malicious
App Mode Change	App Entered onCreate()	Network Scanner	benign
	App Entered onPause()	Network Scanner	benign
	App Entered onDestroy()	Network Scanner	benign
	App Entered onResume()	Network Scanner	benign

Questo implica che le scansioni eccessive effettuate dal primo utente erano causate da un malware presente all'interno del dispositivo, quindi l'analisi effettuata ha rilevato un'anomalia concreta.