ANALYSIS AND SIMULATION OF A PROCESS

Process mining course – University of Padua

Pietro Volpato - 2079419

Q1 - Description of the event log and process model

1.1 – Log exploratory analysis

I started by importing the ***pm4py*** library and importing the log with the ***pm4py.xes_importer_apply()*** function. For some visuals and data analysis the *data frame* data-structure will be used, achieved by converting the log into a data frame with the ***pm4py.convert_to_dataframe(log)*** function.

My analysis started from the initial activities of our log, displayed using ***pm4py.get_start_activities(log)*** From the stakeholder 's model description we can understand that a trace can start only with *Autonomously triage* or *ambulance triage,* but our log shows a different behavior:

```
start_act = pm4py.get_start_activities(log)
print(start_act)
```

```
{'AUTONOMOUSLY TRIAGE': 8079, 'AMBULANCE TRIAGE': 3378, 'VISIT': 3, 'EMERGENCY SERVICE': 8}
```

Out of 11468 traces 11 have a starting activity different from one of the 2 triages: 3 starts with visit and 8 with emergency service. Given their low impact on the whole log I decided to filter them out using , `pm4py.filter_start_activities(log, ['AUTONOMOUSLY TRIAGE', 'AMBULANCE TRIAGE'])` removing them from this analysis and creating a filtered log with 11457 traces.

If we analyze the final activities with the ***pm4py.get_end_activities(filtered_log)*** on the filtered log, we can see that 1447 have and ending activity different from the supposed one: *Discharge*. In this case the sample is statistically significant, so we cannot filter out all the traces.

```
end_act = pm4py.get_end_activities(filtered_log)
end_act
```

```
{'DISCHARGE': 10010,  'X-RAY REPORT': 244, 'EMERGENCY SERVICE': 86,
 'CONSULTATION': 427, 'VISIT': 11,         'ANGIO REPORT': 8,
 'LABORATORY': 617,   'TC REPORT': 49,     'ECHO': 5}
```

Moving on with our analysis we can see that the blood sampling, labeled as optional from the stakeholder, is executed at most once per patient (so no need for a loop) and with a percentage of 11.69% (1339 out of 11457 cases).

The *'Visit*' activity is correctly executed at most once per patient and it is present in all cases, compliant with its mandatoriness. All the traces ending in this way still contain discharge, so we can assume that the order of how activities are recorded is wrong.

Regarding other activities, below is displayed the number of occurrences and their frequency at a trace level. We can see that visit, discharge and the triages are present in all traces while for other activities we can see how consultation, emergency service and laboratory are the most frequent activities. The least frequent ones are some of the radiological examinations and their respective report: TC, MRI and Angio.

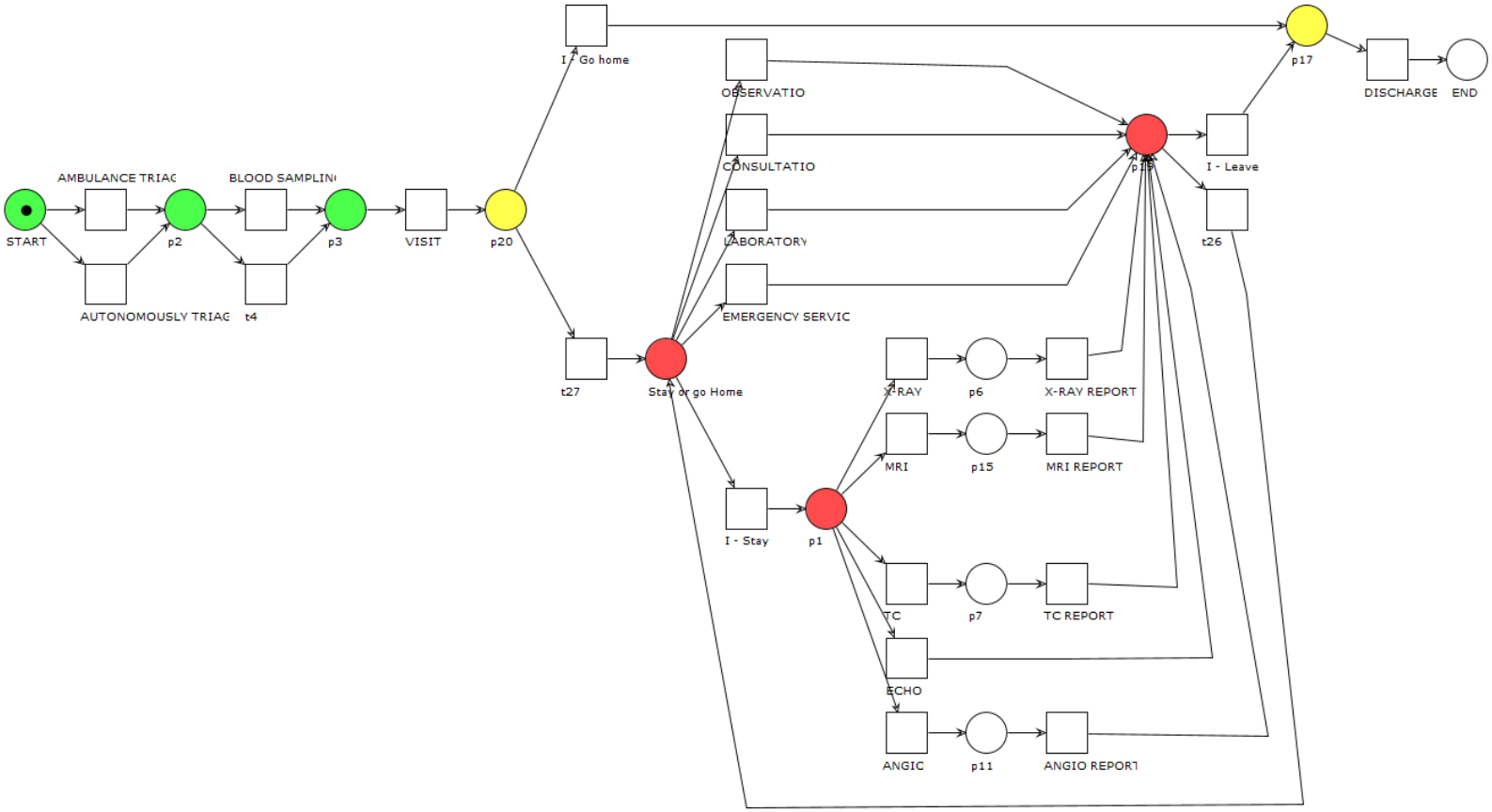
ACTIVITY	FREQUENCY	PERCENTAGE
VISIT	11457	100.0%
DISCHARGE	11457	100.0%
AUTONOMOUSLY TRIAGE	8079	70.52%
CONSULTATION	7222	54.27%
EMERGENCY SERVICE	7196	36.01%
LABORATORY	6499	41.6%

ACTIVITY	FREQUENCY	PERCENTAGE
X-RAY REPORT	4843	40.76%
X-RAY	4818	40.76%
AMBULANCE TRIAGE	3378	29.48%
BLOOD SAMPLING	1339	11.69%
ECHO	1281	11.09%
TC REPORT	1237	10.51%

ACTIVITY	FREQUENCY	PERCENTAGE
TC	1236	10.51%
OBSERVATION	390	3.4%
ANGIO	129	1.13%
ANGIO REPORT	129	1.13%
MRI	5	0.04%
MRI REPORT	5	0.04%

By looking at percentages, we observe a behavior similar to the frequency one. We can, however, see some differences: *Emergency service* has a very high frequency but a low trace percentage, suggesting the repetition of the same activity multiple times in the same trace. The same behavior can be observed also in *'Consultation'* and *'Laboratory'* (with an order of magnitude smaller than the emergency one) again suggesting for loops.

1.2 – Petri net



Performance metrics:

- *Precision*: 71,836 %
- *Generalization*: 99,998 %
- *Accuracy*: 96%

1.3 Process model analysis

The net starts with a place having a XOR split, allowing the 1st activity to be only one of the triages analyzed before.

We then have a XOR split with '*BLOOD SAMPLING'* and an invisible transition, since this activity in considered optional. We then arrive to a mandatory activity, '*VISIT'*, which each patient must undergo.

After the visit we arrive again at a XOR split, with 2 possible outcomes:

- 'DISCHARGE': the doctor thinks that the patient does not need any further examination and can be sent home right away.
- STAY: If the patient has to undergo some exams and cannot leave the hospital right away, an invisible transition allows for this behavior.

If the patient has to stay in the hospital, a token is produced in the *p1* yellow place, where we encounter a XOR split giving us 2 possible paths:

- 'ENTER RADIOLOGY': An invisible transition generates a place that allows to perform the radiological exams one at a time.
- 'OTHER ACTIVITIES': An invisible transition generates a place that allows to perform one of the 4 remaining activities: '*CONSULTATION*', '*OBSERVATION*', '*LABORATORY*' and '*EMERGENCY SERVICE'*.

At the end of each activity a token is produced in the p4 yellow place. Here a XOR split lets the patient perform other activities, thanks to an invisible transition producing another token in the *p1* place (the one having the split between *enter radiology* and *other activities*). If doctors consider the patient ready to leave, from this place an invisible transition can be fired, and the token generated allows the '*DISCHARGE*' activity to be fired, allowing the patient to leave the hospital.

The accuracy of the model is of 96%, with a 71,87% fitness and a 99,98% generalization, performance metrics we can consider happy with, given the other models considered.

1.4 other models

To give an understanding of why I chose this model, I want to briefly present other models considered.

1.4.1 Model 1 – Single XOR spilt

Instead of having a xor split between *enter radiology* and *other activities* I tried with a single place being a XOR split to the whole 9 possible activities, so without the "separation" between radiology and the rest of the hospital. In terms of performance metrics, I obtained the same accuracy and generalization, but a precision slightly lower (0,71756).

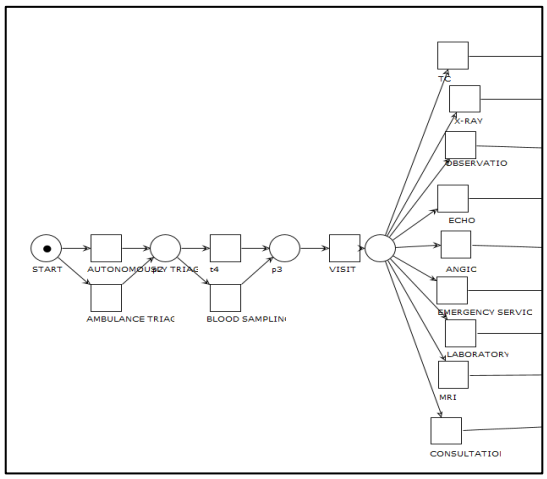
If we convert the 2 models in a BPMN format we can see that they are the same, with a xor - split allowing for the 9 possible activities one at a time. Considering performance metrics models can be considered the same too, with an almost meaningless difference in precision.

I decided to stick with this model because it is conceptually more correct for me to consider radiology separated from the other departments. However in terms of quantitative metrics we can assume the 2 models being equal.

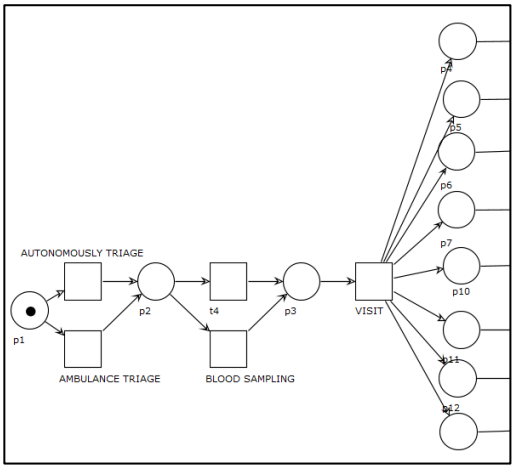
1.4.2 Model 2 – Single AND split

Instead of a xor split I tried with an and split, generating 9 tokens, one for each activity. The concept behind is to allow a patient to undergo other activities while waiting for a report, allowing much more freedom in the model. The accuracy of this model was higher (99%) but, given the increased complexity, the precision was way lower (0.5186). The idea came from some traces in the log, which had the report not directly following the respective activity. Given the increase in accuracy but the net drop in precision I decided to drop this model and consider traces showing this behavior as "wrong" (traces containing log moves).

Model 1



Model 2



Q2 Process discovery

In this part we will go through the analysis of the different processes for each triage color, to understand if there are any communalities or significative differences

2.1 Process models for different triage colors

This analysis of different triage colors will start with an overview of the activities common to all colors and then will focus on the most common activities for each color.
The technique used for filtering logs on triage colors is the ***Filter log on event attribute values*** ProM plug-in, while for discovering process models for different colors I used the ProM plugin ***interactive Data-aware heuristic mine (iDHM),*** this time applied on the log filtered on activities starting with a triage and ending **ONLY** with discharge. After running the plug-in with both the usual log (filtered on starting activities) and this log I noticed that the activities displayed are the same, but the log filtered also on ending activities returns a 'clearer' view of the process model since it avoids the noise of traces where the order of activities is untidy. The log filtered only on starting activities will be used to calculate interarrival times of patients with a given triage color and compute statistics, since even if the order is untidy those activities have been performed.

The parameters for the iDHM plug-in are the default ones, with activities under a 10% frequency at trace level not being displayed.

2.1.1 Common activities

The visit activity is not displayed since it has an execution rate of 100%

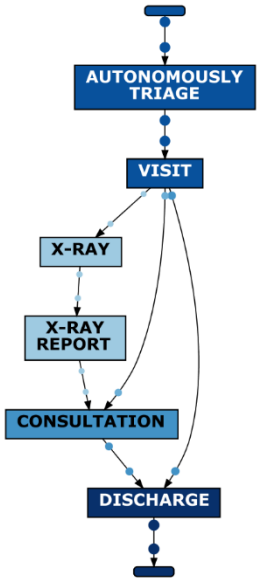
Color	Autonomous triage	Ambulance triage	Blood sampling
RED	[30 - 13.1%]	[199 - 86.9%]	[14 - 0.88%]
YELLOW	[1707 - 51.11%]	[1633 - 48.89%]	[779 - 43.28%]
GREEN	[5030 - 78.0%]	[1419 - 22.0%]	[314 - 17.44%]
BLUE	[1173 - 91.0%]	[116 - 9.0%]	[6 - 0.38%]
WHITE	[139 - 92.67%]	[11 - 7.33%]	[0 - 0.0%]

First of all, it's interesting to notice the way in which patients arrive to the hospital, namely the different Triages. Coherent with expectations white and blue patients arrive mostly in an autonomous way, given the lighter nature of their harm. Green still shows dominance of the autonomous triage while Red patients demonstrate the opposite: predominance of 'Ambulance' as arrival method.

The only even situation among all is the yellow one, where patients are almost equally split between the 2 arrival methods. Unexpectedly also some patients with blue and white triage color arrive by ambulance: this can be due to some minor accidents or people with no possibility to reach the hospital autonomously.

Moving on to blood sampling we discover that the activity is repeated at most once per patient, and spread like this: again yellow patients are the most "balanced" ones, with near-to-50% of them undertaking the blood sampling. Green are the only other triage color where this activity is relevant, with 17.4% of patients undertaking it. For all the other colors blood sampling is almost irrelevant, with less than 1% of patients belonging to each category undergoing it.

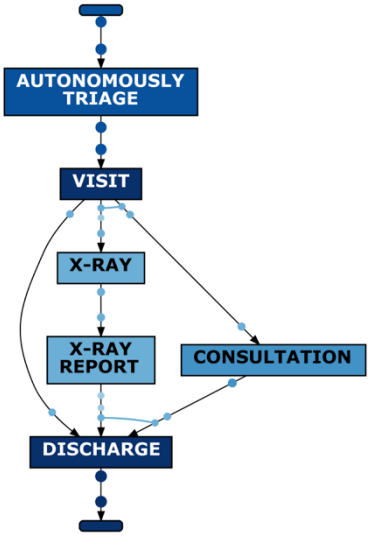
2.1.2 White patients



In our filtered log we have observed 150 white patients, with an average arrival rate of one every 15 hours and 36 minutes. In the 'Radiology' branch we can see that the most common activity is X-RAY and its report, with 14% of patients (21) needing it. From our log we have evidence of another patient asking for other radiological exams (ECHO, TC and ANGIO, with their respective reports) but given the rarity of this event we can consider him some sort of outlier (maybe wrong triage color assigned). Among other activities consultation is definitely the most important, with 45.3% of patients (68) undergoing it. Some emergency service and some laboratory activities have been performed on white patients but given their scarceness they are not displayed in the C-net.

The average number of activities per trace is 3.96, with an average time in the hospital (difference between the time of the triage and the discharge, computed on the log filtered on starting and ending activities) is of 2 hours and 10 minutes.

2.1.3 Blue patients

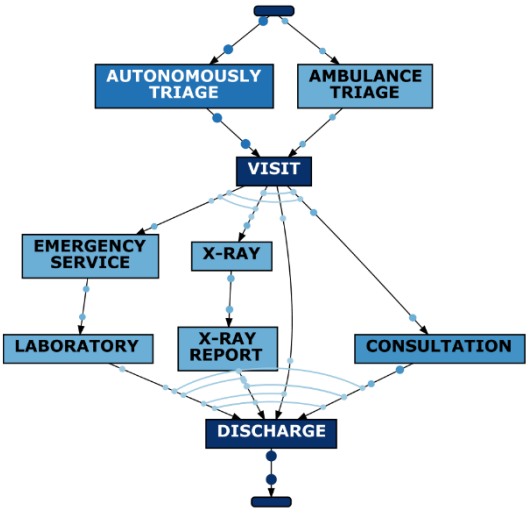


The activities displayed in the C-Net are the same displayed for white patients, but if we consider frequencies we can spot huge differences between the 2 triage colors: In our filtered log we have 1289 blue patients, with an average arrival rate of one patient ever 1 hour and 50 minutes. Again, the most frequent activity is consultation, with 498 patients (38.6%) needing. The frequency of this activity in the blue log is of 830 and it is performed 1.6 times per patient (on average). The total number of different activities we can find in the blue log is greater than the one in the white (16 versus 12), indicating that some activities (*Blood sampling, Observation, Angio and Agio report*) can be considered a discriminator for triage colors: patients needing Observation and Angio will be given blue triage.

Blue patients have an average number of activities per trace of 4.72 and an average permanence time in the hospital of 2 hours and 14 minutes.

Even if the activities are more, if we consider activities happening for more than 10% of patients, we see that the 2 colors are very similar, with some differences in complexity for the blue. The average number of activities is higher even if the permanence time is similar, since blue patients have a lower response time with respect to whites.

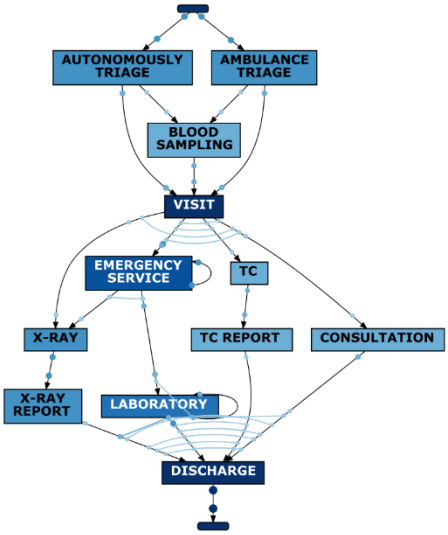
2.1.4 Green patients



From now on we will start to see remarkable differences in the C-nets: here for the 1st time the *Ambulance traige* is visible, since at least 10% of patients arrrive by ambulance. With green patients we start to see also emergency service and laboratory in the net: activities that were performed also before, but that now are fequent enough to be displayed. Green are also the most frequent patients in the hospital, with a total of 6449 (more than 50% of the total) and an arrival average of (approximately) one patient every 22 minutes. Green is also the first triage color where all of the 18 activities are performed, with 3 patients needing MRI and its respective report. Again the most frequent activities are Consultation, X-ray and emergency service, so the ranking (in terms of percentage of patients undergoing it) is the same as other colors. What changes is the length of loops: emergency serivce is performed on average 1.5 times per patient needing it. Loops are present also for other activities but the average of execution per patient does not cross the 1.15 treshold.

Green patients have an average of 5.57 activities per trace, with an average permanence time of 2 hours and 40 minutes. The response time here is even lower than blue but the permanence time gets way higher, due to the needed complexity of analysis performed on those patients.

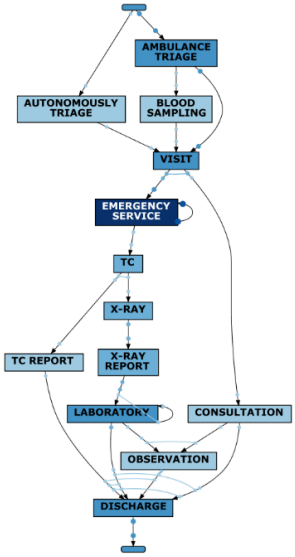
2.1.5 Yellow patients



Yellow patients are the second most frequent patients in this hospital, being 3340 in the time of our analysis, with an average arrival rate of one every 42 minutes.
The most frequent activity is *Laboratory*, with 77% of patients (2581) doing but with an average of 1.46 executions per patient. The second most frequent activity here is *Emergency service*, with 60% of patients (1990) underoing it and with an average of 1.93 executions per patient. Those data are confirmed by the loops present in the C-net, suggesting the high number of patients repeating those activities. For the 1st time we see a graph with the TC and TC report activities, given their high frequency for yellow patients (20% - 666 patients). Also for yellow patients all the 18 activities are performed, with an incraese in frequency for *Observation* and *Angio* (and its report) with respect to blue patients.

Yellow patients on average undergo 7.73 activities, with an average permanence time in the hospital of 4 hours. Yellow is the first traige color whose patients have an average permanence time higher than the hospital's mean of 3 hours and 8 minutes

2.1.6 Red patients



Red patients are the second-least patients in the hospital, being 229.
Here the most frequent activity is *Laboratory*, with 90% of patients (207) needing it but with a low number of executions per patient (1.3). The second most frequent activity is *Emergency service*, with 86% of patients undergoing it but with a very high number of executions per patient: 2,91. For red (and yellow) triage, *Laboratory* is the activity executed on the most number of patients (percentage) but *Emergency service* is the activity executed the most number of times (frequency).
Red patients do not undergo MRI and MRI report, so their total number of activities is 16. It is interesting to notice that they are the only triage color that have Observation in the C-net, meaning that more than 10% of patients need it.
Red patients are subjected to 9.79 activities on average, with a permanence time in the hospital of 3 hours and 42 minutes.

2.1.7 Overview

	RED		YELLOW		GREEN		BLUE		WHITE		TOTAL	
ACTIVITY	PERCENTAGE	FREQUENCY	PERCENTAGE	FREQUENCY	PERCENTAGE	FREQUENCY	PERCENTAGE	FREQUENCY	PERCENTAGE	FREQUENCY	PERCENTAGE	FREQUENCY
VISIT	100.0%	229	100.0%	3340	100.0%	6449	100.0%	1289	100.0%	150	100.0%	11457
DISCHARGE	100.0%	229	100.0%	3340	100.0%	6449	100.0%	1289	100.0%	150	100.0%	11457
AUTONOMOUSLY TRIAGE	13.1%	30	51.11%	1707	78.0%	5030	91.0%	1173	92.67%	139	70,52%	8079
CONSULTATION	52.84%	173	43.32%	1689	59.58%	4452	57.41%	830	45.33%	78	54.27%	7222
EMERGENCY SERVICE	85.59%	571	59.58%	3858	27.86%	2583	10.01%	168	9.33%	16	36.01%	7196
LABORATORY	90.39%	277	77.28%	3758	29.26%	2358	6.83%	103	2.0%	3	41.6%	6499
X-RAY REPORT	47.6%	111	47.57%	1672	38.04%	2529	38.63%	509	14.0%	22	40.76%	4843
X-RAY	47.6%	111	47.57%	1667	38.04%	2514	38.63%	504	14.0%	22	40.76%	4818
AMBULANCE TRIAGE	86.9%	199	48.89%	1633	22.0%	1419	9.0%	116	7.33%	11	29,48%	3378
BLOOD SAMPLING	13.1%	30	27.84%	930	5.78%	373	0.47%	6	0.0%	0	11.69%	1339
ECHO	2.18%	5	13.95%	471	11.63%	755	3.8%	49	0.67%	1	11.09%	1281
TC REPORT	40.17%	98	19.94%	684	6.56%	432	1.71%	22	0.67%	1	10.51%	1237
TC	40.17%	97	19.94%	685	6.56%	431	1.71%	22	0.67%	1	10.51%	1236
OBSERVATION	7.86%	18	7.63%	255	1.72%	111	0.47%	6	0.0%	0	3.4%	390
ANGIO	14.41%	33	2.22%	74	0.33%	21	0.08%	1	0.0%	0	1.13%	129
ANGIO REPORT	14.41%	33	2.22%	74	0.33%	21	0.08%	1	0.0%	0	1.13%	129
MRI	0.0%	0	0.06%	2	0.05%	3	0.0%	0	0.0%	0	0.04%	5
MRI REPORT	0.0%	0	0.06%	2	0.05%	3	0.0%	0	0.0%	0	0.04%	5

Q 2.2 Performance indicator

This part aims at understanding if patients are treated accordingly with the protocol indicators.

```
def response_time(color, log, parameter):
    long_responses = {} #Dictionary containing as keys misstreated cases, and as values time passed for treatment
    response_times = [] #List containing the response time for each patient
    filtered_log = pm4py.filter_event_attribute_values(log, "triage color", [color], level="case", retain=True)
    for trace in filtered_log:
        start = trace[0]['time:timestamp']
        end = trace[1]['start:timestamp']
        delta = (end - start).total_seconds()
        response_times.append(delta)
        if delta > parameter:
            long_responses[trace[0]['case:concept:name']]=delta
        mean = np.round(np.mean(response_times), decimals = 0)
        percentage = np.round(len(long_responses)*100/len(filtered_log), decimals = 2)
    return percentage, mean, response_times
```

To do so built a function, called ***response_time,*** that takes as input the triage color to analyze, the log and a parameter indicating the maximum number of seconds to respond to a patient. This function returns the mean response time for the color, the percentage of mistreated patients and a list containing all the response times.

I used the ***pm4py.filter_event_attribute_values(log)*** to filter the log given as input.
The attribute to filter on is the triage color and the parameter for filtering is the color given as input. The function will return a filtered log, containing only patients who were assigned the triage color given as input.

I iterate over all the traces in the filtered log and assign to the ***start*** variable the ending time of the first activity, while I assign to the ***end*** variable the starting time of the second activity.

The response time is calculated as the difference between the ending time of the 1st activity and the starting time of the 2nd activity, in the function as ***end – start***, and it is saved in the ***delta*** variable. The ***delta*** time is then converted in seconds. We then add to the list ***response_times*** the ***delta*** response time. This variable will be used later for calculating the mean response time and for plotting the waiting times distribution.

I then check if the ***delta*** time exceeds the parameter given as a KPI. If delta is lower I move on to the next trace in the log, while if response time is greater I add to the ***long_responses*** dictionary a record, with as key the *caseID* of the trace, and as value the ***delta*** time, exceeding the limit.

The data structure used to keep record of long responses is a dictionary that, if needed, can be added to the **return** variables to analyze which cases are mistreated and their relative time.

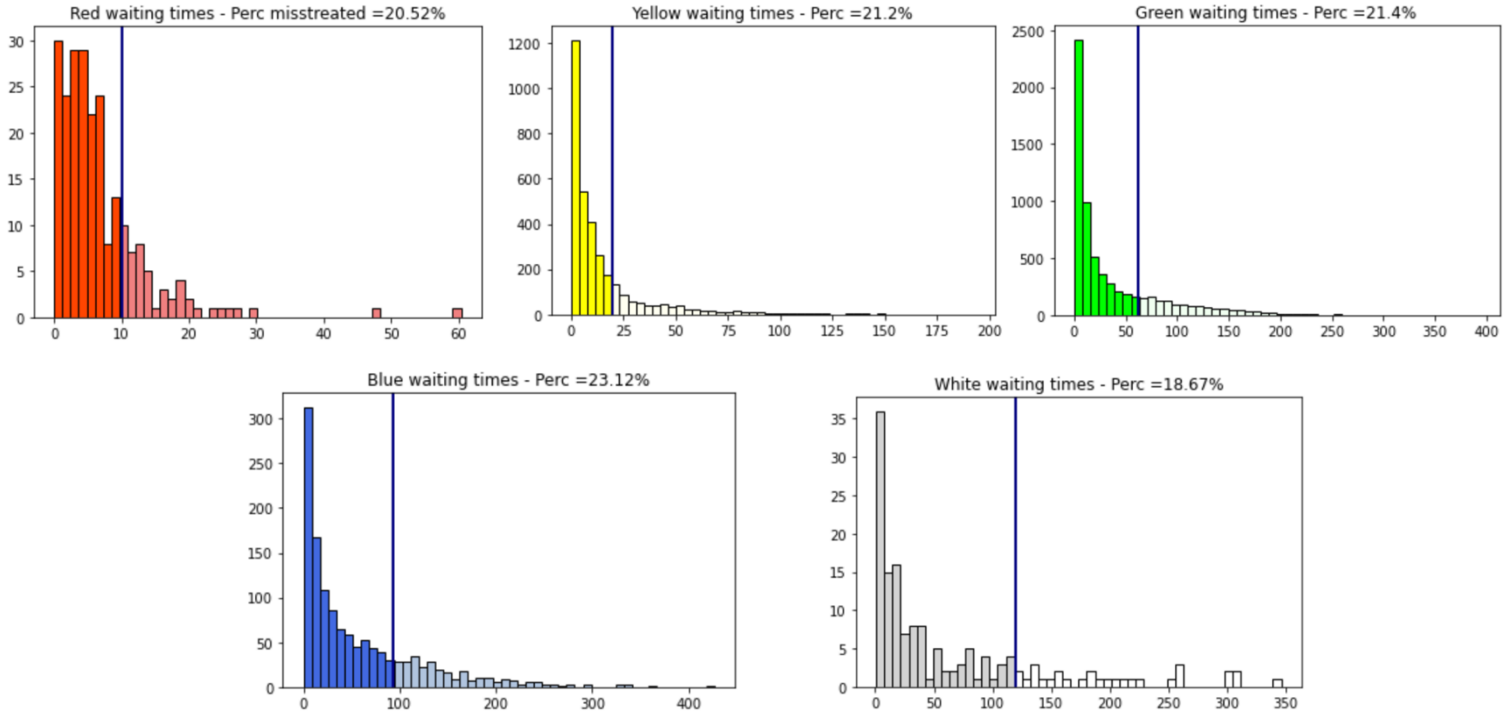
After iterating among all the traces in the log, the ***mean*** is computed as the mean of the ***response_times*** list, while the percentage of exceeding cases is calculated as the length of the ***long_responses*** dictionary over the total length of the log. This means the number of people mistreated over all the people with that given triage color assigned.

The function returns 3 variables: the ***percentage*** of mistreated people***, the mean*** of response times and the ***long_responses*** list**.**

The function is called on the log filtered on starting activities.

Q 2.2.1 Results

To print the results, I called the function on each of the 5 colors, giving as parameter the maximum number of minutes in the KPI table provided in the project introduction. Displayed below the results.



Each graph represents the distribution of the waiting times, with a vertical line having an ordinate equal to the maximum time patients with that color should wait. In a brighter color we have the patients treated in a time below the limit one, while in a paler color the portion of patients who waited longer than supposed.

AVERAGE WAITING TIMES - In minutes				
Red : 6.72	Yellow: 14.53	Green: 35.51	Blue: 55.86	White: 65.11

When looking at the means in the table below we can see that, on average, patients are treated according to the protocol times, but we have a portion of them (roughly 20%), who is “left on hold” for too much time.

The severity of these patients abandonment gets worse in a directly followed relationship with the triage color: waiting too long for a treatment will not cause much harm to a patient with a white or blue code, but could mean a matter of life or death for a patient with a yellow (or even worse) red triage color. The short response times these patients have are due to the severity of their harm: leaving a yellow patient on hold for more than 2 hours or a red patient for more than 30 minutes is something the hospital should definitely fix in the shortest time possible.

Obviously also other patients deserve to be treated in a reasonable amount of time, but the more ‘critic’ patients are the one that should get the most attention.

Q3 Process simulation

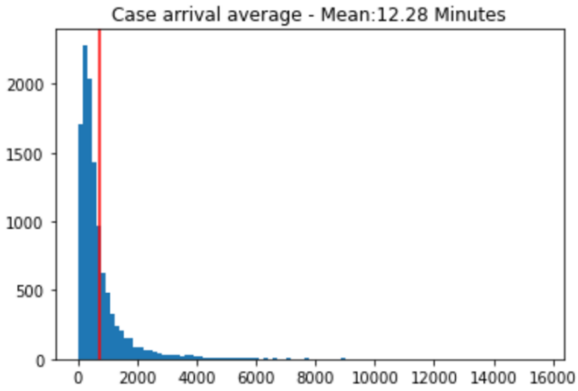
3.1 Process simulation parameters

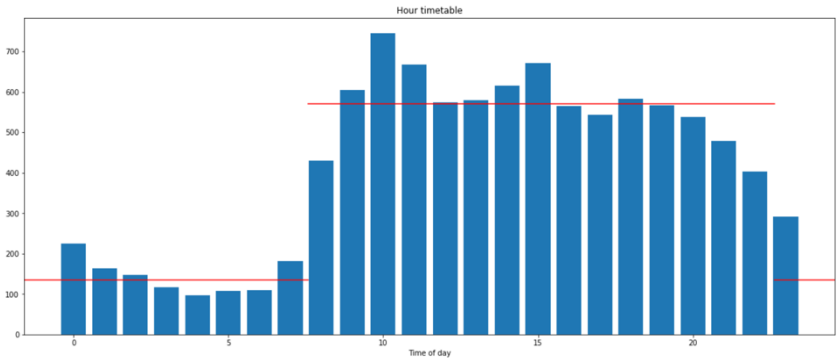
In this section of the report I will go through the process that led me to finding each parameter needed for the simulation, starting from

3.1.1 Case arrival rate

Since the hospital is opened 24/7, we can calculate the case arrival rate as the time difference between the starting activity of each trace in our log. This can be done with a for loop over the whole log or using the ***pm4py.get_case_arrival_average(filtered_log).*** The results with both methods are the same and show an exponential distribution with a mean of 12.28 minutes, meaning that (on average) a patient arrives at the hospital every 12.28 minutes.

Even if the graph resembles a log-normal distribution, I used an exponential distribution since it describers better arrival times and the data's variance is too high with respect to the mean, causing wrong results in the simulation.





From the graph on the right, we can see the hourly distribution of patients in the 3 months our log keeps record. We can clearly see a net distinction between hours from 7 to 23 (with a case arrival rate of one patient every 34 minutes) and hours from 8 to 22 (with a case arrival rate of one patient every 9 minutes). For the purpose of this project, we will consider the case arrival rate as uniform during the day, with the distribution displayed on the left and a mean of one patient every 12.28 minutes.

3.1.2 Resource roles

The resource roles can be found with ***pm4py.discover_organizational_roles(filtered_log)***

```
roles = pm4py.discover_organizational_roles(filtered_log)
for i in range (len(roles)):
    print(roles[i].activities, ':',len(roles[i].originator_importance.values()))
```

In this print the original output has been "adapted" for representational purposes, so instead of returning a dictionary with all the resources for that role here we return the role and the **number** of resources in that specific role.

ROLE	RESOURCES
['AMBULANCE TRIAGE']	1
['ANGIO REPORT']	36
['ANGIO', 'TC', 'X-RAY']	27
['AUTONOMOUSLY TRIAGE']	1
['BLOOD SAMPLING', 'EMERGENCY SERVICE']	30
['CONSULTATION']	34
['DISCHARGE', 'VISIT']	15
['ECHO', 'TC REPORT', 'X-RAY REPORT']	39
['LABORATORY']	96
['MRI']	5
['MRI REPORT']	5
['OBSERVATION']	12

From this initial roles division something jumps to the eye: radiological activities and radiological reports are grouped together but *Angio report, MRI* and *MRI report* are left out. Since that all those activities belong to the same "department" (radiology), we might be able to group some roles together, by checking if people in the minority groups also perform other activities in the majority group. The hypothesis is to end up with 2 roles for the radiology department: a role with all activities belonging to radiological reports and one role with all the other activities belonging to radiology department.

To do so I wrote a script of code to check if the resources performing the activity ANGIO REPORT also perform other activities:

```
ANGIO_REPORT = list(roles[1].originator_importance.keys())
attività_ANGIO_REPORT = []
for trace in filtered_log:
    for event in trace:
        if event['org:resource'] in ANGIO_REPORT and event['concept:name'] not in attività_ANGIO_REPORT:
            attività_ANGIO_REPORT.append(event['concept:name'])
attività_ANGIO_REPORT.remove('ANGIO REPORT')
```

I created a list called **ANGIO_REPORT,** containing all the resources performing the *Angio Report* activity, and another list called **attività_ANGIO_REPORT** in which I will insert activities performed by those resources. After iterating over the whole log and checking the activities performed by each resource, those are the results (I applied the same methodology also for finding the other activities performed by resources in the role *MRI* and *MRI REPORT*)

```
People performing MRI, also perform X-RAY, ANGIO, TC
People performing MRI_REPORT, also perform X-RAY REPORT, TC REPORT, ANGIO REPORT, ECHO
People performing ANGIO-REPORT, also perform X-RAY REPORT, TC REPORT, ECHO, MRI REPORT
```

The results below show that people performing the MRI or ANGIO reporting activities are also involved in the other radiological reporting activities. The same holds for MRI, whose resources also perform all the other activities in the radiology department. This suggests that, for the radiology department, we can create 2 roles:

- Radiological operators: performing *MRI, ANGIO, TC, X-RAY*
- Radiological reporter: performing the reporting activities and *ECHO*

After renaming the roles as in the mapping below (*Table 1*) we obtain this segmentation (*Table* 2)

Table 1

ACTIVITIES	ROLE NAME
['AMBULANCE TRIAGE']	Ambulance
['AUTONOMOUSLY TRIAGE']	Auto
['OBSERVATION']	Bed
['DISCHARGE', 'VISIT']	Doctor
['BLOOD SAMPLING', 'EMERGENCY SERVICE']	Nurse
['CONSULTATION']	Operator 1
['LABORATORY']	Operator 2
['ANGIO', 'TC', 'X-RAY' , 'MRI']	Radio operator
['ECHO', 'TC REPORT', 'X-RAY REPORT', 'MRI REPORT', 'ANGIO REPORT']	Radio reporter

Table 2

ROLE	RESOURCES
Ambulance	1
Auto	1
Bed	12
Doctor	15
Nurse	30
Operator_1	34
Operator_2	96
Radio_operator	27
Radio_reporter	39

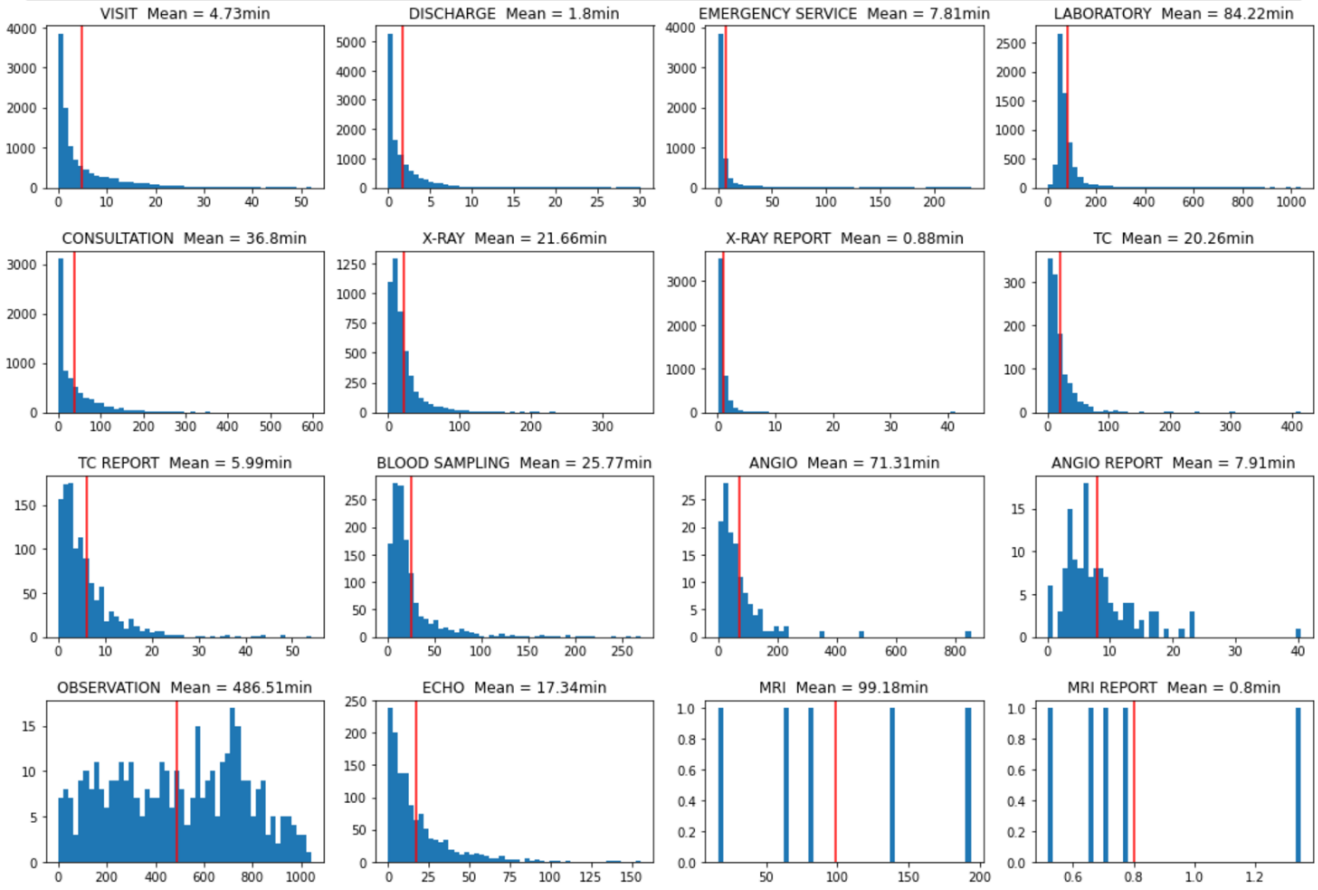
We can see that the total number of resources in the role Radio_operator and Radio_reporter are the same as in the previous roles [*'ANGIO', 'TC', 'X-RAY'*] and [*'ECHO', 'TC REPORT', 'X-RAY REPORT'*], even after merge. This confirms our initial hypothesis of 2 single roles rather than the 5 the algorithm found.

Instead of the previous 12 roles we now have 9 total roles, with 255 resources available (253 if we don't count the ambulance and the Auto, which is the autonomous arrival at the hospital of the patient).

3.1.3 Tasks duration

The most straightforward way to calculate tasks duration would be to use the ***pm4py.soj_time_get.apply(log)*** function, but when analyzing the log we can see that there are some activities recorded with execution time equal to 0 or even negative. The execution time equal to 0 is correct for the 11457 triages activities, but we also have other 2220 activities recorded as instantaneous which in reality are not. The assumption I will carry on during this part is that those activities actually happened but there was an error in the log when recording timestamps, so they will be excluded when calculating the tasks duration. The same reasoning holds for activities with negative execution time (14 in the whole filtered log).

```
times_no_zero = {x:[] for x in list(dataframe.Name.unique())}
for trace in log:
    for event in trace:
        start = event['start:timestamp']
        end = event['time:timestamp']
        delta = end - start
        delta = delta.total_seconds()
        if delta > 0:
            times_no_zero[event['concept:name']].append(delta/60)
```



To keep track of execution times I created a dictionary having as keys the activities and as values an empty list.

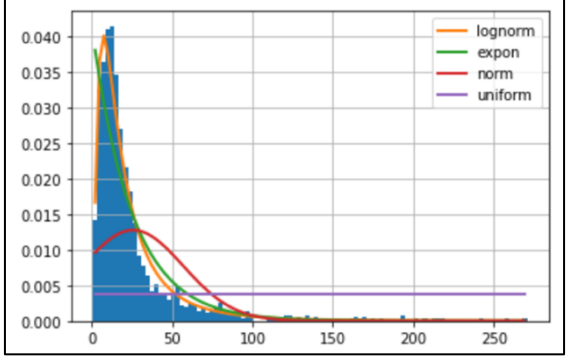
I then iterated through the whole log, and for each activity I calculated the execution time as the difference between the ending and starting timestamps. If this difference is positive, the execution time is appended to the list relative to that activity, otherwise the time is skipped. The results obtained show the distributions displayed in the previous page.

Some of them can be identified straightforwardly, while for some ambiguous one (The *BLOOD SAMPLING* one for example) I used an accessory tool to get a quantitative response: the ***fitter*** library in python.

Provided below the snipped of code used to check distributions

```
from fitter import Fitter, get_common_distributions
temp_var = 'BLOOD SAMPLING'
f = Fitter(times_no_zero[temp_var],
           distributions=['lognorm',
                          "expon",
                          "norm",
                          "uniform"
                         ])
f.fit()
f.summary()
print(list(f.get_best(method = 'sumsquare_error').keys()))
temp_2 = times_no_zero[temp_var]
print(f'Blood sampling mean:{np.mean(temp_2).round(2)}')
print(f'Variance:{np.var(temp_2).round(2)}')
```

```
['lognorm']
Blood sampling mean:25.77
Variance:973.76
```



The output consists in the graph on the right, where the fitter method finds the best distribution for each one of the 4 given as input, and then calculates the sum of squared errors for each distribution, returning the best one fitting our data (in this case *lognorm*). I then calculated the parameters for the lognormal distribution, which are *mean* and *variance*.

After running this code on dubious distributions, I obtained those results:

Activity	Distribution	Parameters	Activity	Distribution	Parameters
['VISIT',	Exponential	Mean = 4.73 min	'TC REPORT',	Exponential	Mean = 6
'DISCHARGE',	Exponential	Mean = 1.8 min	'BLOOD SAMPLING',	Log normal	Mean = 25.77 - Var = 973,76
'EMERGENCY SERVICE',	Exponential	Mean = 7,81 min	'ANGIO',	Log normal	Mean = 71,31 - Var = 9392
'LABORATORY',	Log normal	Mean = 84.22 - Var = 5919.47	'ANGIO REPORT',	Log normal	Mean = 7.91 - Var = 32,71
'CONSULTATION',	Exponential	Mean = 36.8	'OBSERVATION',	Uniform	Min = 0,18 - Max = 1041,68
'X-RAY',	Exponential	Mean = 21.66	'ECHO',	Exponential	Mean = 17,34
'X-RAY REPORT',	Log normal	Mean = 0.88 - Var = 4.2	'MRI',	Uniform	Min = 16,33 - Max = 193,52
'TC',	Exponential	Mean = 20,26	'MRI REPORT']	Uniform	Min = 0,52 - Max = 1,35

3.1.4 Branching probabilities

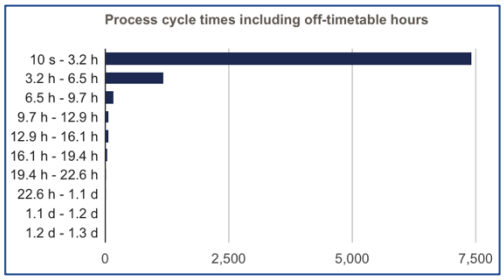
The branching probabilities can be found by running the ***multi-perspective miner*** plug-in in ProM. The log used is the one filtered on starting activities.

For space purposes I will not include in the report a picture with branching probabilities, but they can be found in the .bpmn file when imported in BIMP.

3.1.5 Simulation

After plugging the simulation parameters found above I obtained "coherent" results in BIMP. I decided to exclude 5% of patients from both the nose and the tail of our distribution, to consider cool-down and warm-up. By changing the arrival time calendar, I allowed patient arrival at any time of the day.

Given those parameters the simulation is run over 9000 total instances, with a total simulation time of approximately 3 months.



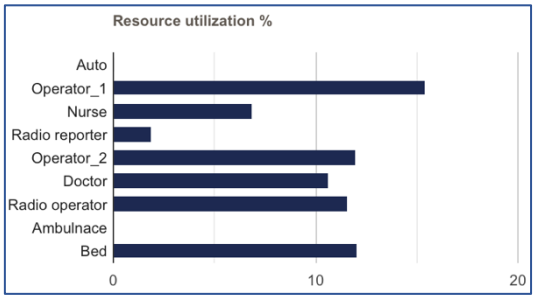
From this graph we can see that patients, on average, spend between 10 seconds and 3.2 hours in the hospital, with a process instance cycle average of 1.9 hours. This parameter is lower with respect to what seen in the log (188 minutes) given the fact that in my simulation the waiting times are equal to 0. The only positive value is in the '*Visit*' activity, where some patients wait for a maximum time of 2.5 minutes.

A cause for this behavior might be the high number of activities following an exponential distribution for their execution time. This kind of distribution tend to assign values very close (or even equal) to 0, which are difficult to see in a real-world situation: I expect a TC exam to last at least 20-30 minutes, and difficultly seconds. This hypothesis however is dismissed by our log, where observe TC activities being performed in the order of seconds.

If we also consider the whole process lifecycle, in reality it is difficult to see a patient being in the hospital for less than a couple of minutes, while in our simulation a considerable portion of them spend less than a minute in the hospital. By looking at the log we can see that the minimum time spent by a patient is 76 seconds (again, strange).

A solution would be to discuss with the shareholder, to understand if there are problems with the log (and so with the distributions found). Another solution might be to shift the exponential distributions to the right, 'penalizing' values that are too low and avoiding having traces (patients) staying in the hospital for a low amount in time.

The other activities' frequency and average duration are coherent with what observed in the log, so we can be satisfied with that.



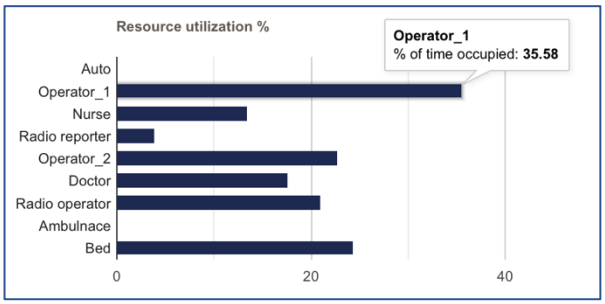
By looking at resource's utilization, we can see that they never overstep the 16% threshold. This shows a clear situation of resources underutilization: there too many people employed at the hospital. The number of resources given as parameter in the BIMP simulation is 1/3 of what seen in the log, given that people in real life perform 8-hours shifts, while in BIMP we consider them working 24/7.

We haven't considered costs in this simulation so this part will not be covered in the report.

3.2 Process improvement

By looking at the results obtained in our simulation, the first order of business is to re-size the number of resources: a resource utilization of 15% is way too low for an organization be efficient. Obviously, we cannot expect to reach 100% because we have to consider coffee and lunch breaks, extra-working activities, sick days, vacations and also with those parameters we are considering a fixed productivity rate of our employees. To reach satisfactory efficiency parameters we aim a resource utilization percentage between 35% and 45%.

To start downsizing I firstly tried with 50% of the initial resources and those are the results obtained:



The only acceptable result is the Operator_1 (performing '*Consultation*'), which crosses the 35% threshold. Waiting times are still equal to 0, which is not a bad thing but in terms of efficiency we can still do better. All the other resources are still underutilized, so there is still room for improvement by downsizing again.

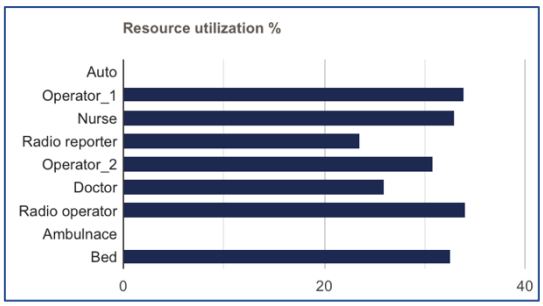
3.2.1 Final results

By implementing the downsizing displayed in *Table 3* we obtain the results in *Figure 2*.

Table 3

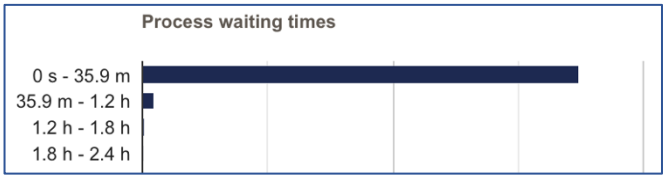
Activity	Inital resources	Current resources	% of inital resources
Bed	12	4	33%
Auto	1	1	100%
Operator_2	96	36	38%
Nurse	30	6	20%
Doctor	15	6	40%
Radio_reporter	39	3	8%
Radio_operator	27	9	33%
Ambulance	1	1	100%
Operator_1	34	15	44%

Figure 2



By reducing the number of employees by 68% we obtain reasonable resources utilization time, all around the 35% threshold.

By looking at the waiting times in the table below we can see that the majority of patients (around 97%) wait a maximum amount of time of 36 minutes, with no patient having to wait longer than 1.8 hours. The average permanence time in the hospital is now of 2 hours, similar to the results obtain with previous parameters.



We can be happy with those results, but we have to make some considerations.

3.2.2 Final considerations

We saw how by cutting 70% of the personnel in the hospital we still obtain results showing a very efficient utilization of resources. In real life this downsize is unfeasible from a practical and ethical point of view. From data on-hand we can clearly state that the hospital can cut some resources, but with an order of magnitude very different from our result.

We have to take into account that our waiting and process life-cycle times are slightly different from the one showed in the log, so in reality having this few resources will lead to huge lines, high waiting times and it might cost some patient's life.

When dealing with healthcare, having an 'inefficient' utilization of resources is acceptable, since we are talking about saving people's life. Having more resources than what theoretically needed might mean saving a people's life and this spans beyond the concept of saving money.

Having an overview on costs would be useful for making a more accurate decision on which resources to cut, allowing us to have a 360-degree overview at performances. With information on-had this is the optimal solution that balances cost reduction and efficiency.