PISA UNIVERSITY

PROCESS MINING AND INTELLIGENCE

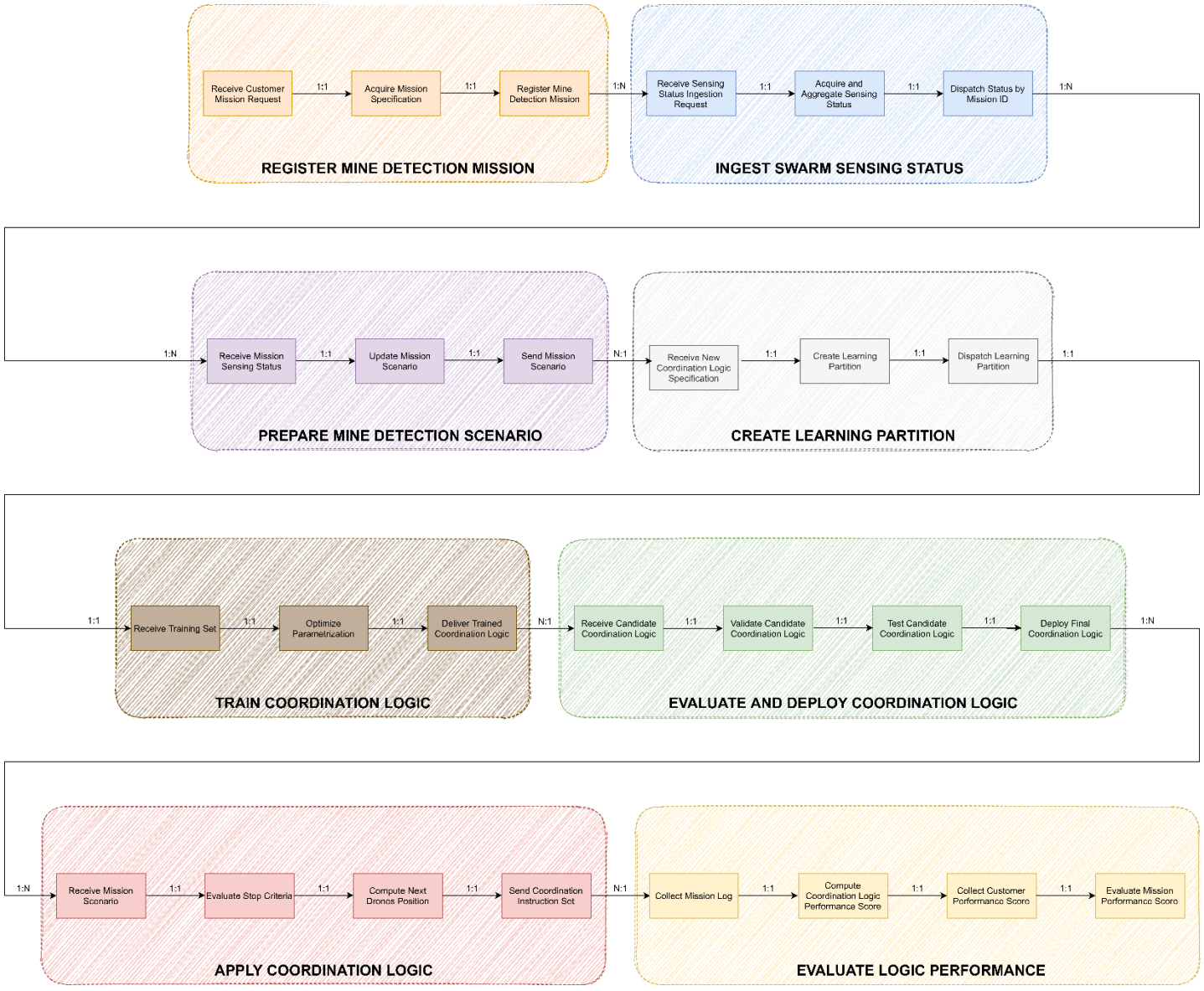
DRONES SWARMS PROJECT DOCUMENTATION

ACADEMIC YEAR 2020-2021

BALESTRI FEDERICO, PETROCCHI STEFANO, QUINTAVALLA MIRCO, SILVESTRI GIULIO

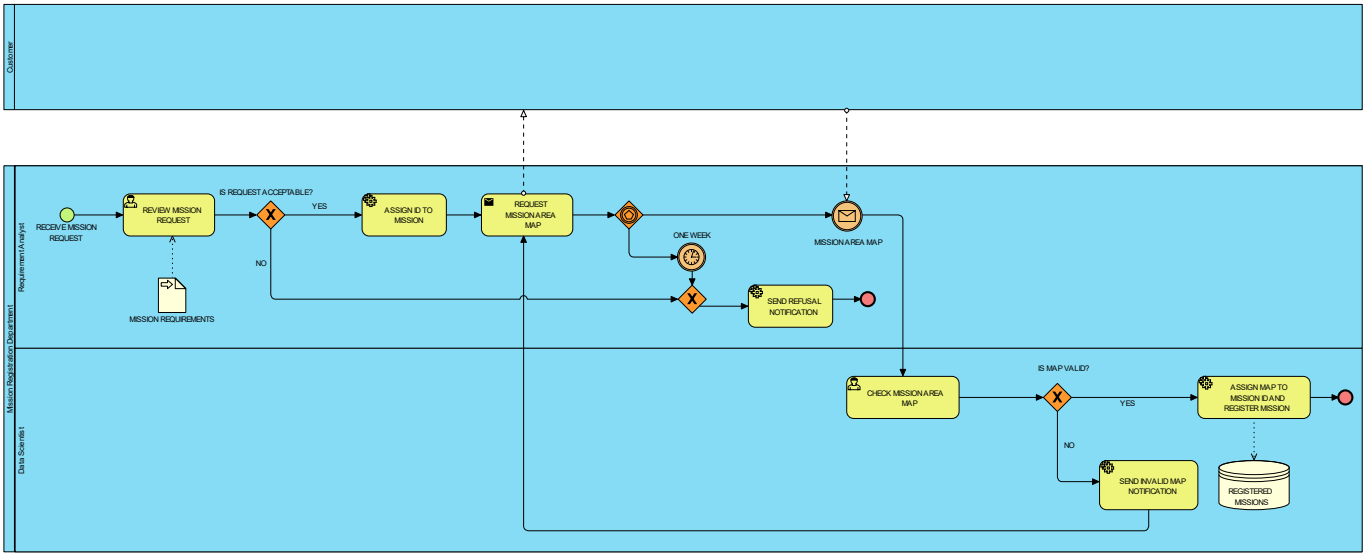
SUMMARY

PROCESS LANDSCAPE

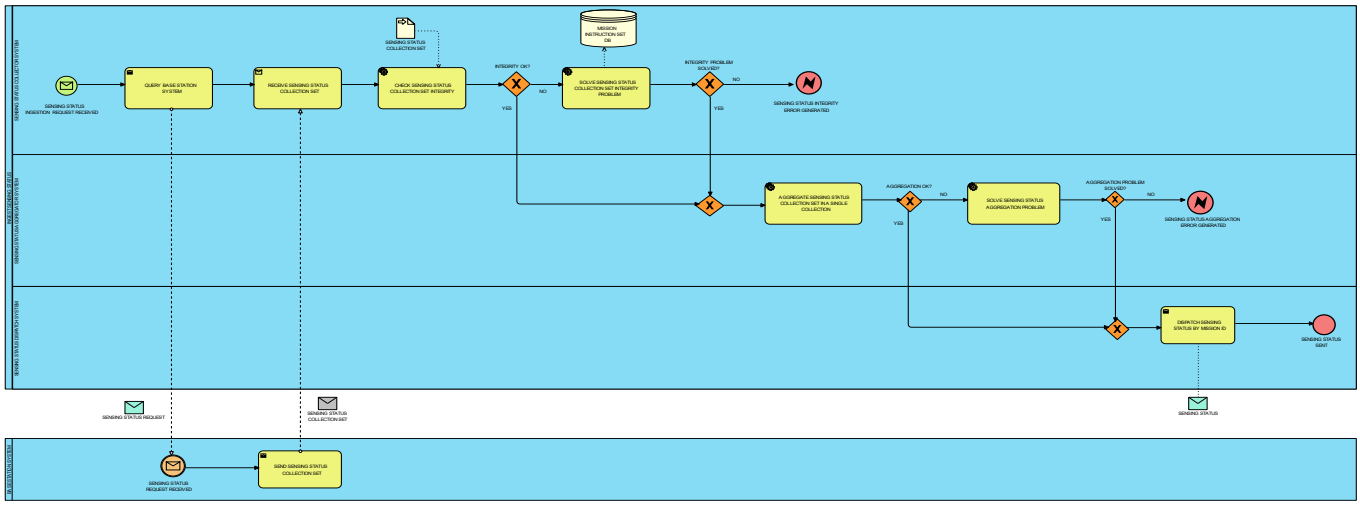


BPMN DIAGRAMS

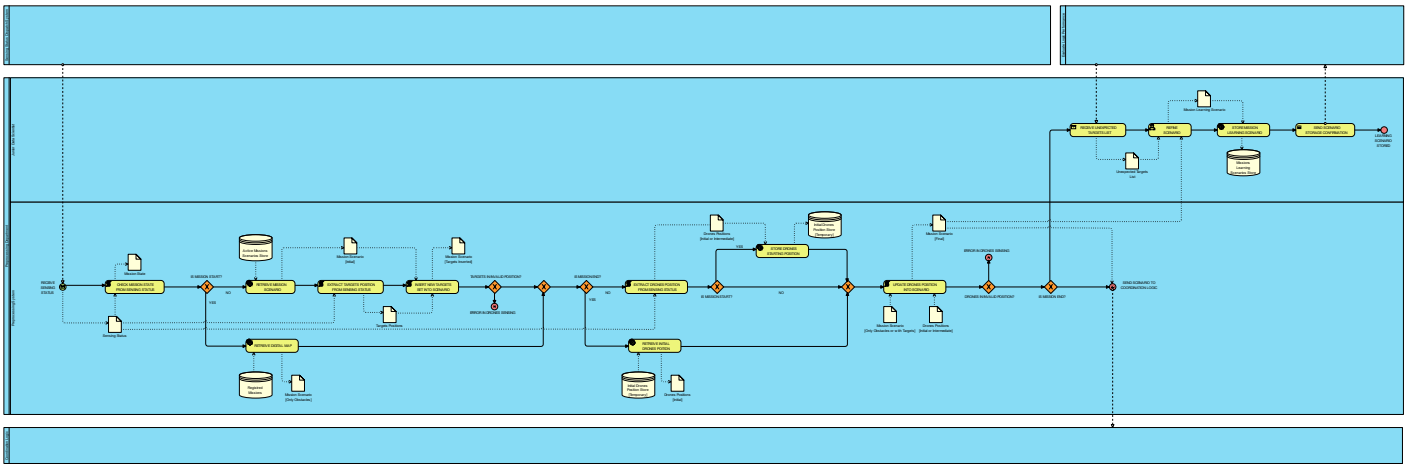
REGISTER MINE DETECTION MISSION



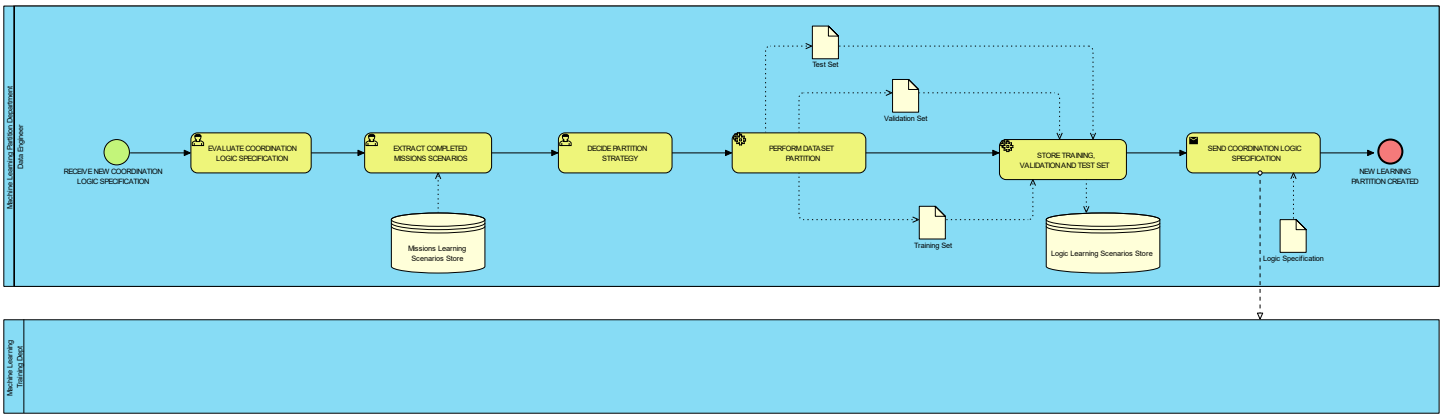
INGEST SWARM SENSING STATUS



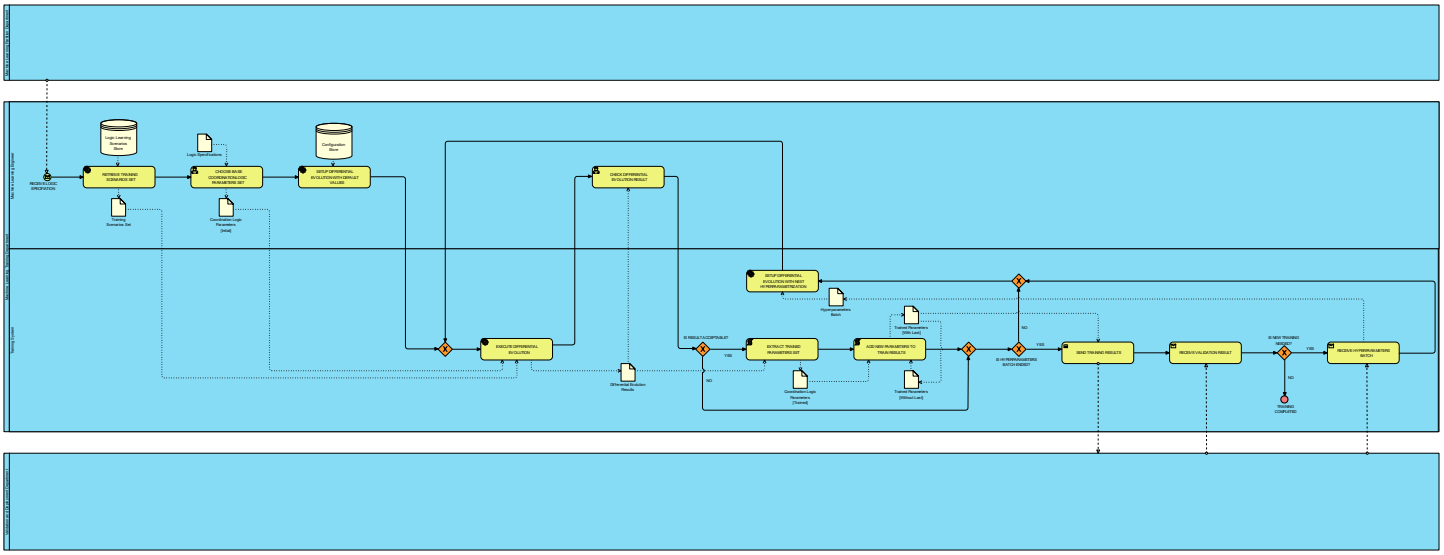
PREPARE MINE DETECTION SCENARIO



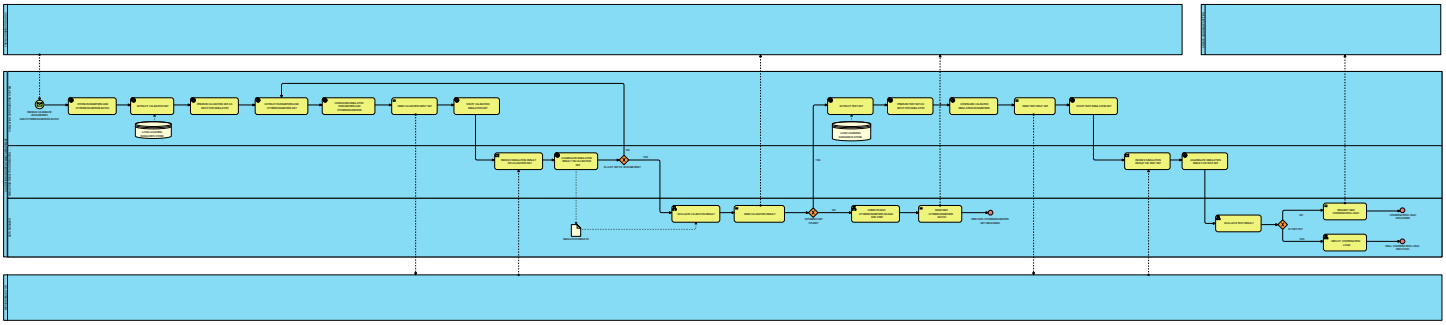
CREATE LEARNING PARTITION



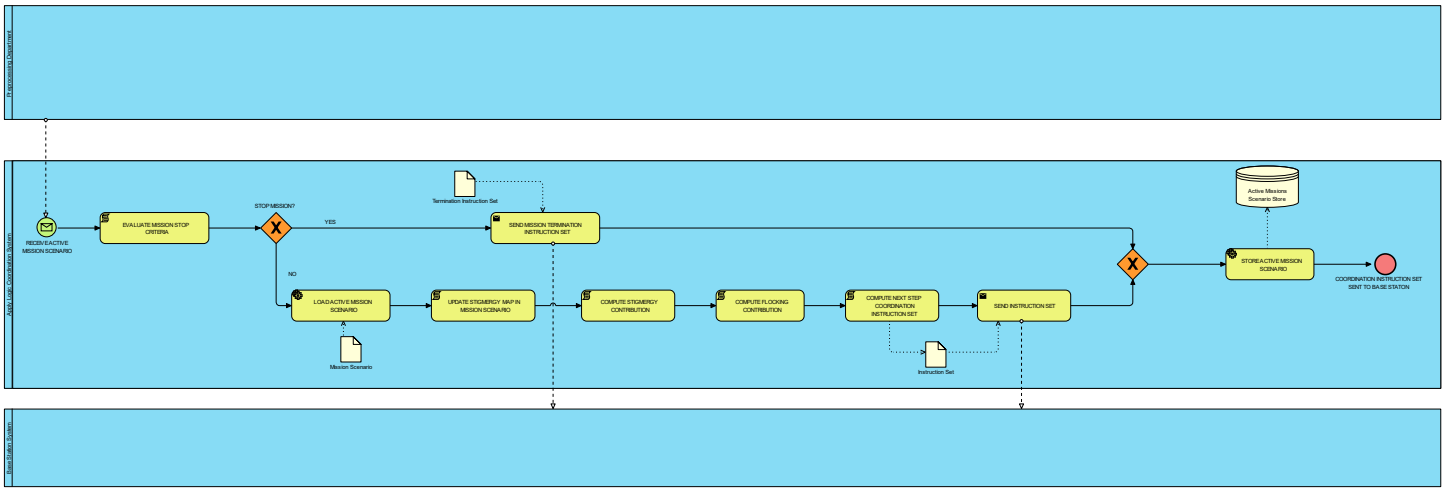
TRAIN COORDINATION LOGIC



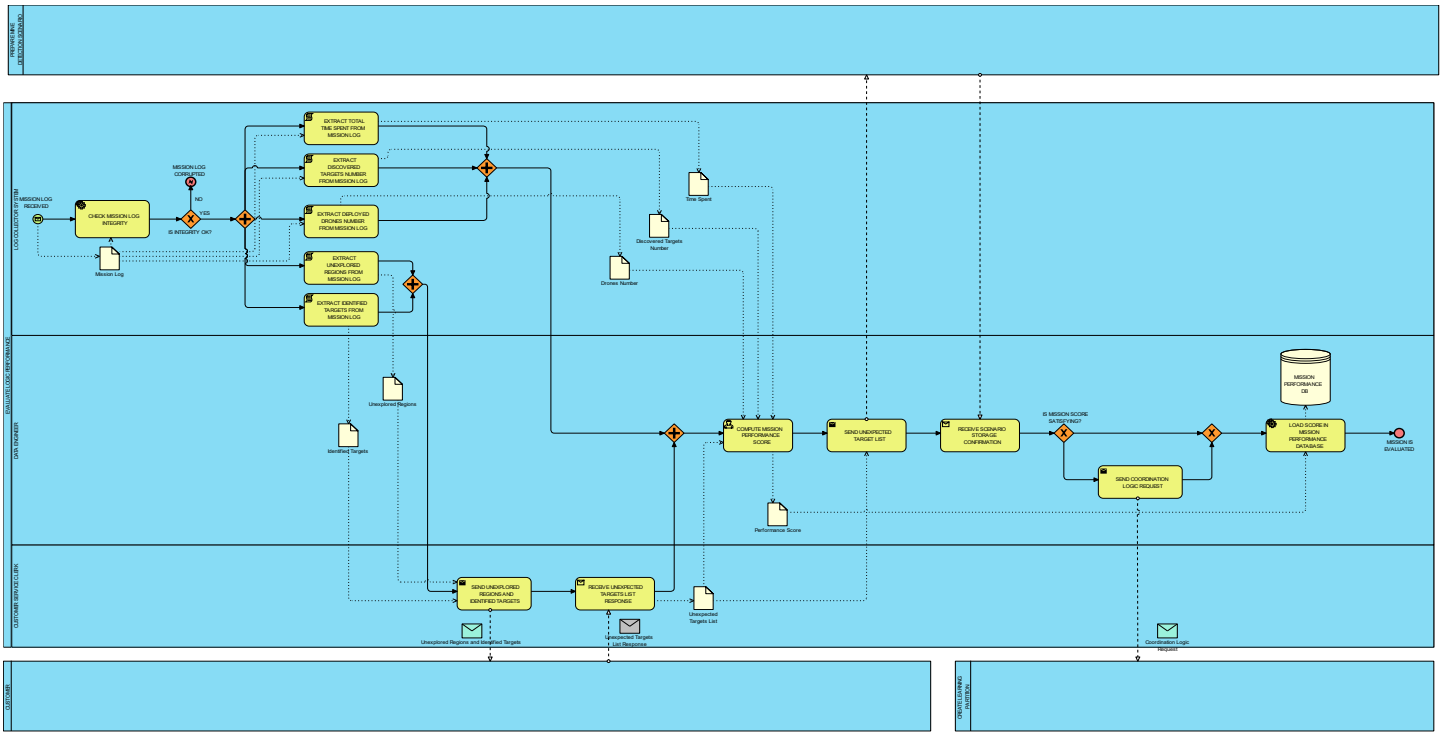
EVALUATE AND DEPLOY COORDINATION LOGIC



APPLY COORDINATION LOGIC



EVALUATE LOGIC PERFORMANCE



PROBLEM IDENTIFICATION AND RESOLUTION

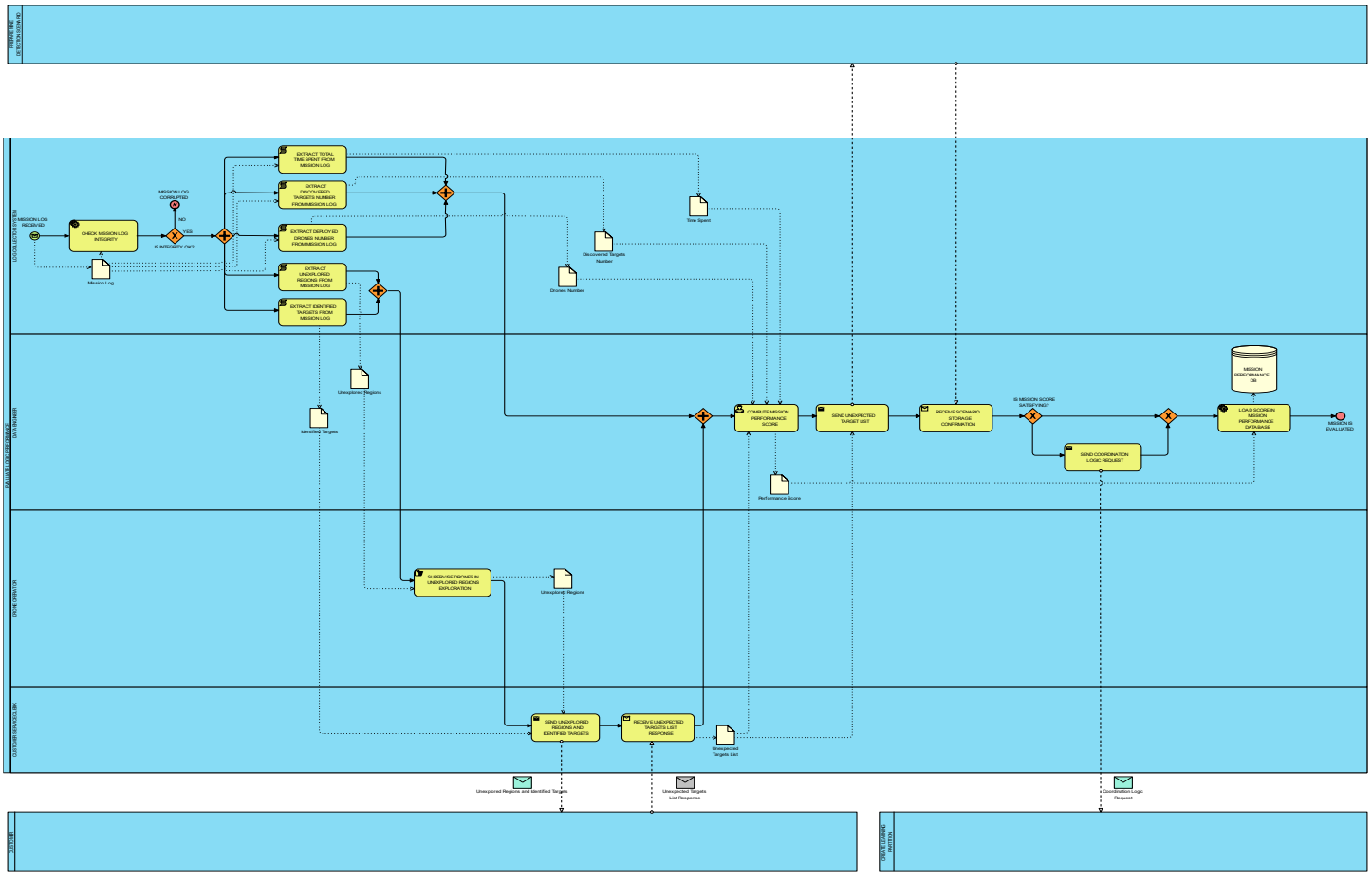
UNSUITABLE TRAINING SCENARIOS

A possible problem in the current pipeline is the reception of a map from the customer for which are not present suitable scenarios to be used in the training stage. Less similar scenarios will be used, not exactly matching the new map's characteristics and for that reason the coordination logic obtained will not produce satisfying results. The number of discovered targets will be low and the total time spent on the mission will be high and unproductive. This issue will produce a very low mission performance score for the logic making the problem evident during performance evaluation.

Unfortunately, performance evaluation is currently performed only in the final pipeline ‘s process, at the end of the mission. An earlier discovery of the problem may mitigate this waste of time and resources.

TO BE MODEL

EVALUATE MISSION PERFORMANCE



COGNITIVE COST EVALUATION

SALARIES PROPORTION

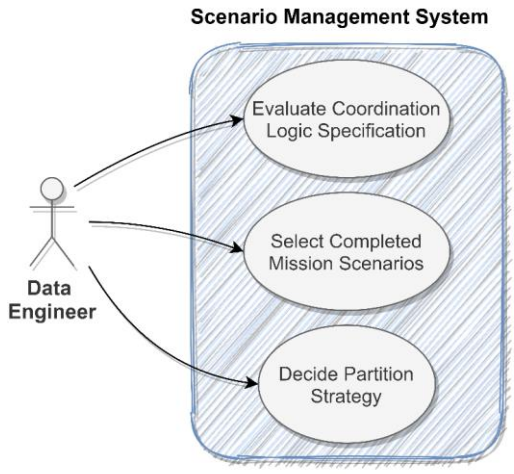
Task	Link	Cost
Requirement Analyst	https://www.glassdoor.com/Salaries/requirement-analyst-salary-SRCH_KO0,19.htm	5
Junior Data Scientist	https://www.glassdoor.com/Salaries/junior-data-scientist-salary-SRCH_KO0,21.htm	8
Data Engineer	https://www.glassdoor.com/Salaries/data-engineer-salary-SRCH_KO0,13.htm	10
Machine Learning Engineer	https://www.glassdoor.com/Salaries/machine-learning-engineer-salary-SRCH_KO0,25.htm	11
Customer Service Clerk	https://www.glassdoor.com/Salaries/customer-service-clerk-salary-SRCH_KO0,22.htm	2
Drone Operator	https://www.glassdoor.com/Salaries/uav-operator-salary-SRCH_KO0,12.htm	3
Mine-Detection (Cost per Square Meter)	https://www.gichd.org/fileadmin/GICHD-resources/info-documents/Manual-mine-clearance-Aug2005/Manual_Mine_Clearance_Book5.pdf	0.1

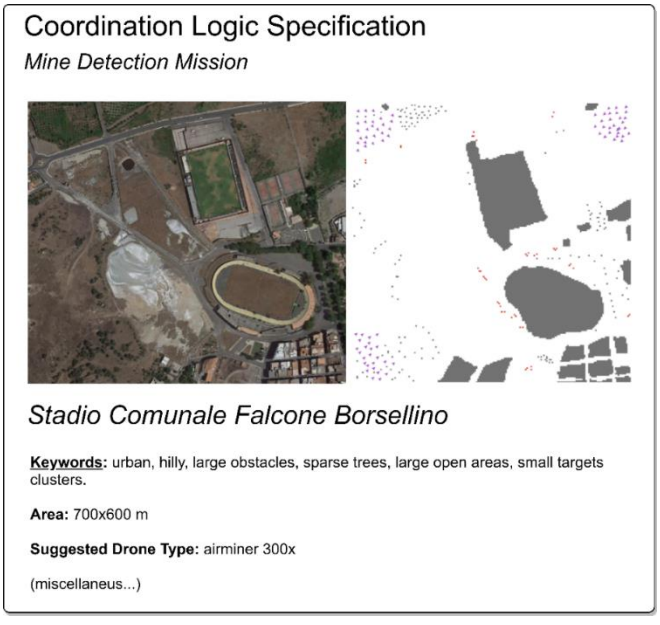
Salaries are computed using the following algorithm:

1. *Take the annual salary*
2. *Round it down*
3. *Divide it by 10000*

Mine detection cost is deducted from the average (between min and max) cost for manual mine clearance with ground preparation, rounded down and divided by 10000. The result is then multiplied by the estimated number of mission per year (20) to be consistent with the salaries.

CREATE LEARNING PARTITION

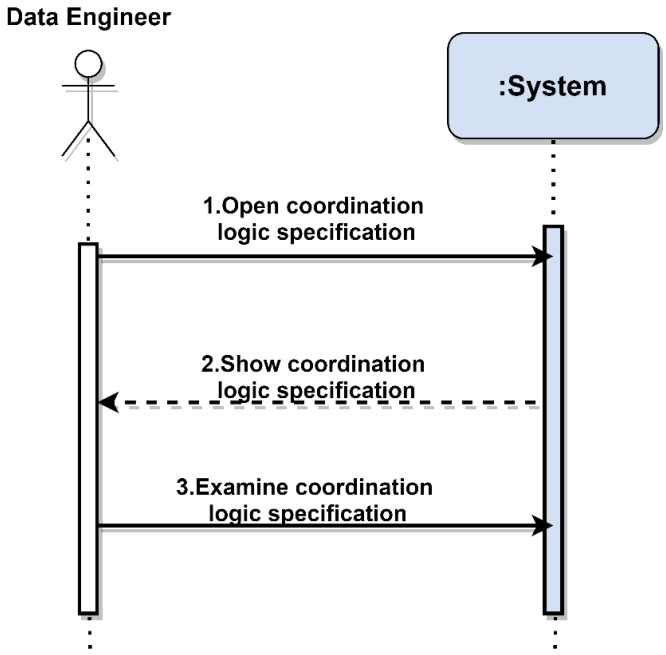




USE CASES

EVALUATE COORDINATION LOGIC SPECIFICATION

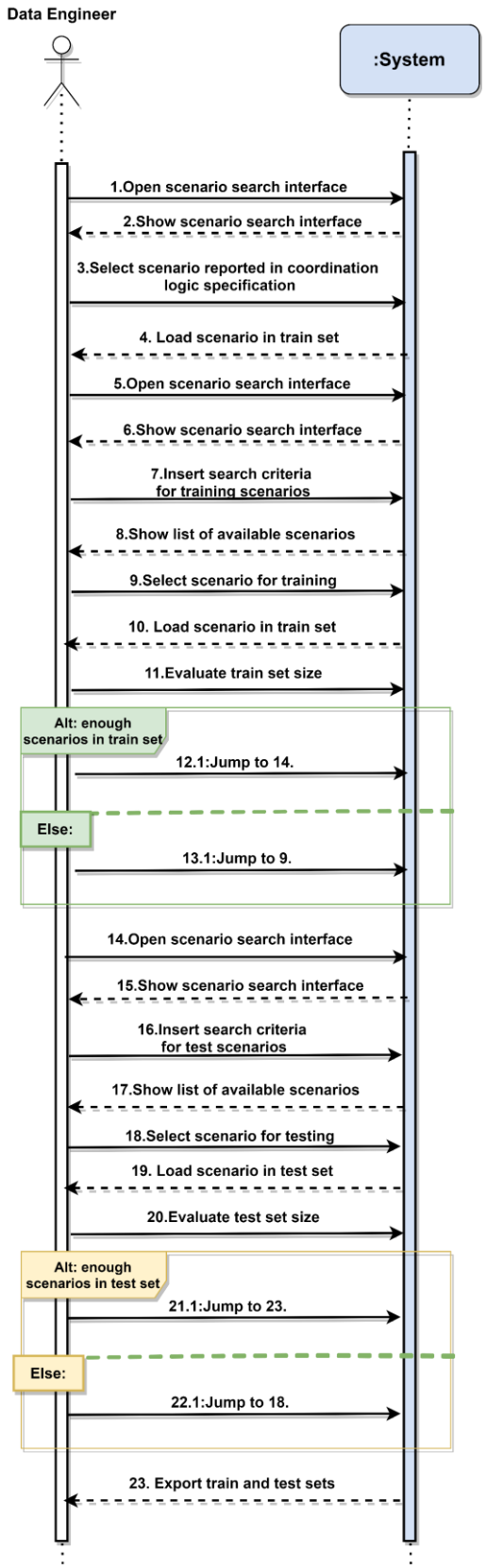
1. *Data Engineer opens coordination logic specification*
2. ***System:** shows coordination logic specification*
3. *Data Engineer examines coordination logic specification*



SELECT COMPLETED MISSIONS SCENARIOS

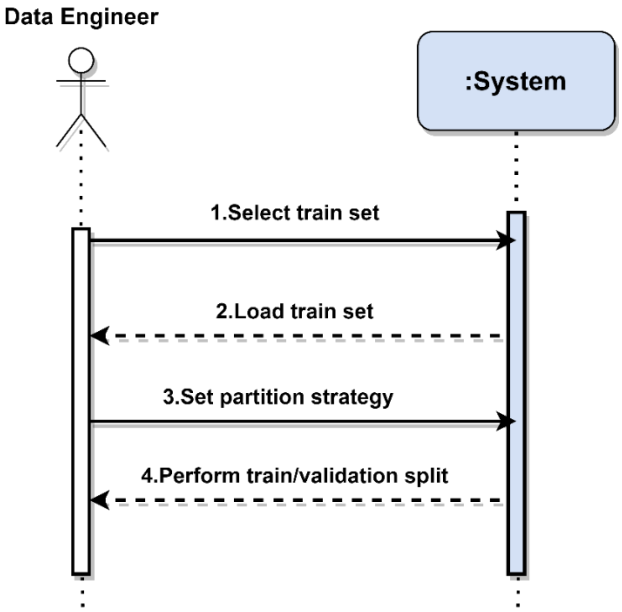
1. *Data Engineer opens scenario search interface*
2. ***System:** shows scenario search interface*
3. *Data Engineer selects scenario reported in Coordination Logic Specification*
4. ***System:** loads scenario in train set*
5. *Data Engineer opens scenario search interface*
6. ***System:** shows scenario search interface*
7. *Data Engineer inserts search criteria for training scenarios*
8. ***System**: shows list of available scenarios*
9. *Data Engineer selects scenario for training*
10. ***System:** loads scenario in train set*
11. *Data Engineer evaluates train set size*
12. ***IF** enough scenarios in train set*
    - *12.1. **GOTO**-> 14.*
13. ***ELSE***
    - *13.1. **GOTO->** 9.*
14. *Data Engineer opens scenario search interface*

15. System: *shows scenario search interface*
16. Data Engineer inserts search criteria for testing scenarios
17. ***System****: shows list of available scenarios*
18. Data Engineer selects scenario for testing
19. ***System:*** *loads scenario in test set*
20. Data Engineer evaluates test set size
21. ***IF*** *enough scenarios in test set*
21.1. ***GOTO->*** *23.*
22. ELSE
22.1. ***GOTO****->18.*
23. ***System:*** *exports train and test set*



DECIDE PARTITION STRATEGY

1. *Data Engineer selects train set*
2. ***System****: loads train set*
3. *Data Engineer sets partition strategy*
4. ***System:*** *perform train/validation split*

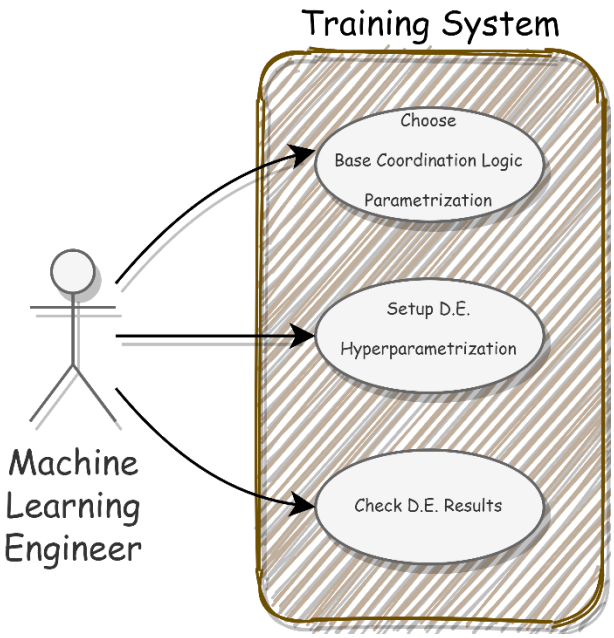


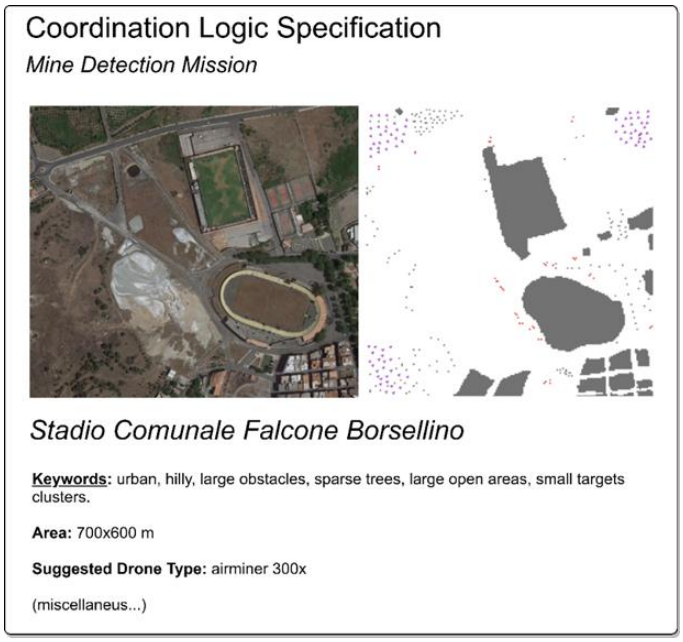
COST EVALUATION

Task	Step	Actor	Cognitive Effort	Execution Probability	Total Cost
Evaluate Coordination Logic Specification	Examine logic specification	*Data engineer*	Understand (2)	1	$10 \cdot 2 \cdot 1 = 20$
Select Completed Missions Scenarios	Insert scenario reported in specification	*Data Engineer*	Remember(1)	1	$10 \cdot 1 \cdot 1 = 10$
	Insert search criteria for training scenarios	*Data engineer*	Understand (2)	1	$10 \cdot 2 \cdot 1 = 20$
	Select scenario for training	*Data engineer*	Analyze (4)	3	$10 \cdot 4 \cdot 3 = 120$
	Check train set size	*Data engineer*	Understand (2)	3	$10 \cdot 2 \cdot 3 = 60$
	Insert search criteria for testing scenarios	*Data engineer*	Understand (2)	1	$10 \cdot 2 \cdot 1 = 20$
	Select scenario for testing	*Data engineer*	Analyze (4)	3	$10 \cdot 4 \cdot 3 = 120$
	Check test set size	*Data engineer*	Understand (2)	3	$10 \cdot 2 \cdot 3 = 60$
Decide Partition Strategy	Set partition strategy	*Data engineer*	Understand (2)	1	$10 \cdot 2 \cdot 1 = 20$
				Total Cost	**450**

TRAIN COORDINATION LOGIC

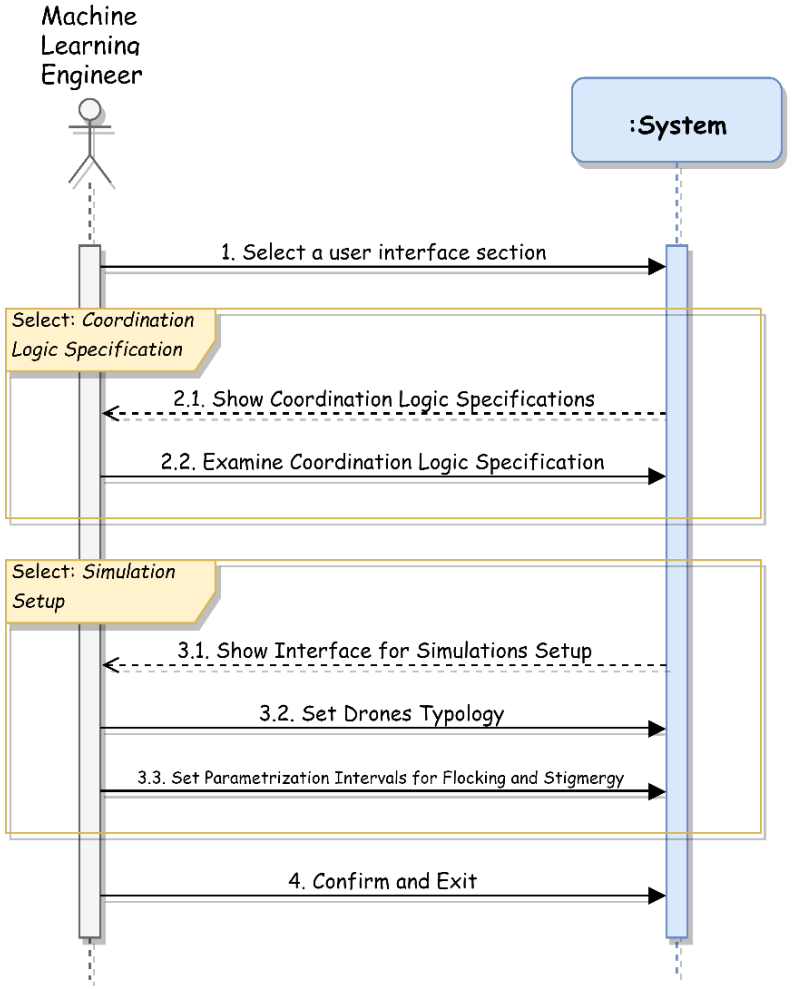
USE CASES





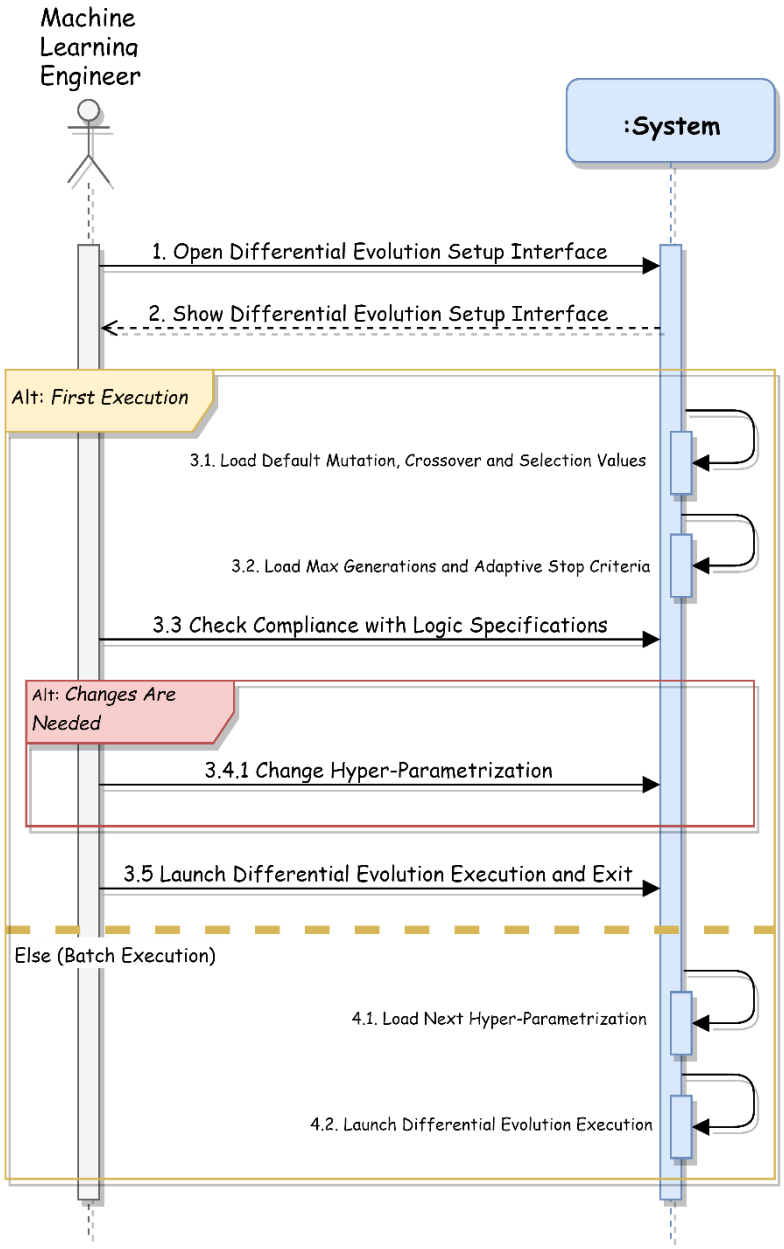
CHOOSE BASE COORDINATION LOGIC PARAMETRIZATION

1. *M.L. engineer* selects a user interface section
2. **IF** *coordination logic specifications* is selected
   - 2.1. ***System****:* shows coordination logic specifications
   - 2.2. *M.L. engineer* examines coordination logic specifications
3. **IF** *simulations setup* is selected
   - 3.1. ***System***: shows interface for simulation setup
   - 3.2. *M.L. engineer* sets drones typology
   - 3.3. *M.L. engineer* sets parametrization intervals for *flocking* and *stigmergy*
4. *M.L. engineer* confirms and exits



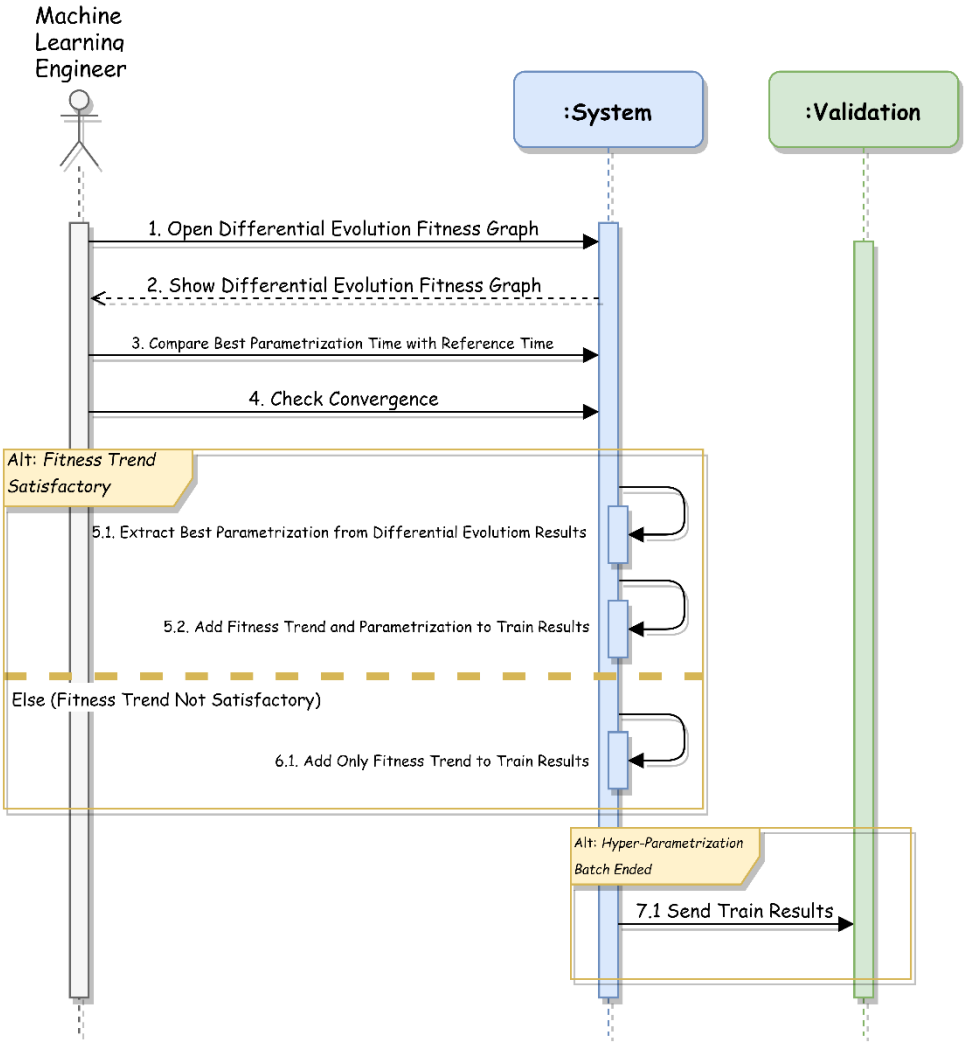
SETUP DIFFERENTIAL EVOLUTION HYPER-PARAMETRIZATION

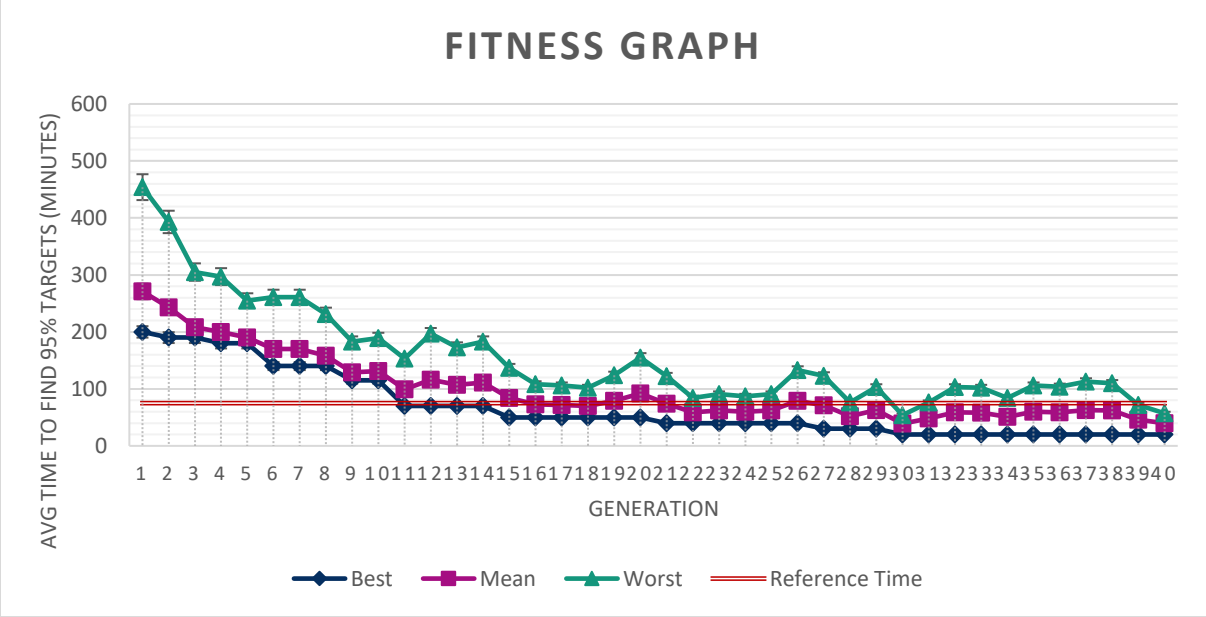
1. *M.L. engineer* opens differential evolution setup interface
2. ***System***: shows differential evolution setup interface
3. **IF** first execution
   - 3.1. ***System***: loads default *mutation*, *crossover* and *selection* values
   - 3.2. ***System***: loads default *max generations* and *adaptive stop criteria*
   - 3.3. *M.L. engineer* checks compliance with logic specifications
   - 3.4. **IF** changes are needed
     - 3.4.1. *M.L. engineer* manually changes hyper-parametrization
   - 3.5. *M.L. engineer* launches differential evolution execution and exits
4. **ELSE** (batch execution)
   - 4.1. ***System***: loads next hyper-parametrization
   - 4.2. ***System***: launches differential evolution execution



CHECK DIFFERENTIAL EVOLUTION RESULTS

1. *M.L. engineer* opens differential evolution *fitness graph*
2. ***System***: shows differential evolution *fitness graph*
3. *M.L. engineer* compares best parametrization time with reference time
4. *M.L. engineer* checks convergence
5. **IF** fitness trend satisfactory
   - **5.1. *System***: extracts best parametrization from differential evolution results
   - **5.2. *System***: adds fitness trend and parametrization to *train results*
6. **ELSE** (fitness trend not satisfactory)
   - **6.1. *System***: adds only fitness trend to *train results*
7. **IF** hyper-parametrizations batch ended
   - 7.1. ***System***: sends *train results* to **<u>validation</u>**



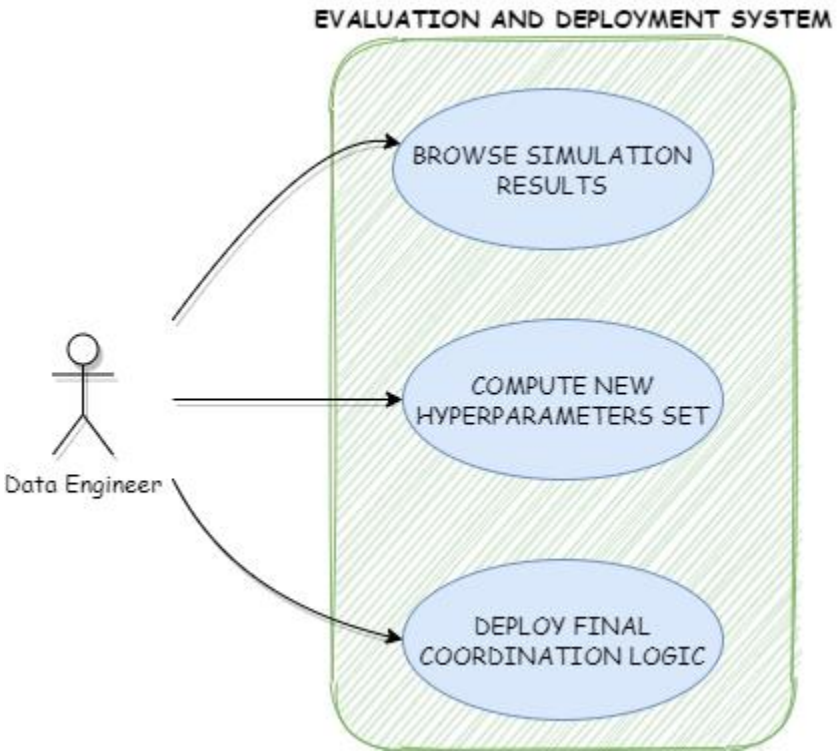


COST EVALUATION

Task	Step	Actor	Cognitive Effort	Execution Probability	Total Cost
Choose Base Coordination Logic Parametrization	Examine logic specification	*M.L. engineer* (11)	Understand (2)	1	$11 \cdot 2 \cdot 1 = 22$
	Set drones typology	*M.L. engineer* (11)	Remember (1)	1	$11 \cdot 1 \cdot 1 = 11$
	Set parametrization intervals	*M.L. engineer* (11)	Apply (3)	1	$11 \cdot 3 \cdot 1 = 33$
				Total Task Cost:	66
Setup Differential Evolution Hyper-Parametrization	Check compliance with logic specifications	*M.L. engineer* (11)	Understand (2)	1	$11 \cdot 2 \cdot 1 = 22$
	Manually changes hyper-parametrization	*M.L. engineer* (11)	Apply (3)	0.2	$11 \cdot 3 \cdot 0.2 = 6$
				Total Task Cost:	28
Check Differential Evolution Results	Compares best time and reference time	*M.L. engineer* (11)	Understand (2)	1	$11 \cdot 2 \cdot 1 = 22$
	Checks convergence	*M.L. engineer* (11)	Understand (2)	1	$11 \cdot 2 \cdot 1 = 22$
				Total Task Cost:	44

EVALUATE AND DEPLOY COORDINATION LOGIC

USE CASE

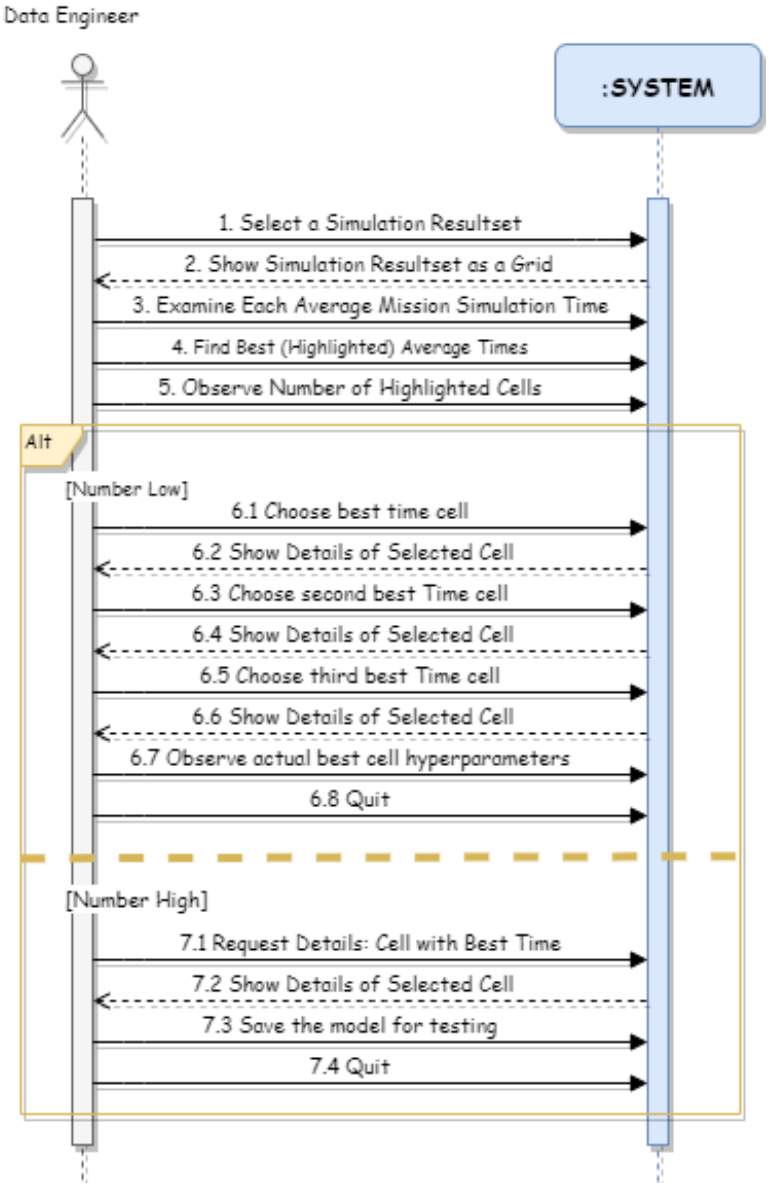


GRID EXAMPLE

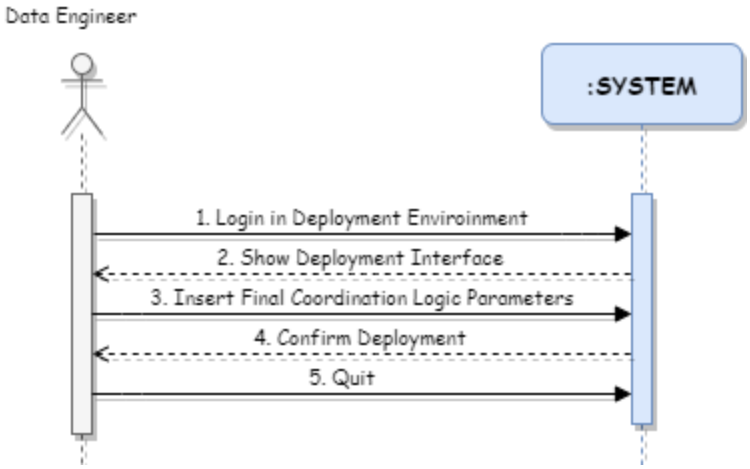
CR \ F	0.65	0.7	0.75	0.8
0.45	180	140	155	170
0.5	170	136	138	167
0.55	198	186	177	200

The cells with score close enough to the best are highlighted automatically, using color saturation to identify their closeness to the best score of the simulation.

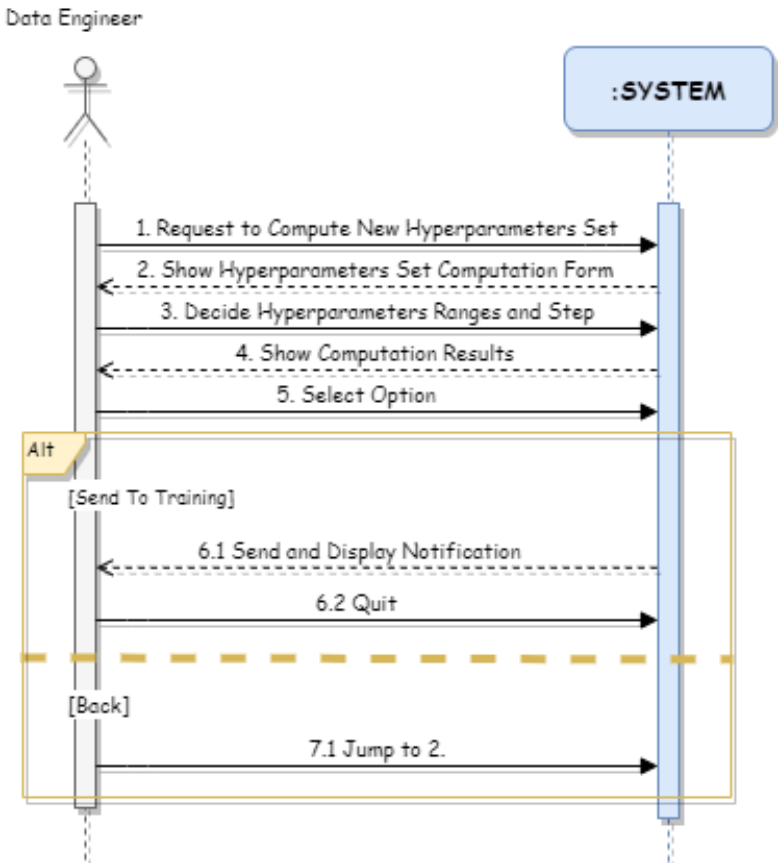
BROWSE SIMULATION RESULTS



DEPLOY FINAL COORDINATION LOGIC



COMPUTE NEW HYPERPARAMETERS SET



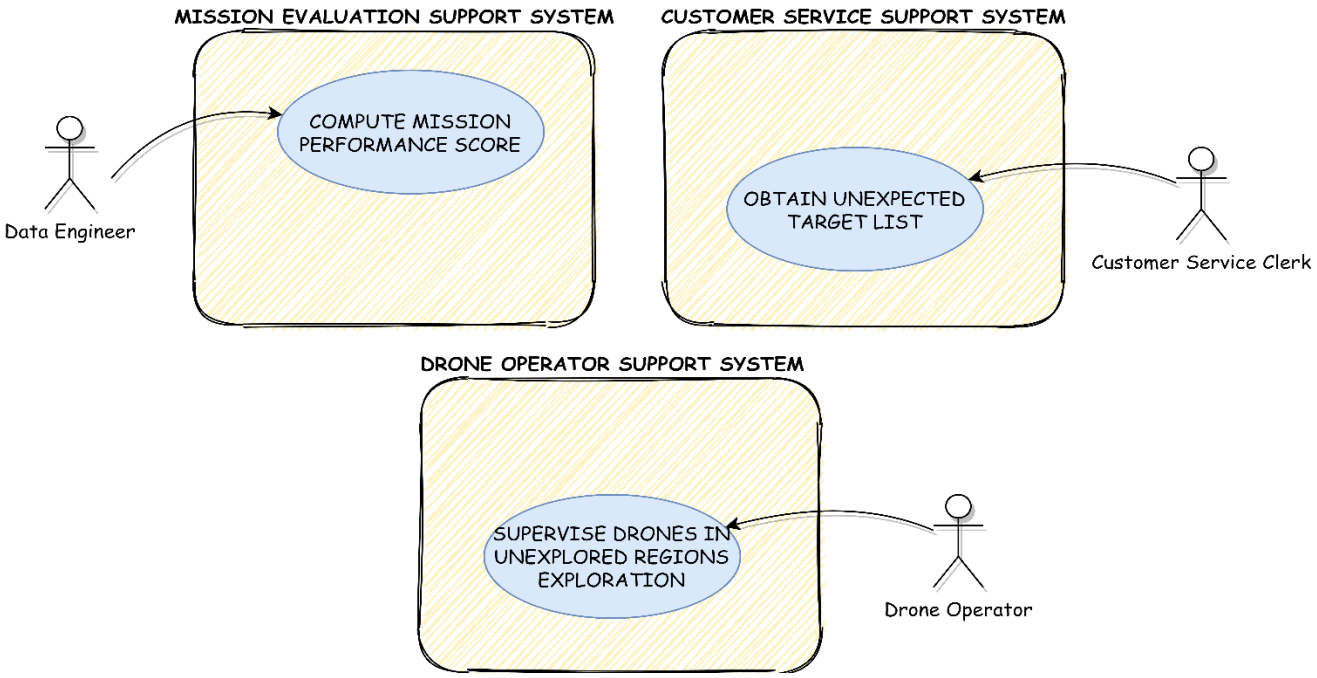
COST EVALUATION

Use Case	Step	Actor	Cognitive Effort	Execution Probability	Total Cost
Browse Simulation Results	Examine Average Mission Simulation Time	*Data engineer*	Remember (1)	*1*	10*1*1=10
	Find Best Average Time	*Data engineer*	Understand (1)	*1*	10*1*1=10
	Evaluate number of Highlighted Cells	*Data engineer*	Understand (2)	*1*	10*2*1=20
	Examine Cell Details	*Data engineer*	Analyze (4)	*3 Validation* *1 Test*	Validation: 10*4*3=120 Test: 10*4*1=40
Compute New Hyperparameters Set	Decide Hyperparameters Range and Step	*Data engineer*	Understand (2)	*1*	10*2*1=20
	Inspect Computation Result	*Data engineer*	Remember (1)	*1*	10*1*1=10
Deploy Final Coordination Logic	Insert Final Parameters Set	*Data engineer*	Remember (1)	*1*	10*1*1=10

For accurate cost computation, we need to consider that a one-cell grid is presented by the system for the first training result and test simulations.

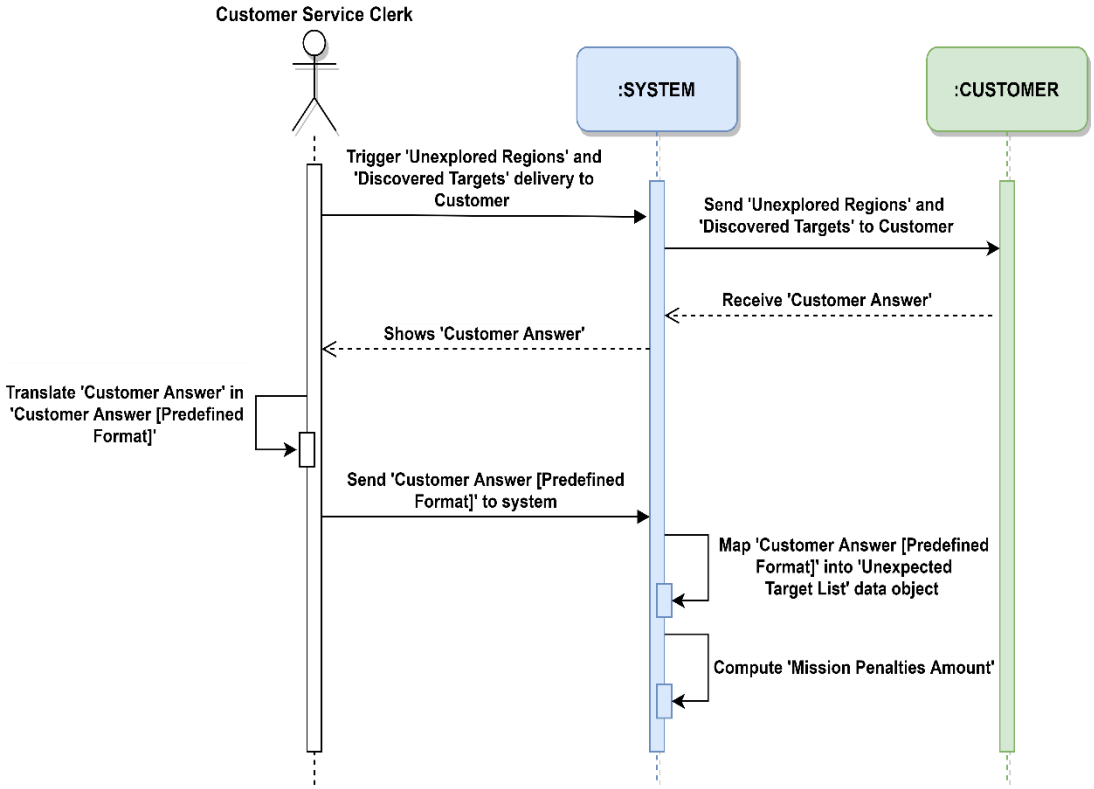
EVALUATE LOGIC PERFORMANCE

USE CASES



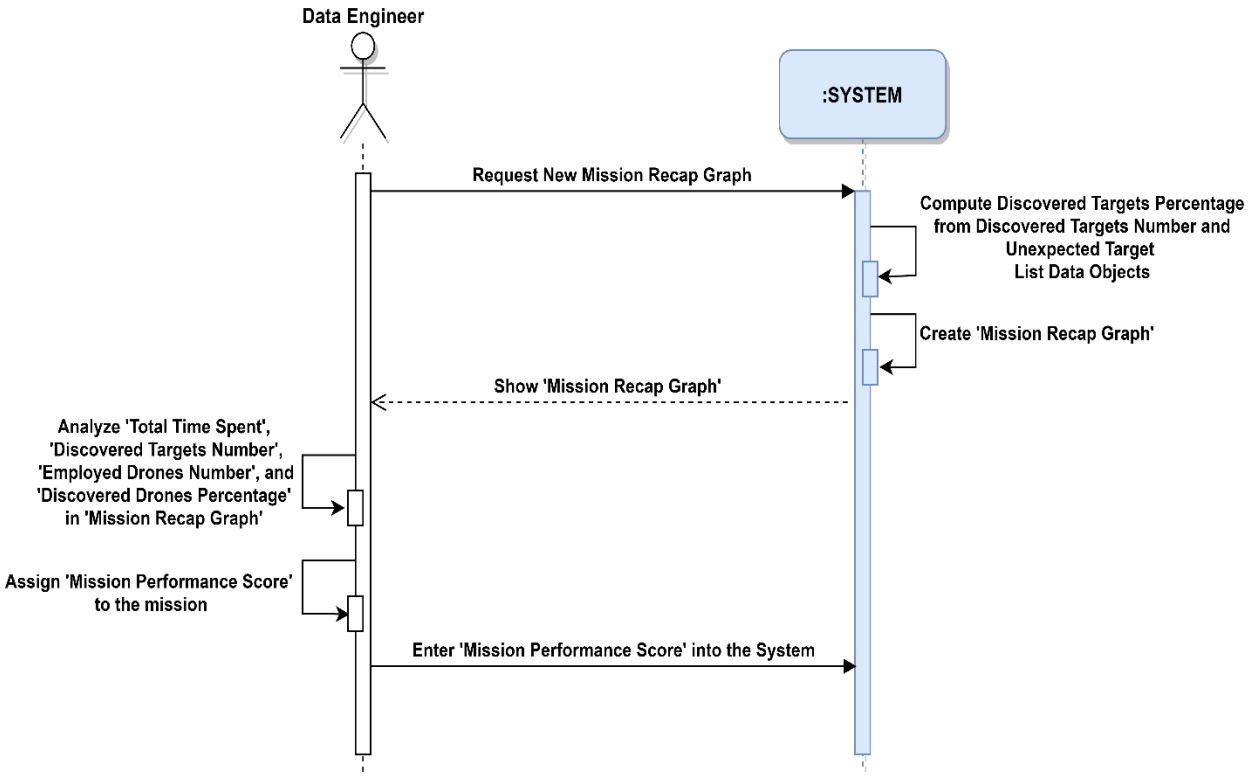
OBTAIN UNEXPECTED TARGET LIST

1. *Customer Service Clerk triggers 'Unexplored Regions' and 'Identified Targets' delivery to Customer*
2. ***System:** Sends 'Unexplored Regions' and 'Identified Targets' to Customer*
3. ***System:** receives 'Customer Answer'*
4. ***System:** shows 'Customer Answer'*
5. *Customer Service Clerk translates 'Customer Answer' in 'Customer Answer [Predefined Format]'*
6. *Customer Service Clerk sends 'Customer Answer [Predefined Format]' to system*
7. ***System:** maps 'Customer Answer [Predefined Format]' into 'Unexpected Target List' data object*
8. ***System:** computes 'Mission Penalties Amount'*



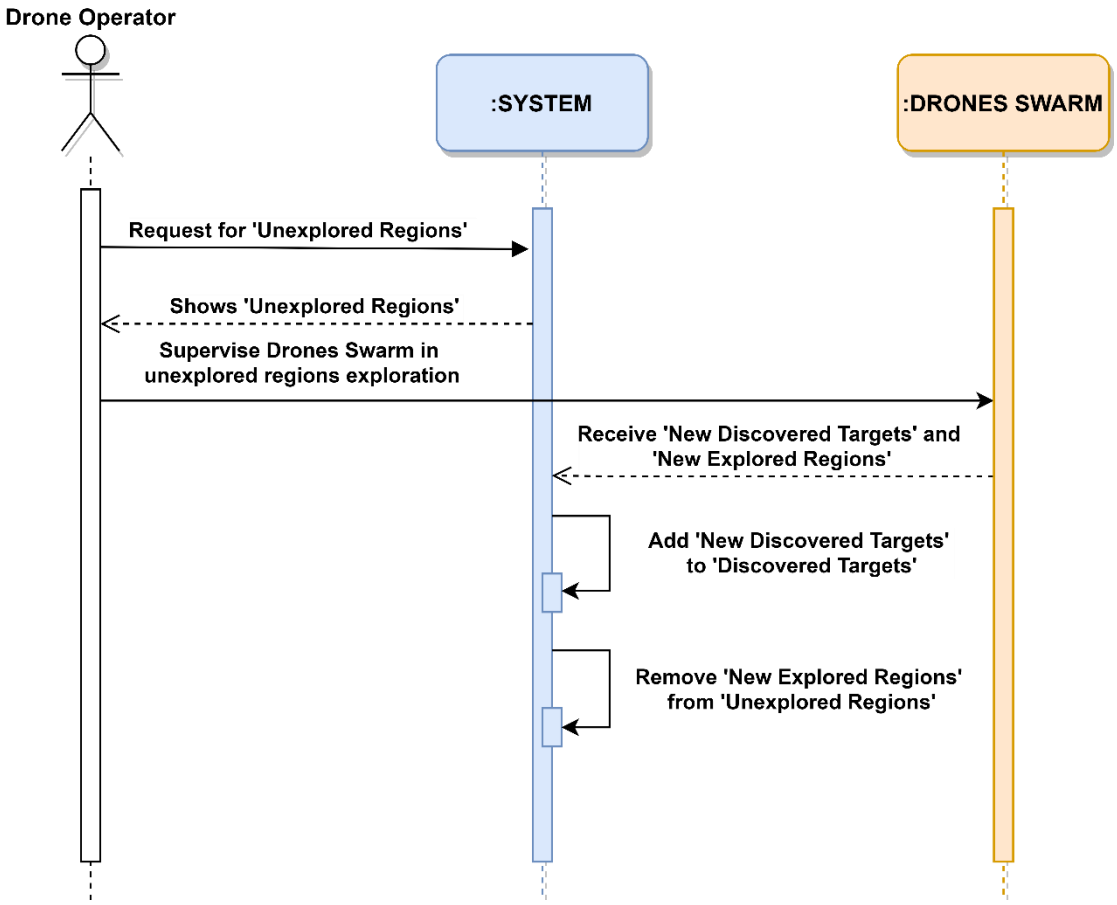
COMPUTE MISSION PERFORMANCE SCORE

1. *Data Engineer requests for new 'Mission Recap Graph'*
2. ***System:** compute Discovered Targets Percentage from Discovered Targets Number and Unexpected Target List Data Objects*
3. ***System:** creates 'Mission Recap' Graph*
4. ***System:** shows 'Mission Recap' Graph*
5. *Data Engineer analyzes 'Total Time Spent', 'Discovered Targets Number', 'Employed Drones Number' and 'Discovered Drones Percentage' in 'Mission Recap Graph'*
6. *Data Engineer assigns a 'Mission Performance Score' to the mission*
7. *Data Engineer enters 'Mission Performance Score' into the System*

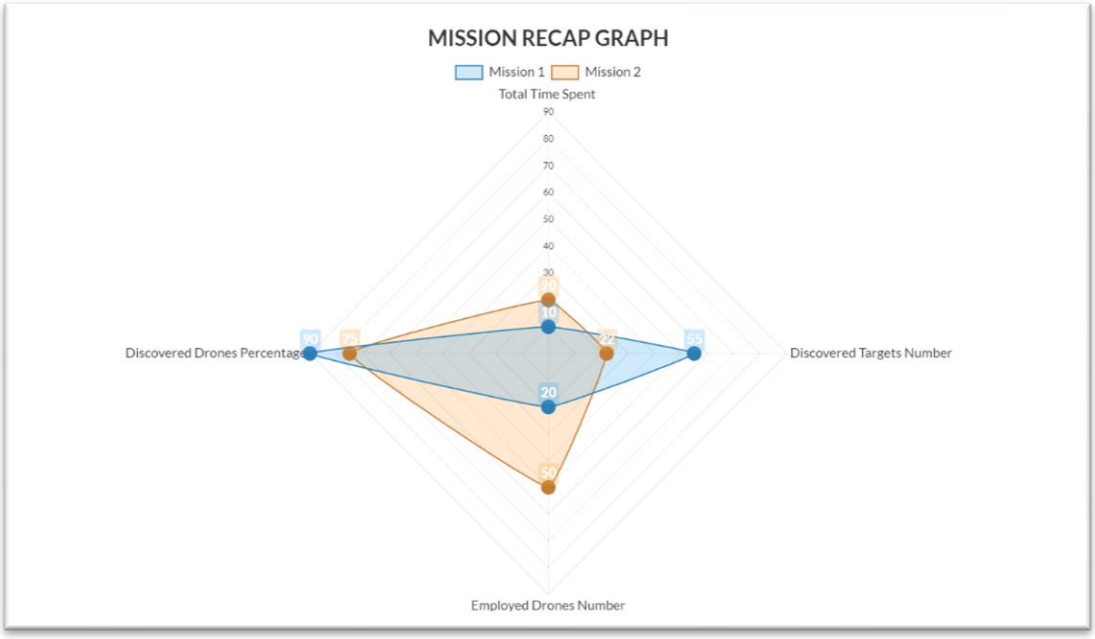


SUPERVISE DRONES IN UNEXPLORED REGIONS EXPLORATION (“TO BE” MODEL ONLY)

1. *Drone Operator requests for ‘Unexplored Regions’*
2. ***System:** Shows ‘Unexplored Regions’*
3. *Drone Operator pilots Drones Swarm in unexplored regions*
4. ***System:** receive ‘New Discovered Targets’ and ‘New Explored Regions’ from Drones Swarm*
5. ***System:** add ‘New Discovered Targets’ to ‘Discovered Targets’*
6. ***System:** remove ‘New Explored Regions’ from ‘Unexplored Regions’*



MISSION RECAP GRAPH



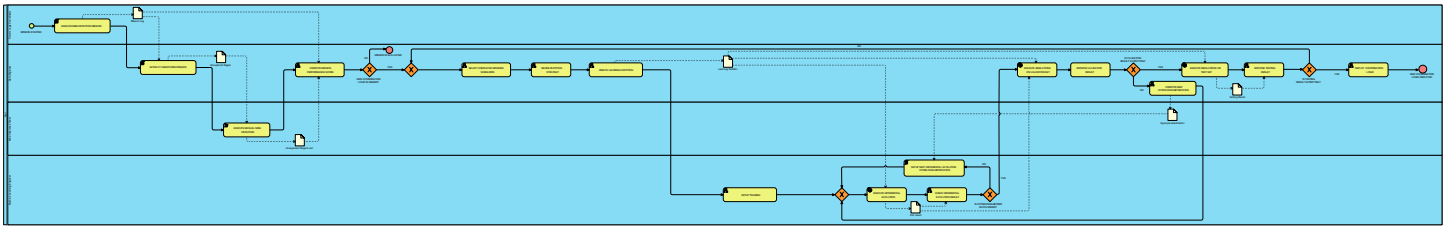
COST EVALUATION

Task	Step	Actor	Cognitive Effort	Total Cost
Obtain Unexpected Target List	Customer Service Clerk translates 'Customer Answer' into a 'Customer Answer [Predefined Format]'	*Customer Service Clerk*	Understand (2)	*2*2 = 4*
	Customer Service Clerk sends 'Customer Answer [Predefined Format]' to system	*Customer Service Clerk*	Remember (1)	*2*1 = 2*
			Total Task Cost:	6
Compute Mission Performance Score	Data Engineer analyzes 'Total Time Spent', 'Discovered Targets Number', 'Employed Drones Number' and 'Discovered Drones Percentage' from 'Mission Recap Graph'	*Data Engineer*	Apply (3)	*10*3 = 30*
	Data Engineer assigns a 'Mission Performance Score' to the mission	*Data Engineer*	Understand (2)	*10*2 = 20*
	Data Engineer enters 'Mission Performance Score' into the System	*Data Engineer*	Remember (1)	*10*1 = 10*
			Total Task Cost:	*60*
Pilot Drones in Unexplored Regions	Drone Operator supervises Drones Swarm in unexplored regions	*Drone Operator*	*Apply (3)*	*3*3 = 9*
			Total Task Cost:	*9*

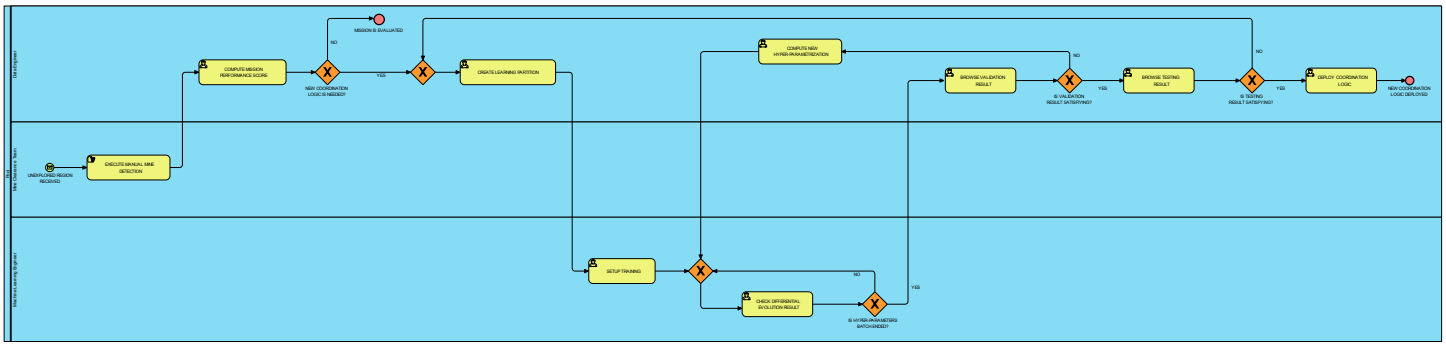
SIMULATIONS

MODELS

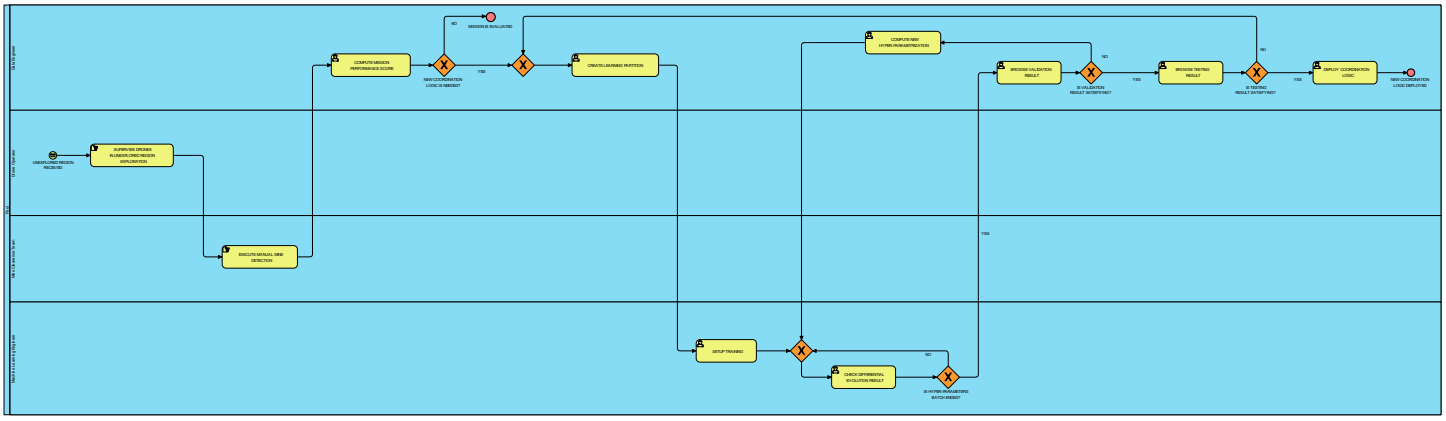
SIMULATION MODEL "AS IS" OVERVIEW



SIMULATION MODEL "AS IS"



SIMULATION MODEL "TO BE"



SIMULATIONS SETUP

COSTS EXPLANATION

This subsection contains the explanation of the costs used in the simulation phase.

- **Browse Testing Result: 80**. The cost for the task *Browse Testing Result* is fixed to 80 according to the evaluation previously done. Execution probability of the step *Examine Cells Details* is considered equal to 1.

- **Browse Validation Result: 160.**
  The cost for the task *Browse Validation Result* is fixed to 160 according to the evaluation previously done. This time, execution probability of the step *Examine Cells Details* is considered equal to 3, leading to an higher total cost.
- **Check Differential Evolution Result: 44.**
  The cost for the task *Check Differential Evolution Result* is fixed to 44 according to the evaluation previously done.
- **Create Learning Partition: 450**
  The cost for the task *Create Learning Partition* is fixed to 450 according to the evaluation previously done (considering 4 training scenarios and 3 test scenarios).
- **Compute Mission Performance Score: 60.**
  The cost for the task *Compute Mission Performance Score* is fixed to 60 according to the evaluation previously done.
- **Compute New Hyper-Parametrization: 30.**
  The cost for the task *Compute New Hyper-Parametrization* is fixed to 30 according to the evaluation previously done.
- **Deploy Coordination Logic: 10.**
  The cost for the task *Deploy Coordination Logic* is fixed to 10 according to the evaluation previously done.
- **Execute Manual Mine Detection ("As Is"): 10 500.**
  The cost is calculated considering a non-performing coordination logic, which makes it necessary to manually detect mines in an area equal to 25% ($\frac{1}{4}$) of the map, it is also assumed that a map has an average size of $700\ m\ x\ 600\ m = \ 420\ 000\ m^2$. The average cost of mine clearance per square meter is 0.1 using our scale. The *standard deviation* is set to $2\ 100$ (difference between $\frac{1}{3}$ and $\frac{1}{4}$ of the map's exploration)
  $$420\ 000 \cdot 0.25 \cdot 0.1 = 10\ 500\ \pm\ 2\ 100$$
- **Execute Manual Mine Detection ("To Be"): 5 250.**
  Using the same reasoning as the previous point and considering a performing logic, we can hypothesize an area of 12.5% ($\frac{1}{8}$) on which to perform manual mine detection. The *standard deviation* is set to $1\ 758$ (difference between $\frac{1}{8}$ and $\frac{1}{6}$ of the map's exploration)
  $$420\ 000 \cdot 0.125 \cdot 0.1 = 5\ 250\ \pm\ 1\ 758$$
- **Setup Training: 94.**
  The cost for the task *Setup Training* is fixed to 94 according to the evaluation previously done. This is the sum between the costs of the subtasks *Choose Base Coordination Logic Parametrization* and *Setup Differential Evolution Hyper-Parametrization.* Execution probability of *Manually changes hyper-parametrization* step is set to 0.2.
- **Supervise Drones in Unexplored Regions Exploration: 9.**
  The cost for the task *Supervise Drones in Unexplored Regions Exploration* is fixed to 9 according to the evaluation previously done.

BRANCHING PROPORTIONS

- **"Is New Coordination Logic Needed?" ("As Is"):** *5% → yes, 95% → no.*
  Performing training fewer times leads to less performing logic as described previously. For every 100 missions 5 different logics are created.
- **"Is New Coordination Logic Needed?" ("To Be"):** *30% → yes, 70% → no.*
  To decrease unexplored regions by 12.5%, a 25% increase in probability of training a new logic was assumed. For every 100 missions 30 different logics are created. This value is justified by the fact that the effectiveness of the training decreases as we approach 0% of unexplored areas and therefore an exponentially greater number of training is required to improve an already performing logic.
- **"Is Validation Result Satisfying?":** *20% → yes, 80% → no.*
  We assumed that on average 5 hyper-parametrization optimization are required.
- **"Is Hyper-Parameters Batch Ended?":** *4% → yes, 96% → no.*
  We assumed that on average the check is done on 20 different hyper-parameters configurations. The validation results are not optimal in 4/5 of the cases. The training is done on 5 batches, the first batch trained is the default hyper-parametrization, other 4 batches are 5x5 grid of hyper-parameters.
  $$\frac{(1\ +(5x5\cdot 4)\ )}{5} \cong 20$$
- **"Is Testing Result Satisfying?":** *98% → yes, 2% → no.*
  We have estimated that out of 100 trainings only 2 must be re-run from scratch due to a bad choice of scenarios.

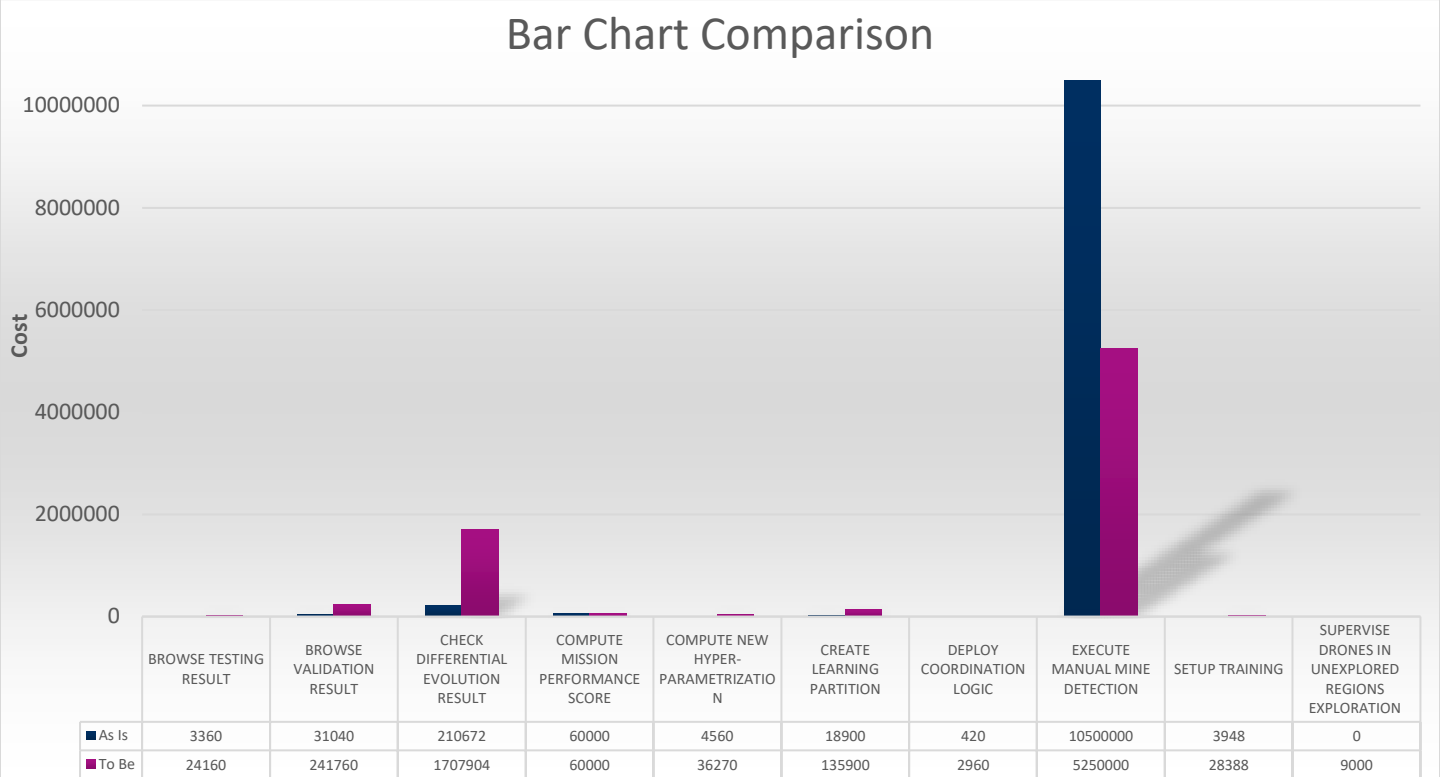
SIMULATIONS RESULT

"To Be" model offers better results if compared to ***"As Is"*** model and led to an **improvement** of $\mathbf{31.03 \pm 1.26\%}$ with a confidence of *95%*.

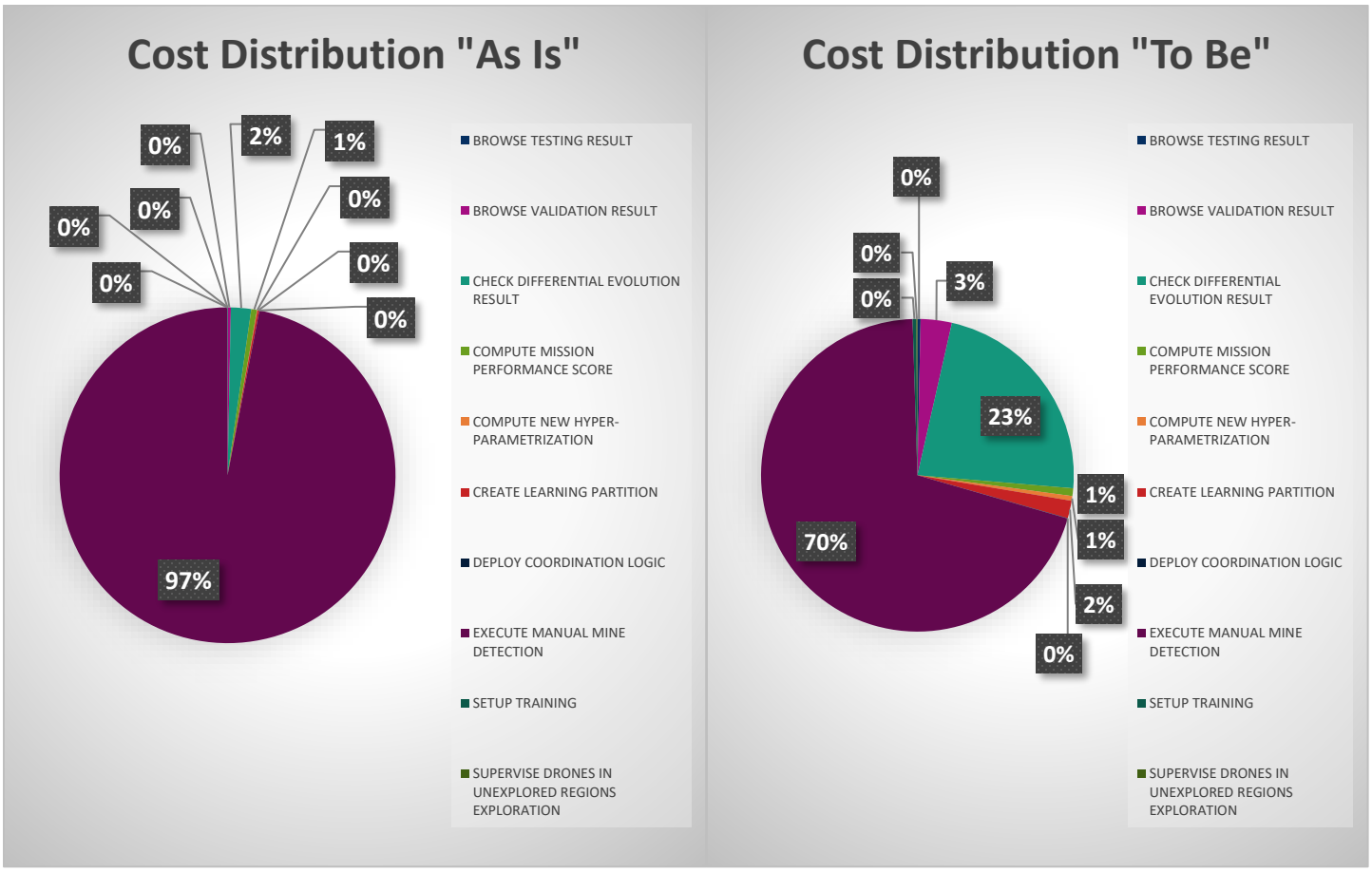
Results	Mean	Standard Deviation	95% Confidence Interval
"As Is" Model	10 922 857	62 831	$10\ 922\ 857 \pm 55\ 100$ $(10\ 900\ 000\ to\ 11\ 000\ 000)$
"To Be" Model	7 533 958	162 856	$7\ 533\ 958 \pm 143\ 000$ $(7\ 390\ 000\ to\ 7\ 680\ 000)$
Difference	3 388 899	159 694	$3\ 388\ 899 \pm 1\ 40\ 000$ $(3\ 250\ 000\ to\ 3\ 530\ 000)$
Percentage	31.03 %	1.44 %	$31.03 \pm 1.26\%$ $(29.8\%\ to\ 32.3\%)$

MODELS COMPARISON

The following diagrams show and compare costs distribution in ‘As Is’ and ‘To Be’ models.

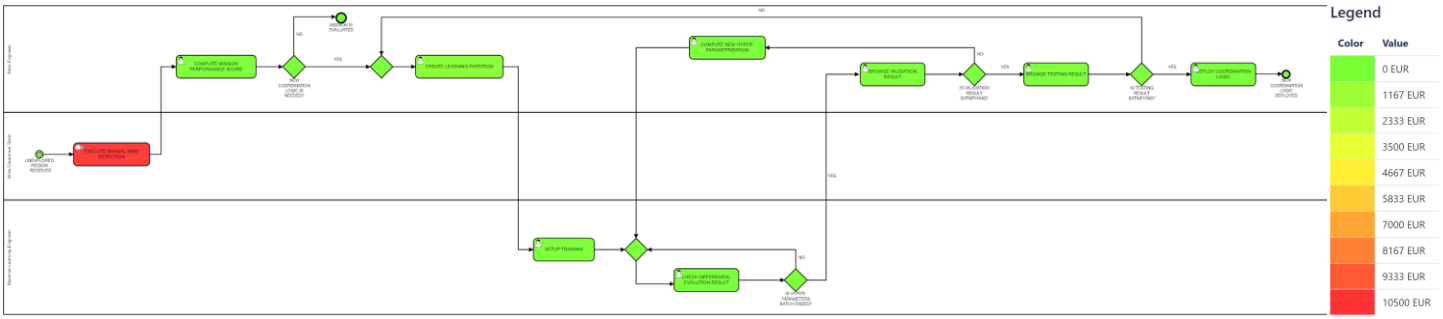


	BROWSE TESTING RESULT	BROWSE VALIDATION RESULT	CHECK DIFFERENTIAL EVOLUTION RESULT	COMPUTE MISSION PERFORMANCE SCORE	COMPUTE NEW HYPER-PARAMETRIZATION	CREATE LEARNING PARTITION	DEPLOY COORDINATION LOGIC	EXECUTE MANUAL MINE DETECTION	SETUP TRAINING	SUPERVISE DRONES IN UNEXPLORED REGIONS EXPLORATION
As Is	3360	31040	210672	60000	4560	18900	420	10500000	3948	0
To Be	24160	241760	1707904	60000	36270	135900	2960	5250000	28388	9000

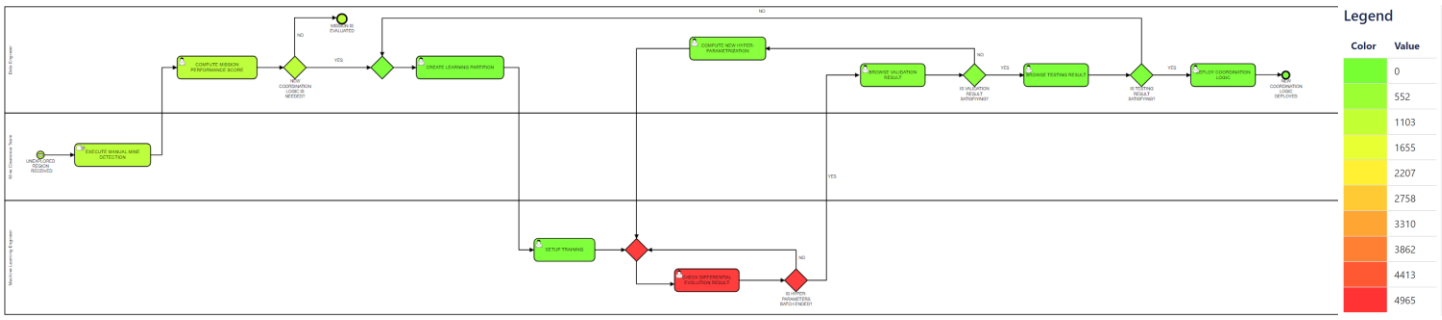


HEATMAPS "AS IS"

Cost:

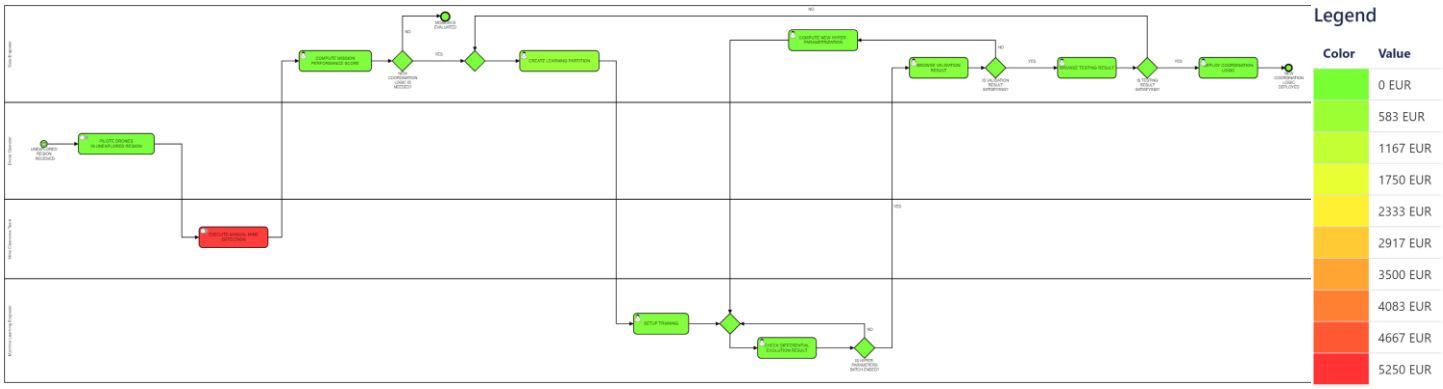


Counts:

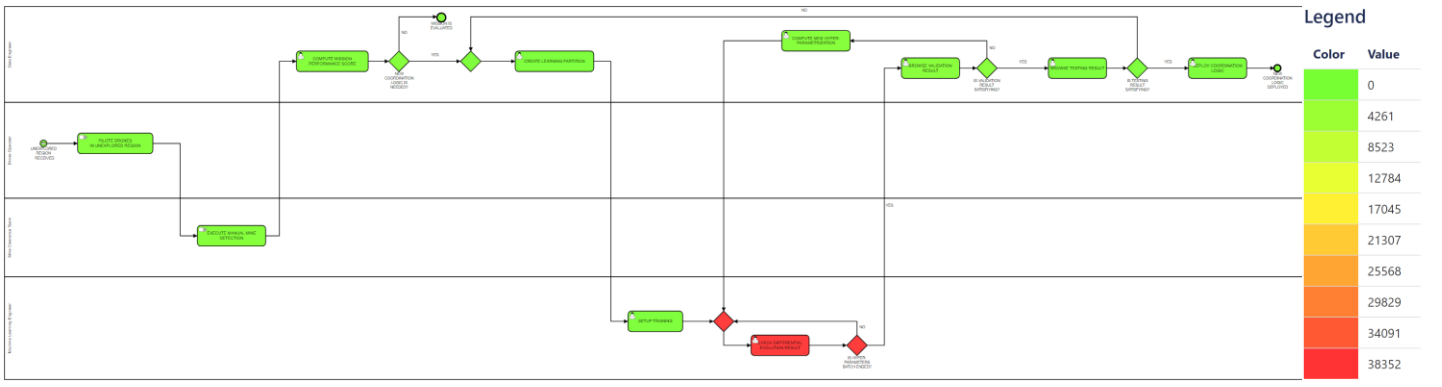


HEATMAP "TO BE"

Cost:



Counts:



PROCESS MINING

REGISTER MINE DETECTION MISSION

NORMATIVE MODEL

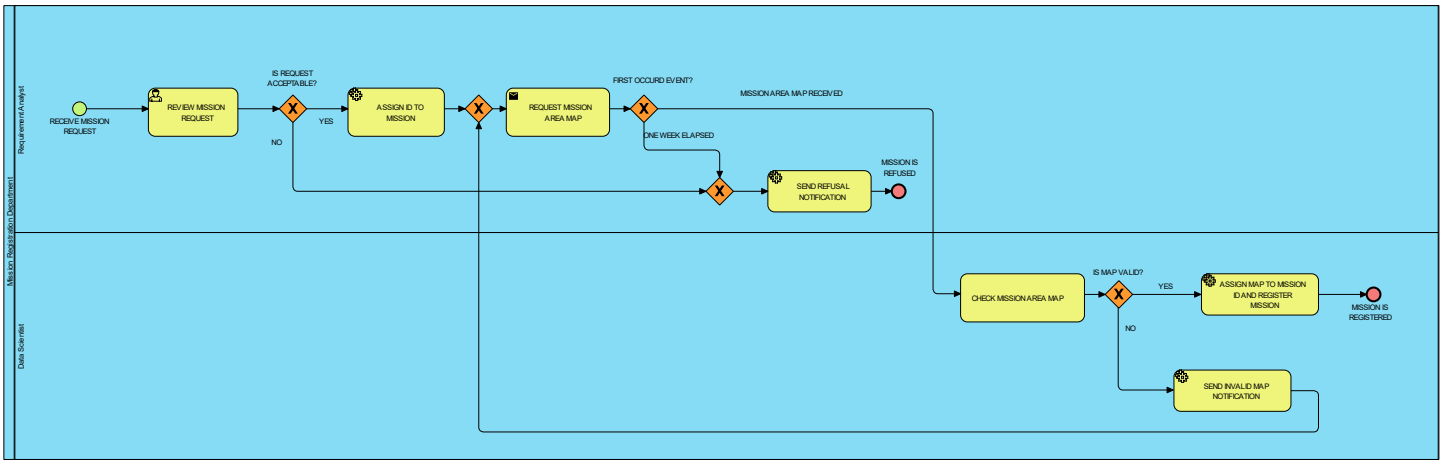


Figure 1: Normative Model

ORIGINAL LOG EXPERIMENTS

In order to obtain the original log, normative model is simulated using BIMP considering 100 tokens and 50% as branching proportion for each gateway. Task cost is set to 1€, and task duration is set to 1s.

BASELINE EXPERIMENT: THE FOUR METRICS FOR ORIGINAL LOG AND NORMATIVE MODEL

In order to have some baseline values to compare with the following results, a first experiment that can be done is evaluating the four metrics considering the original log and the normative model.

Fitness is shown in the following figure.

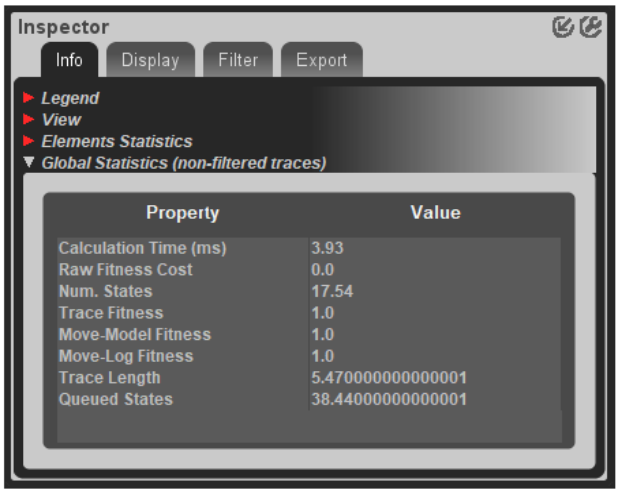


Figure 2: Fitness obtained considering normative model and original log

As expected, fitness is equal to 1. This is because original log is obtained from normative model, so they show perfect fitness to that model. Using the specific plugin, **precision** and **generalization** are computed: their values are 0,96995 and 0,92107 respectively. Precision is not equal to one because other behaviors are permitted: for example, the model can be permit the loop “Check Mission Area Map”, “Send Invalid Map Notification”, “Request Mission Area Map” an undefined number of times. However the number of loops in the original log is limited because 100 tokens are used, and branching proportion is set to 50% for each gateway. Generalization is high, meaning that the model can reproduce additional future behaviors of the process. In other words, it is related to likeliness that the next log will fit the already present variants. For example, if the next log will perform an additional cycle “Check Mission Area Map”, “Send Invalid Map Notification”, “Request Mission Area Map”, this variant is not already present in the original log. **Simplicity**, computed as the sum of the number of gateways, sequence flows and activities, should be equal to 28 (5 gateways, 16 sequence flows, 7 activities). However, in order to have a value comparable to the next ones, three additional events (single start, two ends) are considered in the count leading to a total of 31.

TRANSITION MAP GENERATED FROM ORIGINAL LOG USING DISCO

The transition map extracted from the original log is shown in the following figure. *Disco* has been used to generate it. All possible paths are represented.

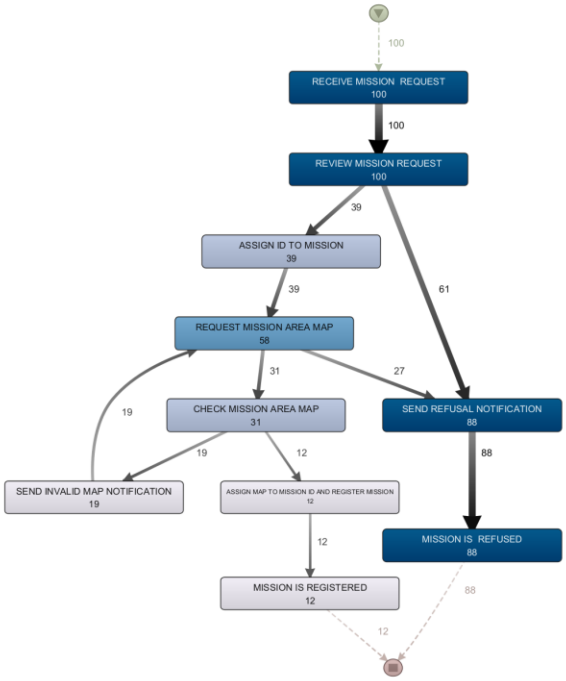


Figure 3: Transition Map generated from original log

BPMN MODEL GENERATED FROM ORIGINAL LOG USING PROM

The BPMN model obtained from the original log is shown in the following figure. It is obtained using *ProM*, and in particular Inductive Miner algorithm (*Infrequent* version with *noise=0* to guarantee maximum fitness).

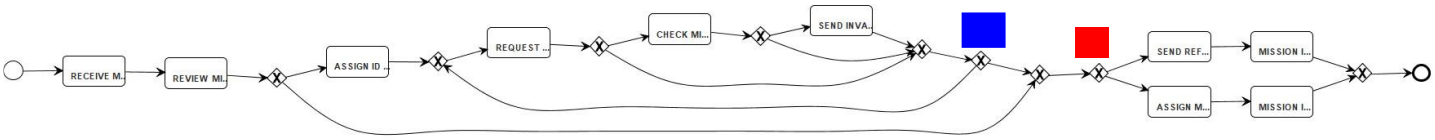


Figure 4: BPMN diagram generated from original log using ProM

Mined model is different from the normative one, as clearly shown by the four consecutive gateways (blue marker), and the final exclusive gateway that leads to “Send Refusal Notification” and “Assign Map to Mission ID and Register Mission” (red marker).

Fitness obtained considering original log and mined model from *ProM* is shown in the following figure. As expected, it is equal to 1 because *noise=0* was set in Inductive Miner algorithm: the original log perfectly fits the mined model.

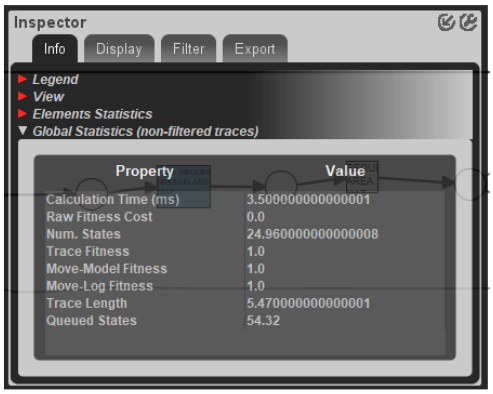


Figure 5: Fitness obtained considering original log and model mined from ProM and original log

Precision and **Generalization** values, computed through the specific plugin, are 0,60765 and 0,92107 respectively. This means that the model is not so precise, allowing a bit much extra behavior. For example, the log “Receive Mission Request”, “Review Mission Request”, “Assign Map to Mission ID and Register Mission” is possible. If compared with precision value obtained from original log and normative model (0,96995), the value obtained in this experiment is far less. On the other hand, generalization is high, meaning that it does not restrict behavior just to the log. It is important to notice that the value obtained is exactly the same obtained from the normative model. It means that they are capable to generalize at the same manner. For example, in this case the variant with multiple cycles “Assign ID to Mission”, “Request Mission Area Map” is possible but not present in the log. **Simplicity**, computed as the sum of the number of gateways, sequence flows and activities, is equal to 44 (9 gateways, 25 sequence flows, 10 activities), so the model is more complex with respect to the normative one (31).

TRANSITION MAP GENERATED FROM ORIGINAL LOG USING APROMORE

The transition map obtained from the original log using *Apromore* is shown in the following. All paths are represented. The map is identical to the one obtained from *Disco*.

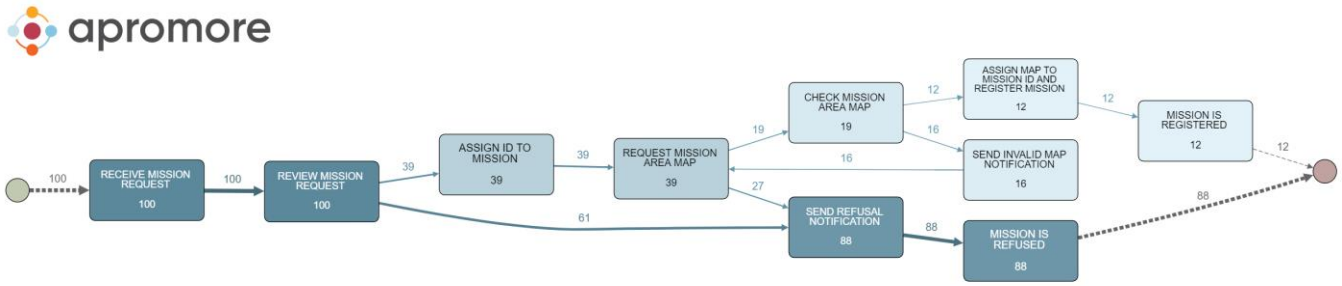


Figure 6: transition map obtained from original log using Apromore

BPMN MODEL GENERATED FROM ORIGINAL LOG USING APROMORE

The BPMN model obtained from original log using *Apromore* is shown in the following figure.

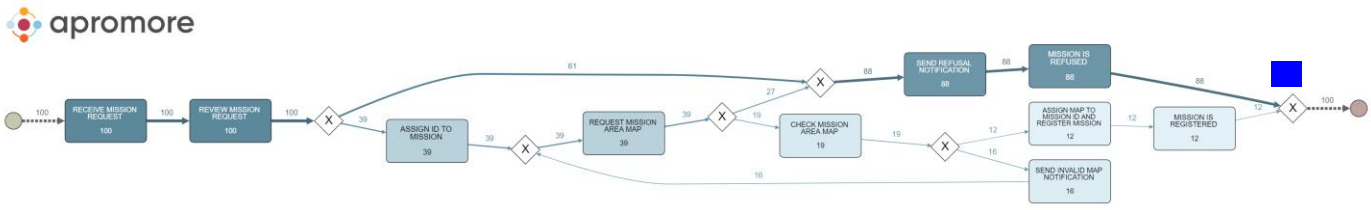


Figure 7: BPMN model obtained from original log using Apromore

BPMN model generated considering original log using *Apromore* is identical to the normative one, the only exception is the management of the two ending events. As a matter of fact, *Apromore* generates a gateway (blue marker) for the merge of the two final events ("Mission is Registered" and "Mission is Refused").

Fitness obtained considering the original log and the model mined through *Apromore* is shown in the following figure.

Figure 8: fitness obtained considering original log and model mined from Apromore

As expected, fitness is equal to 1 (the mined model is identical to the normative model, which has fitness equal to 1 because it was the model from which original logs are extracted). For the same reason, **Precision** and **Generalization** have the same values of the experiment with the normative model (0,96995 and 0,92107 respectively), so the same considerations already made for that model can be replicated for this one. Precision in not equal to one because other behaviors are permitted: for example, the model can be permit the loop "Check Mission Area Map", "Send Invalid Map Notification", "Request Mission Area Map" an undefined number of times. However the number of loops in the original log is limited because 100 tokens are used, and branching proportion is set to 50% for each gateway. Generalization is high, meaning that the model can reproduce additional future behaviors of the process. In other words, it is related to the likeliness that the next log will fit the already present variants. For example, if the next log will perform an additional cycle "Check Mission Area Map", "Send Invalid Map Notification", "Request Mission Area Map", this variant is not already present in the original log. **Simplicity**, computed as the sum of the number of gateways, sequence flows and activities, is equal to 36 (6 gateways, 20 sequence flows, 10 activities). The difference with the normative model (31) is related to the different management of the starting/ending event mentioned above.

SUMMARY TABLE OF EXPERIMENTS WITH ORIGINAL LOG

This table summarizes the results of the experiments performed considering original log.

Model	*Fitness*	*Simplicity (g,sf,a,e)*	*Precision*	*Generalization*
Original Log + Normative Model	**1**	5 + 16 + 7 + 3 = **31**	**0,96995**	**0,92107**
Original Log + ProM Inductive Miner Model	**1**	9 + 25 + 10 = **44**	**0,60765**	**0,92107**
Original Log + Apromore Model	**1**	6+20+10 = **36**	**0,96995**	**0,92107**

MODIFIED LOG EXPERIMENTS

VIOLATIONS LIST

The list of violations considered for this experiment are listed below.

- **"Review Mission Request" skipped (*Case 1*):** under the assumption that customers can send multiple and consecutive requests for similar missions in similar locations, the task "Review Mission Request" can be skipped and being executed only for a subset of the requests. In order to achieve this violation, the activity "Review Mission Request" is deleted from logs 20, 24 and 62.
- **"Assign ID to Mission" postponed (*Case 2*):** under the assumption of having some black-listed customers, namely customers that always send requests but retard in sending the map (leading to request refusal), the activity "Assign ID to Mission" can be postponed to a moment in which we are

sure the mission and the map are already positively evaluated (just before "Assign Map to Mission ID and Register Mission" task. In order to achieve this violation, the activity "Assign ID to Mission" is postponed and put just before the activity "Assign Map to Mission ID and Register Mission" in logs 27, 41, 66.

- **"Request Mission Area Map" skipped in particular variants (*Case 3*):** under the assumption of having some trusted customers, "Request Mission Area Map" can be skipped in the case the flow comes from "Send Invalid Map Notification", taking that the customer already knows the re-send the map without delivering another message. In order to achieve this violation, the activity "Request Mission Area Map" is skipped instead of being executed the second (logs 1, 31, 60) and third (log 1) times.

TRANSITION MAP GENERATED FROM MODIFIED LOGS USING DISCO

The transition map generated from the modified logs is shown in the following figure. *Disco* has been used to generate it. Changes caused by modified logs are clearly shown. Case 1 ("Review Mission Request" skipped) is highlighted by the blue marker. The flow passes directly to the following activity ("Assign ID to Mission"). Case 2 ("Assign ID to Mission" postponed) is highlighted by the red marker. "Assign ID to Mission" is put just before "Assign Map to Mission ID and Register Mission". Finally, case 3 ("Request Mission Area Map" skipped in particular variants) is highlighted by the green marker. Instead of performing another "Request Mission Area Map", the flow goes directly into "Check Mission Area Map" or "Send Refusal Notification".

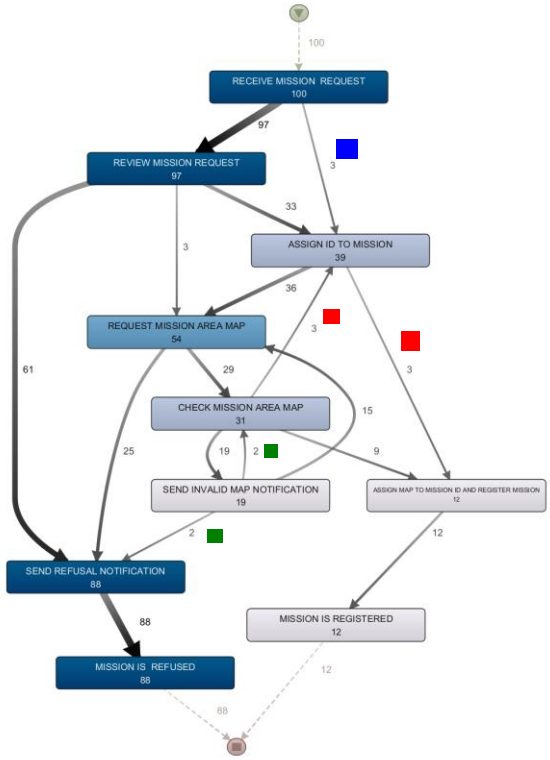


Figure 9: transition map generated from modified logs using Disco

FITNESS OF VIOLATIONS

In order to perform conformance checking, the modified log and the normative model. The model is converted in Petri Net using *ProM*, and then the latter is used to perform conformance checking with the modified log. **Fitness** is equal to 0.9893 as shown in the following figure.

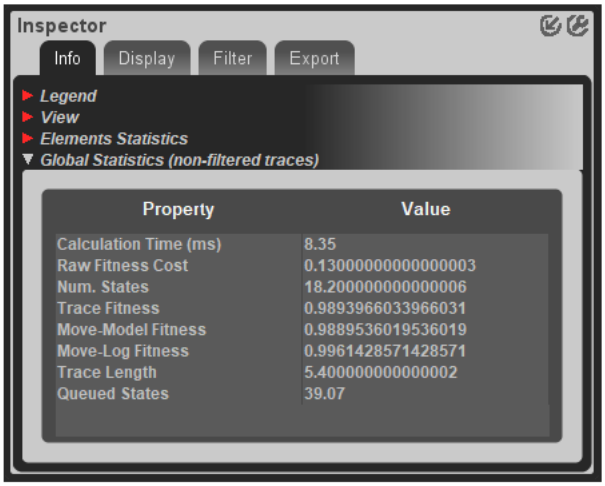


Figure 10: fitness obtained considering modified log and normative model

As expected, fitness is not equal to one because some logs have been modified in order to achieve the violations of the model.

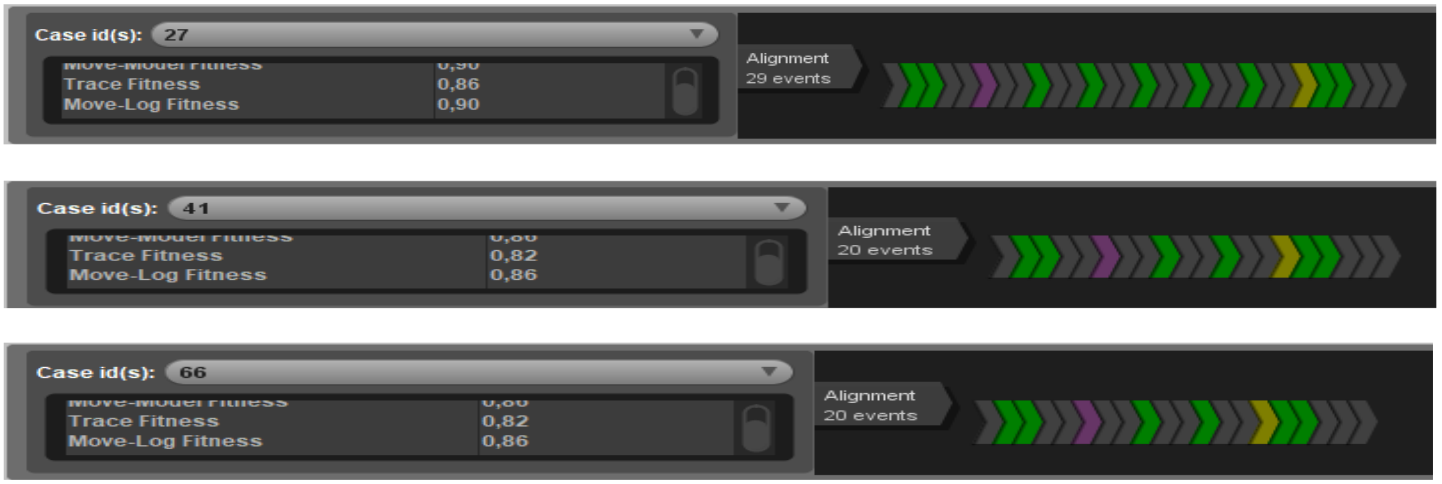
CASE 1: REVIEW MISSION REQUEST SKIPPED

As shown in the following figures, logs 20, 24 and 62 are modified by skipping the activity "Review Mission Request". This is highlighted by the fact that the task is marked with purple color, meaning that it is present in the model but not in the log. Trace fitness is shown on the left side of the images. As expected, they are not equal to 1 because modifies on the log have been applied.

CASE 2: ASSIGN ID TO MISSION POSTPONED

In this case, the task "Assign Mission ID" is postponed in logs 27, 41 and 66. This is highlighted by the fact that the task is skipped in the log (purple marker) and then is performed on the log only (yellow marker). Trace

fitness is shown on the left side of the images. As expected, they are not equal to 1 because modifies on the log have been applied.



CASE 3: REQUEST MISSION AREA MAP SKIPPED IN PARTICULAR VARIANTS

In case 3, "Request Mission Area Map" can be skipped under the assumptions mentioned above. In log 1 the task is skipped twice (because there is a cycle) as highlighted by the purple markers, while in log 31 and 60 it is skipped only once. Trace fitness is shown on the left side of the images. As expected, they are not equal to 1 because modifies on the log have been applied.

BPMN MODEL GENERATED FROM MODIFIED LOG USING PROM

The BPMN model represented in figure is obtained by applying Inductive Miner (Infrequent version with *noise=0*) on the modified log. As shown, there are some differences with the BPMN model obtained from the original log. In particular, the task "Review Mission Request" can be skipped (it is preceded and followed by an exclusive gateway) according to *Case 1* (blue marker). The fact that the task "Assign ID to mission" can be postponed is shown by the exclusive gateway just before the task (red marker). Finally, the fact that the task "Request Mission Area Map" can be skipped in certain situations (*Case 3*) is represented by the cycle on the tasks "Check Mission Area Map" and "Send Invalid Map Notification", which does not include the task "Request Mission Area Map" itself (green marker).

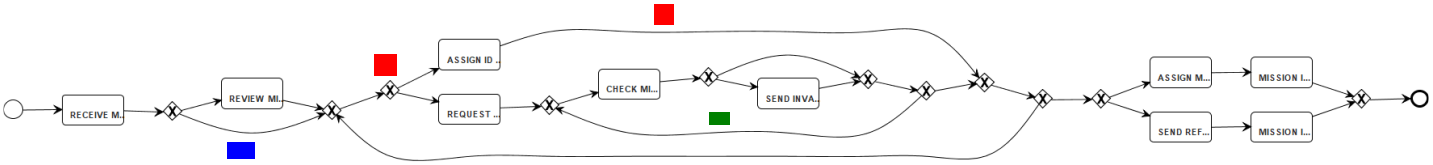


Figure 11: BPMN model generated from modified log using ProM

Fitness value is shown in the following figure. The value is not perfect (0.90): the model for example does not guarantee the behavior in which there is no response after "Request Mission Area Map", and the flow should go into "Send Refusal Notification" activity. So in order to improve the model that sequence flow should be added.

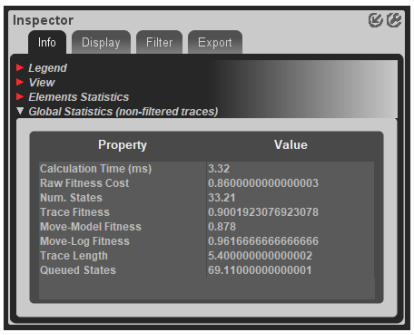
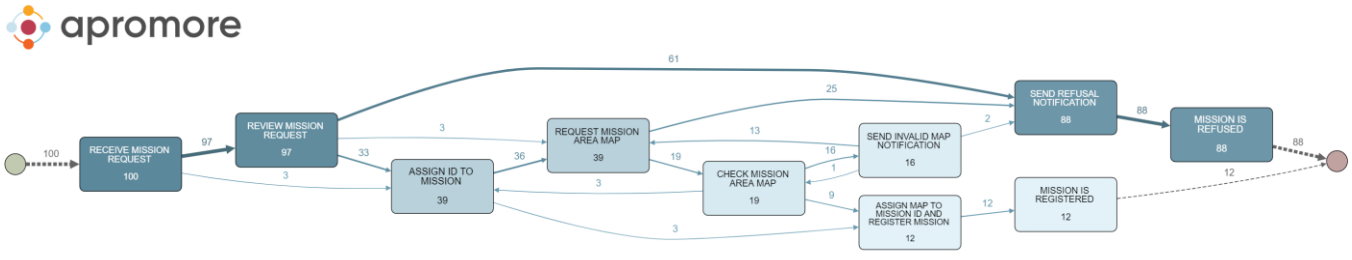


Figure 12: fitness obtained considering modified log and BPMN model from ProM

Precision and **Generalization**, computed with the specific plugin, are 0,47143 and 0,94632 respectively. This means that the model permits a bit much of extra behavior and has a satisfying capability to generalize. For example, the variant "Receive Mission Request", "Assign ID to Mission", "Send Refusal Notification", "Mission is Refused" is possible accordingly to the model (it should not in real cases), but it is not present in the log. Generalization is high because the model does not restrict the behavior just to the log: for example, an undefined number of cycles "Request Mission Area Map", "Check Mission Area Map" and "Send Invalid Notification" is possible. **Simplicity**, computed as the sum of the number of gateways, sequence flows and activities, is equal to 49 (11 gateways, 28 sequence flows, 10 activities).

TRANSITION MAP GENERATED FROM MODIFIED LOG USING APROMORE

Transition map generated from modified log using *Apromore* is shown in the following figure.



As shown, transition map is identical to the one generated from *Disco*.

BPMN MODEL GENERATED FROM MODIFIED LOG USING APROMORE

BPMN model generated from modified log using *Apromore* is shown in the following figure. With respect to the one generated from *ProM*, there is the sequence flow that solves the problem mentioned above (green marker).

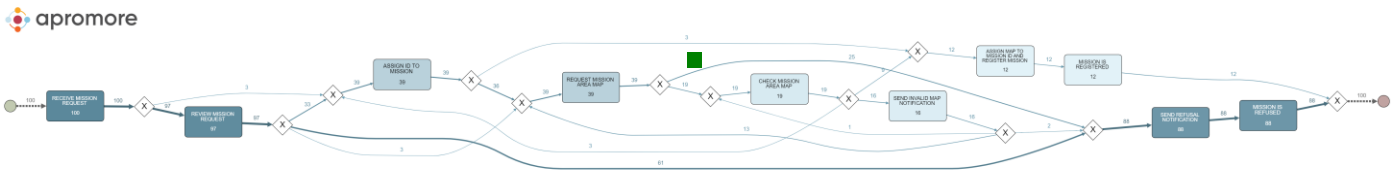


Figure 13: BPMN model obtained from modified log using Apromore

Fitness is shown in the following figure. As expected, it is equal to 1.

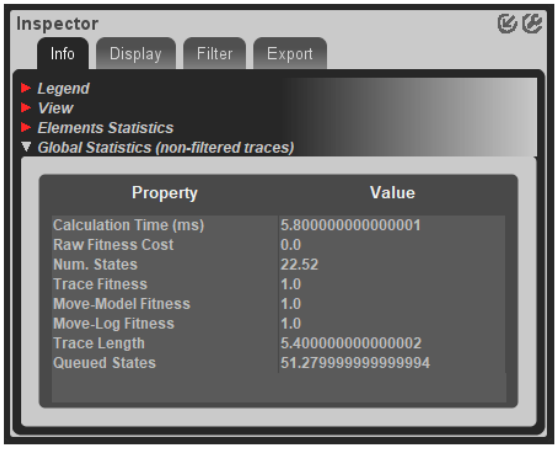


Figure 14: Fitness obtained using modified log and BPMN model generated from Apromore

Precision and **Generalization** are 0,62083 and 0,94796 respectively. Precision is higher with respect to the one obtained from modified log: this model permits less extra behavior with respect to the previous one. For example, the variant described for the model generated from *ProM* ("Receive Mission Request", "Assign ID to Mission", "Send Refusal Notification", "Mission is Refused") it is not possible now. Generalization is still high, meaning that the model can generalize well (the same example of multiple "Request Mission Area Map", "Check Mission Area Map" and "Send Invalid Notification" loops is still valid). **Simplicity**, computed as the sum of the number of gateways, sequence flows and activities, is equal to 54 (12 gateways, 32 sequence flows, 10 activities), higher than the one obtained with *ProM* model.

SUMMARY TABLE OF EXPERIMENTS WITH MODIFIED LOG

This table summarizes the results of the experiments performed considering modified log.

Model	*Fitness*	*Simplicity (g,sf,a)*	*Precision*	*Generalization*
Modified Log + ProM Inductive Miner Model	**0,9001**	11 + 28 + 10 = **49**	**0,47143**	**0,94632**
Modified Log + Apromore Model	**1**	12 + 32 + 10 = **54**	**0,62083**	**0,94796**

FINAL CONSIDERATIONS

This table summarizes all the results obtained so far.

Model	*Fitness*	*Simplicity*	*Precision*	*Generalization*
Original Log + Normative Model	**1**	**31**	**0,96995**	**0,92107**
Original Log + ProM Inductive Miner Model	**1**	**44**	**0,60765**	**0,92107**
Original Log + Apromore Model	**1**	**36**	**0,96995**	**0,92107**
Modified Log + ProM Inductive Miner Model	**0,9001**	**49**	**0,47143**	**0,94632**
Modified Log + Apromore Model	**1**	**54**	**0,62083**	**0,94796**

Considering the **original log**, the best result is obtained from the BPMN model generated from *Apromore*. As a matter of fact, it presents perfect fitness, and generalization and precision values are the same as the normative model ones. Simplicity is higher with respect to the normative model because of a different management of starting/ending activities. It is however lower than the one obtained considering the model generated from *ProM.* The latter presents also a lower precision value, 0.60765 vs. 0.96995.

Furthermore, the best model obtained considering the experiments with **modified log** is again the one generated from *Apromore*. In contrast to the results achieved by the model generated from *ProM*, it

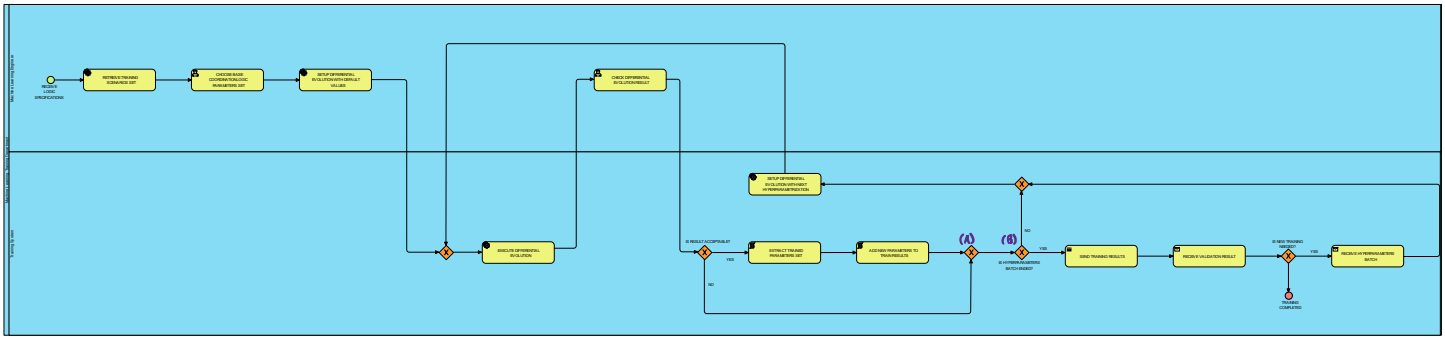
guarantees perfect fitness, and the higher complexity is balanced by better results in terms of precision and generalization (0.62083 vs. 0.47143 and 0.94796 vs. 0.94632).

Talking about **customer perspective**, the model obtained considering the violations is better in terms of responsiveness and speed, because the fact that some tasks are skipped under certain assumptions (for example “Review Mission Request” when continuous requests are coming from the same customer) simplifies the workflow leading to a speedup of the process. In addition, a similar achievement is obtained by skipping the activity “Request Mission Area Map” when “Send Invalid Map Notification” has previously been sent. Customer has not to wait another message and can directly send the map after the notification that the previous one was invalid. This speeds up the whole process.

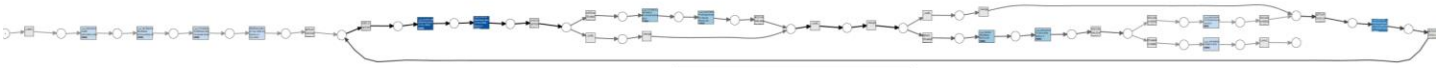
From an **internal perspective**, the process obtained from the modified log provides some useful optimizations. For example, the creation of the ID for a mission can be postponed and be performed only after it is certain that the mission is accepted. So the process will be less expensive in all the cases in which mission is refused after the creation of the ID, namely when the customer does not send the map in one week. A similar argument can be done for the deletion of the activity “Review Mission Request”: the saving is both in terms of time and cost.

TRAIN COORDINATION LOGIC

NORMATIVE MODEL USED FOR SIMULATION

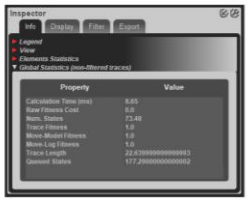


QUALITY ESTIMATION FOR REFERENCE



Precision : 0,86008

Generalization : 0,99982



These measurements refer to the ***normative model*** used to run the simulation and the ***original log***.

Model	*Fitness*	*Simplicity (g,sf,a)*	*Precision*	*Generalization*
Normative Model	**1**	6 + 23 + 13 = **42**	**0.86008**	**0.99982**

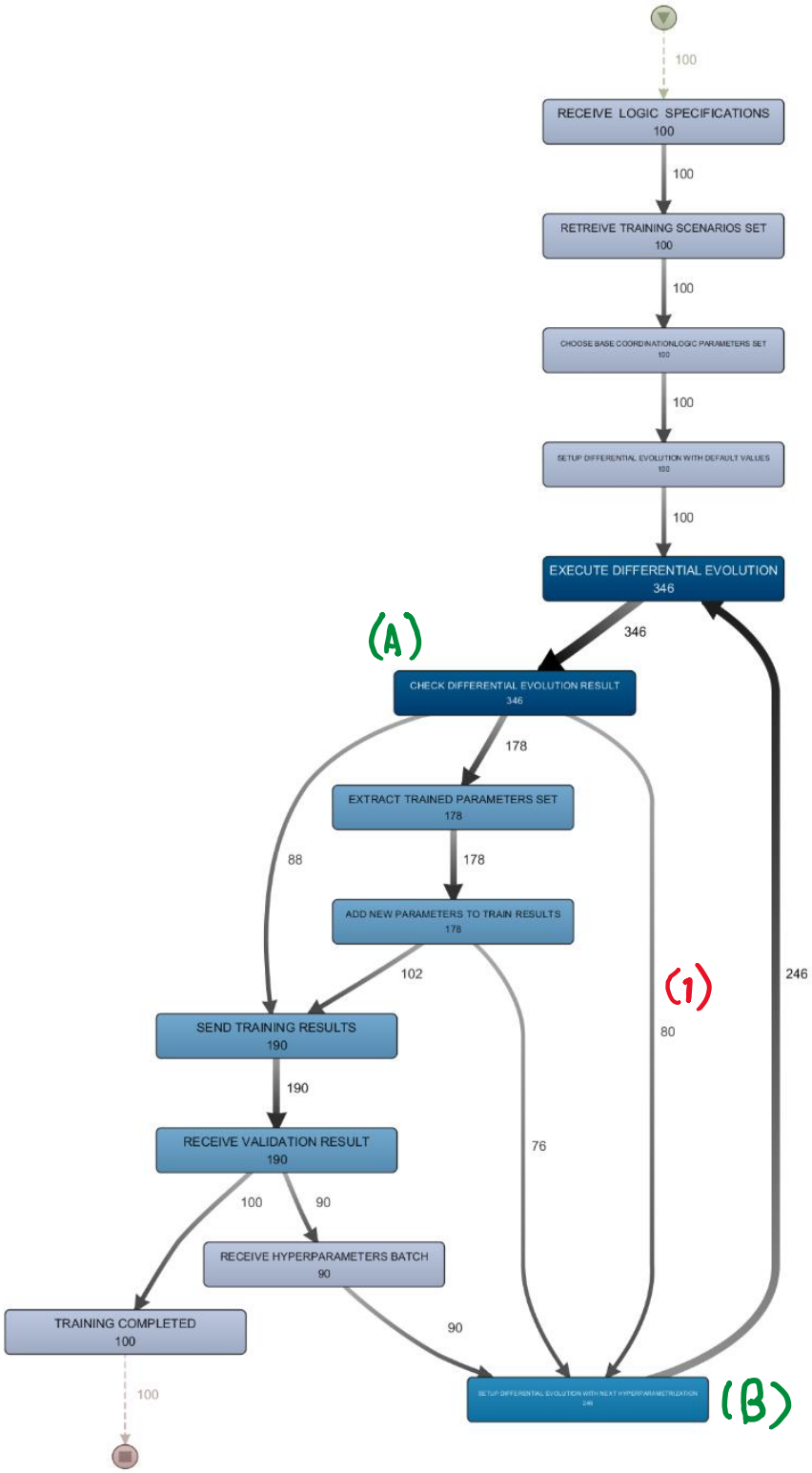
Note that **g** = *gateways*, **sf** = *sequence flows* and **a** = *activities*.

Note: Only for the normative model *START* end *END* events are counted as an activity (because are replaced by activities in mined models).

Note: At the number of control flows in this model have been also added 2 arch to compensate the fictitious ones present in the mined models

ORIGINAL LOG

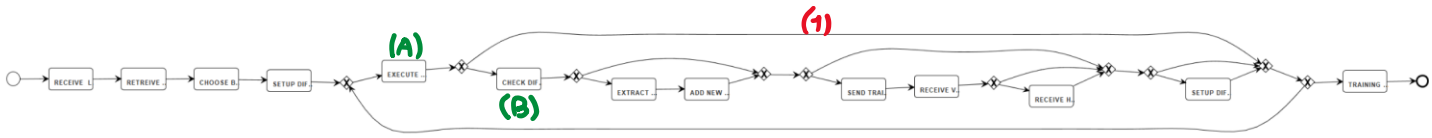
TRANSITION MAP



The ***transition map*** created with *Disco* using the ***original log*** almost perfectly mirrors the structure of the original model, using the log created by it. Note arc **(1)** from activity **(A)** to activity **(B)** which must be present in a transition map since gateways are not modeled.

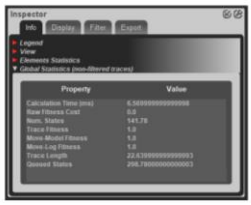
MINED MODEL USING INDUCTIVE MINER

In order to crate this BPMN model the ***Inductive Miner-Infrequent*** tool of *ProM* was used over the **original log**, a *noise threshold* set to *0* ensures to maximize the *fitness*.



Precision : 0,61766

Generalization : 0,99982



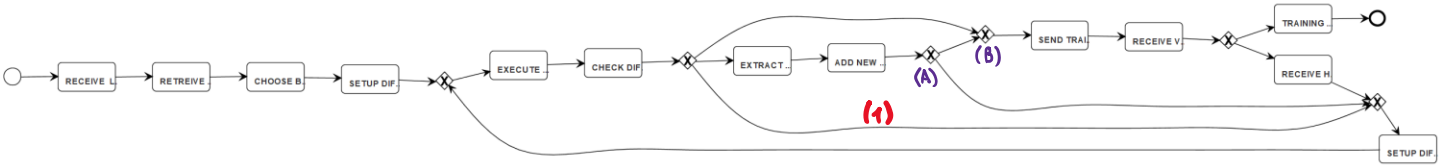
Model	*Fitness*	*Simplicity (g,sf,a)*	*Precision*	*Generalization*
Normative Model (Reference)	**1**	6 + 23 + 13 = **42**	**0.86008**	**0.99982**
Original Log + Inductive Miner	**1**	10 + 28 + 13 = **51**	**0.61766**	**0.99982**

Explanation:

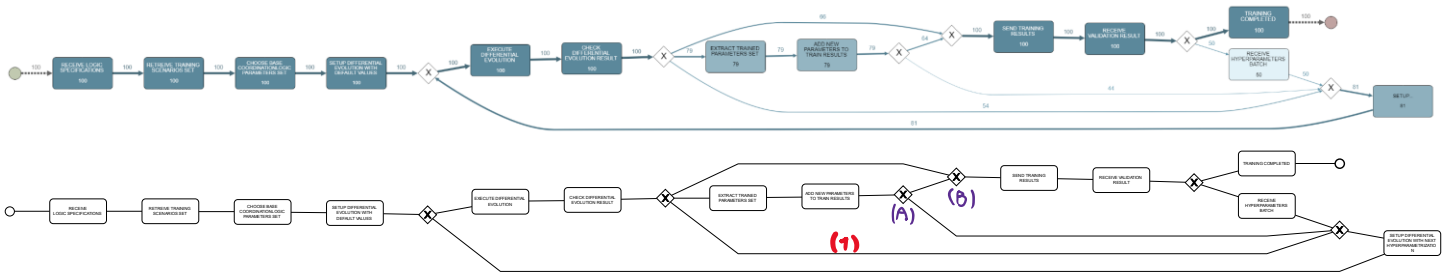
- **Fitness:** fitness is high because a zero threshold was used for noise.
- **Simplicity:** the simplicity is worse than the *basic model* as more *gateways* and more *control flows* are mapped in the *mined* one.
- **Precision:** precision is lower because the model allows behaviors not mapped in the normative one, for example using the arch **(1)** from task **(A)** it is possible to skip task **(B)** and subsequent cycling on itself but this would be prohibited in the original model.
- **Generalization:** the model allows more behaviors than the normative one so it is normal to have an almost maximum generalization.

MINED MODELS USING HEURISTICS PROM6 MINER AND APROMORE

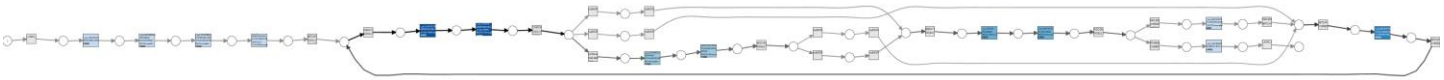
In order to crate this BPMN model the ***Heuristics ProM6 Miner*** tool of *ProM* was used with the *default parameters* over the ***original log***. The same results were obtained using ***Apromore***.



Above, the model obtained using ***ProM***.



Above, the model obtained using ***Apromore***.



Precision : 0,86008

Generalization : 0,99982

Model	*Fitness*	*Simplicity (g,sf,a)*	*Precision*	*Generalization*
Normative Model (Reference)	**1**	6 + 23 + 13 = **42**	**0.86008**	**0.99982**
Original Log + Heuristics Miner/Apromore	**1**	6 + 24 + 13 = **43**	**0.86008**	**0.99982**

Explanation:

- **Fitness:** fitness is high because the model permits the same behaviors permitted in the normative model, the two models are almost the same (see simplicity for the differences).
- **Simplicity:** the simplicity is worse than the *basic model* as one more *control flow* is mapped in the *mined* ones. In the *mined models*, if the gateways **(A)** and **(B)** were swapped, would be possible to suppress the arch **(1)**.
- **Precision:** precision is the same as the basic model as the mined ones permit the same behaviors.
- **Generalization:** the model allows more behaviors than the normative one, so it is normal to have an almost maximum generalization.

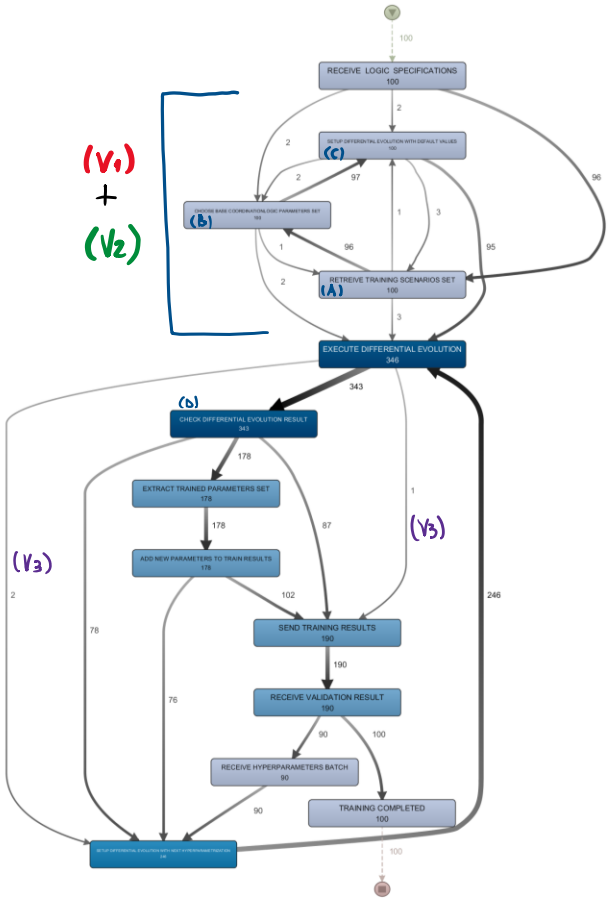
MODIFIED LOG

VIOLATIONS TO THE MODEL

The cases modified to achieve violations of the original model are listed below:

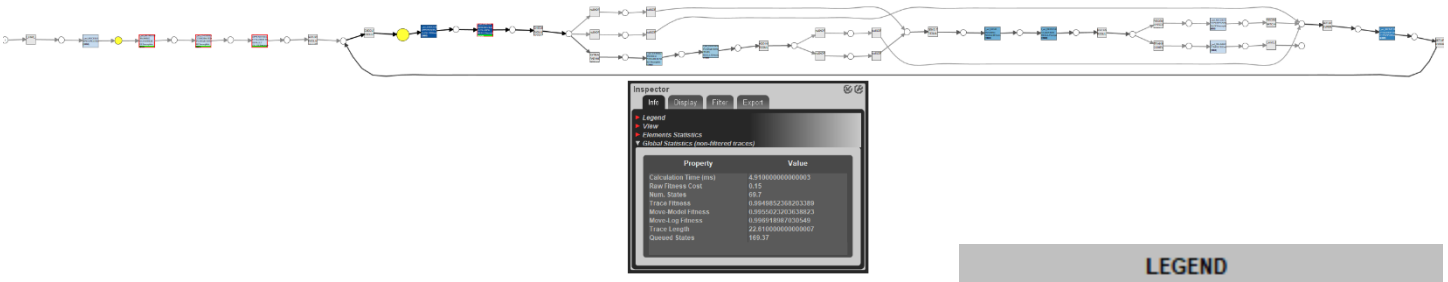
- **"Retrieve Training Scenarios Set" postponed (*V1*):** it is possible to postpone this task as theoretically it could be performed in parallel with the next two, a sequential order was initially used because it would be more appropriate to execute the tasks in order, especially if they are performed by the same actor who can obtain useful information for subsequent tasks by executing the previous ones first.
- **"Setup Differential Evolution with Default Values" anticipated (*V2*):** Following the same reasoning described above, this task can also be anticipated, it was taken into consideration with the previous one in order to create, by modifying 3 log records for both tasks (in total 6), all the possible permutations of the three consecutive tasks ($3! = 6$ permutations), in order to highlight their complete parallelism within the logs.
- **"Check Differential Evolution Result" not executed (*V3*):** For extremely high values of the *fitness function,* the training result can be directly discarded without being interpreted by the *M.L. Engineer* in order to avoid employing him even in the control of trivial cases. To keep the model consistent, 3 records were changed, all of which related to cases where the fitness was considered not satisfactory in the original log.

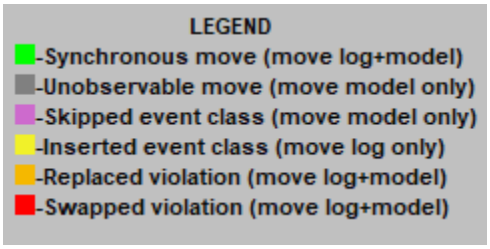
TRANSITION MAP



Consider the permutations of the three initial tasks as **A→B→C** in the non-violated case *("Retrieve Training Scenarios Set" → "Choose Base Coordination Logic Parameters Set" → "Setup Differential Evolution with Default Values")*. In the ***transition map*** created using Disco over the ***modified log*** is clearly visible the new parallelism between the tasks **(A),(B),(C)** introduced using violations **(V1)** and **(V2)**. Can be also noticed that task **(D)** can be avoided using **(V3)**.

FITNESS OF VIOLATIONS





Fitness = 0.99498

The ***modified log*** was replayed using a *Petri Net* extracted from the ***normative model*** to test the ***fitness*** of the violations. it is possible to notice how the *fitness* is no longer perfect, due to them. However, violations remain a small percentage decreasing *fitness* by less than *1%.*

Consider the permutations of the three initial tasks as **A→B→C** in the non-violated case *("Retrieve Training Scenarios Set" → "Choose Base Coordination Logic Parameters Set" → "Setup Differential Evolution with Default Values")*.

Violation 1

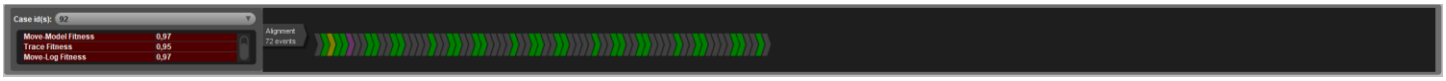
Fitness = 0.97

This is the permutation **B→C→A** where *"Retrieve Training Scenarios Set"* was postponed after *"Setup Differential Evolution with Default Values"*.

Fitness = 0.94

This is the permutation **B→A→C** where *"Retrieve Training Scenarios Set"* was postponed after *"Choose Base Coordination Logic Parameters Set"*.

Violation 2



Fitness = 0.95

This is the permutation **C→A→B** where *"Setup Differential Evolution with Default Values"* was anticipated before *"Retrieve Training Scenarios Set"*.

Fitness = 0.90

This is the permutation **A→C→B** where *"Setup Differential Evolution with Default Values"* was anticipated before *"Choose Base Coordination Logic Parameters Set"*. (Was recognized as *"Choose Base Coordination Logic Parameters Set"* was postponed).

Violation 1 + 2

Fitness = 0.85

Finally, this is the permutation **C→B→A** were *"Retrieve Training Scenarios Set"* and *Setup Differential Evolution with Default Values"* were swapped (changing *2* records in the same case). (Was recognized as *"Retrieve Training Scenarios Set"* and *"Choose Base Coordination Logic Parameters Set"* were postponed).

Violation 3

Fitness = 0.98

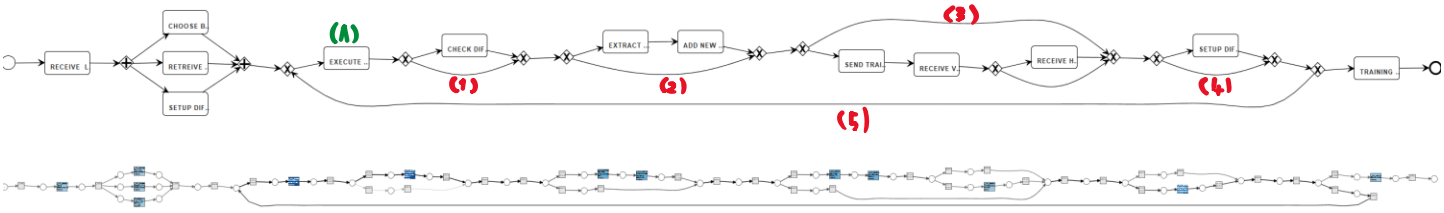
Fitness = 0.97

Fitness = 0.95

These are the cases where *"Check Differential Evolution Result"* task was skipped.

MINED MODEL USING INDUCTIVE MINER

In order to crate this BPMN model the ***Inductive Miner-Infrequent*** tool of *ProM* was used over the ***modified log***, a *noise threshold* set to *0* ensures to maximize the *fitness*.



Precision : 0,46946

Generalization : 0,99578

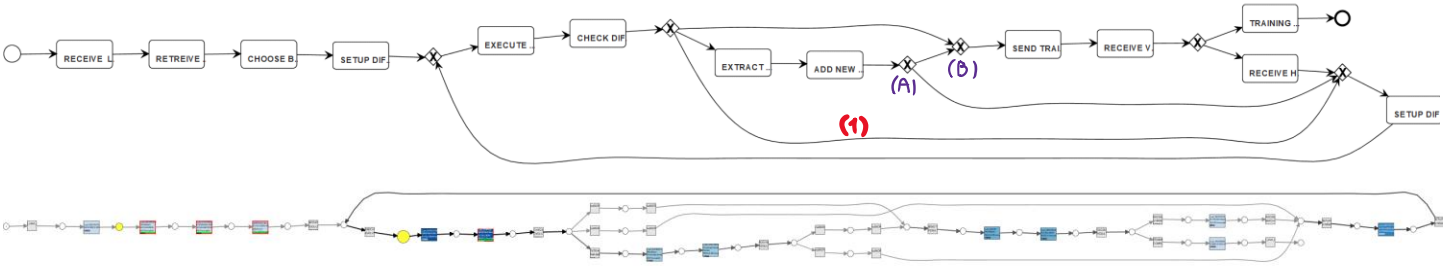
Model	Fitness	Simplicity (g,sf,a)	Precision	Generalization
Normative Model (Reference)	**1**	6 + 23 + 13 = **42**	**0.86008**	**0.99982**
Modified Log + Inductive Miner	**1**	13 + 35 + 13 = **61**	**0.46964**	**0.99578**

Explanation:

- **Fitness:** fitness is high because a zero threshold was used for noise.
- **Simplicity:** the simplicity is worse than the *normative model* as more *gateways* and more *control flows* are mapped in the *mined* one.
- **Precision:** precision is lower because the model allows behaviors not mapped in the normative one, for example using the arch **(1)**,**(2)**,**(3)**,**(4)**,**(5)** task **(A)** can cycle on itself but this would be prohibited in the original model.
- **Generalization:** the model allows more behaviors than the normative one, so it is normal to have a high generalization.

MINED MODEL USING HEURISTICS MINER PROM6

In order to crate this BPMN model the ***Heuristics ProM6 Miner*** tool of *ProM* was used over the ***modified log*** with the default parameters.



Precision : 0,86246

Generalization : 0,99983

NOTE: the model is exactly the same as the ***"Original Log + Heuristics Miner/Apromore"***, this is due to the fact that the violations are only less the 1% respect to all records.

Model	Fitness	Simplicity (g,sf,a)	Precision	Generalization
Normative Model (Reference)	**1**	6 + 23 + 13 = **42**	**0.86008**	**0.99982**
Modified Log + Heuristics Miner	**0.99499**	6 + 24 + 13 = **43**	**0.86246**	**0.99983**

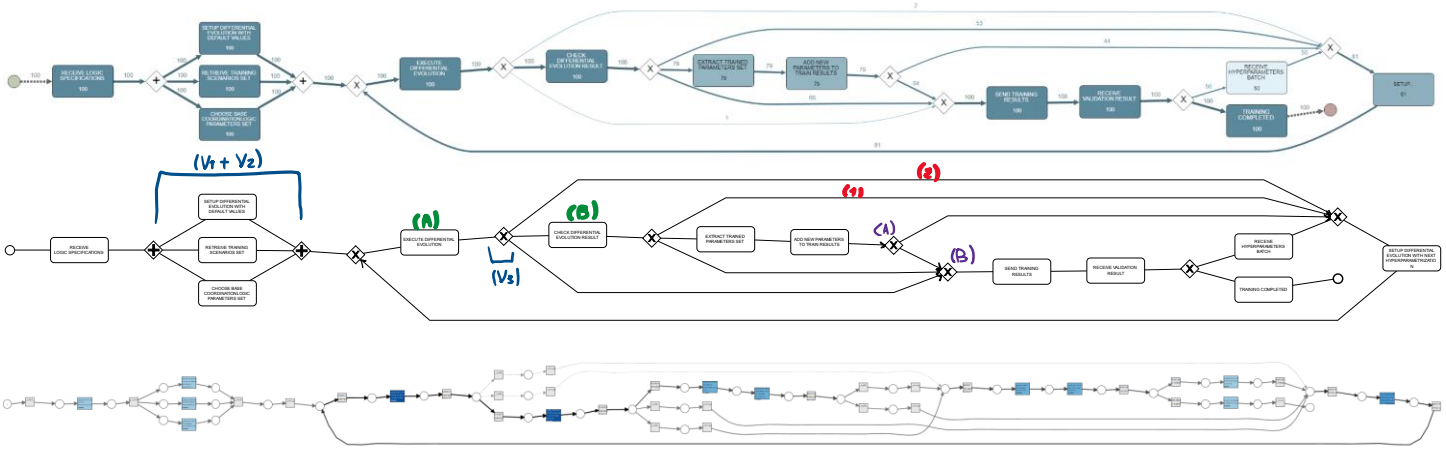
Explanation:

- **Fitness:** fitness is not 1 because the model ignores the changes introduced in the *modified log*, so those behaviors are not mapped in this model.

- **Simplicity:** the simplicity is worse than the *normative model* as one more *control flow* is mapped in the *mined* one. In the *mined model*, if the gateways **(A)** and **(B)** were swapped, would be possible to suppress the arch **(1)**.
- **Precision:** precision is almost the same as the normative model as the mined ones permit the same behaviors and the *modified log* is only less the 1% different from the original one.
- **Generalization:** the model allows more behaviors than the normative one, so it is normal to have an almost maximum generalization.

MINED MODEL USING APROMORE

This model was created using ***Apromore*** and the ***modified log***



Precision : 0,73244

Generalization : 0,99578

NOTE: In this model the effects of the changes introduced in the log are evident. The violations 1 and 2 permits to execute in parallel the three tasks in **(V1 + V2)**. The violation 3 permits to jump the execution of activity **(B)** after activity **(A)** using the gateway in **(V3)**.

Model	*Fitness*	*Simplicity (g,sf,a)*	*Precision*	*Generalization*
Normative Model (Reference)	**1**	6 + 23 + 13 = **42**	**0.86008**	**0.99982**
Modified Log + Apromore	**1**	9 + 31 + 13 = **53**	**0.73244**	**0.99578**

Explanation:

- **Fitness:** fitness is 1 because all the behaviors of the modified log are correctly mapped.
- **Simplicity:** the simplicity is worse than the *normative model* as the modifications **(V1 + V2)** and **(v3)** require more control flows and gateways. Nevertheless, two control flows can be omitted **(1)** and **(2)** if the gateways **(A)** and **(B)** were swapped.
- **Precision:** precision is lower than the *normative model* maybe because the new behaviors mapped (that were intentionally put in the modified log) regards only a small minority of cases (less the 1% of the total records)

- **Generalization:** the model allows more behaviors than the normative one, so it is normal to have a high generalization.

FINAL CONSIDERATIONS

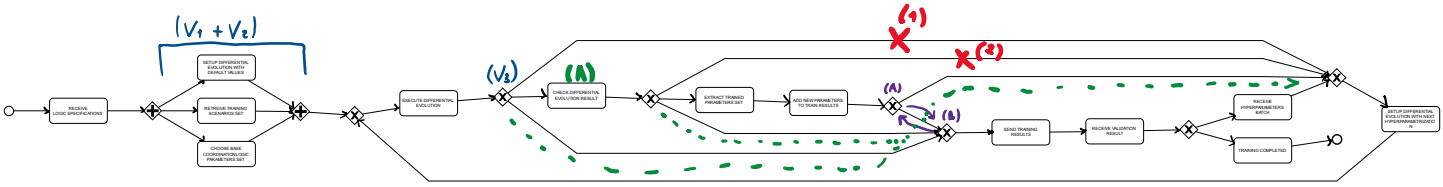
Below is a table summarizing the various mined models and their quality:

Model	*Fitness*	*Simplicity (g,sf,a)*	*Precision*	*Generalization*	*Overall Quality*
Normative Model	**1**	6 + 23 + 13 = **42**	**0.86008**	**0.99982**	**Reference**
Original Log + Inductive Miner	**1**	10 + 28 + 13 = **51**	**0.61766**	**0.99982**	**Poor**
Original Log + Heuristics/Apromore Miner	**1**	6 + 24 + 13 = **43**	**0.86008**	**0.99982**	**Optimal**
Modified Log + Inductive Miner	**1**	13 + 35 + 13 = **61**	**0.46964**	**0.99578**	**Poor**
Modified Log + Heuristics Miner	**0.99499**	6 + 24 + 13 = **43**	**0.86246**	**0.99983**	**Good**
Modified Log + Apromore Miner	**1**	9 + 31 + 13 = **53**	**0.73244**	**0.99578**	**Optimal**

NOTE: To evaluate the quality of the models it is necessary to highlight how real logs can present incomplete data and therefore a greater *generalization* could be appreciated to also include behaviors not registered in the logs but actually done in the real process. The logs used instead in this project perfectly recreate the behavior that should be predicted by the model, so in this case any kind of *generalization* is superfluous and only makes the model lose *precision*. Indeed, the ***fitness*** and ***precision*** are the most important aspect in this situation.

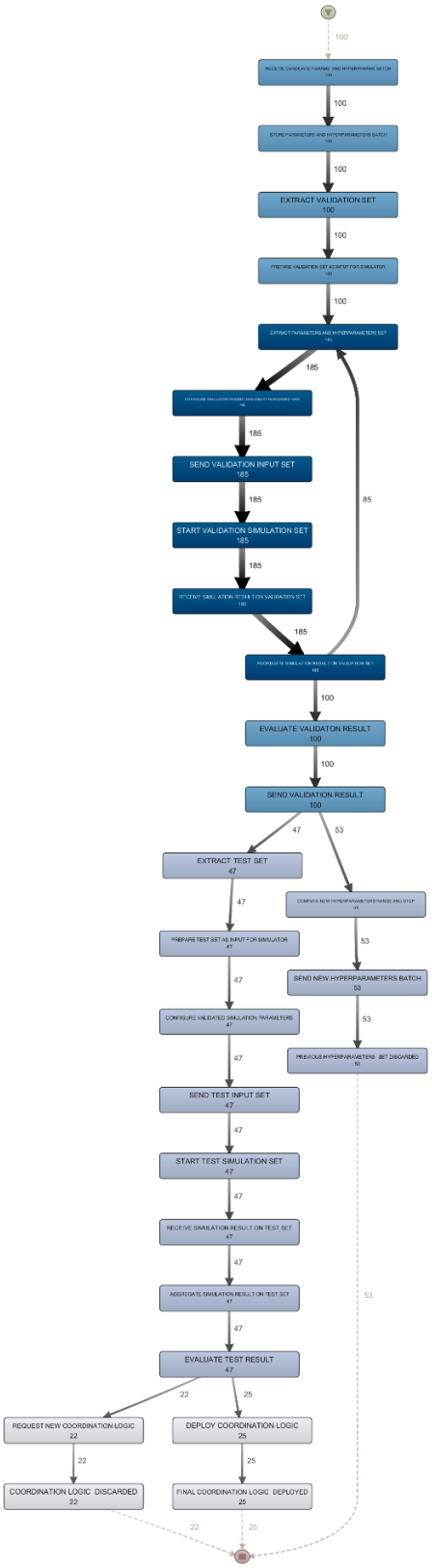
The model mined by ***Apromore*** on the ***modified log*** is almost the ideal model. Unfortunately, it also falls into the finesse of not swapping the two gateways **(A, B)** to omit two useless control flows **(1, 2)**, but this is only a detail and the model is excellent, probably better than the original one since it allows the parallel execution of the three initial tasks **(V1 + V2)** and the optional execution of the task **(A)** adding the gateway **(V3)**, exactly the changes introduced in the log.

*The conclusion is that the best model for a client would be achieved modifying the last mined model using **Apromore** from the **modified log**, by adding the changes described above.*



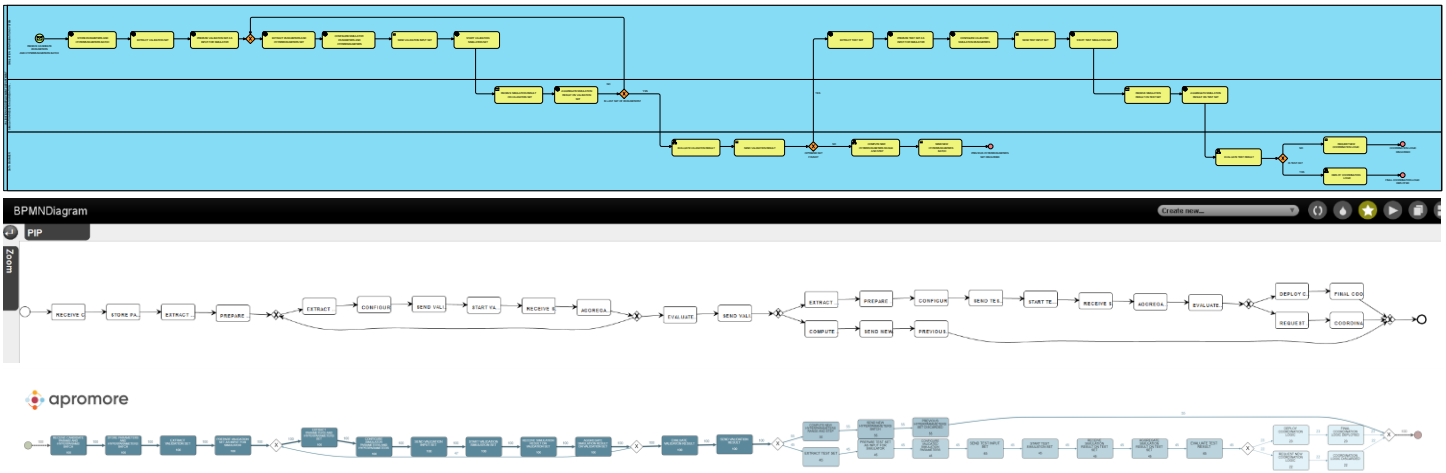
EVALUATE AND DEPLOY COORDINATION LOGIC

TRANSITION MAP OF THE SIMULATION LOG

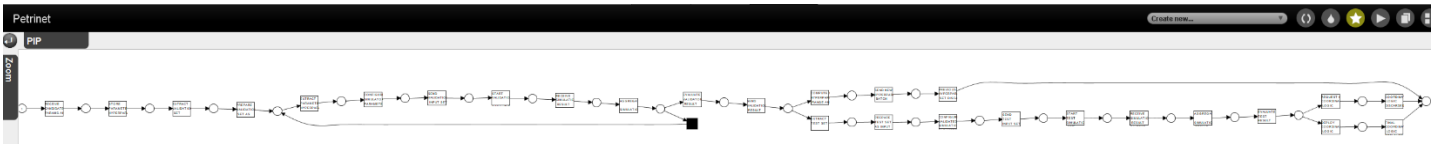


BUSINESS PROCESS DISCOVERY USING ORIGINAL LOG

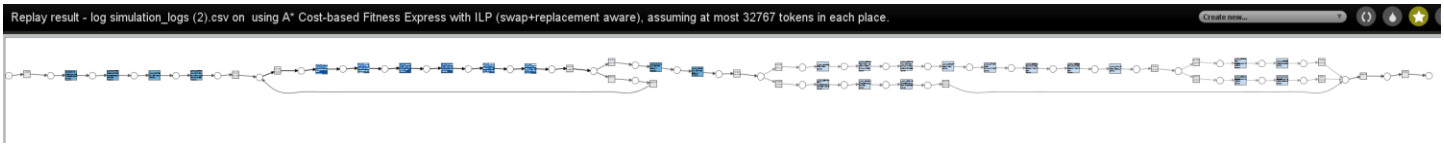
A simulation was executed assigning a default 1 euro cost and 1 sec duration to each task, 10 resources per lane, 50% to each gateway, and 100 input tokens. This simulation log was imported in Disco and exported as csv. The original log was converted to XES and a BPMN model was mined from the BPMN Miner tool in ProM using Inductive Miner-Infrequent with noise threshold set to 0 to ensure maximized fitness. The log was also imported in Apromore to discover a BPMN diagram. In both cases the model mined is the same as the original one (The only differences are the ending activities and the final merge gateway to have only one end event as ProM seems to have trouble with multiple end events).



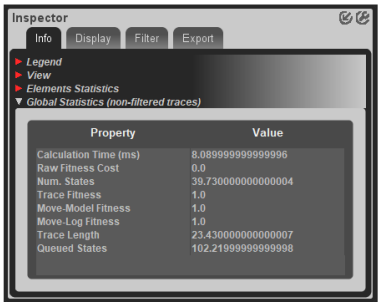
The original BPMN was then converted to a Petri net using the "Convert BPMN Diagram to Petri net (control-flow)" tool.



The Petri net and the log were used on the "Replay Log on Petri Net for Conformance Analysis" tool to measure fitness that we already know should be 1 from the mining algorithm settings.



The replay result, the log and the petri net were then used as input for the measurements obtained by the "Measure Precision/Generalization" tool. The four measures of the model are:



Fitness: 1

Simplicity: 27 activities+36 sequence flows+5 gateways=68

Precision: 0,97695

Generalization: 0,99958

CREATION OF THE MODIFIED LOG

A modified log was obtained starting from the original log, adding 3 different alterations to 3 cases each removing, anticipating or postponing tasks.

Alteration 1: Removal of COMPUTE NEW HYPERPARAMETERS RANGE AND STEP task.

The new batch of parameters could be automatically calculated by the system using a fixed step and taking in consideration the best scoring (lowest average execution time) cell of the grid. *[Case 0,1,64]*

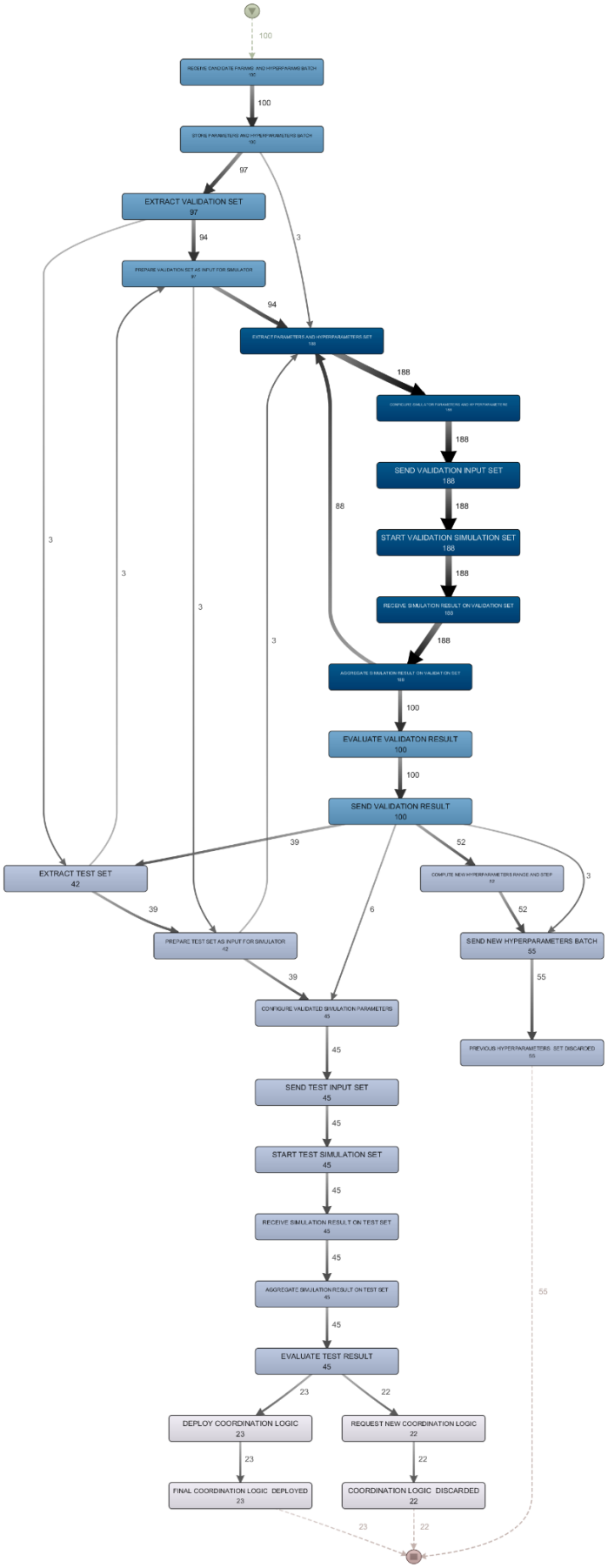
Alteration 2: Anticipation of EXTRACT TEST SET and PREPARE TEST SET AS INPUT FOR SIMULATOR tasks to be executed respectively after EXTRACT VALIDATION SET and PREPARE VALIDATION SET AS INPUT FOR SIMULATOR.

Assuming we have freshly updated starting Hyperparameters for the training process, we could have a high probability to proceed to test at the first execution of this process. We could prepare our simulation data for test as we prepare the data for validation to use the service software only once. *[Case 3,16,30]*

Alteration 3: Removal of EXTRACT TEST SET, PREPARE TEST SET AS INPUT FOR SIMULATOR, EXTRACT VALIDATION SET and PREPARE VALIDATION SET AS INPUT FOR SIMULATOR tasks.

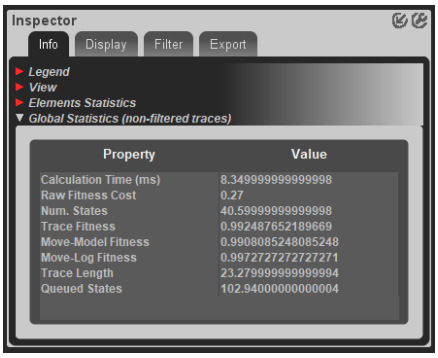
If we consider that we receive a very high number of requests for mine detection in similar environments we could use the same validation and test sets. *[Case 5,7,12]*

TRANSITION MAP OF THE MODIFIED LOG



CONFORMANCE CHECKING ON THE MODIFIED LOG

A conformance analysis was then performed on the modified log and previously extracted petri net using the "Replay Log on Petri Net for Conformance Analysis" tool to measure fitness. We introduce the **move-log fitness** as the measure of the fitness calculated considering only the mismatches solved with the trace moving a step while the model remains in the current position and the **move–model fitness** as the measure of the fitness considering only the mismatches solved with the trace remaining in the current position while the model moves a step.

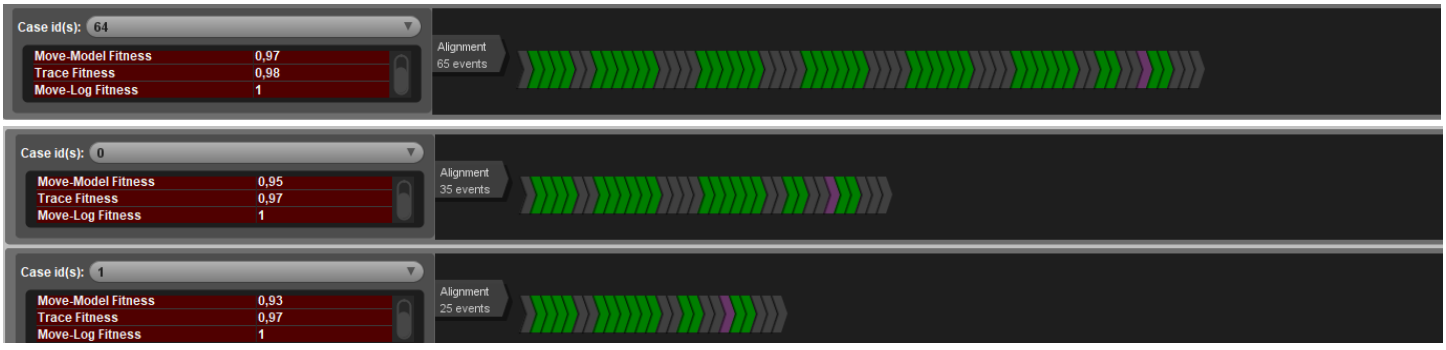


General Trace Fitness: 0,992487652189669

General Move-Model Fitness: 0,9908085248085248

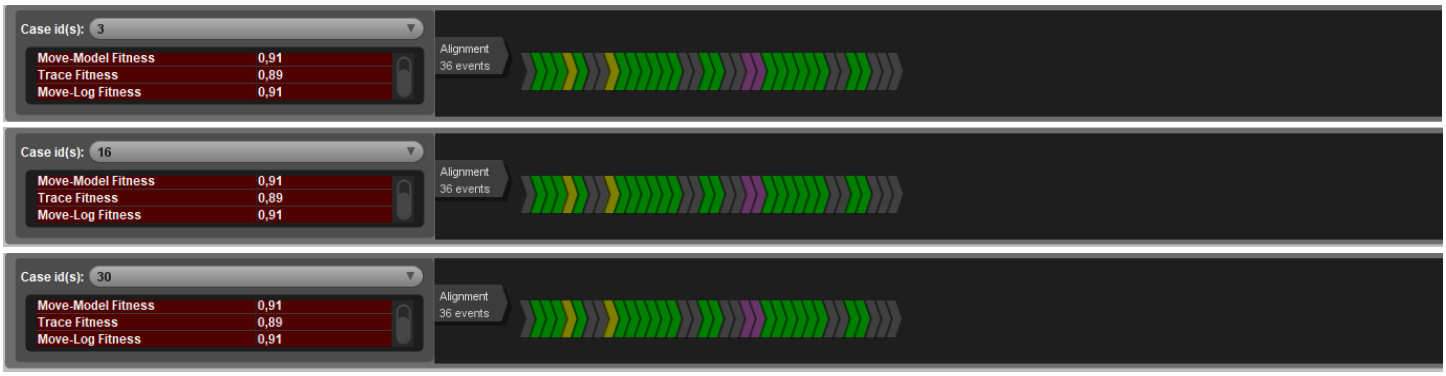
General Move-Log Fitness: 0,9972727272727271

1) [Case 0,1,64]



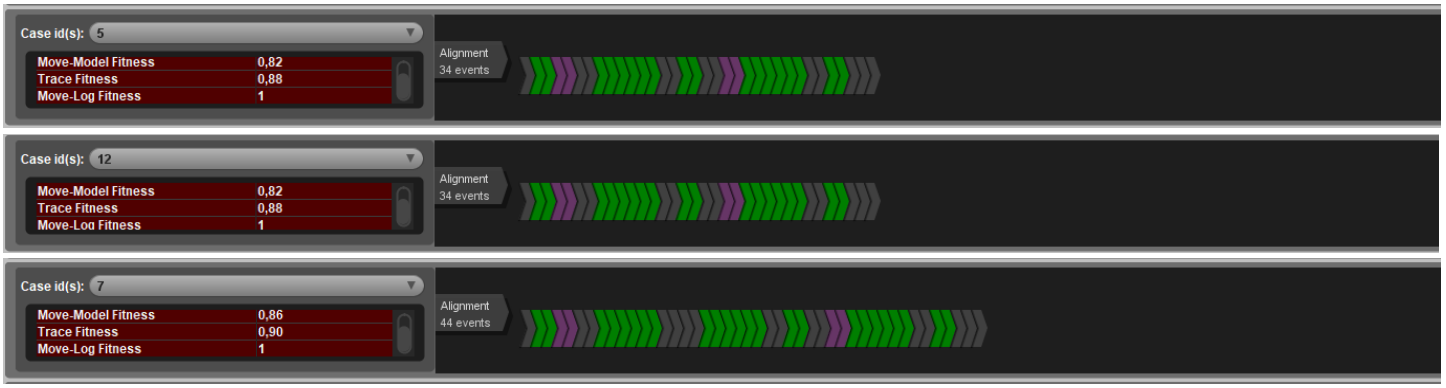
We can observe move-model fitness was affected by the violations while move-log fitness is 1: that is because a task was skipped producing a move-model violation.

2) [Case 3,16,30]



We can observe that both move-model and move-log fitness were affected by the violations: as we anticipated two tasks, we have a move-log violation for the insertion and a move-model violation for the skip.

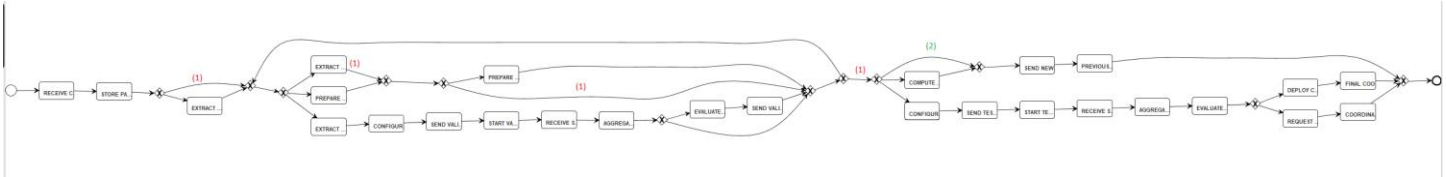
3) [Case 5,7,12]



We can observe again move-model fitness was affected by the violations. Since the tasks skipped were significantly more we can observe that the fitness is lower for these cases.

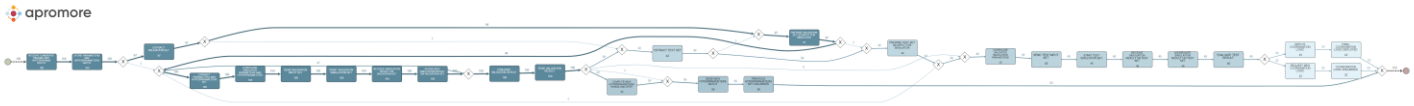
BUSINESS PROCESS DISCOVERY ON MOFIFIED LOG

The modified log was used to mine a BPMN diagram, using the BPMN Miner tool with Inductive Miner with noise threshold set to 0 to ensure maximized fitness.



In this figure we can see that some unexpected gateways and sequence flows that can lead to the acceptance of the model of strange behaviours were mined. To make an example following flows (1) we can see that this model could permit a behaviour where we completely skip the validation phase and extract the test set without processing it. In (2) we could also see that the eventual skip of the “Compute New Hyperparameters Range and Step” case was mapped as expected.

The same log was uploaded in Apromore to discover a BPMN diagram. We can observe that the diagram obtained in Apromore is less readable and has more unexpected gateways and sequence flows.



The ProM diagram was then used to extract a Petri net, on which the log was replayed to calculate fitness that we already know should be 1 from the mining algorithm settings.

The replay, the Petri net and the log were used in the "Measure Precision/Generalization" tool.

The four measures for the new model are:

Fitness: 1

Simplicity: 27 activities+ 49 sequence flows + 12 gateways=88

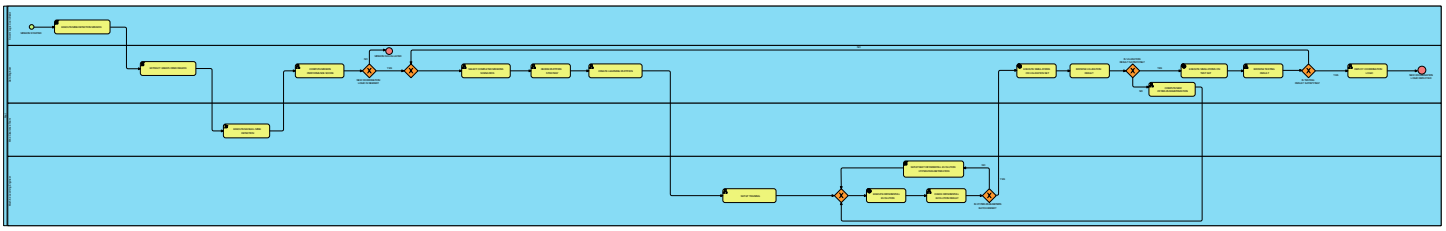
Precision: 0,78731

Generalization: 0,99054

From the comparison of the two models, we can say that the original one is a good fit while the one mined from the modified log is suffering of underfitting as it has extra sequence flows that permit unexpected behaviours. This consideration is enforced by a comparison of the precision measurements that are 0,97695 for the original and 0,78731 for the new one. We can also visually deduct that the original model is simpler, this can be also proved by the simplicity measures that are 68 for the model mined from the original log (that is a one end event version of the original model) and 88 for the one mined from the modified log.

SIMULATION MODEL AS IS OVERVIEW

BASE MODEL FOR SIMULATION



MODEL SIMULATION

A simulation was executed assigning a default 1 euro cost and 1 sec duration to each task, 10 resources per lane, 50% to each gateway, and 100 input tokens. The simulation log (called "**original log**") has been used to mine a model using both ProM and Apromore.

NORMATIVE MODEL QUALITY ESTIMATION

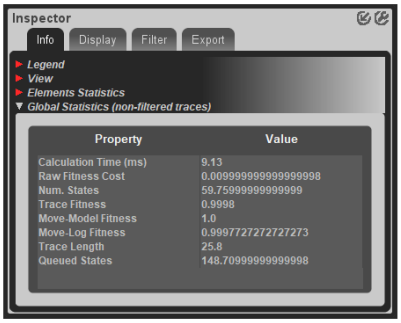
The four qualities have been measured for the normative model using the **original log**. Fitness has been computed after obtaining a Petri Net from the model and then using the "Replay a Log on Petri Net for conformance analysis" tool in ProM. Precision and Generalization have been computed with the "Measure Precision/Generalization" tool in ProM. These tools have been used for computing the qualities for all the subsequent models. Simplicity has been computed as *"#gateways + #sequence flows + #activities".*

Fitness: 0.9998

Simplicity: 7 + 32 + 20 = 59*

Precision: 0,89639

Generalization: 0,99963



*In the base model, we must consider 3 extra activities, representing the Start and End events, since they will be replaced by activities in the mined models. We must consider 3 extra sequence flows, to compensate the fictitious ones added in the mined models for the Start/End activities. We also must consider 1 extra gateway and sequence flow, that connect the two end activities to the end event.

Model	Fitness	Simplicity	Precision	Generalization
Normative Model	0.9998	59	0.89639	0.99963

The Fitness is very close to 1, as expected since this is the model that generated the original logs. One unexpected sequence of events ((1) in the transition map in *figure 15*) has been generated and the model was not able to obtain a fitness equal to 1. Precision is high but not very close to 1 because the model is very complex (with 3 loops one inside the other) and don't enforce constrains on the number of cycles. Generalization is high because the model has many loops and many other behaviours are possible (In the simulations there were only 100 token so the logs represent only a part of the possible behaviours).

TRANSITION MAP OF THE ORIGINAL LOG

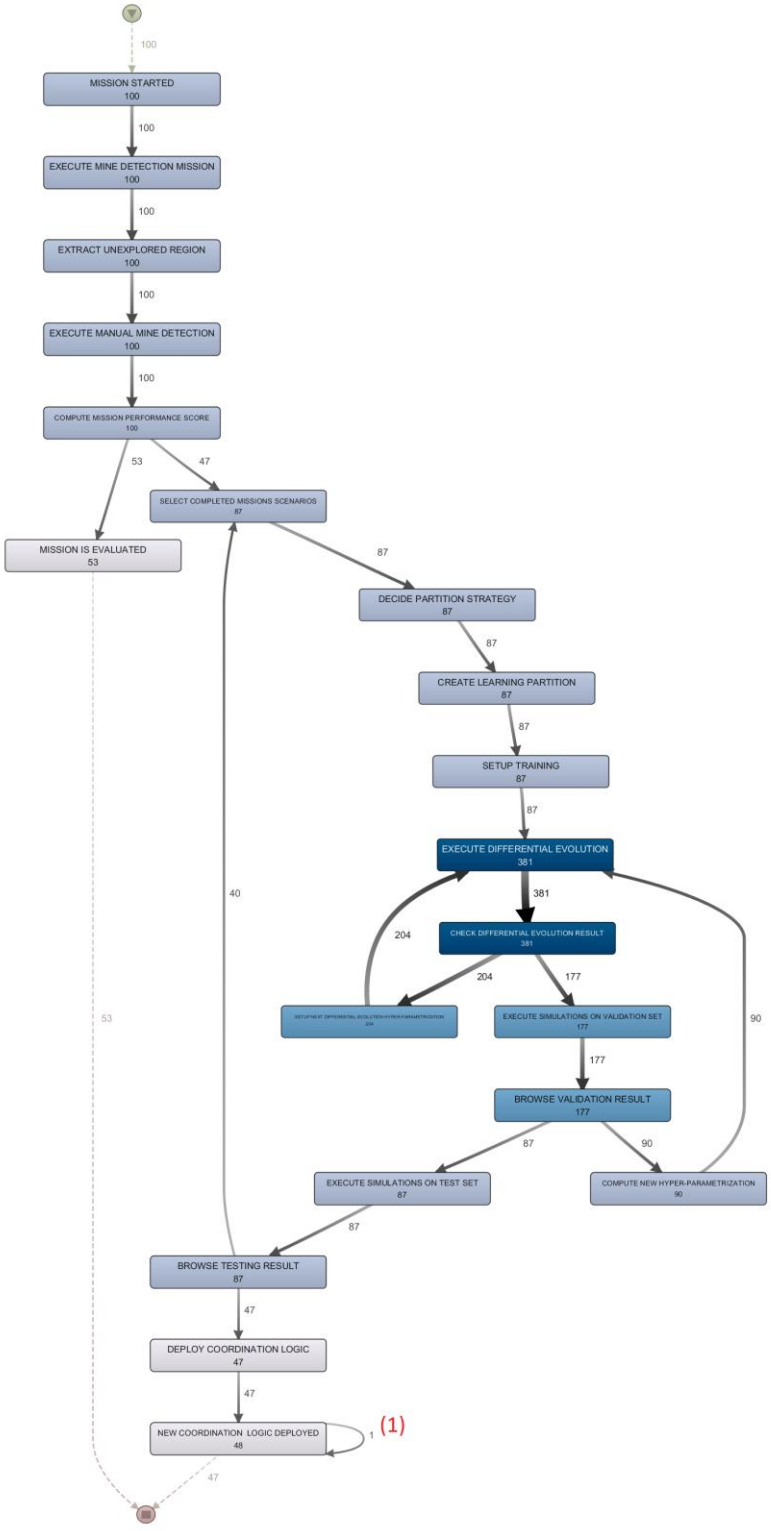


Figure 15: transition map of the original log

The transition map has been obtained by importing the original logs in Disco and setting the paths visibility to 100%. We can notice that there is an unexpected path (1) that is not present in the normative diagram.

MINED MODEL FROM THE ORIGINAL LOGS (PROM)

The **original log** were then used to obtain a model, using the "Bpmn Miner" tool (Inductive Infrequent Miner with noise threshold equal to 0, in order to maximize the fitness).

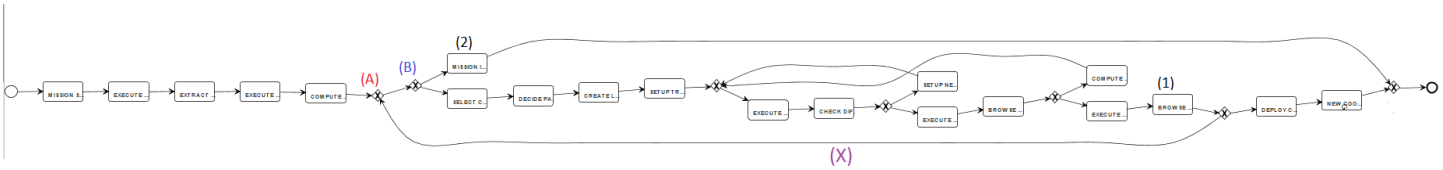
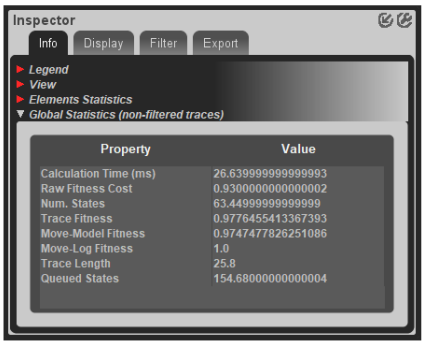


Figure 16: mined model from the original log (ProM)

We can notice that the mined model is very similar to the normative one, except for the two gateways (A) and (B) that are in inverted positions. We should not be able to come back to "Mission is evaluated" (2) from "Browse Testing Result" (1).



Fitness: 0.9776

Simplicity: 7 (gateways) + 32 (sequence flows) + 20 (activities) = 59

Precision: 0,89650

Generalization: 0,99959

Model	Fitness	Simplicity	Precision	Generalization
Mined Model original log (ProM)	0.9776	59	0.89650	0.99959

Fitness, Precision and Generalization are very similar to the Normative model qualities: this is expected since the two models are almost identical, and we used a noise threshold of 0. Simplicity is the same as the Normative model.

MINED MODEL FROM THE ORIGINAL LOG (APROMORE)

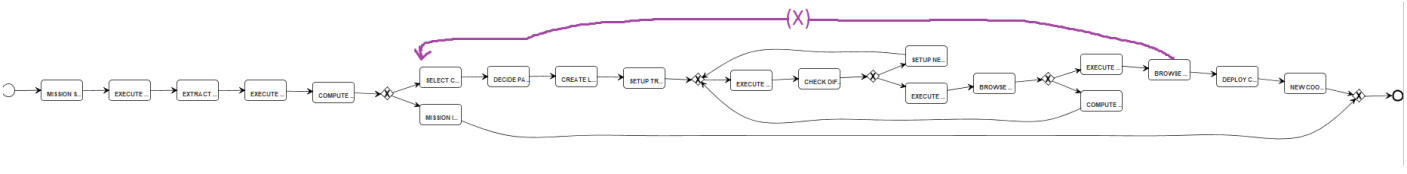
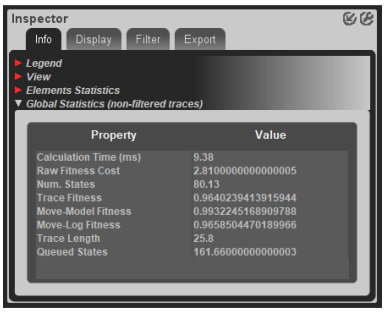


Figure 17: mined model from the original log (Apromore), with default parameters.

The model mined using Apromore (*figure 17*) on the **original log** is similar to the mined model using ProM (*figure 16*) but does not present the gateway (A) (*figure 16*)and the sequence flow (X), so it doesn't allow one of the loops of the normative model. This means that this model enforces more constraints on the possible behaviors.



Fitness: 0.9640

Simplicity: 5 (gateways) + 29 (sequence flows) + 20 (activities) = 54

Precision: 0.92247

Generalization: 0.99955

Model	Fitness	Simplicity	Precision	Generalization
Mined Model original log (Apromore)	0.9640	54	0.92247	0.99955

Fitness is not 1 because the model, as previously highlighted, does not allow the sequence flow (X) (*figure 16)* and so is not able to perfectly capture the original logs. Since the model lacks this sequence flow, it is simpler than both the mined model and the normative model and has a higher precision because enforce more constrains on the behavior.

Generalization is still very high, since the model has two loops one inside the other and allows for many other behaviors than the ones in the logs (that were limited to 100 token)

MODIFIED LOG

The original log were modified using a text editor in order to insert three realistic violations to the model, obtaining the "**modified log**".

- **"Decide Partition Strategy" not executed (*V1*):** The same partition strategy, used to split the original train set into the training set and the validation set, could be fixed at some standard values (e.g. 80% training and 20% validation) when we have enough scenarios for training. Another possibility is to re-use the same partition strategy every time a new learning set is requested after the first one. *Cases[25,61,69]*
- **Moving of "Execute Simulations on Validation Set" and "Execute Simulations on Test Set" to execute them in parallel(*V2*):**
  In order to save some time, the execution of the simulations on the validation set and the test set could be performed in parallel. This applies to cases in which a trained logic is already been simulated on the validation set a one or more times with results close to be satisfactory, so that it is unlikely that the new logic will not perform well on the validation set. *Cases[2,8,84]*
- **"Check Differential Evolution Result" not executed (*V3*):** Instead of checking the simulation result for each set of hyper-parameters in the batch, if we have a very small hyper-parametrization batch, we could only check them after the complete execution of the batch with an aggregate fitness graph. *Cases[9,82,92]*

TRANSITION MAP OF THE MODIFIED LOG

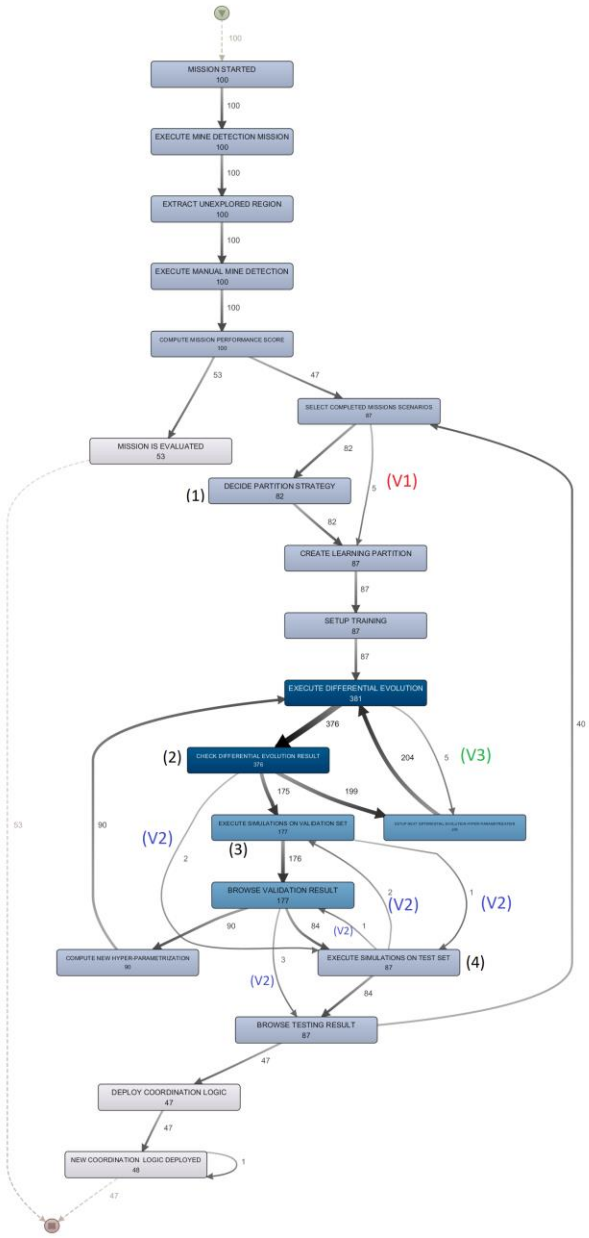
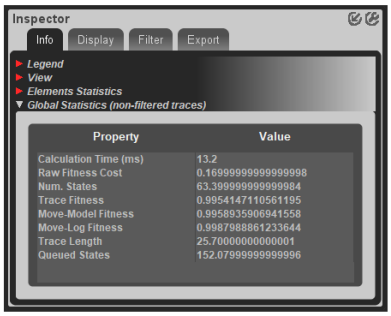


Figure 18: transition map of the modified log

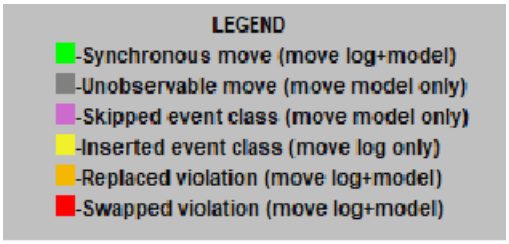
In the transition map of the **modified log**, obtained with Disco (setting paths visibility to 100%), we can notice the three violations. Task (1) can be avoided with (V1) and task (2) can be avoided with (V3). The parallelization of Tasks (3) and (4) is represented by the new sequences (V2).

FITNESS OF VIOLATIONS

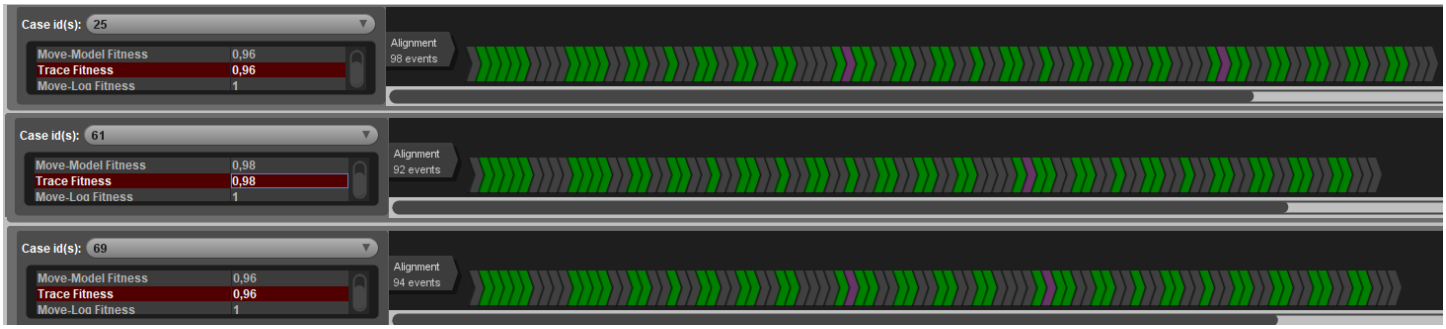
Using the ProM, a Petri Net has been extracted from the Normative Model and then we used the “Replay a Log on Petri Net for Conformane Analysis” ProM tool to replay the modified logs and compute the fitness.



We can notice that the fitness decreased from 0.9998 (normative model with original logs) to 0.9954. The violations represent a very small %, so the changes in fitness are lower than 0.5%.

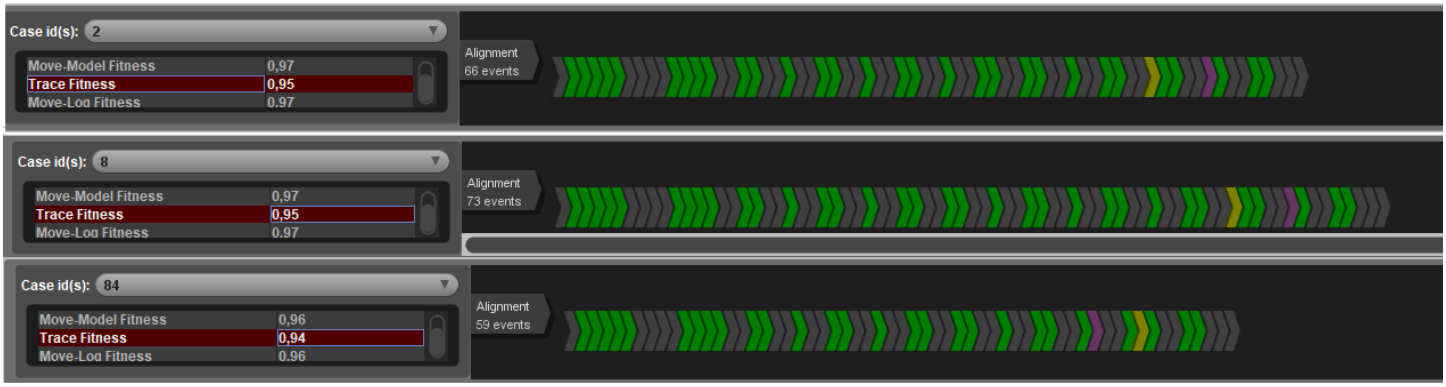


INSTANCES VIOLATION 1



The “Decide Partition Strategy” task has been skipped (purple color) two times in Case 25 and 69 (fitness: 0.96) and one time in Case 61 (fitness 0.98)

INSTANCES VIOLATION 2



In Case 2 and 8 (fitness: 0.95), the “Execute Simulations on Test Set” tasks has been anticipated and executed just before “Execute Simulations on Validation Set” (red line in *figure 19*, at next page), while in Case 84 (fitness 0.94) the “Execute Simulations on Test Set” has been executed right after “Execute simulations on Validation Set” (orange line in *figure 19*, at next page).

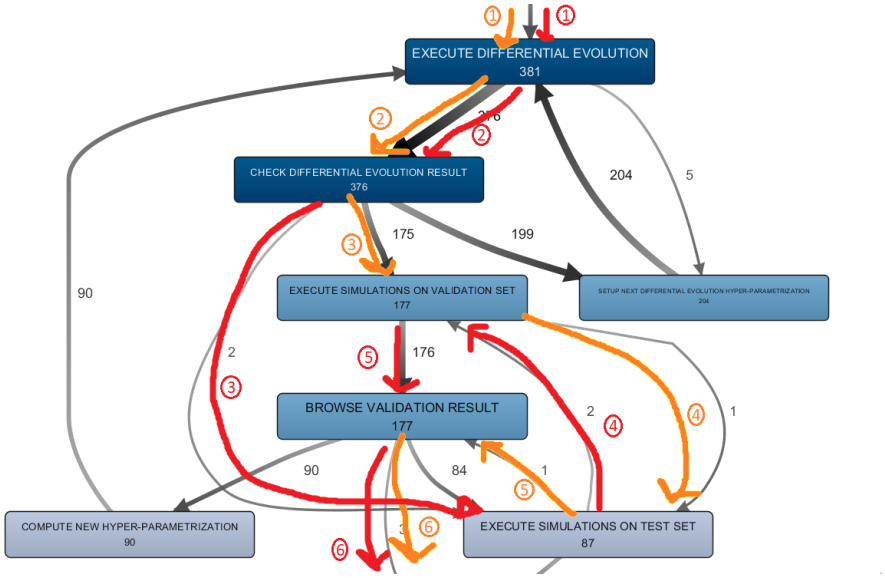


Figure 19: detail of the execution of violation 2

INSTANCES VIOLATION 3

In Case 9 (fitness 0.90), the “Check Differential Evolution Result” task has been skipped 3 times, while in case 82 (fitness 0.97) and 92 (0.96) only one time.

MINED MODEL FROM THE MODIFIED LOGS (PROM)

Following the same procedure used for the original logs, a model has been obtained from the **modified log** using ProM Inductive Miner (Infrequent & life cycle with noise threshold equal to 0)

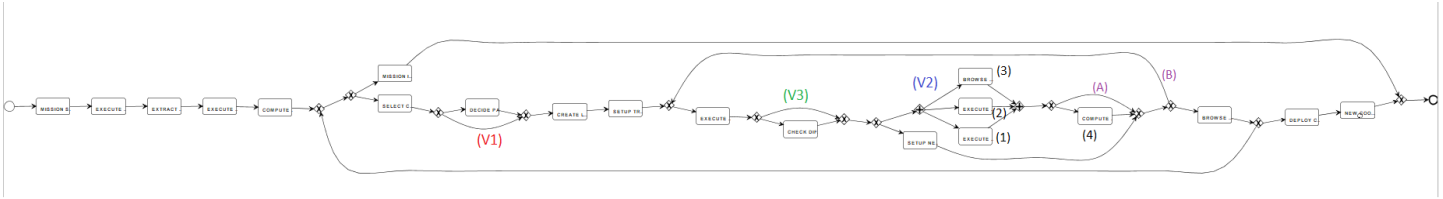
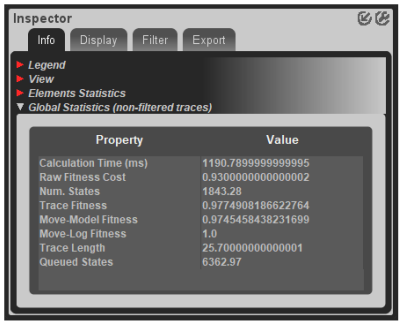


Figure 20: mined model from the modified log (ProM)

The model was able to capture violations (V1) and (V3), but there are some problems with (V2) because the model force the parallelism between "Execute Simulations on Validation Set" (1), "Execute Simulations on Test Set" (2) and "Browse Validation Result" (3).



Fitness: 0.9774

Simplicity: 15 (gateway) + 45 (sequence flows) + 20 (activities) = 80

Precision: 0.51862

Generalization: 0.99828

Model	Fitness	Simplicity	Precision	Generalization
Mined Model modified log (ProM)	0.9774	80	0.51862	0.99828

Fitness is inferior with respect of the normative model because the mined model couldn't capture some behaviours. In particular, the executions in which the "Compute new hyper-parametrization" task (4) is executed before "Execute Simulations on Test Set" (2).

Being a very complex model, simplicity is 80. There are 9 gateways more than in the normative model.

Precision is also bad since there are many possible extra behaviours. For example, we could use arcs (A) and (B) to come back from validation to training without ever updating the hyper-parameter set. Generalization is still high because 3 loops are present.

MINED MODEL FROM THE MODIFIED LOG (APROMORE)

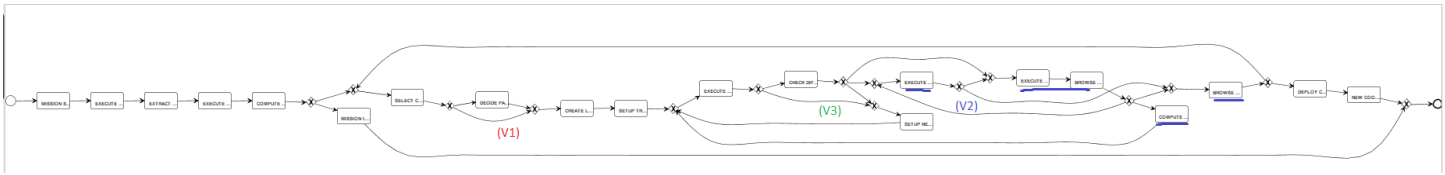
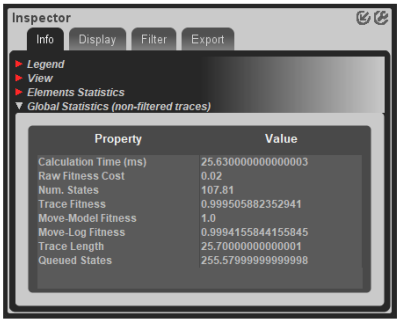


Figure 21: model mined from the modified logs (Apromore). Parallelism 100% and Arc sets to 95%

The model mined with Apromore on the **modified log** can capture violation (V1) and (V3), but not violation (V2). Some tasks (highlighted in blu in the figure) can be executed in different order or skipped with respect of the normative model, but it doesn't present a true parallelism.

Model	Fitness	Simplicity	Precision	Generalization
Mined Model modified log (Apromore)	0.9995	79	0.67178	0.99948



Fitness: 0.9995

Simplicity: 16 (gateway) + 43 (sequence flows) + 20 (activities) = 79

Precision: 0.67178

Generalization: 0.99948

The model obtained with Apromore has better qualities than the one generated with ProM. Fitness is slightly better (0.2%) and Simplicity is only 1 unit lower (79 vs 80), while Precision is significantly higher (0.67178 vs 0.51862). Generalization remains close to 1, as for all the other models.

FINAL CONSIDERATIONS

The results obtained are summarized in the following table

Model	Fitness	Simplicity	Precision	Generalization
Normative model	0.9998	57	0.89639	0.99963
Mined Model original log (ProM)	0.9776	59	0.89650	0.99959
Mined Model original log (Apromore)	0.9640	54	0.92247	0.99955
Mined Model modified log (ProM)	0.9774	80	0.51862	0.99828
Mined Model modified log (Apromore)	0.9995	79	0.67178	0.99948

Considering the models generated from the **original log**, the one generated with ProM (*figure 16*) is better than the one obtained with Apromore (*figure 17*), since it has better qualities and allows for the same behaviours of the normative model.

Regarding the models generated from the **modified log**, both were able to capture violations (V1) and (V3), but none did manage to well capture violation (V2). In the model mined with ProM (*figure 20*), it was enforced to always perform in parallel the simulations on both the validation set and the test set, but this would probably translate in waste of resources for the client most of the time. The model generated with Apromore (*figure 21*) doesn't allow to perform the parallelism required and is still very complex, but it is the best of the two.

The best solution for the client would be achieved using the normative model with the little additions required for address violations (V1) and (V3), consisting in two gateways and a sequence flow each (*figure 22*). Violation (V2) remains very complex to address and would probably require some human BP modelling.

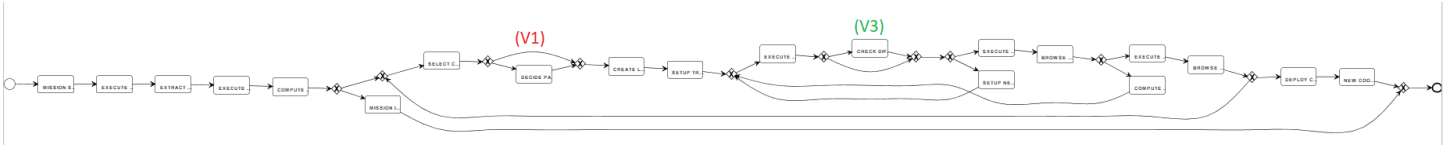


Figure 22: normative model with additions for violation (V1) and (V3)