Brain-based Wheelchair Control

Biagio Cornacchia

Salvatore Lombardi

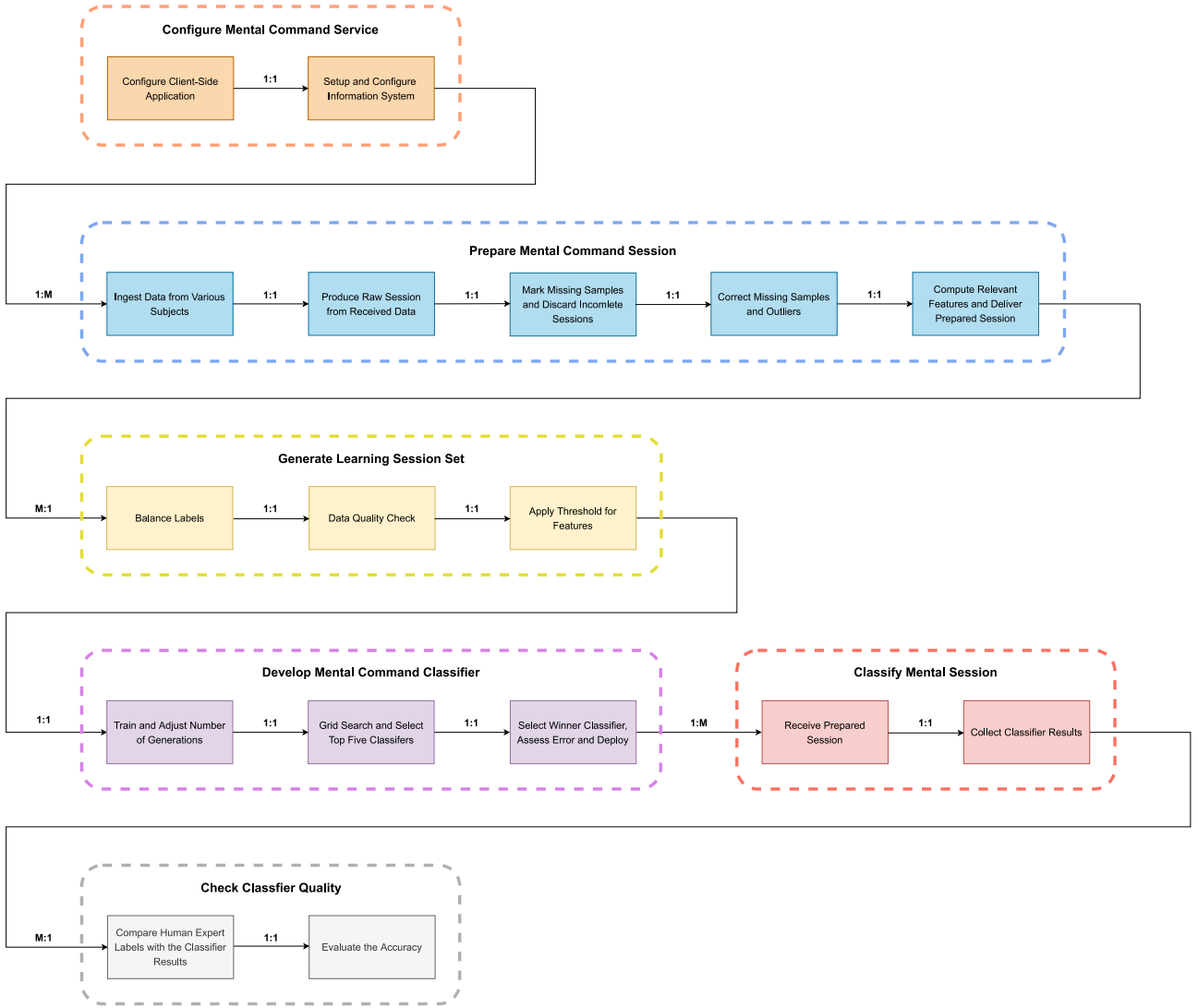
Francesco Martoccia

Matteo Abaterusso

Gianluca Gemini

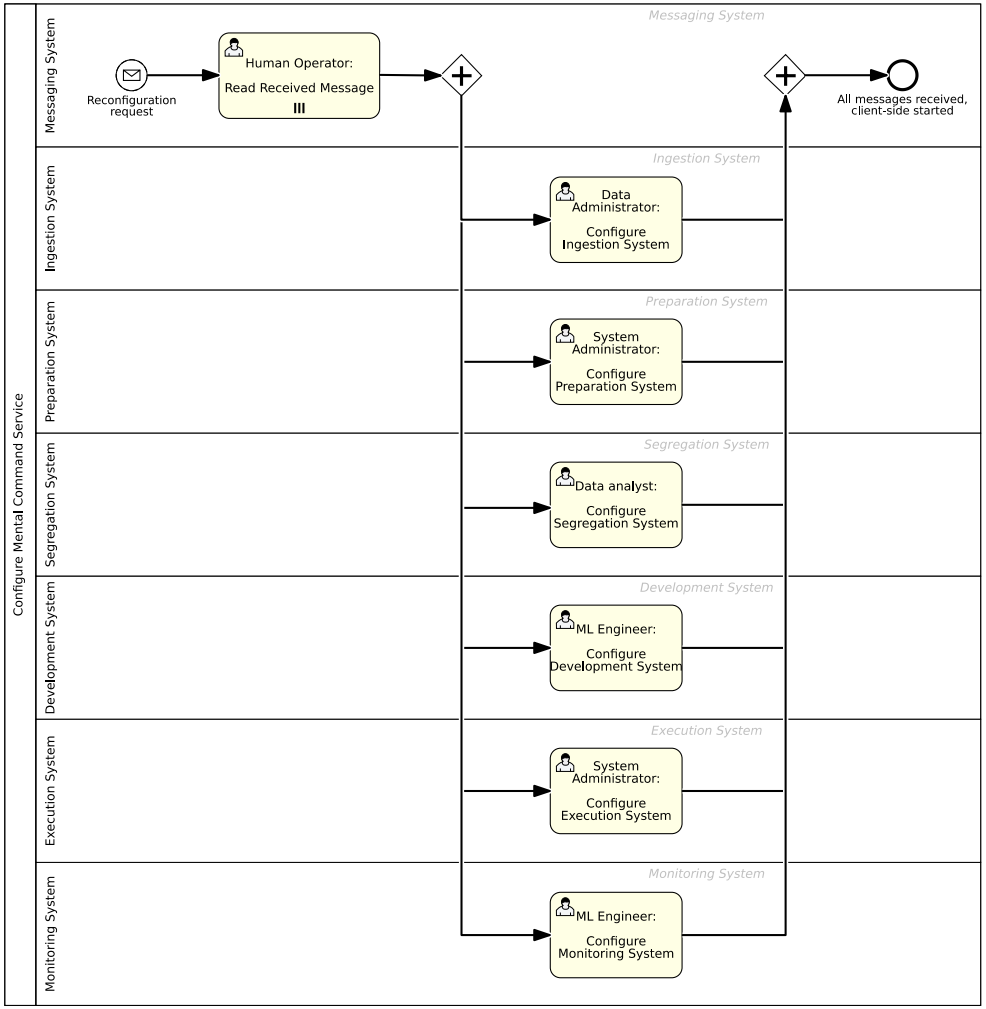
Luca Tartaglia

Process Landscape

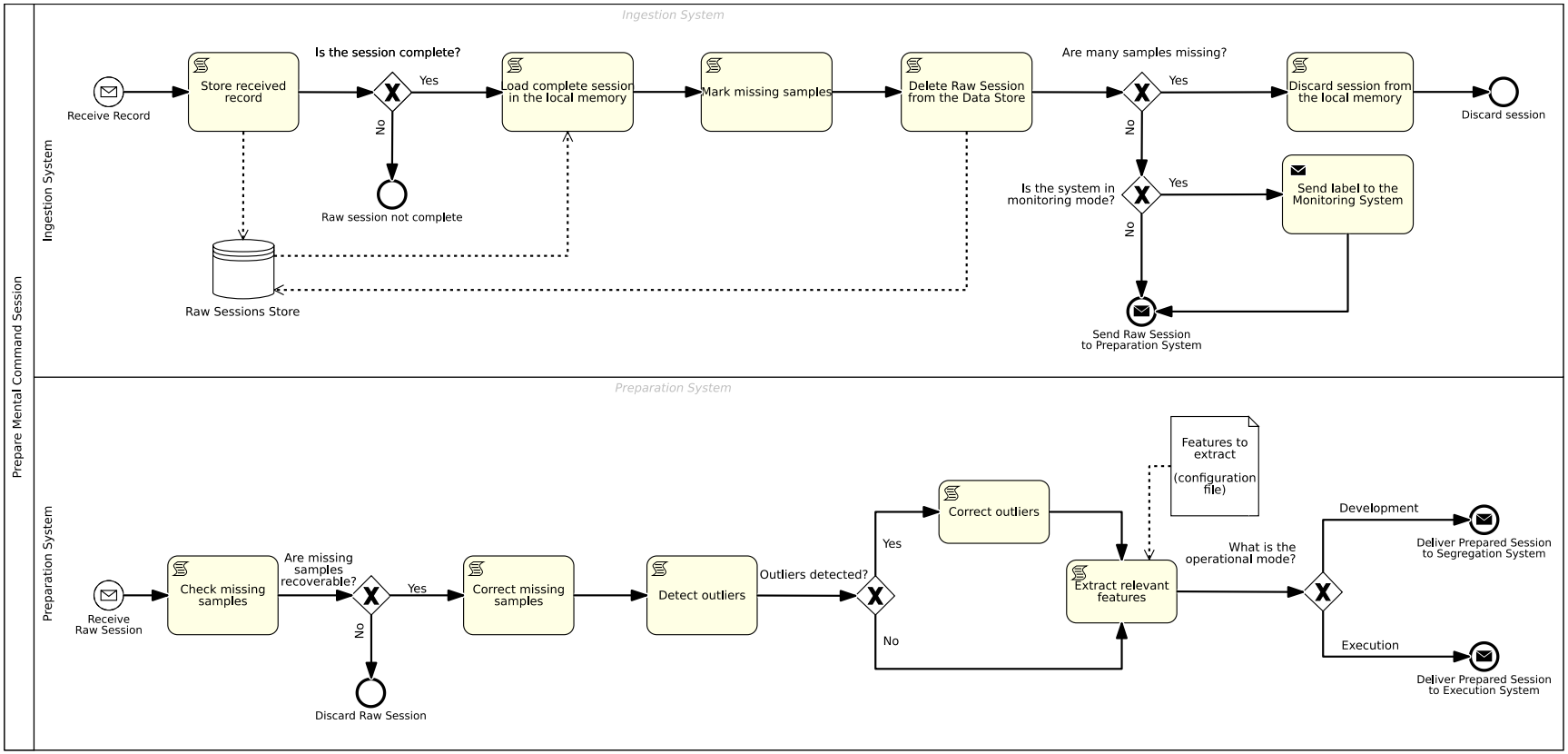


BPMN Modeling

Configure Mental Command Service (Salvatore Lombardi)

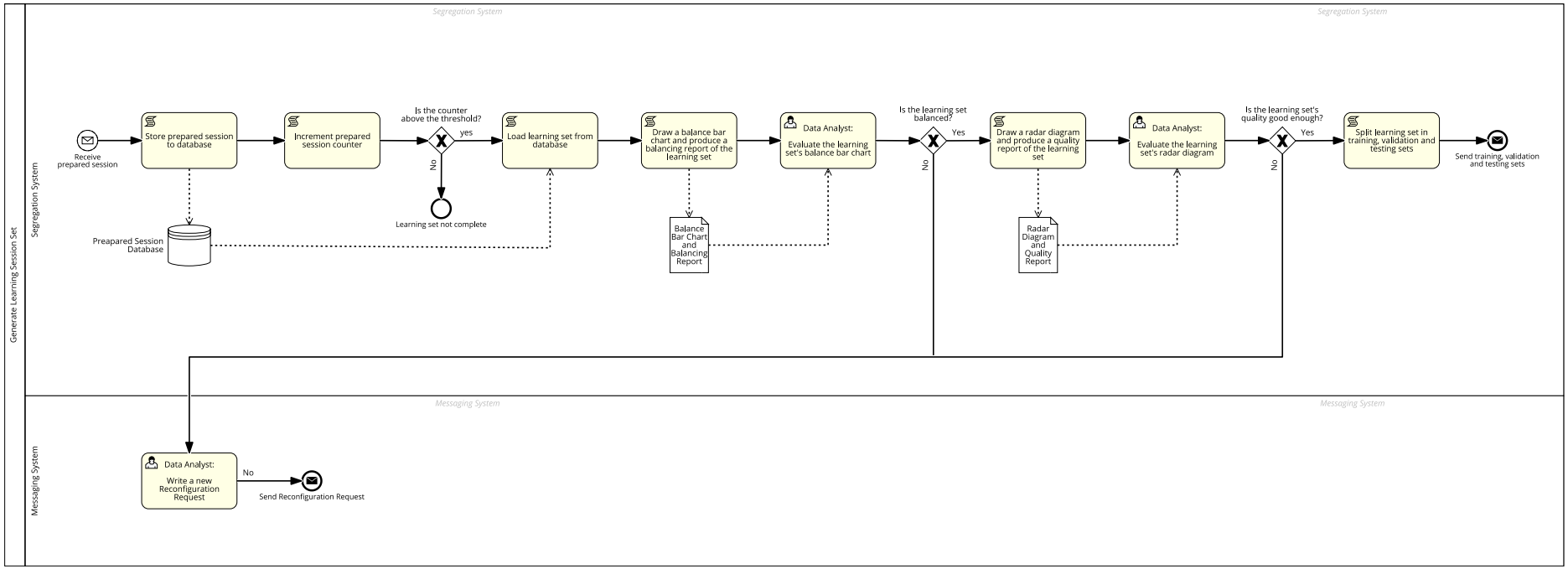


Prepare Mental Command Session (Biagio Cornacchia)

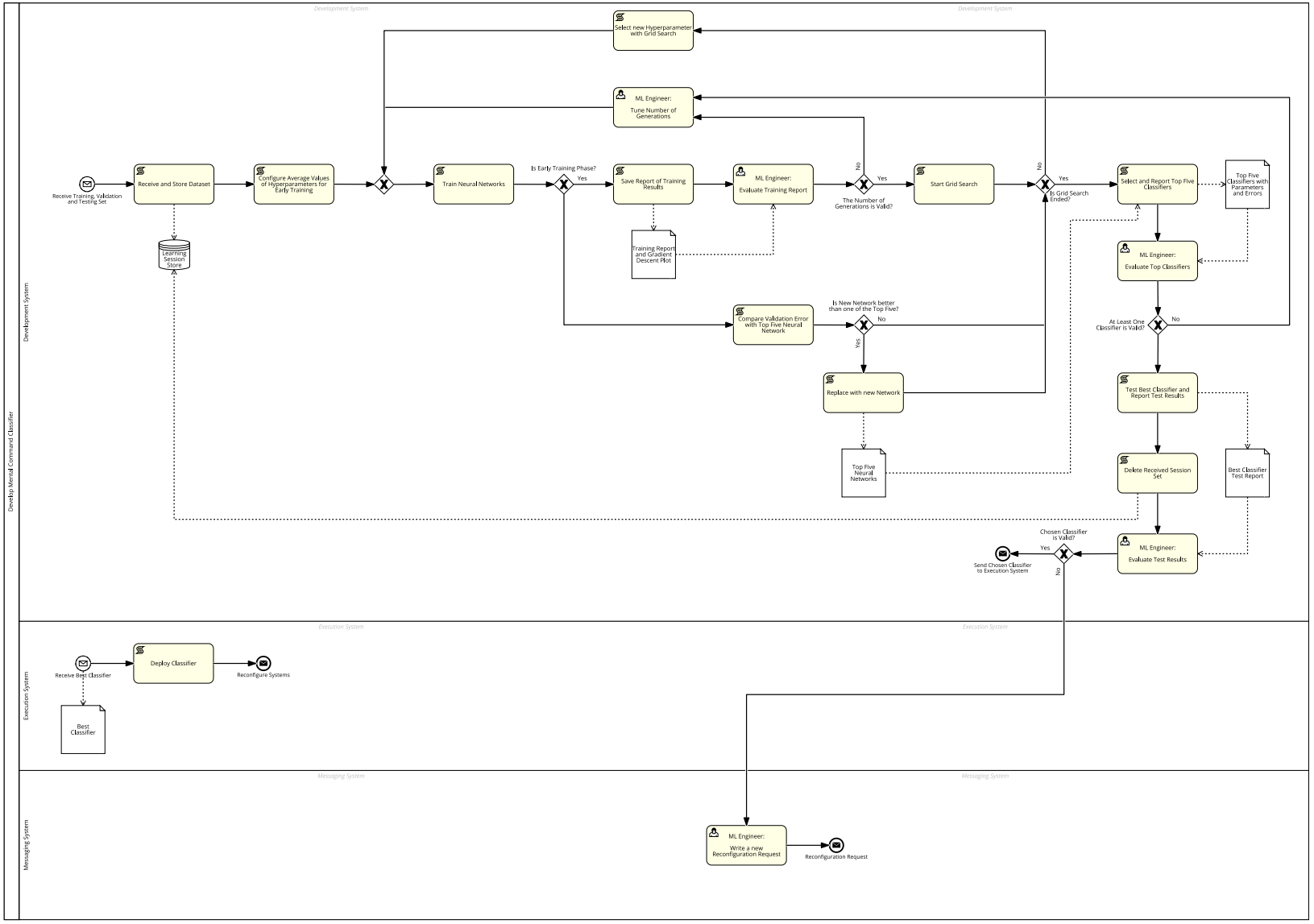


+

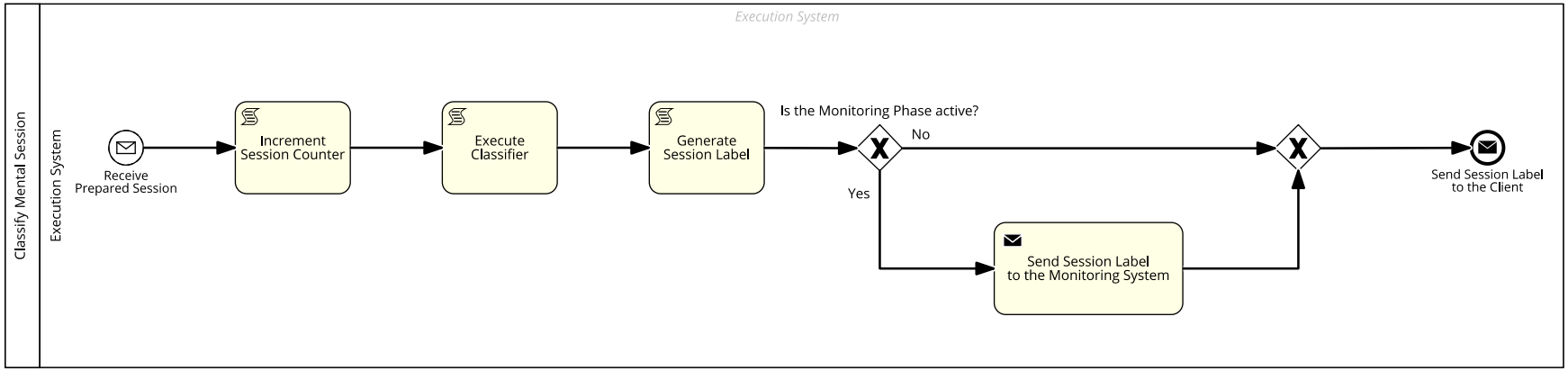
Generate Learning Session Set (Gianluca Gemini)



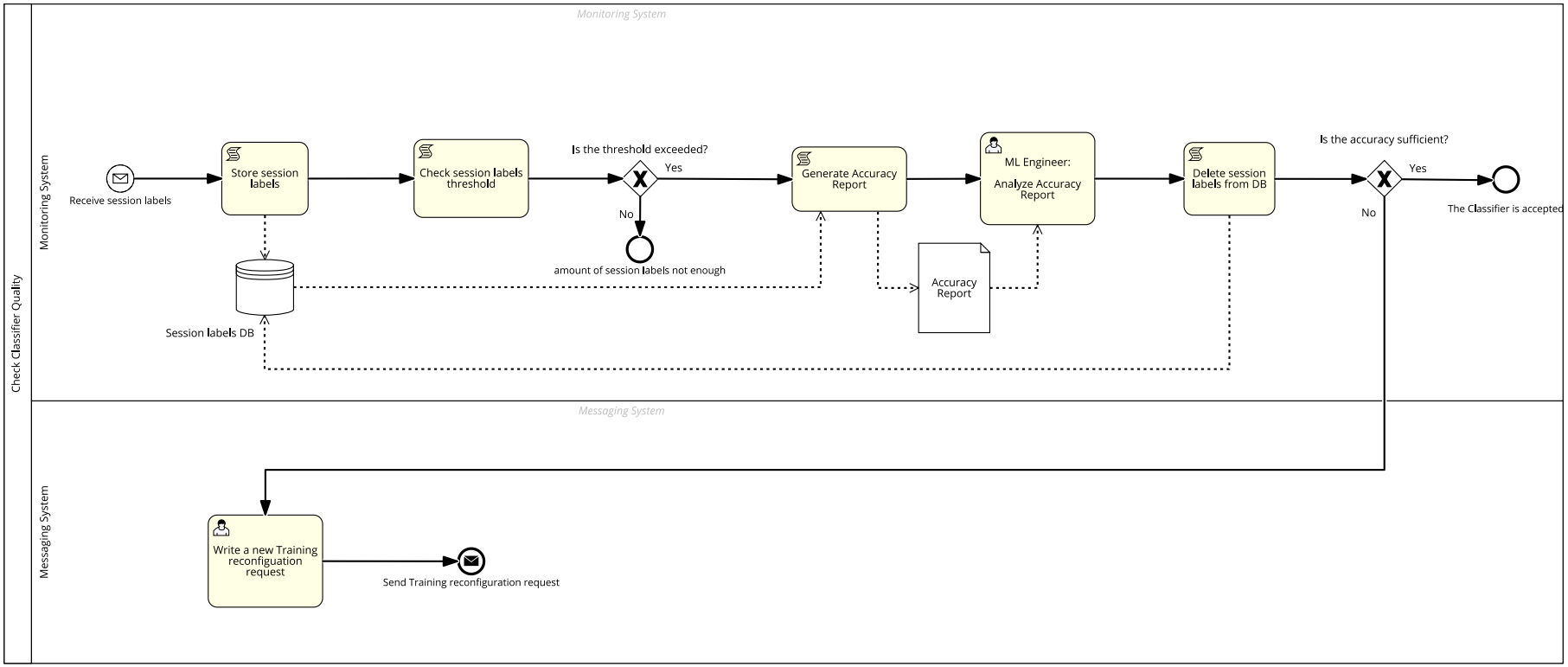
Develop Mental Command Classifier (Matteo Abaterusso)



Classify Mental Session (Francesco Martoccia)



Check Classifier Quality (Luca Tartaglia)

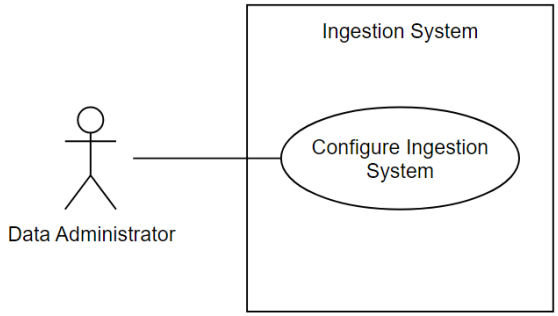


Use Cases

Ingestion System (Biagio Cornacchia)

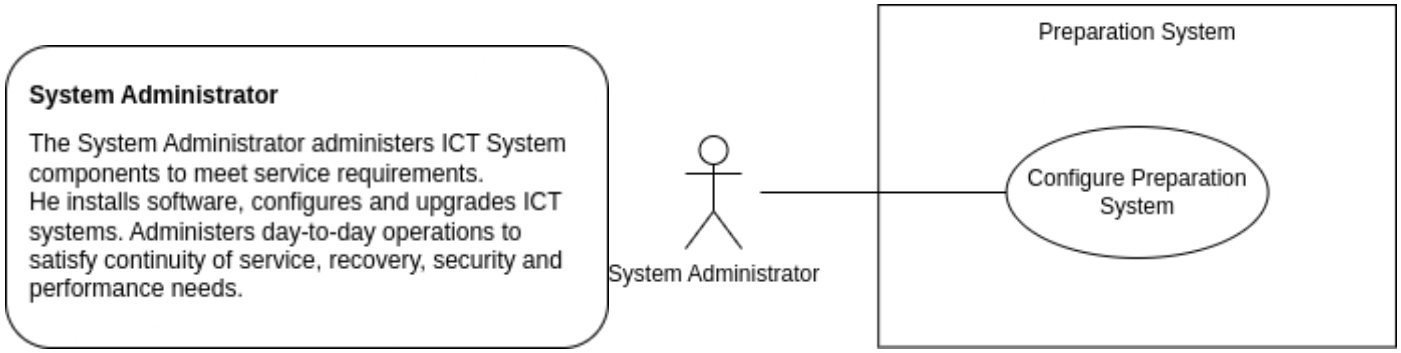
Data Administrator:
The Data Administrator designs, implements or monitors and maintains data sets, structured (databases) and unstructured (big data). Its main activities are:

- Apply standards methods and tools for measuring and reporting on wide set of relevant performance indicators (response time, availability, safety, integrity)
- Produce data set procedures and instructions for other analysts or administrators
- Monitor and maintain data management systems
- Ensure the integrity and security of existing data management systems
- Identify, investigate and correct problems or incidents related to data management systems



Use Case: Configure Ingestion System
ID: UCIS1
Actors: Data Administrator
Preconditions: 1. Ingestion system needs to be configured.
Flow of events: 1. The use case starts when the Data Administrator wants to configure the system. The reasons of the configuration can be: - System initialization - Update of the system configuration parameters 2. The Data Administrator edits the IngestionSystemConfig.json through a Text Editor as needed. 3. The Data Administrator starts the Ingestion System.
Postconditions: 1. The Ingestion System can run with the new parameters.

Preparation System (Salvatore Lombardi)



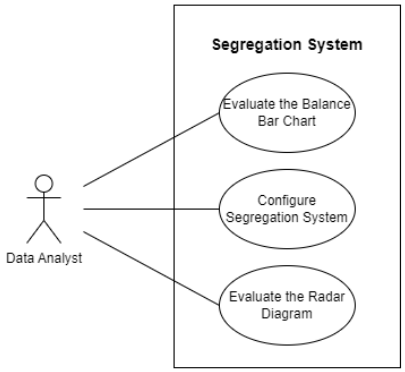
Use case: Configure Preparation System
ID: UCPS1
Actors: System Administrator
Preconditions: 1. The Preparation System has to be configured.
Flow of events: 1. The use case starts when the Preparation System has to be configured. 2. The System Administrator edits the configuration parameters as needed in the file preparation_system_configuration.json using a TextEditor. 3. The System Administrator starts the Preparation System.
Postconditions: 1. The Preparation System started with the updated parameters.

preparation_system_configuration.json

```
{
  "operative_mode": "execution",
  "segregation_endpoint_IP": "172.16.0.12",
  "segregation_endpoint_port": 5000,
  "execution_endpoint_IP": "172.16.0.13",
  "execution_endpoint_port": 5000,
  "max_eeg": 15,
  "min_eeg": -15,
  "features": {
    "delta_wave": {"start_frequency": 0.5, "end_frequency": 4},
    "theta_wave": {"start_frequency": 4, "end_frequency": 8},
    "alpha_wave": {"start_frequency": 8, "end_frequency": 12},
    "beta_wave": {"start_frequency": 12, "end_frequency": 30},
    "environment": {"indoor": 0, "outdoor": 1}
  }
}
```

Segregation System (Gianluca Gemini)

Data Analysts use data analysis to produce accessible data, charts, tables and reports.
The ability to pay attention to detail, communicate well and be highly organised are essential skills for data analysts.
They not only need to understand the data but be able to provide insight and analysis through clear visual, written and verbal communication.
In addition, the professional figure needed in this system must have knowledge in the field of EEG data.



Use Case: Configure Segregation System
ID: UCSS1
Actor: Data Analyst
Preconditions: 1. Segregation System needs to be configured
Flows of Events: 1. The use case starts when the Segregation System has to be configured 2. The Data Analyst edits the parameters in the segregation_system_config.json file using a text editor 3. The Data Analyst runs the Segregation System
Postconditions: 1. The Segregation System can run with the correct configuration

segregation_system_config.json

```
{
    "operative_mode":   "collecting_op_mode",
    "user_id" : 0,
    "testing_mode": "off",
    "endpoint_ip: "127.0.0.1",
    "endpoint_port: "4000",
    "db_name": "segregation.db",
    "collecting_threshold: 300,
    "training_set_size": 0.75,
    "validation_set_size": 0.15,
    "testing_set_size": 0.10
}
```

Use Case: Evaluate the Balance Bar Chart
ID: UCSS2
Actor: Data Analyst
Preconditions: 1. The number of sessions is enough and a new Balance Bar Chart is available 2. The operative_mode is "balancing_evaluation_mode"
Flows of Events: 1. The use case starts when the Data Analyst opens the Balance Bar Chart 2. The Data Analyst views the Balance Bar Chart using an image viewer 3. The Data Analist views the balancing_report.json file using a text editor 4. If the learning set is balanced 4.1 The Data Analyst writes 'balanced' in the balancing_report.json file using a text editor 5. If the learning set is not balanced 5.1 The Data Analyst writes 'not balanced' in the balancing_report.json file using a text editor 6. The Data Analyst starts the Segregation System
Postconditions: 1. The Segregation System can continue its flow

Use Case: Evaluate the Radar Diagram
ID: UCSS3
Actor: Data Analyst
Preconditions: 1. The learning set is balanced and a new Radar Diagram is available 2. The operative_mode is "quality_evaluation_mode"
Flows of Events: 1. The use case starts when the Data Analyst opens the Radar Diagram 2. The Data Analyst views the Radar Diagram using an imager viewer 3. If the learning session set's quality is good 3.1 The Data Analyst writes 'good quality' in the quality_report.json file using a text editor 4. If the learning session set's quality is bad 4.1 The Data Analyst writes 'bad quality' in the quality_report.json file using a text editor 5. The Data Analyst starts the Segregation System
Postconditions: 1. The Segregation System can continue its flow

balancing_report.json

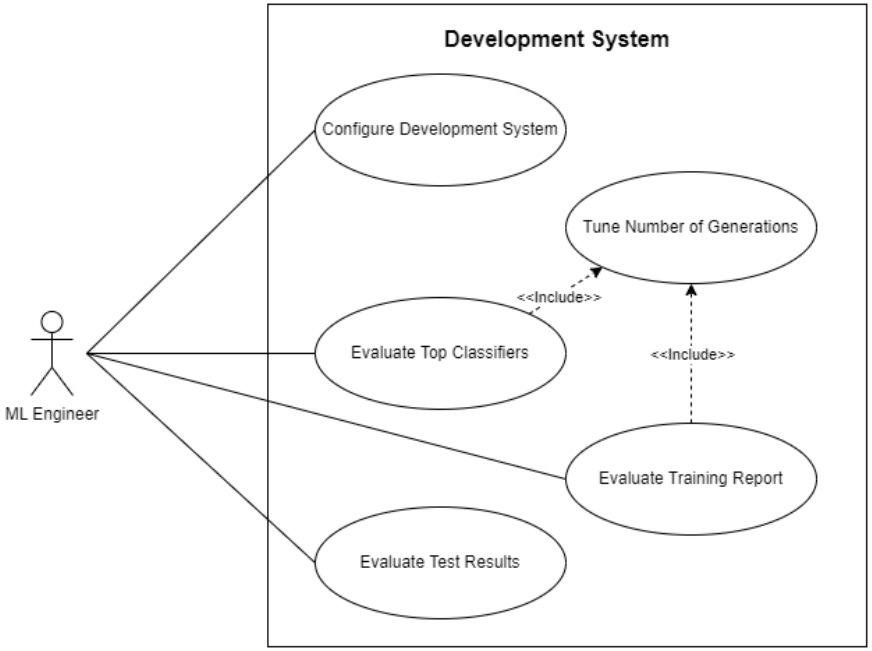
```
{
    "alpha" : 25,
    "beta" : 21,
    "delta": 22,
    "theta : 20,
    "evaluation": "balanced"
}
```

quality_report.json

```
{
    "evaluation": "good quality"
}
```

Development System (Matteo Abaterusso)

A **Machine Learning Engineer** is responsible for creating programmes and algorithms that enable machines to take actions without being directed. Some important skills are to produce project outcomes and isolate the issues that need to be resolved, in order to make programmes more effective, and to build algorithms based on statistical modelling procedures and build and maintain scalable machine learning solutions in production.



Use case: Configure Development System
ID: UCDS1
Actor: ML Engineer
Preconditions: 1. Development System should be configured.
Flows of events: 1. The use case starts when the Development System has to be configured. 2. The ML Engineer edits with a text editor development_system_config.json. 3. The ML Engineer starts the Development System.
Postconditions: 1. The Development System can run with the correct paramenters.

Use case: Tune Number of Generations
ID: UCDS2
Actor: ML Engineer
Preconditions: 1. The number of Generations is not valid according to gradient descent plot, or no valid classifier is found in the top five.
Flows of events: 1. The use case starts when a new number of generations is needed. 2. The ML Engineer edits with a text editor the number of generations in the number_of_generations.json file. 3. The ML engineer restarts the systems.
Postconditions: 1. The system starts in training mode with a new Number of Generations.

Use case: Evaluate Training Report
ID: UCDS3
Actor: ML Engineer
Preconditions: 1. An early training phase is ended. 2. A training report are created by the system.
Flows of events: 1. The use case starts when a new training report is produced during the early training phase. 2. The ML Engineer reads with a text editor in early_training_report.json the hyperparameters used. 3. The ML Engineer watches with a image viewer gradient_descent_plot.png. 4. If the gradient descent plot is good: 4.1 The ML Engineer sets the valid_generations parameters as "true" with a text editor in early_training_report.json 4.2 The ML Engineer restarts the system. 5. If the gradient descent plot is not good: 5.1 The ML Engineer sets the valid_generations parameters as "false" with a text editor in early_training_report.json. 5.2 The ML Engineer modifies the Number of Generations (use case **UCDS2**).
Postconditions:
Alternative flow 1: 1. The ML Engineer sets the valid_generations parameter as "true".
Postconditions: 1. The system starts the Grid Search.
Alternative flow 1: 1. The ML Engineer sets the valid_generations parameter as "false".
Postconditions: 1. The system starts a new Early Training Phase.

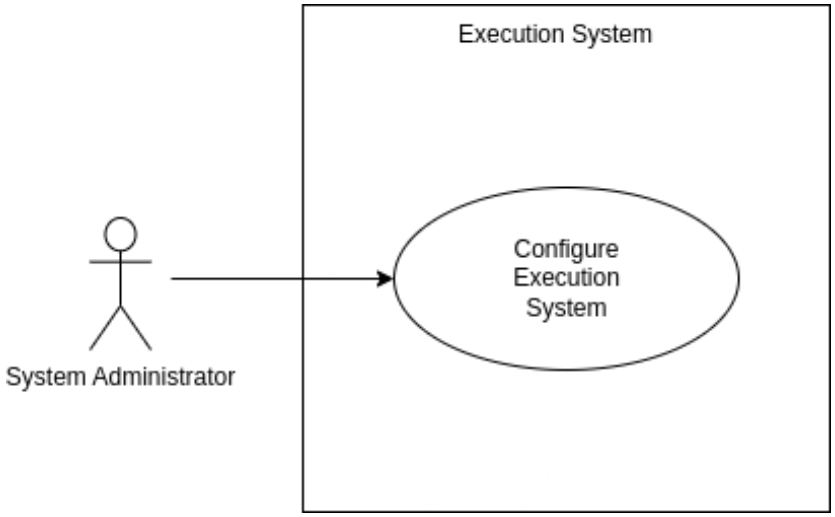
Use case: Evaluate Top Classifiers
ID: UCDS4
Actor: ML Engineer
Preconditions: 1. A new report containing the top five classifiers is available.
Flows of events: 1. The use case starts when the Grid Search phase is ended and the system has collected the top five produced classifiers. 2 The ML engineer reads with a text editor the top five classifiers in top_five_classifiers_report.json. 3. For each classifier in the report: 3.1 The ML Engineer evaluate the difference between training and validation error. 3.2 If the difference is under a given threshold and errors are better than previous classifiers: 3.2.1 The ML Engineer sets the paramenter actual_best as "true" with a text editor in top_five_classifiers_report.json. 3.2.1 The ML Engineer sets the paramenter actual_best of the previous classifiers as "false" with a text editor in top_five_classifiers_report.json. 4. If one Classifier has the actual_best parameter as "true": 4.1 The ML Engineer restarts the system. 5. If no Classifier has the actual_best parameter as "true": 5.1 The ML Engineer modifies the Number of Generations (use case **UCDS2**).
Postconditions:
Alternative flow 1: 1. The ML Engineer restarts and one classifier has the parameter actual_best as "true".
Postconditions: 1. The system starts the testing of the Best Classifier.
Alternative flow 2: 1. The ML Engineer modifies the Number of Generations.
Postconditions: 1. The system starts a new Early Training Phase.

Use case: Evaluate Test Results
ID: UCDS5
Actor: ML Engineer
Preconditions: 1. The Best Classifier test report is available.
Flows of events: 1. The use case starts when the Best classifier is tested with the Test Set and a report is created. 2. The ML Engineer reads the training and test error with a text editor from test_best_classifier_report.json. 3. The ML Engineer evaluate the difference between training and test error. 4. If the difference is under a given threshold: 4.1 The ML Engineer sets the valid_classifier parameters as "true" with a text editor in test_best_classifier_report.json. 4.2 The ML Engineer restarts the system. 5. If the difference exceeds the threshold: 5.1 The ML Engineer sets the valid_classifier parameters as "false" with a text editor in test_best_classifier_report.json. 5.2 The ML Engineer requests a Reconfiguration of the systems to the Messaging System 5.3 The ML Engineer restarts the system.
Postconditions:
Alternative flow 1: 1. The ML Engineer sets the valid_classifier parameters as "true".
Postconditions: 1. The system starts the deployment of the Best Classifier to the Execution System.
Alternative flow 2: 1. The ML Engineer sets the valid_classifier parameters as "false".
Postconditions: 1. The system waits for new Training, Validation an Test sets.

Execution System (Francesco Martoccia)

System Administrator

The System Administrator administers ICT System components to meet service requirements.
He installs software, configures and upgrades ICT systems. Administers day-to-day operations to satisfy continuity of service, recovery, security and performance needs.



Use case: Configure Execution System
ID: UCES1
Actors: System Administrator
Preconditions: 1. The configuration_execution_system.json file contains parameters that need to be updated.
Flow of events: 1. The use case starts when the System Administrator must configure the Execution System. 2. The System Administrator updates the parameters of the configuration_execution_system.json file through a Text Editor. 3. The System Administrator launches the Execution System.
Postconditions: 1. The configuration_execution_system.json file contains the updated parameters. 2. The Execution System is running.

configuration_execution_system.json

```
{
    "operating_mode" : "execution",
    "execution_window_value" : 2000,
    "monitoring_window_value" : 50,
    "endpoint_IP" : "127.0.0.1",
    "endpoint_port" : 5000,
    "testing" : true
}
```

Monitoring System (Luca Tartaglia)

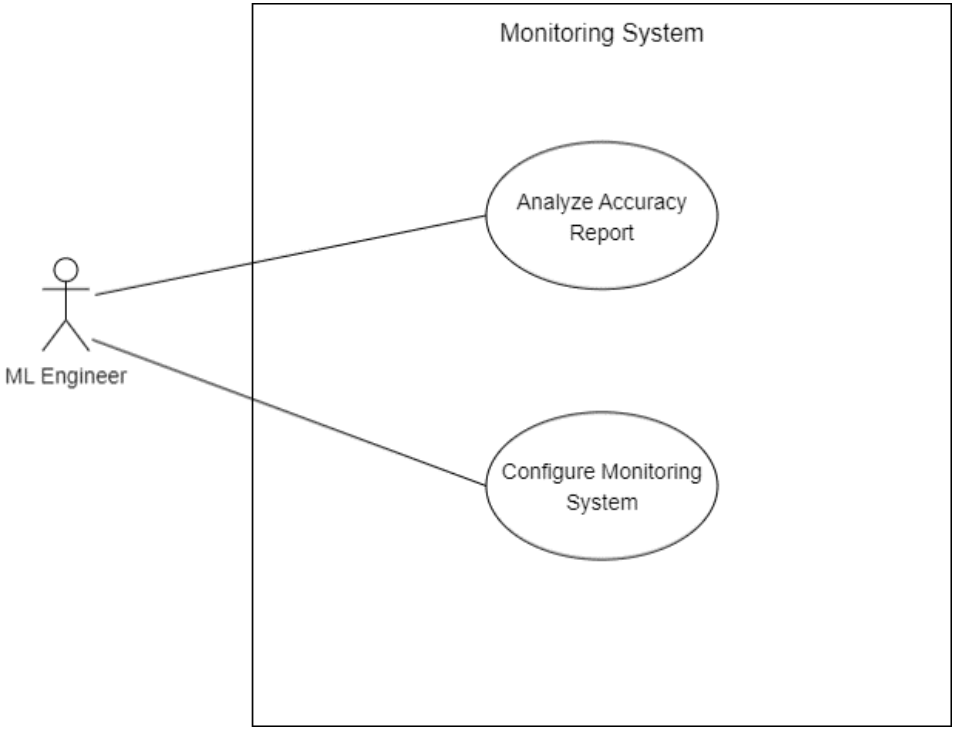
ML Engineer Definition:

The ML Engineer is an experienced programmer who manages Machine Learning projects, then verifies the proper implementation and application of the development algorithms.

The ML Engineer is primarily responsible for:

- **Assisting the team in finding and analyzing the data** needed to be able to create ML models.
- **Designing and developing ML models** based on analysis performed.
- **Monitor and take care of the data flow** and the model.
- Intervene promptly in case of performance drops.

Finally, a ML model must be able to run in production environments at peak performance and, most important be durable and repeatable over time. This involves significant ongoing monitoring and debugging by the ML Engineer.

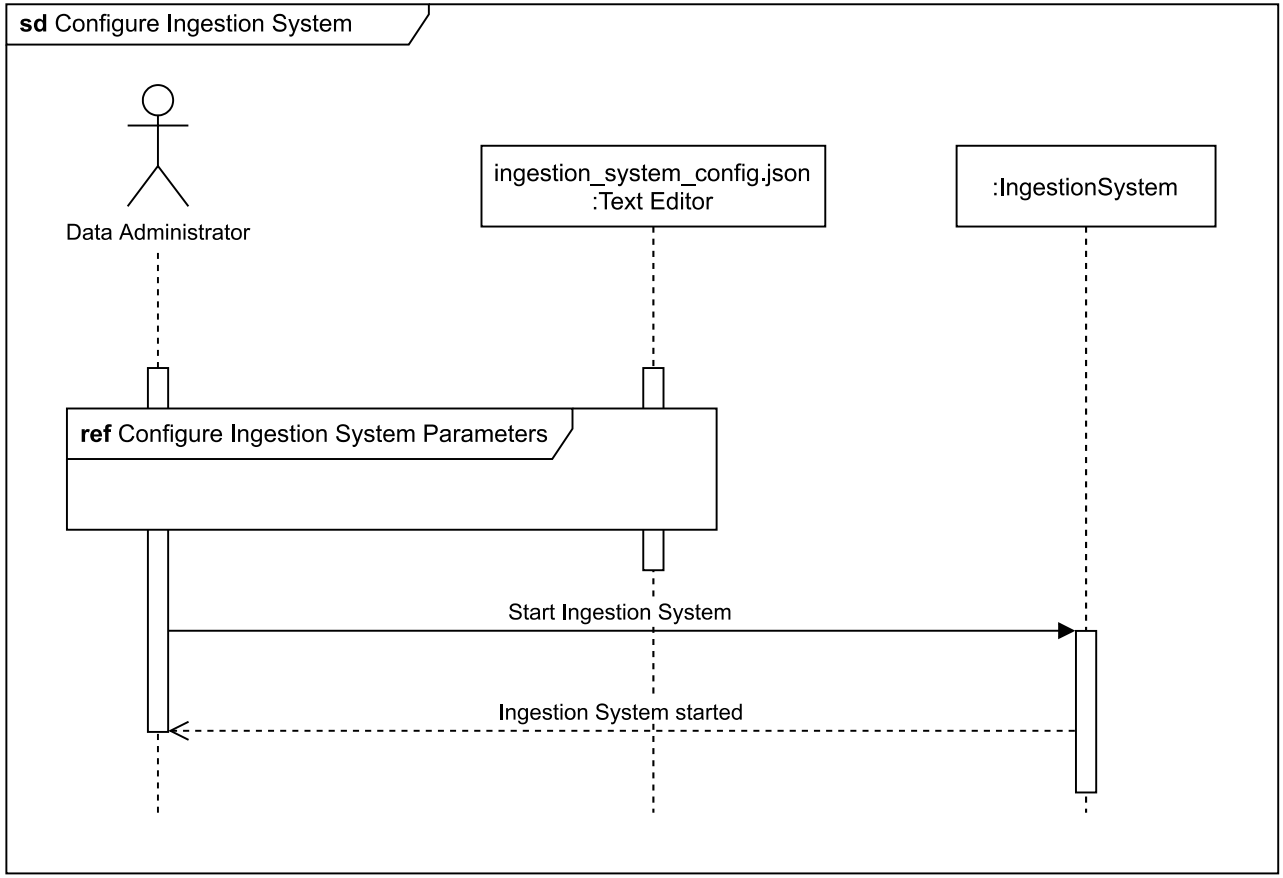


Use Case: Analyze Accuracy Report
ID: UCMS1
Actors: ML Engineer
Preconditions: 1. The Accuracy Report is generated by the system
Flow of Events: 1. The use case starts when the system produces the Accuracy Report. 2. The ML Enigneer reads the file "accuracy_report.json" through the Text Editor. 3. If the Accuracy is sufficient 3.1 The ML Engineer sets up the parameter "classifier_accepted " as "Yes" by using the Text Editor. 4. Else 4.1 The ML Engineer sets up the parameter "classifier_accepted " as "No" by using the Text Editor.
Postconditions:

Use Case: Configure Monitoring System
ID: UCMS2
Actors: ML Engineer
Preconditions: 1. Monitoring System must be configured
Flow of Events: 1. The use case starts when the Monitoring System must be configured. 2. The ML Engineer sets up the Monitoring System parameters with the Text Editor. 3. The ML Engineer runs the Monitoring System.
Postconditions: The Monitoring System runs with the correct parameters.

Sequence Diagrams

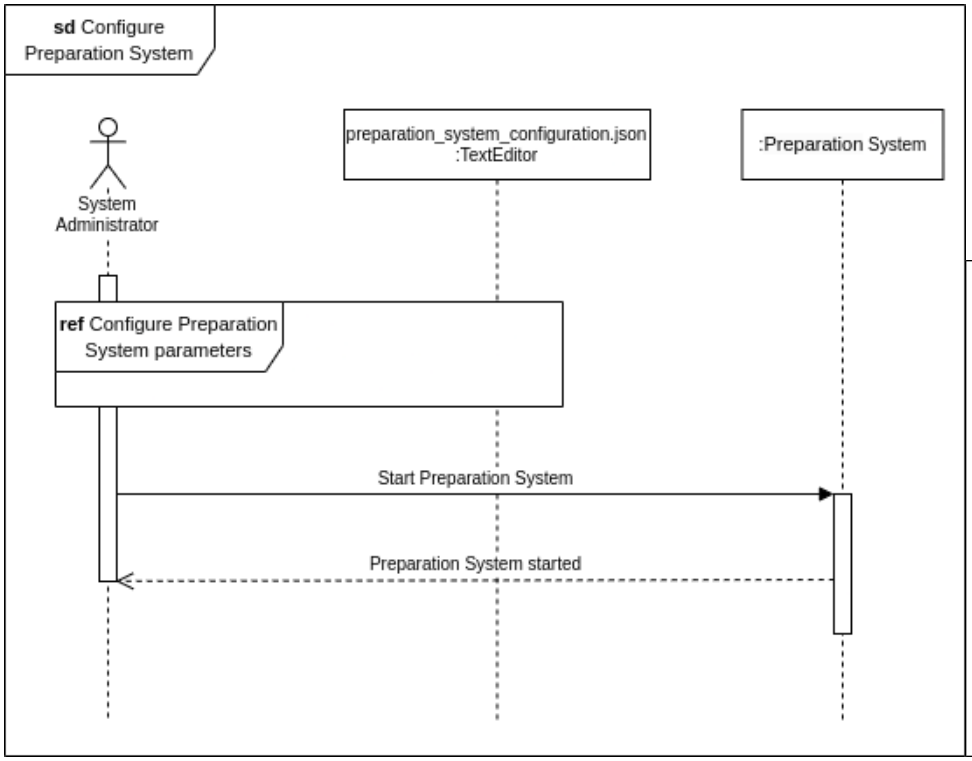
Ingestion System (Biagio Cornacchia)



ingestion_system_conf.json

```
{
operative_mode: "development"
preparation_system_ip: "172.16.0.10"
preparation_system_port: 4000
monitoring_system_ip: "172.16.0.20"
monitoring_system_port: 4000
execution_window: 2000
monitoring_window: 50
missing_samples_threshold: 2
}
```

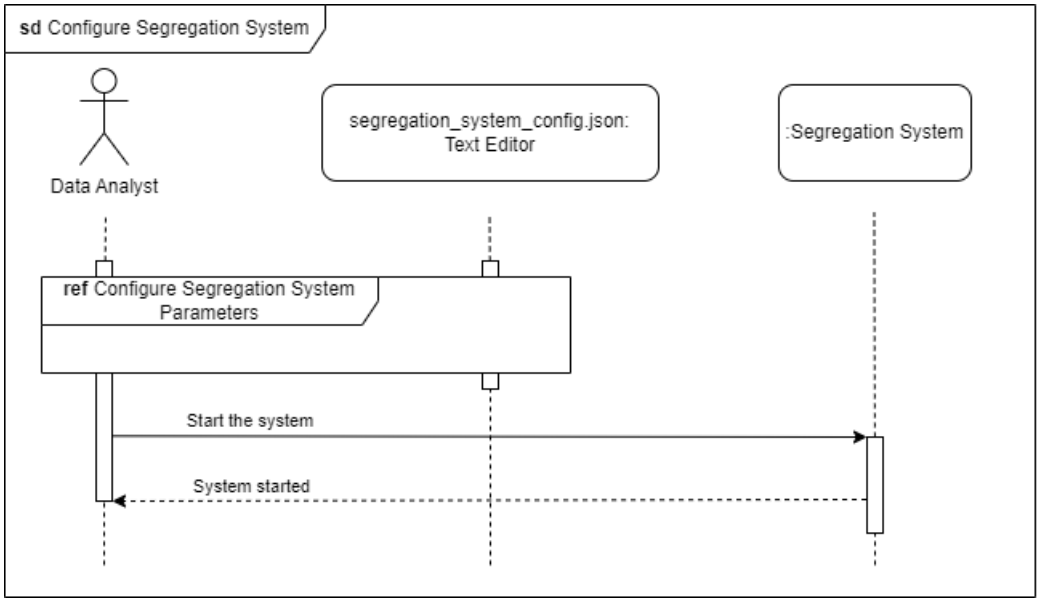
Preparation System (Salvatore Lombardi)



preparation_system_configuration.json

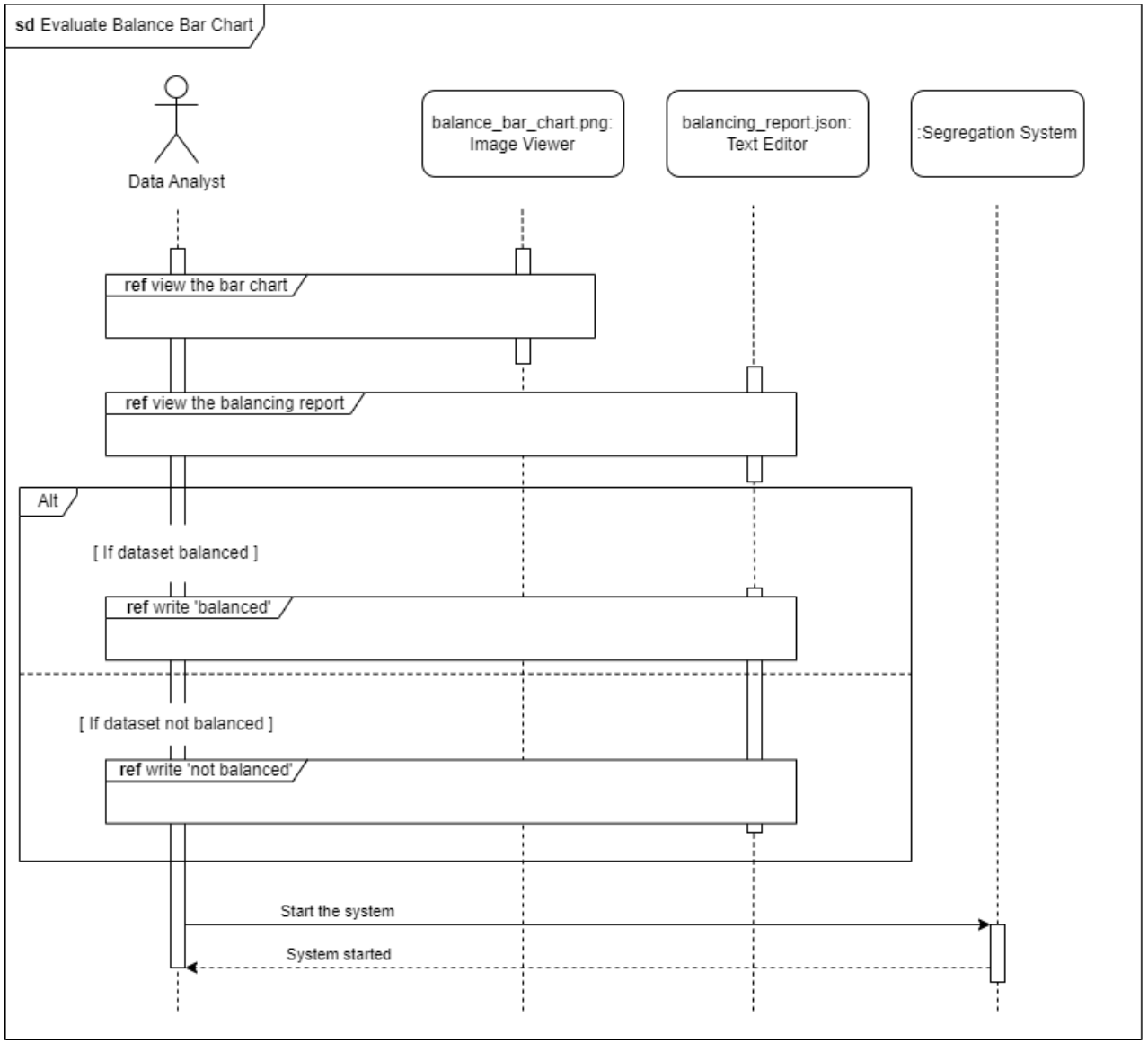

```
{
  "operative_mode": "execution",
  "segregation_endpoint_IP": "172.16.0.12",
  "segregation_endpoint_port": 5000,
  "execution_endpoint_IP": "172.16.0.13",
  "execution_endpoint_port": 5000,
  "max_eeg": 15,
  "min_eeg": -15,
  "features": {
    "delta_wave": {"start_frequency": 0.5, "end_frequency": 4},
    "theta_wave": {"start_frequency": 4, "end_frequency": 8},
    "alpha_wave": {"start_frequency": 8, "end_frequency": 12},
    "beta_wave": {"start_frequency": 12, "end_frequency": 30},
    "environment": {"indoor": 0, "outdoor": 1}
  }
}
```

Segregation System (Gianluca Gemini)



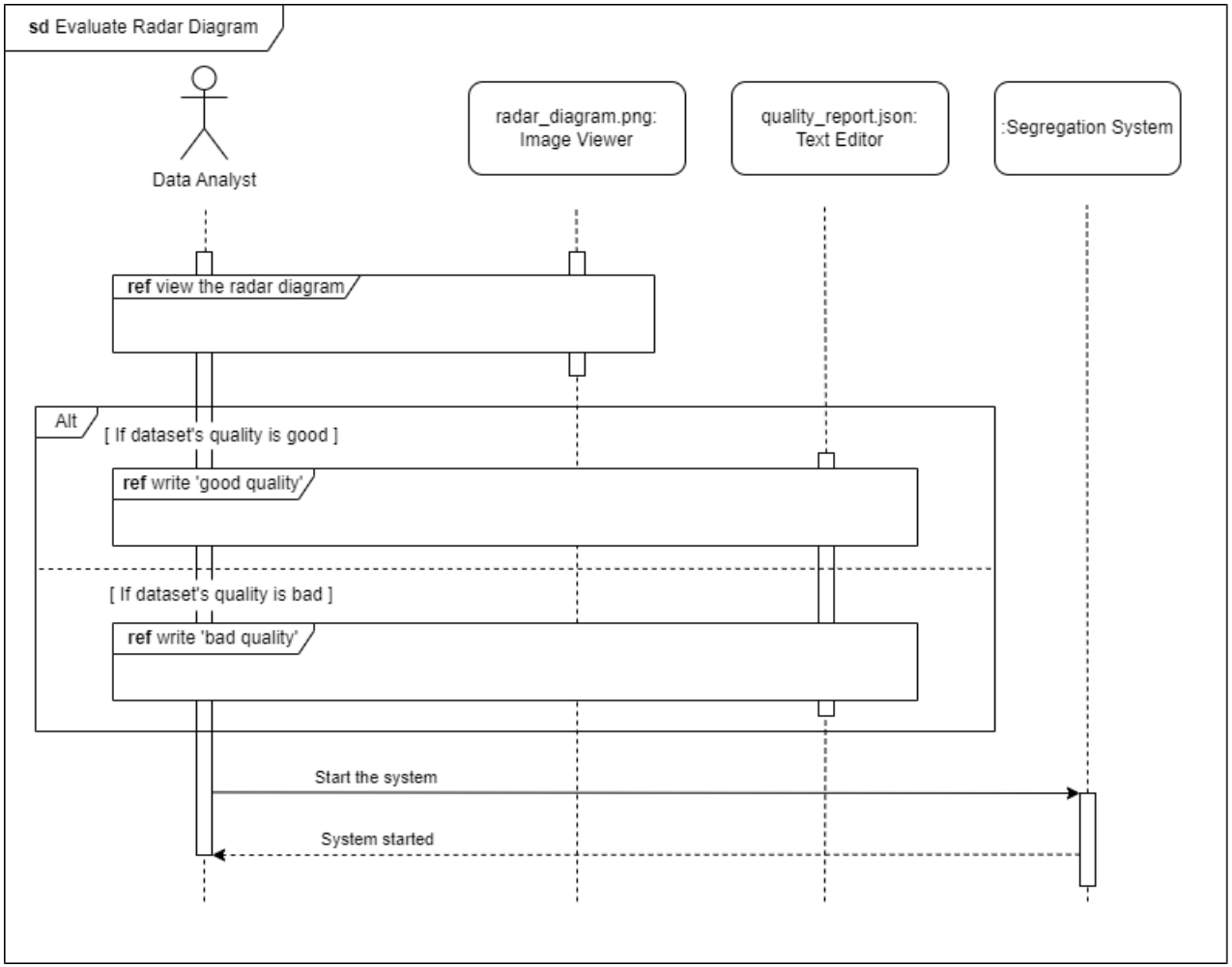
segregation_system_config.json

```
{
    "operative_mode" : "collecting_op_mode",
    "user_id" : 0,
    "testing_mode": "off",
    "endpoint_ip": "127.0.0.1",
    "endpoint_port": "4000",
    "db_name": "segregation.db",
    "collecting_threshold": 300,
    "training_set_size": 0.75,
    "valiation_set_size": 0.15,
    "testing_set_size": 0.10
}
```



balancing_report.json


```
{
    "alpha: 25,
    "beta": 21,
    "delta": 22.
    "tetha": 20,
    "evaluation": "balanced"
}
```



quality_report.json

```
{
    "evaluation": "good quality"
}
```

Development System (Matteo Abaterusso)

development_system_config.json

```
{
    "ip_endpoint": "172.16.0.10",
    "port_endpoint": 5000,
    "operational_mode": "waiting_for_dataset",
    "initial_number_of_generations": 100,
    "number_of_hidden_layer_range": [1, 8],
    "number_of_hidden_neurons_range": [1, 64],
    "validation_error_threshold": 0.3,
    "test_error_threshold": 0.3,
    "testing_mode": false
}
```

number_of_generations.json

```
{
    "number_of_generations": 10
}
```

early_training_report.json

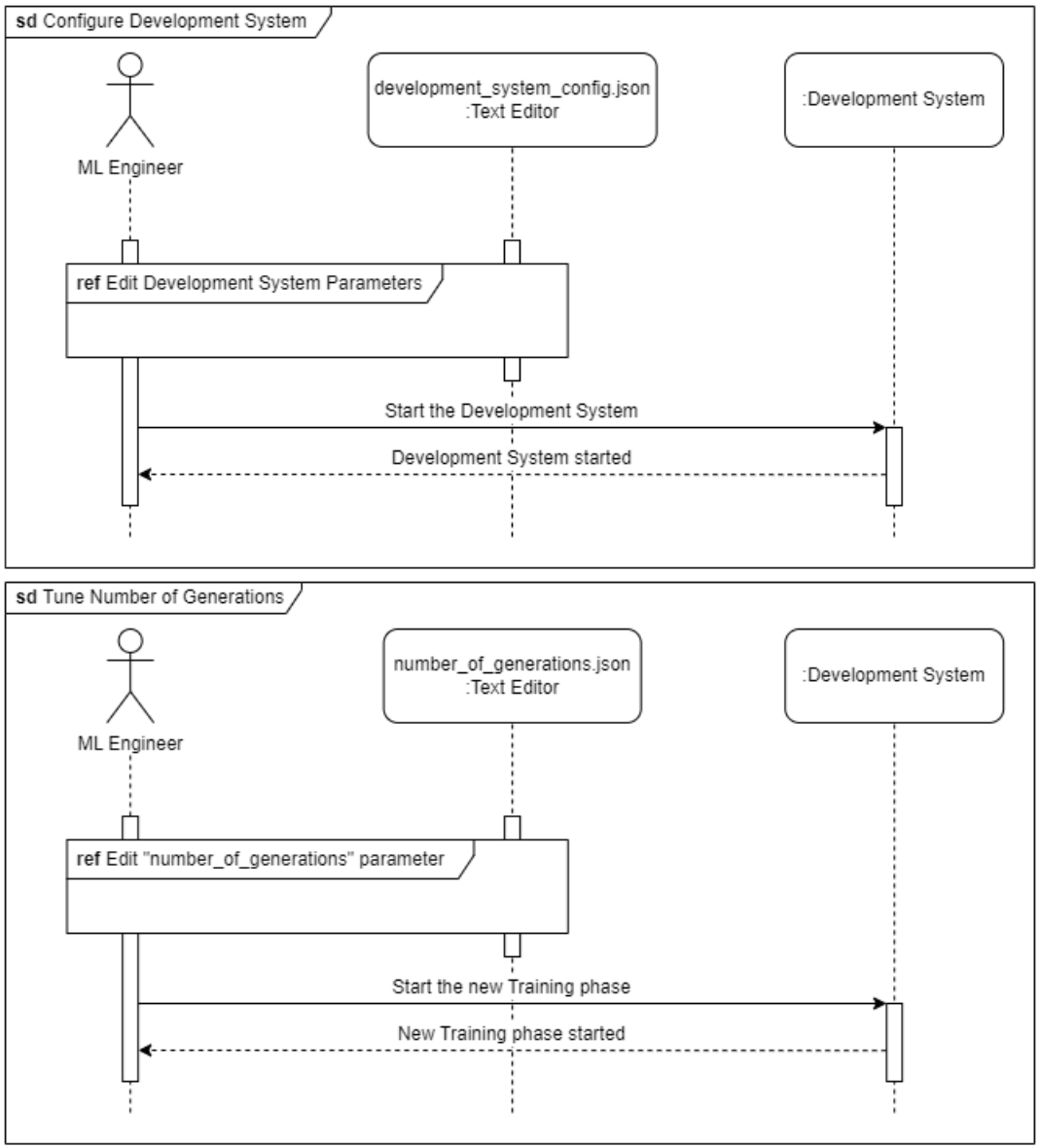
```
{
    "number_of_generations": 100,
    "hidden_layer_sizes": [32, 16, 8, 4],
    "training_error": 0.4,
    "valid_generations": true
}
```

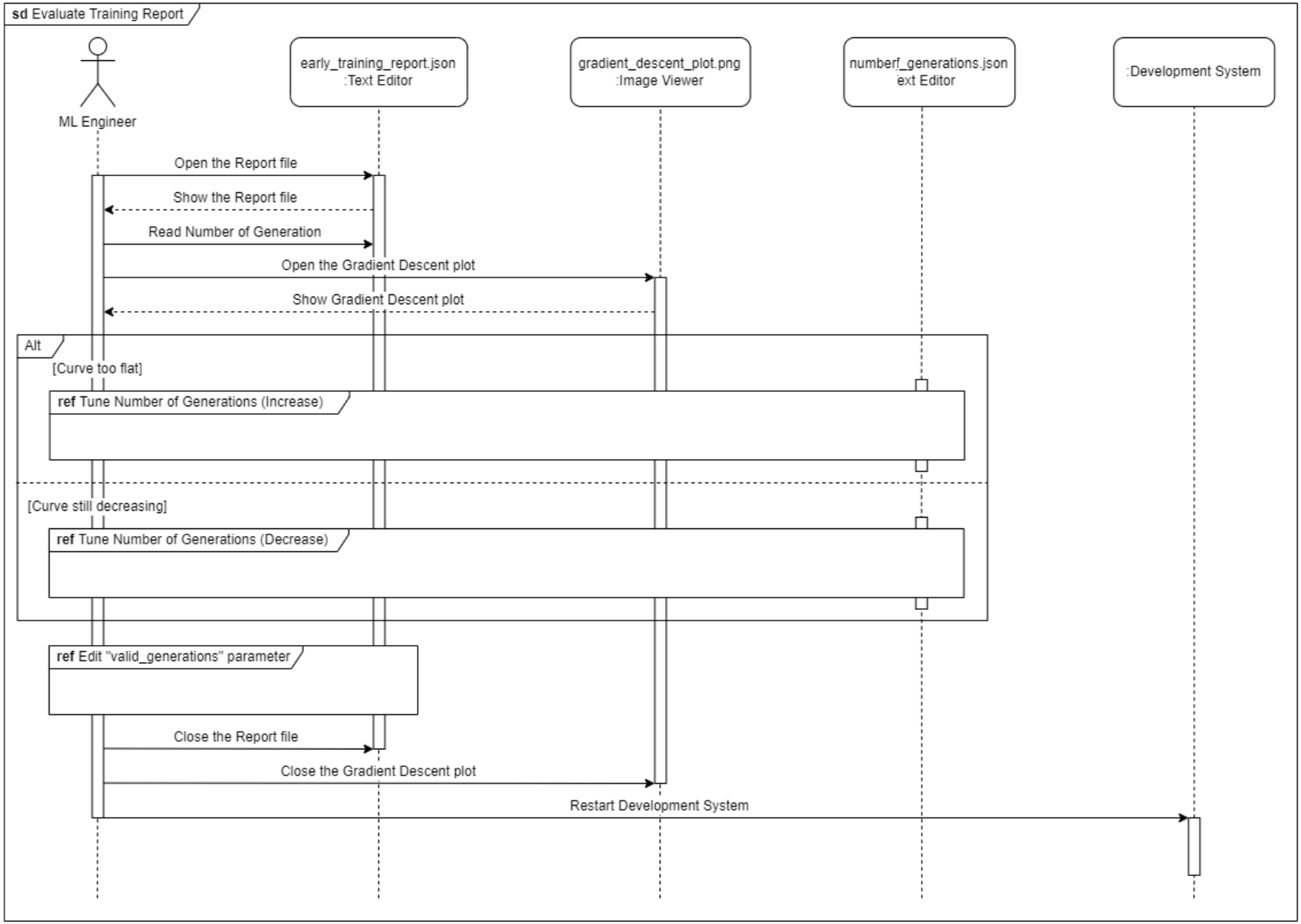
top_five_classifiers_report.json

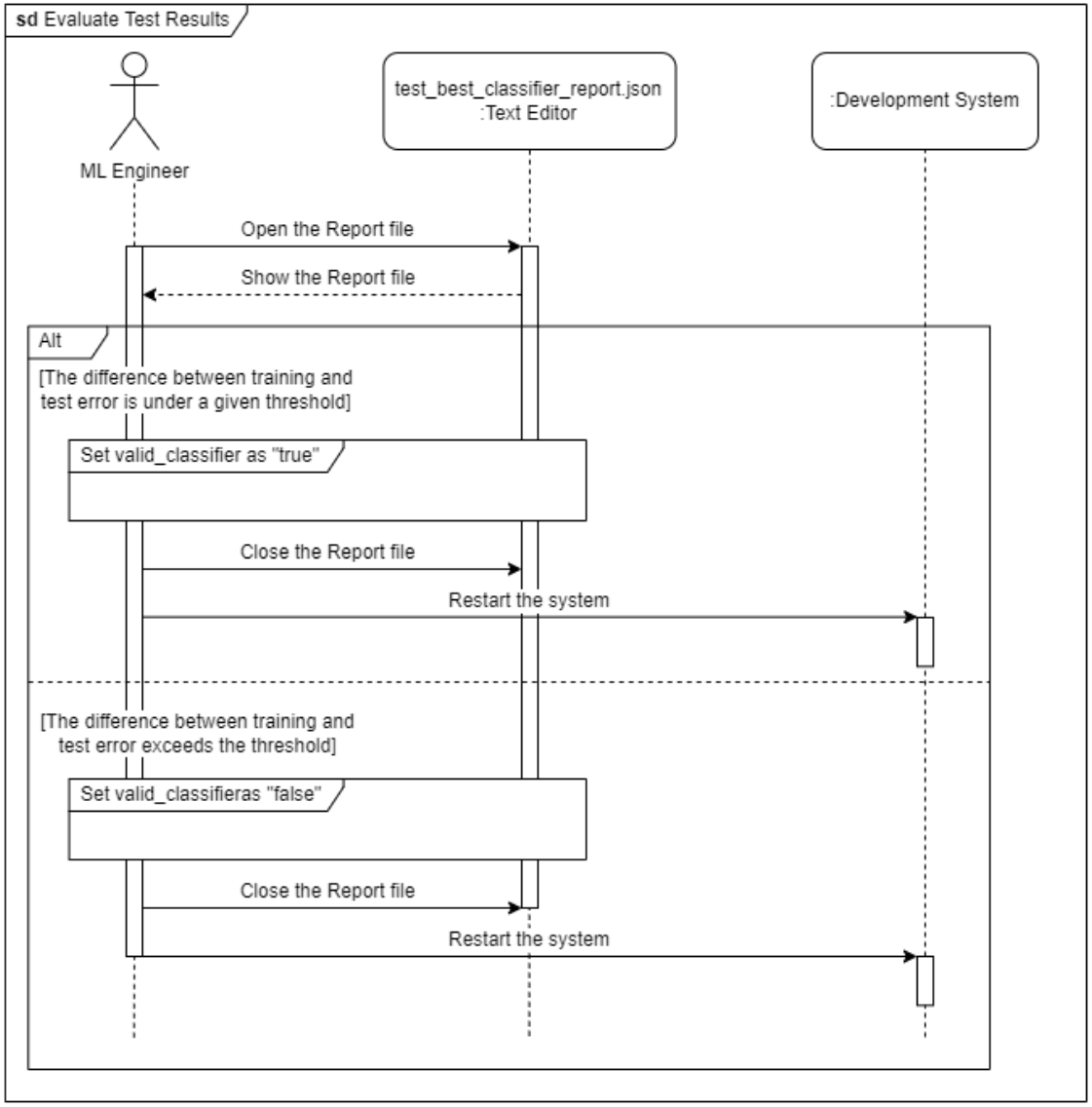
```
{
    "number_of_generations": 10,
    "validation_error_threshold": 0.3,
    "classifiers" : [
        {
            "uuid": 9,
            "hidden_layer_sizes": [16, 8],
            "training_error": 0.67,
            "validation_error": 0.4,
            "actual_best": false
        },
        {
            "uuid": 4,
            "hidden_layer_sizes": [8],
            "training_error": 0.31,
            "validation_error": 0.35,
            "actual_best": false
        },
        ...
    ]
}
```

test_best_classifier_report.json

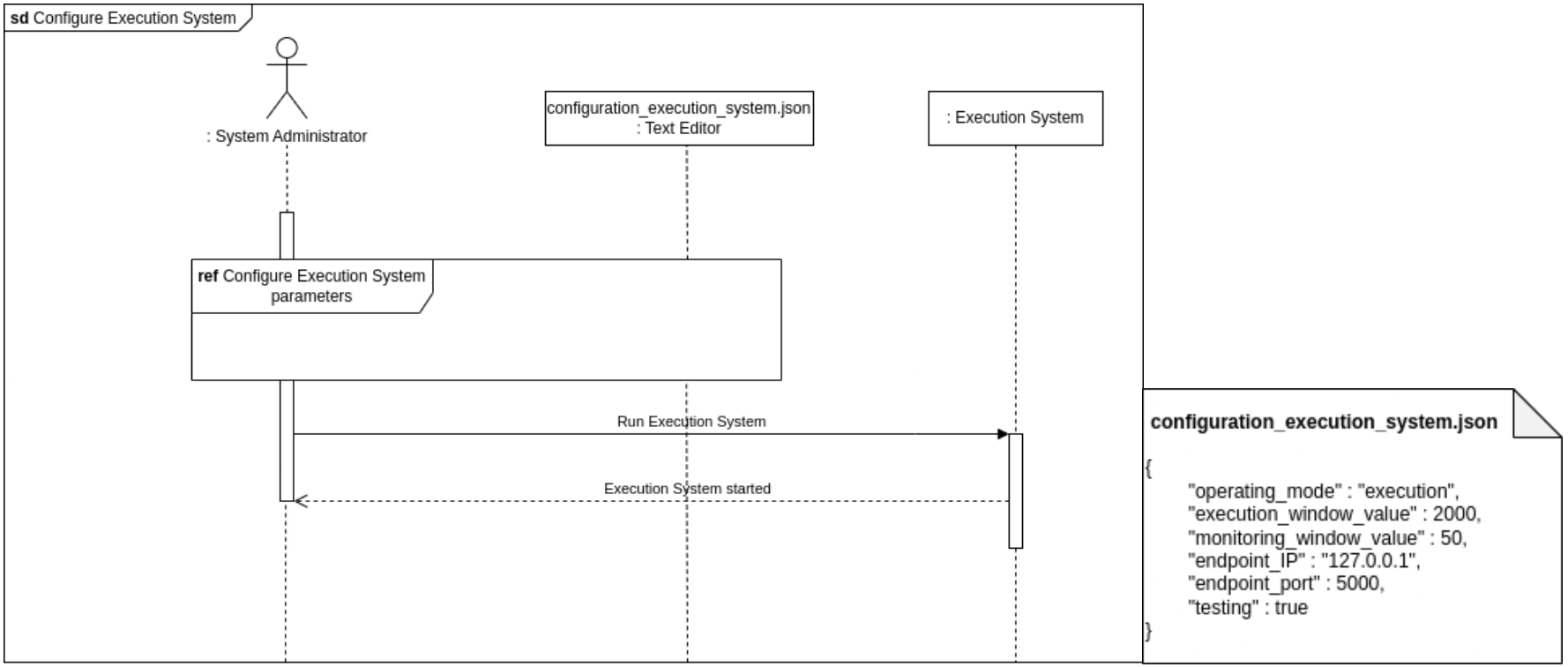
```
{
    "number_of_generations": 200,
    "hidden_layer_sizes": [16, 8, 4, 2],
    "test_error_threshold": 0.3,
    "training_error": 0.44,
    "test_error": 0.2,
    "valid_classifier": true
}
```


sd Configure Development System
ML Engineer
development_system_config.json
:Text Editor
:Development System
ref Edit Development System Parameters
Start the Development System
Development System started
sd Tune Number of Generations
ML Engineer
number_of_generations.json
:Text Editor
:Development System
ref Edit "number_of_generations" parameter
Start the new Training phase
New Training phase started

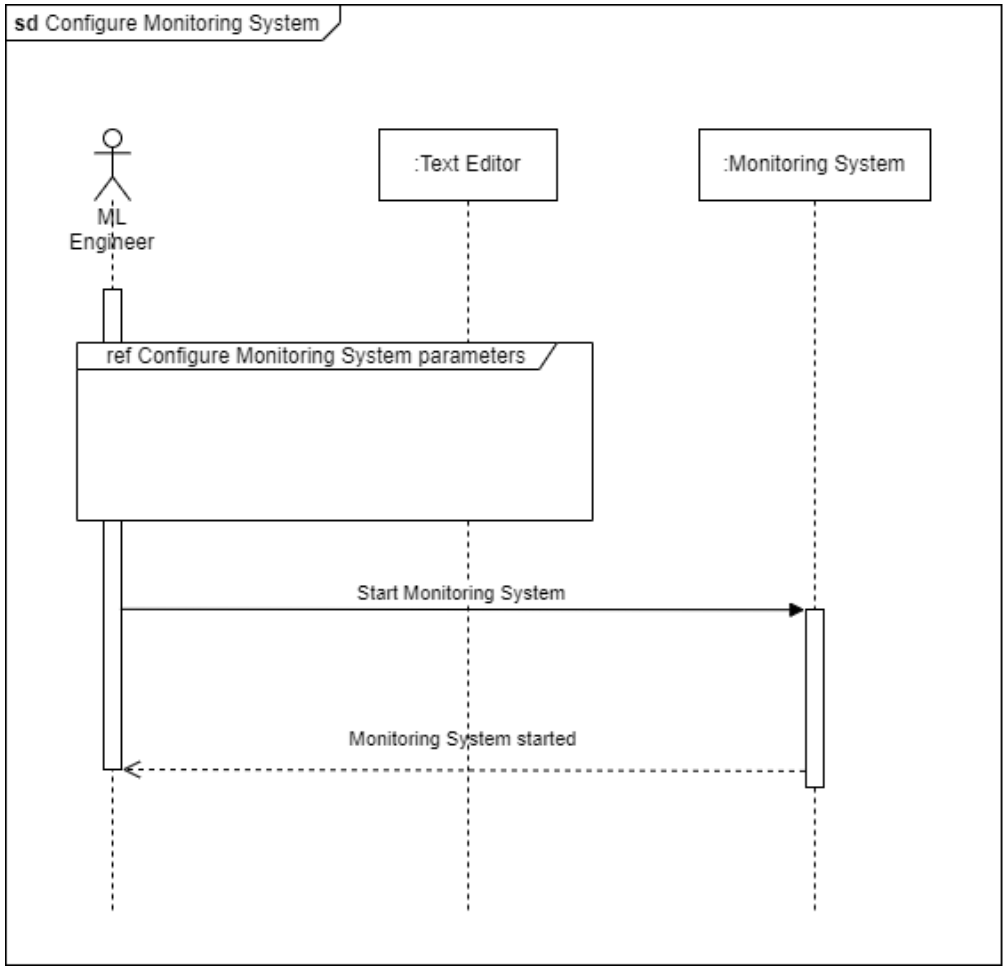

sd Evaluate Training Report
ML Engineer
early_training_report.json
:Text Editor
gradient_descent_plot.png
:Image Viewer
numberf_generations.json
ext Editor
:Development System
Open the Report file
Show the Report file
Read Number of Generation
Open the Gradient Descent plot
Show Gradient Descent plot
Alt
[Curve too flat]
ref Tune Number of Generations (Increase)
[Curve still decreasing]
ref Tune Number of Generations (Decrease)
ref Edit "valid_generations" parameter
Close the Report file
Close the Gradient Descent plot
Restart Development System


sd Evaluate Test Results
ML Engineer
test_best_classifier_report.json
:Text Editor
:Development System
Open the Report file
Show the Report file
Alt
[The difference between training and
test error is under a given threshold]
Set valid_classifier as "true"
Close the Report file
Restart the system
[The difference between training and
test error exceeds the threshold]
Set valid_classifieras "false"
Close the Report file
Restart the system

Execution System (Francesco Martoccia)

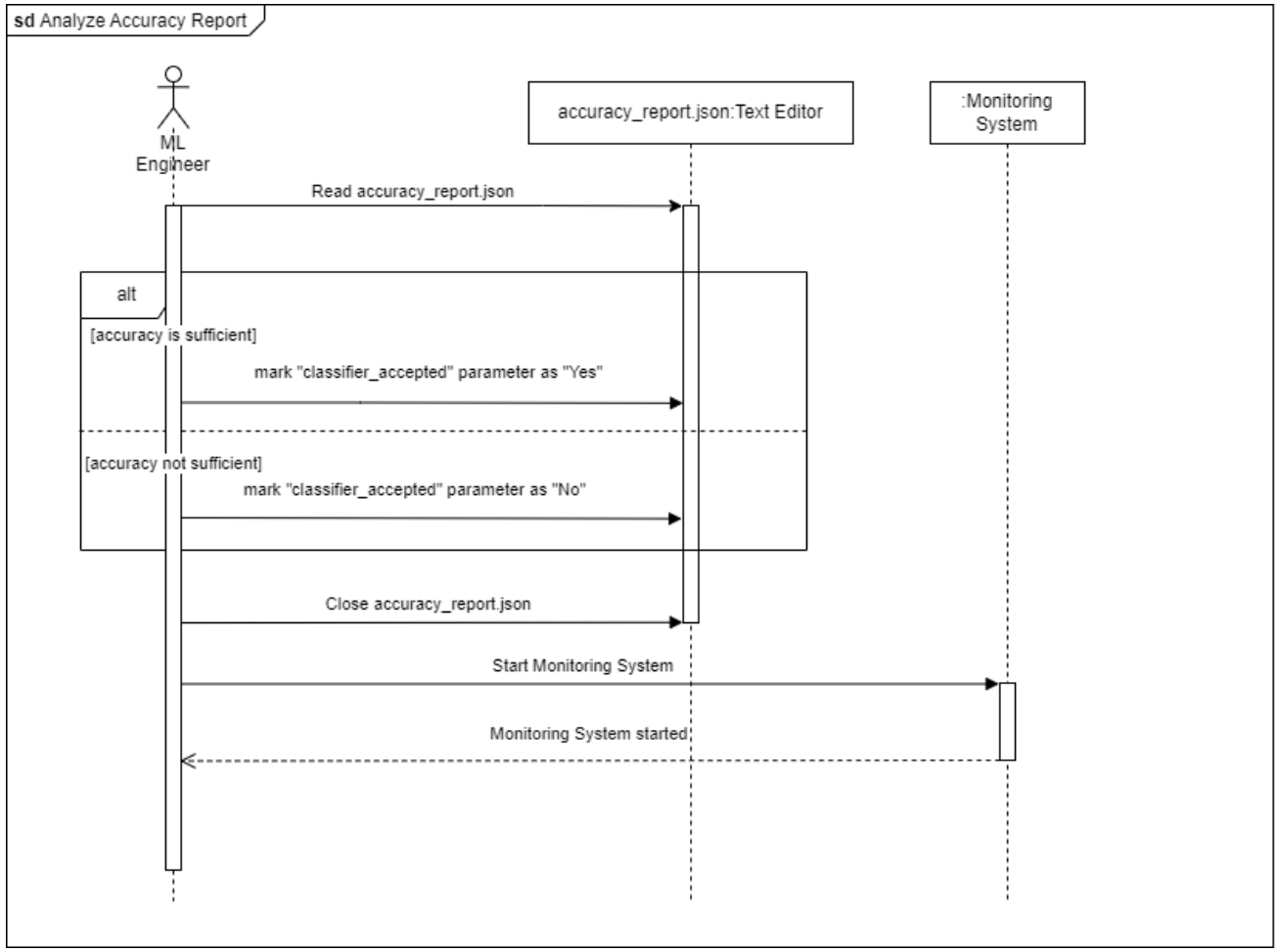


Monitoring System (Luca Tartaglia)



monitoring_system_config.json

```
{
"labels_threshold" : 50,
"max_errors_tolerated": 5
"testing_mode" : "false",
}
```



monitoring_system_config.json

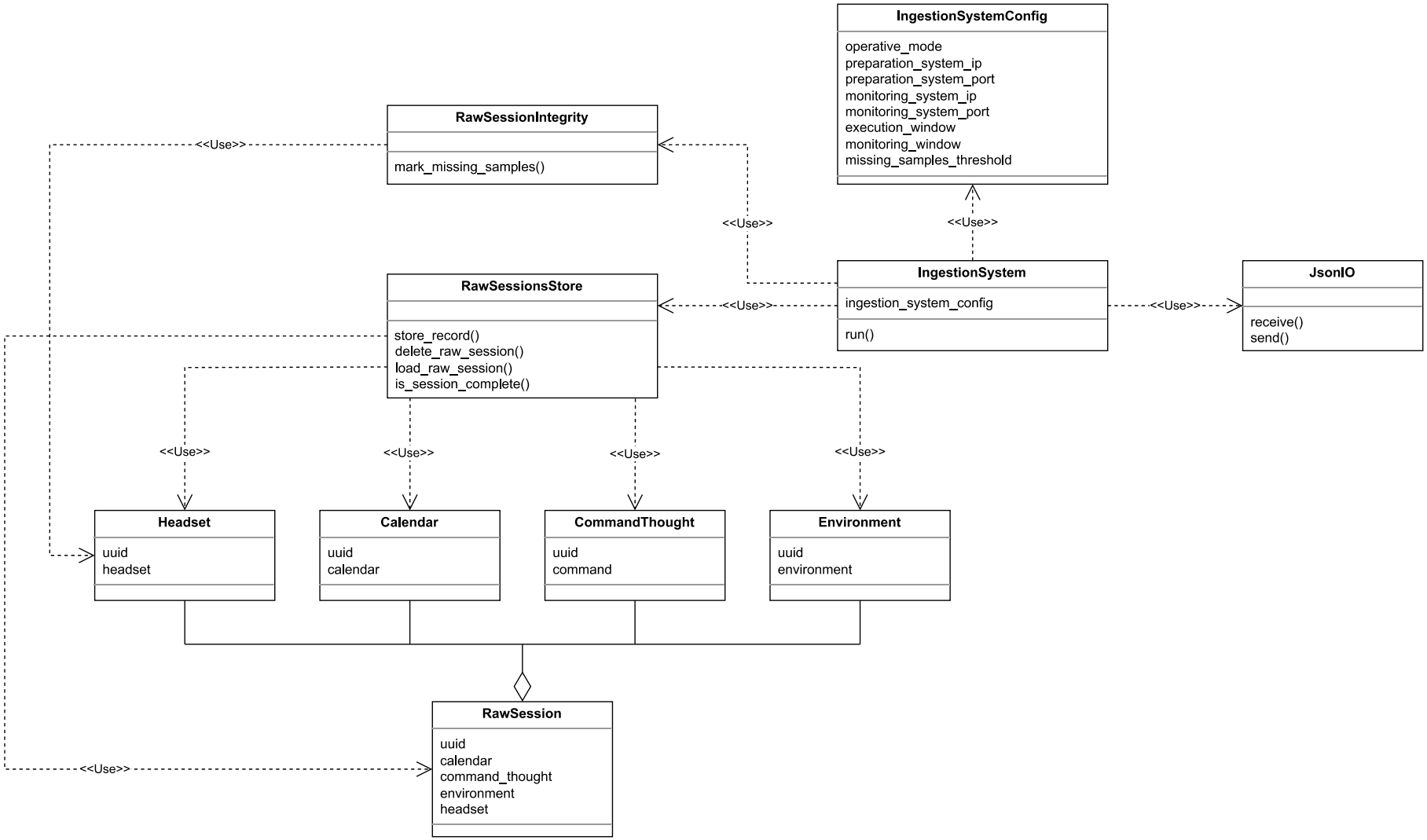
```
{
"labels_threshold" : 50,
"max_errors_tolerated": 5
"testing_mode" : "false",
}
```

accuracy_report.json

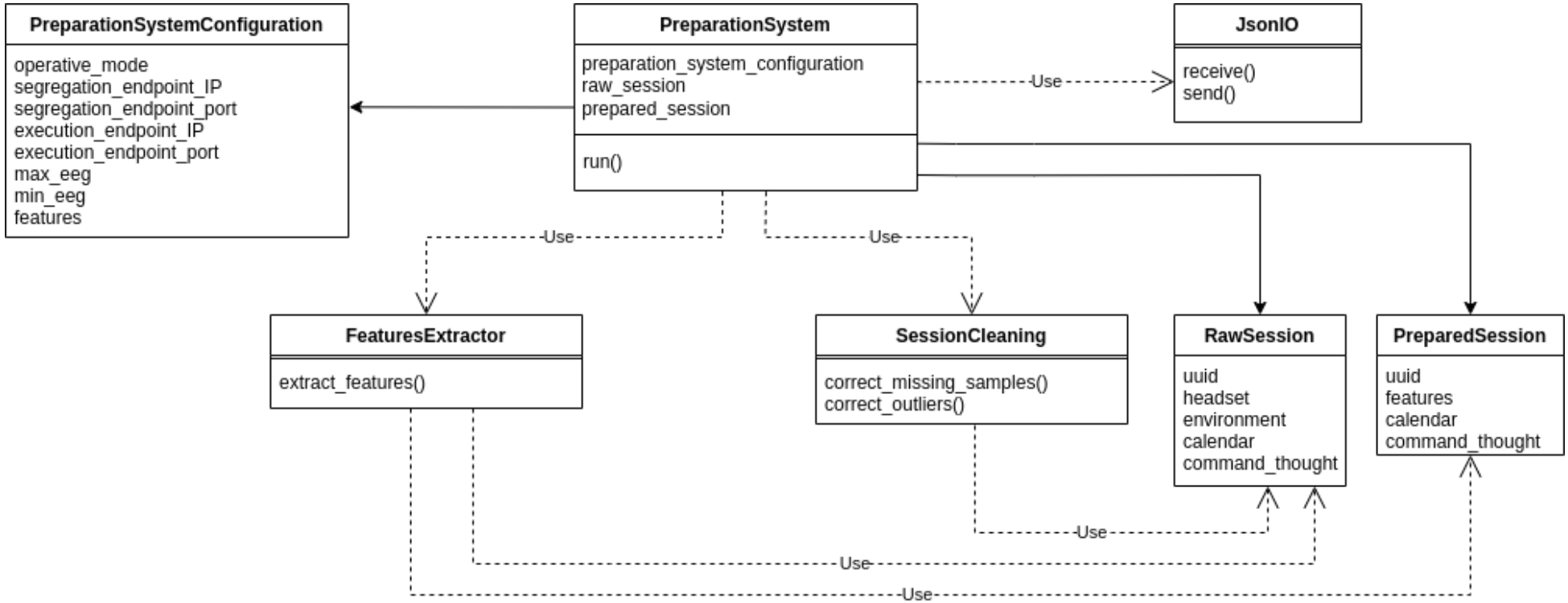
```
{
"evaluated_labels" : 50,
"max_errors_tolerated" : 5,
"errors" : 10,
"accuracy" : 0.80,
"classifier_accepted" : " "
}
```

Class Diagrams (Analysis)

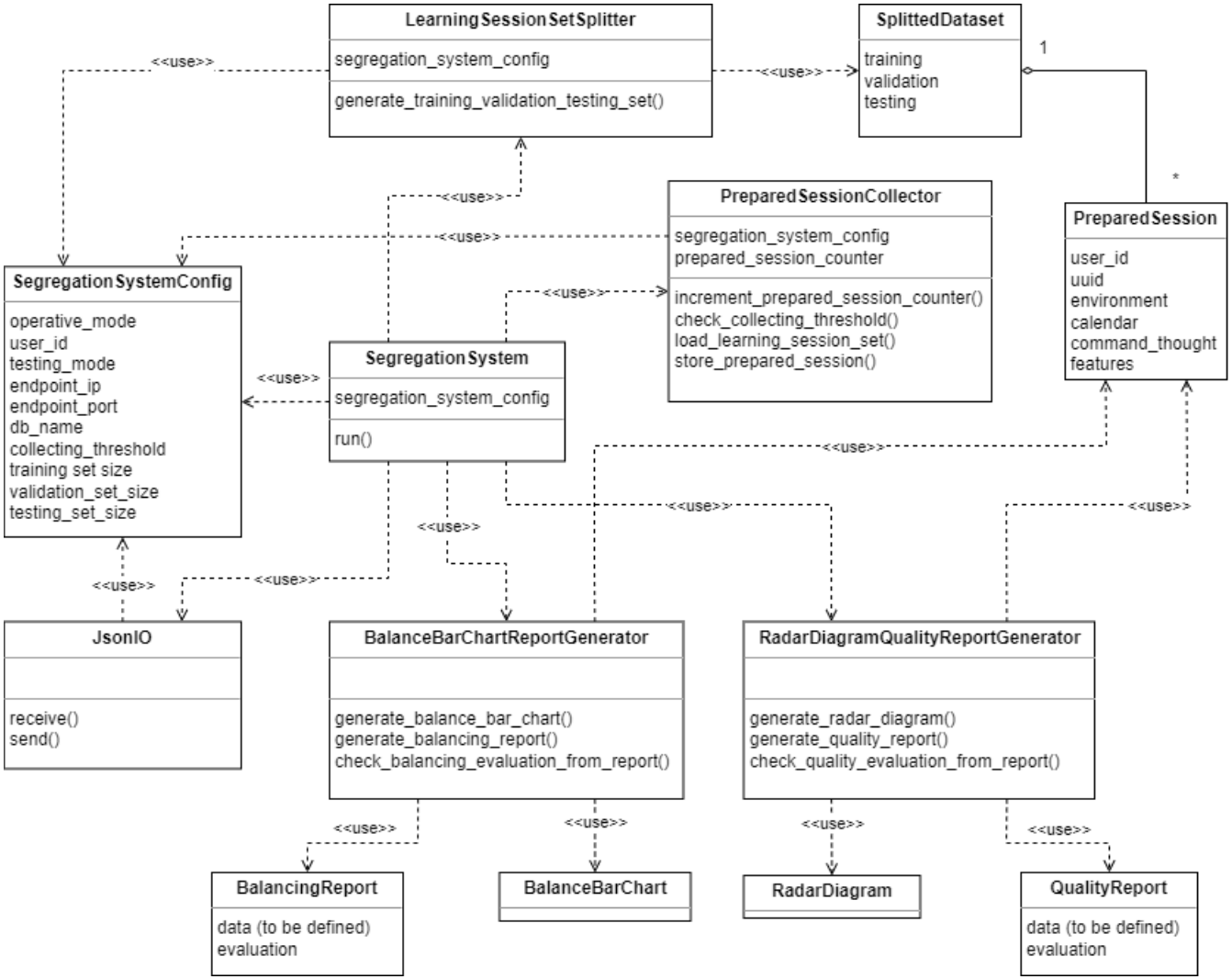
Ingestion System (Biagio Cornacchia)



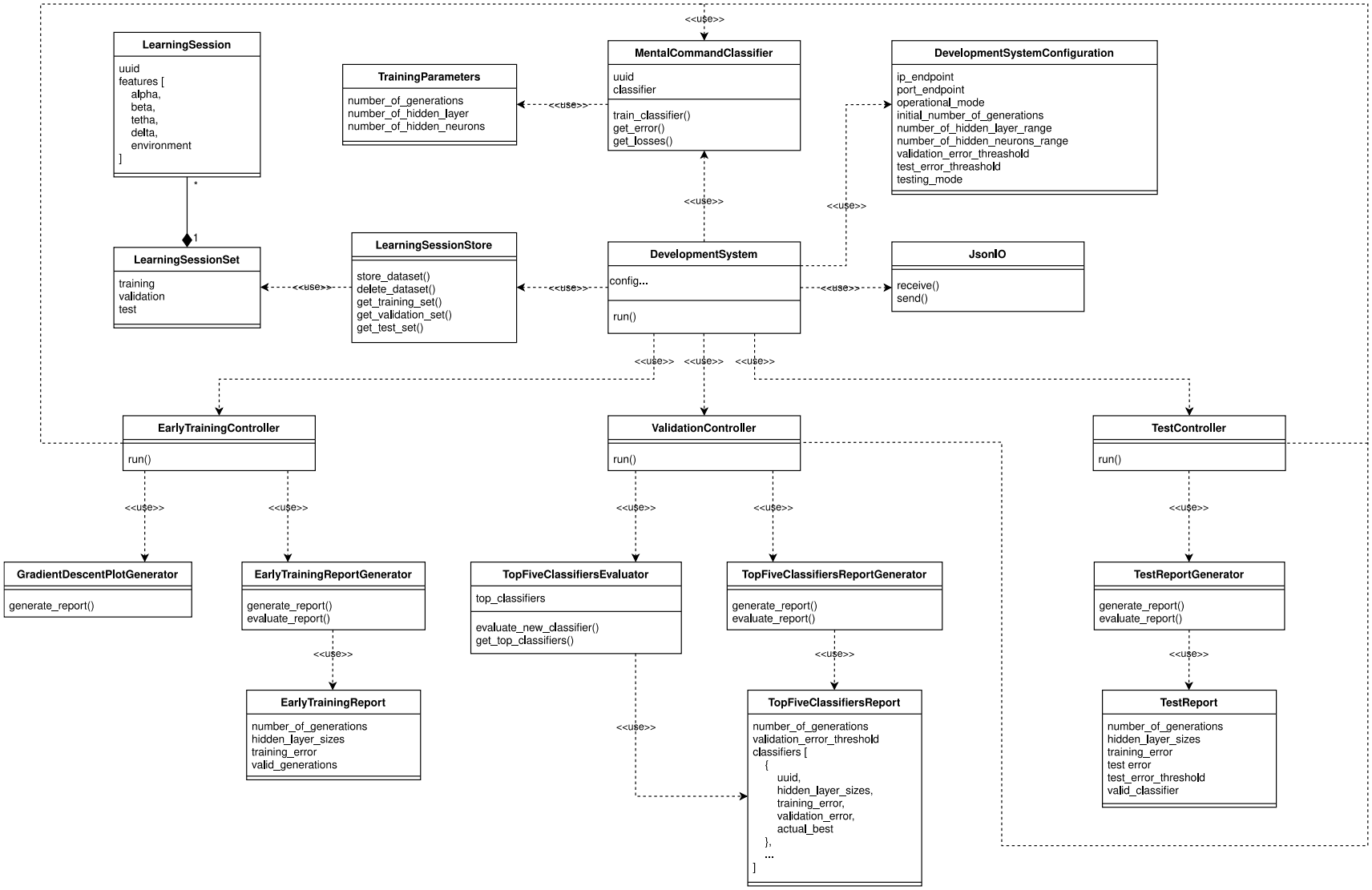
Preparation System (Salvatore Lombardi)



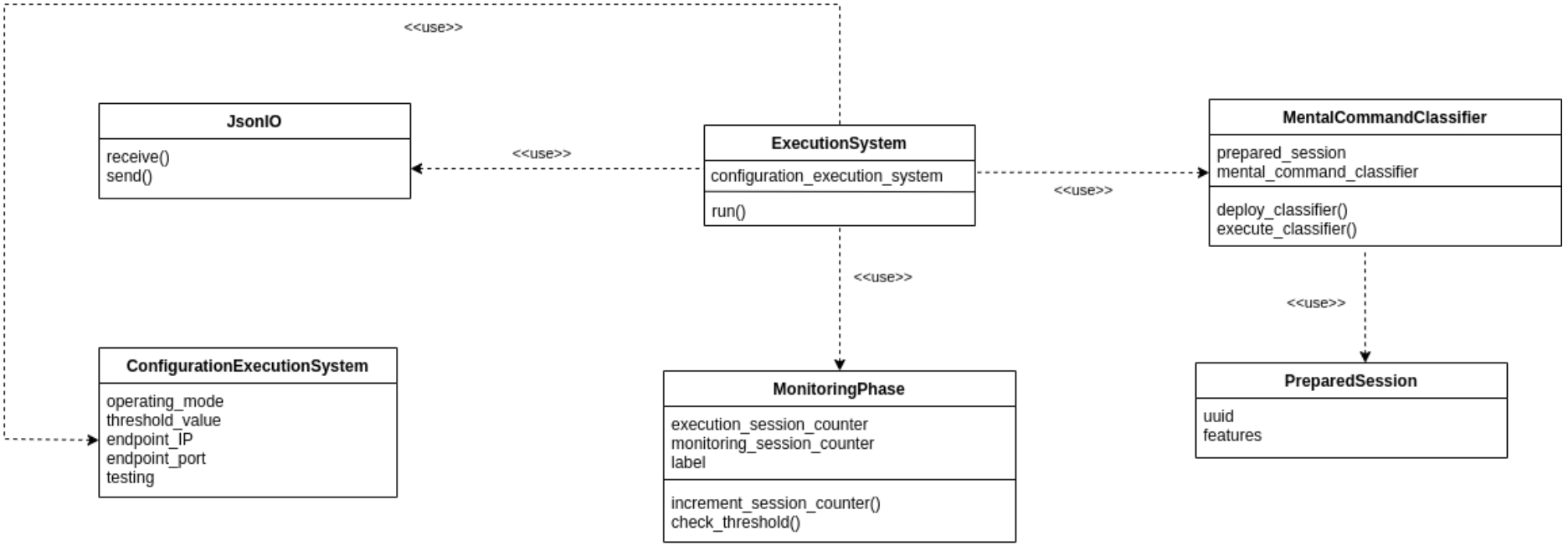
Segregation System (Gianluca Gemini)



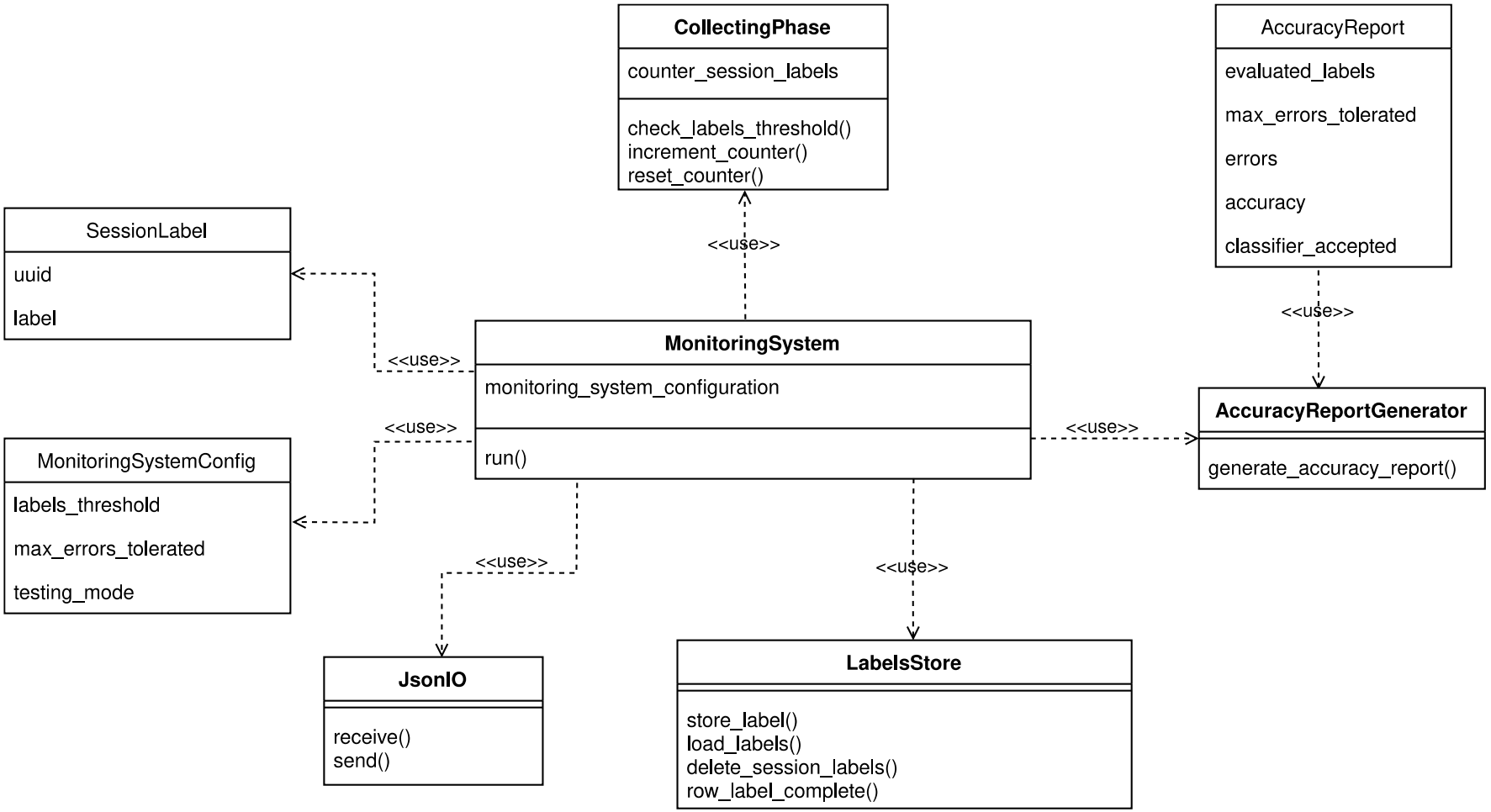
Development System (Matteo Abaterusso)



Execution System (Francesco Martoccia)

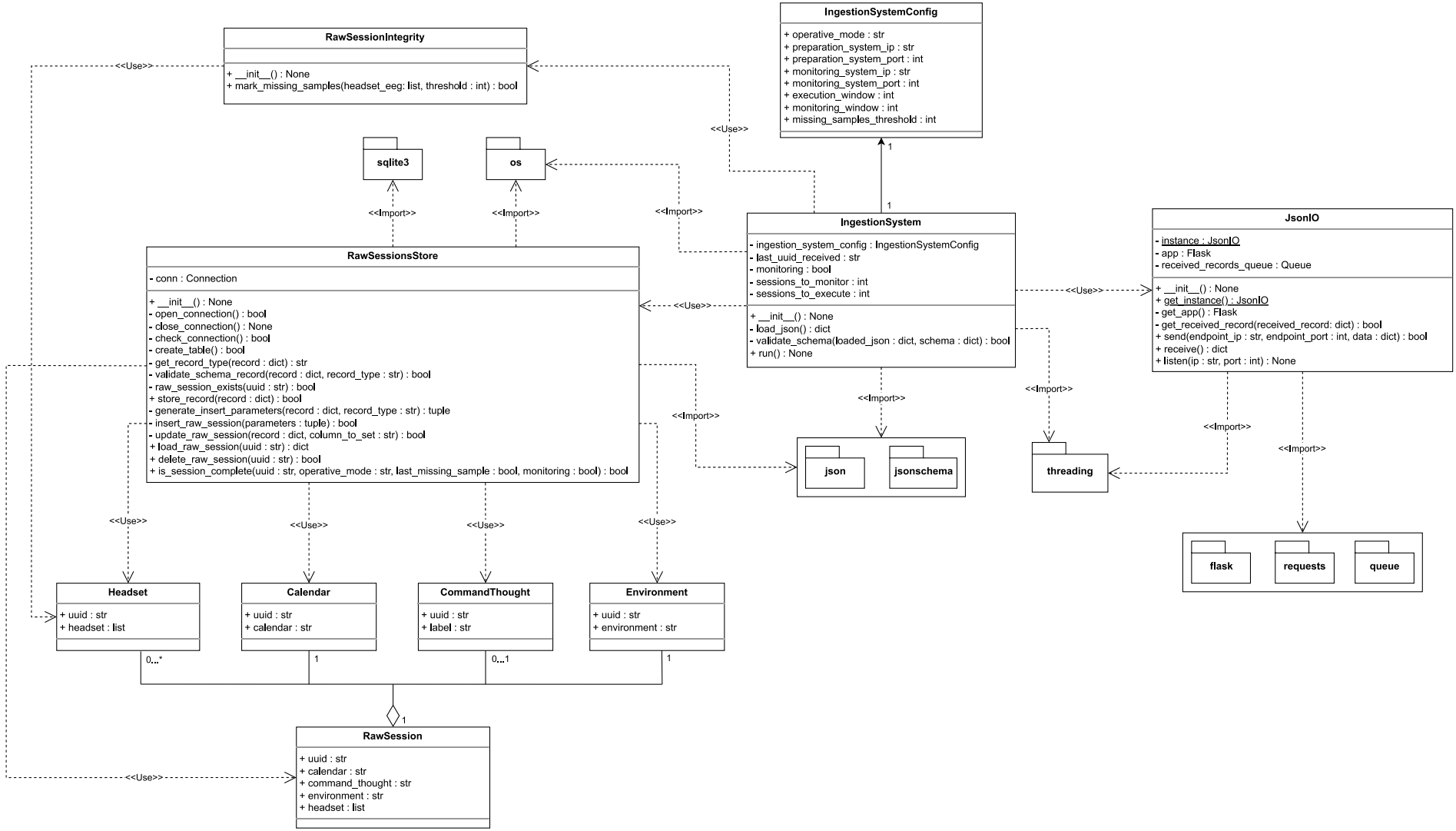


Monitoring System (Luca Tartaglia)

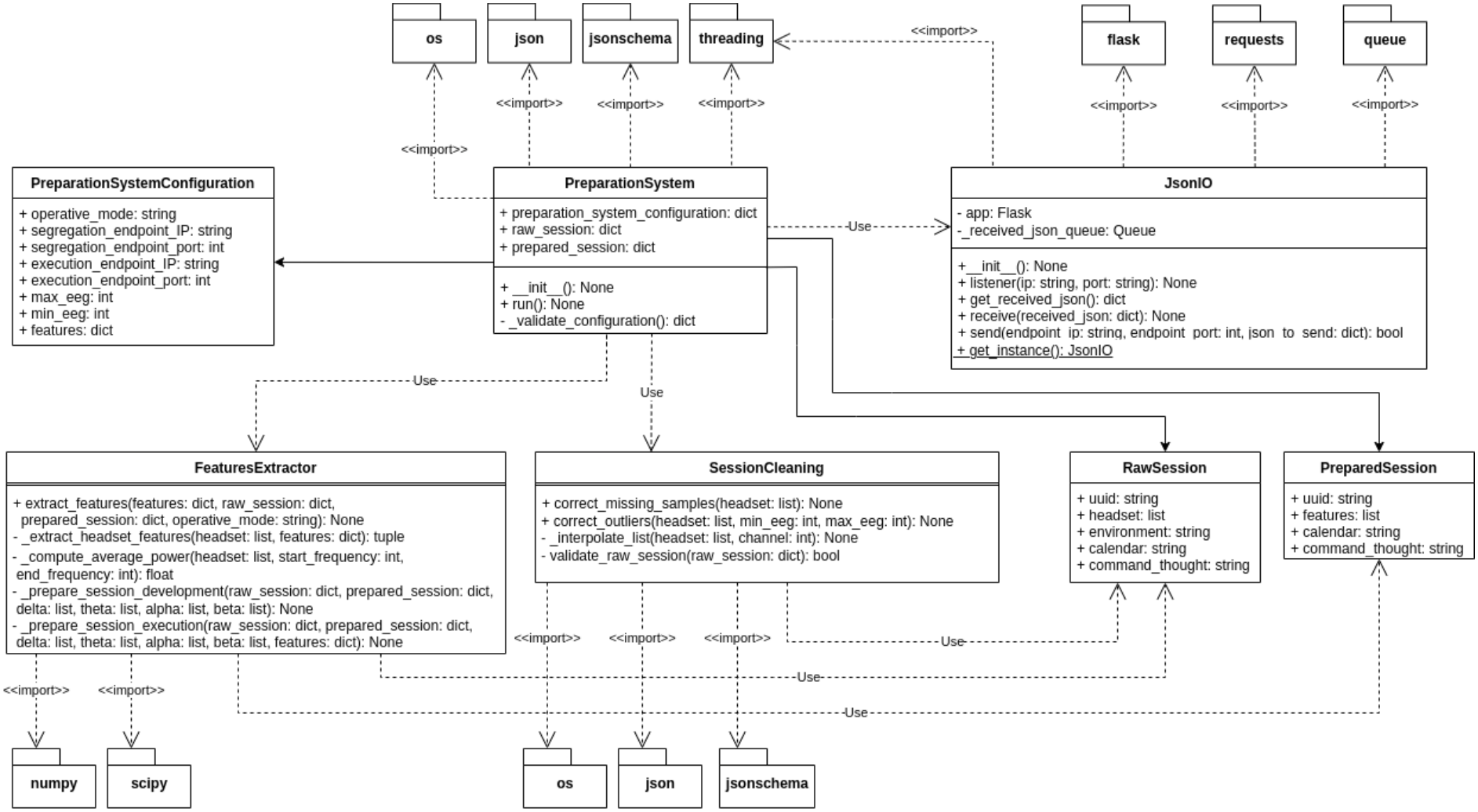


Class Diagrams (Design)

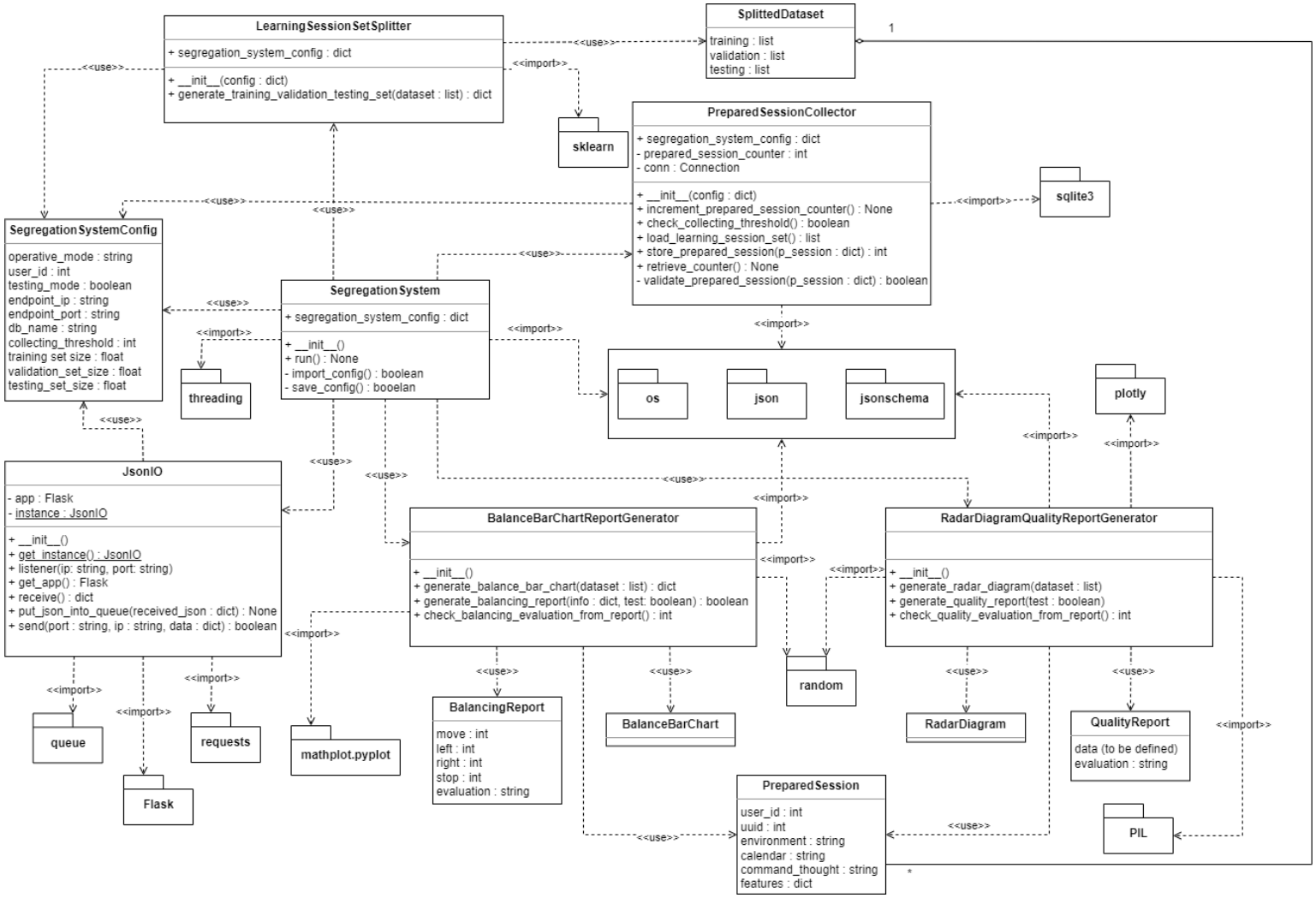
Ingestion System (Biagio Cornacchia)



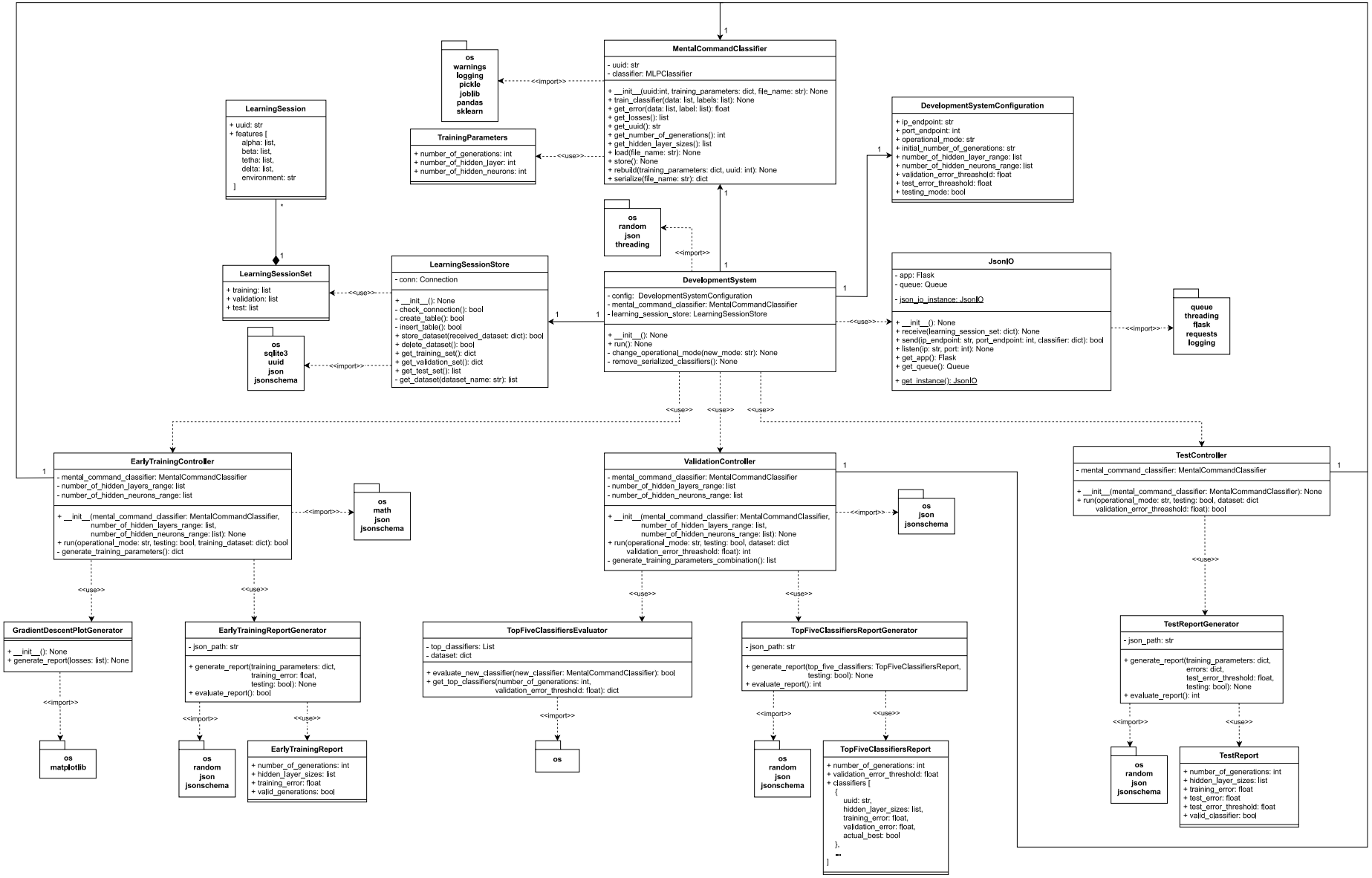
Preparation System (Salvatore Lombardi)



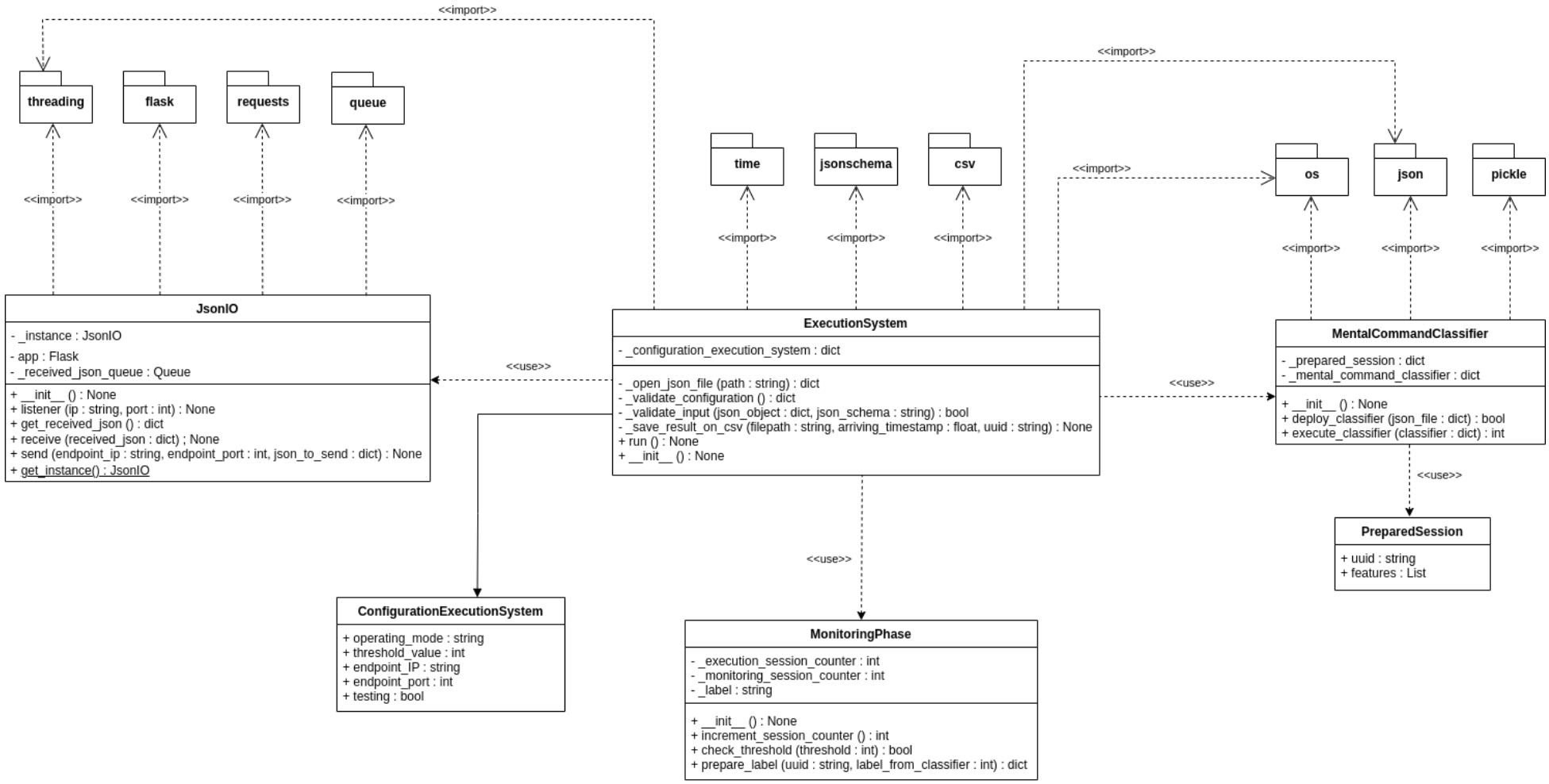
Segregation System (Gianluca Gemini)



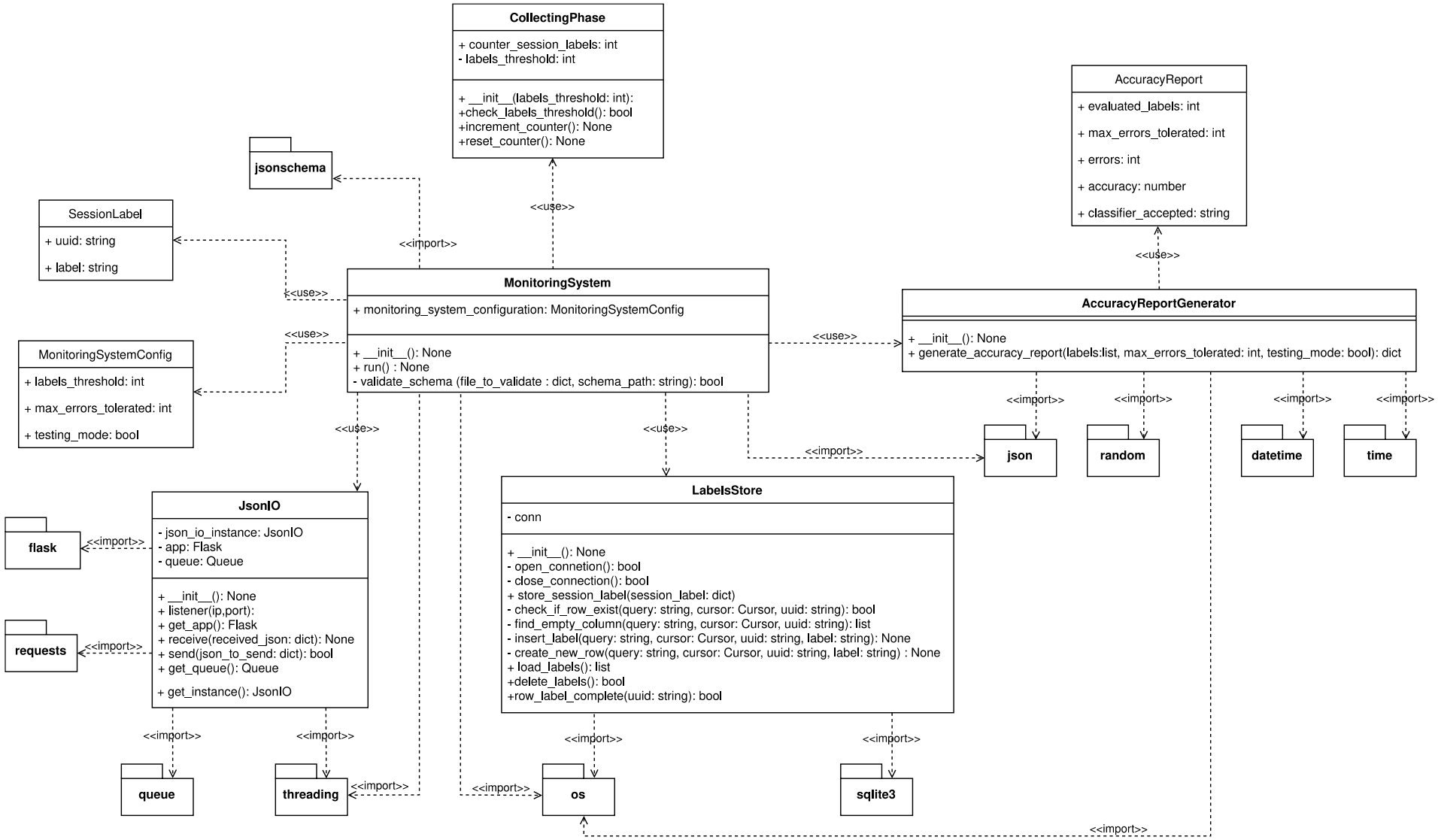
Development System (Matteo Abaterusso)



Execution System (Francesco Martoccia)

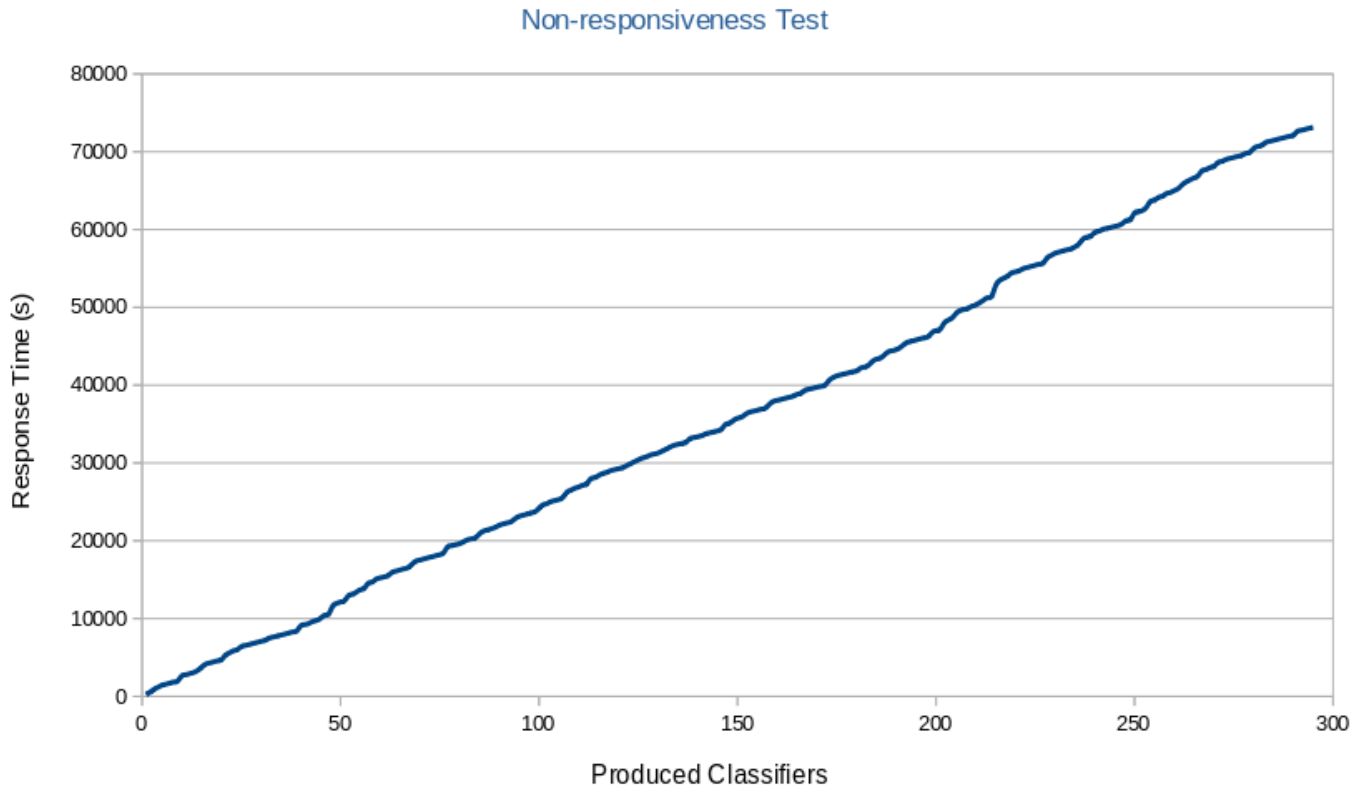


Monitoring System (Luca Tartaglia)



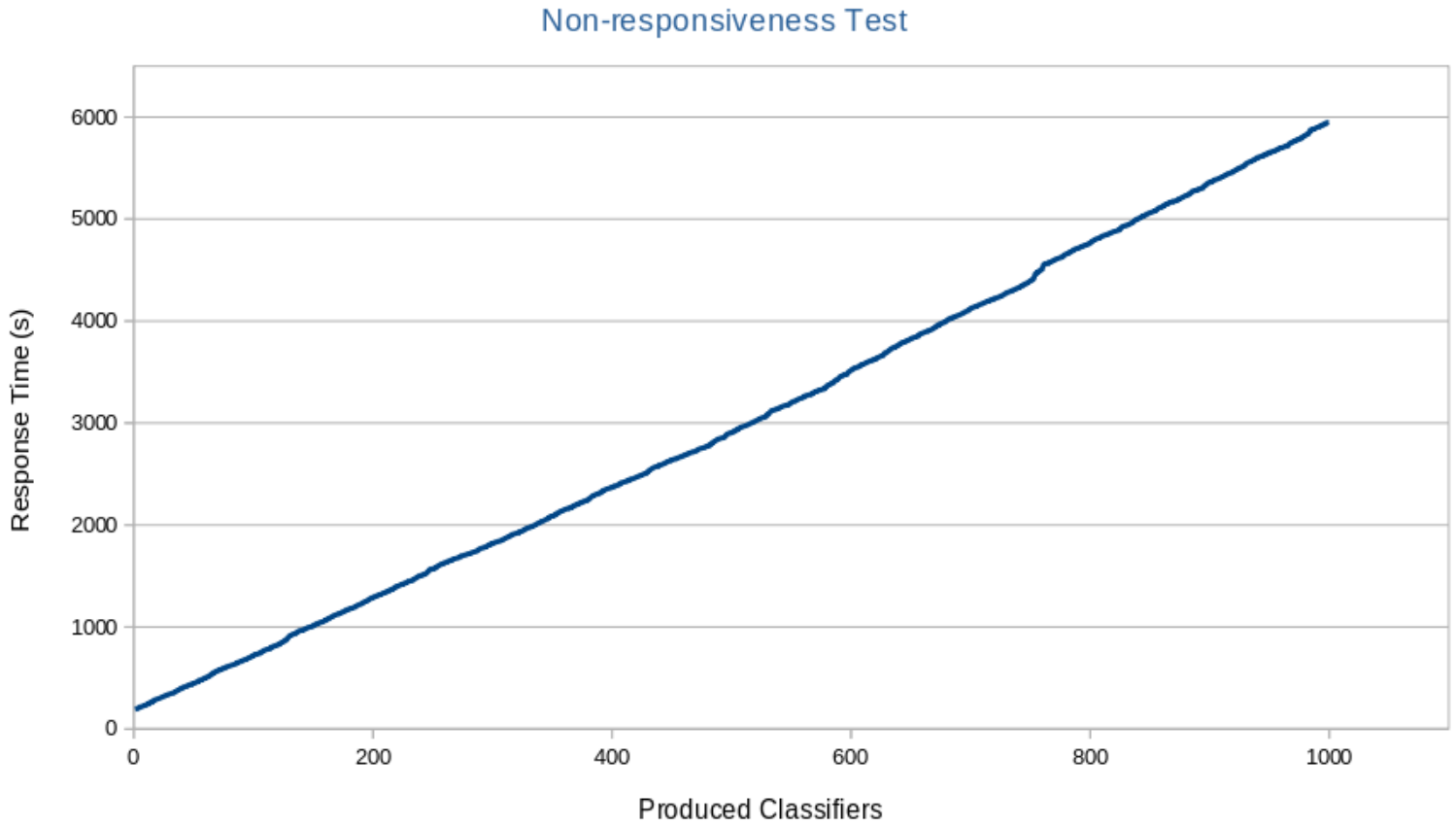
Testing (Non-Responsiveness and Non-Elasticity)

The non-responsiveness test has been performed for the development phase. It consists in producing 300 consecutive classifiers, each one trained using datasets of 100 sessions. The evaluated metric is the time between the sending of the first data and the production of the i-th classifier.



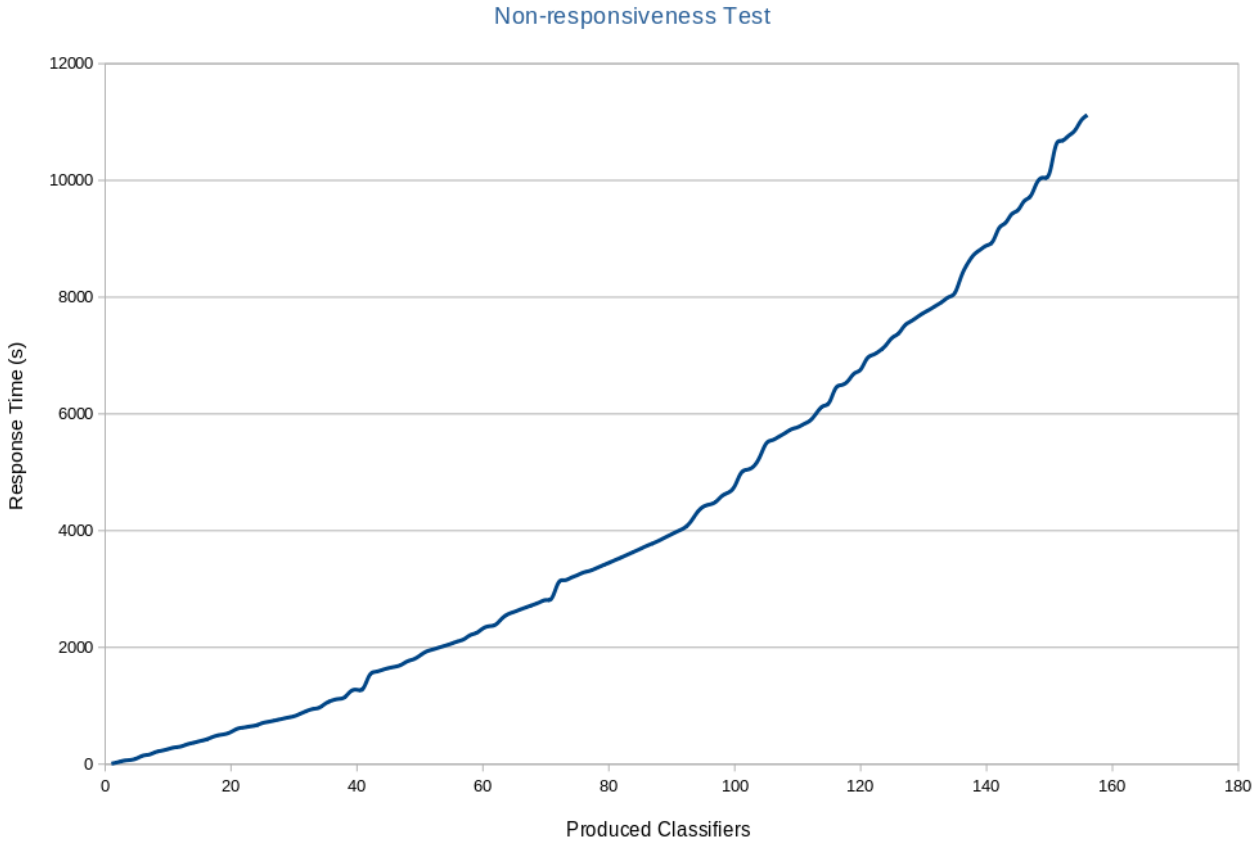
The linear trend in the graph is caused by the arrival time of the development system that is greater than the service time, so there is no queueing. In fact, the results show that it takes an average of four minutes to produce a classifier, which is mainly used by the segregation system to accumulate the 100 sessions that make up a dataset.

To stress the development system a new test has been performed, in which the segregation system produces a new dataset as each new session arrives. Each dataset consists of the new session received plus the 99 sessions already accumulated. The purpose of this test is to evaluate the response of the development system as the queue increases. The metric evaluated is the same as in the previous test.



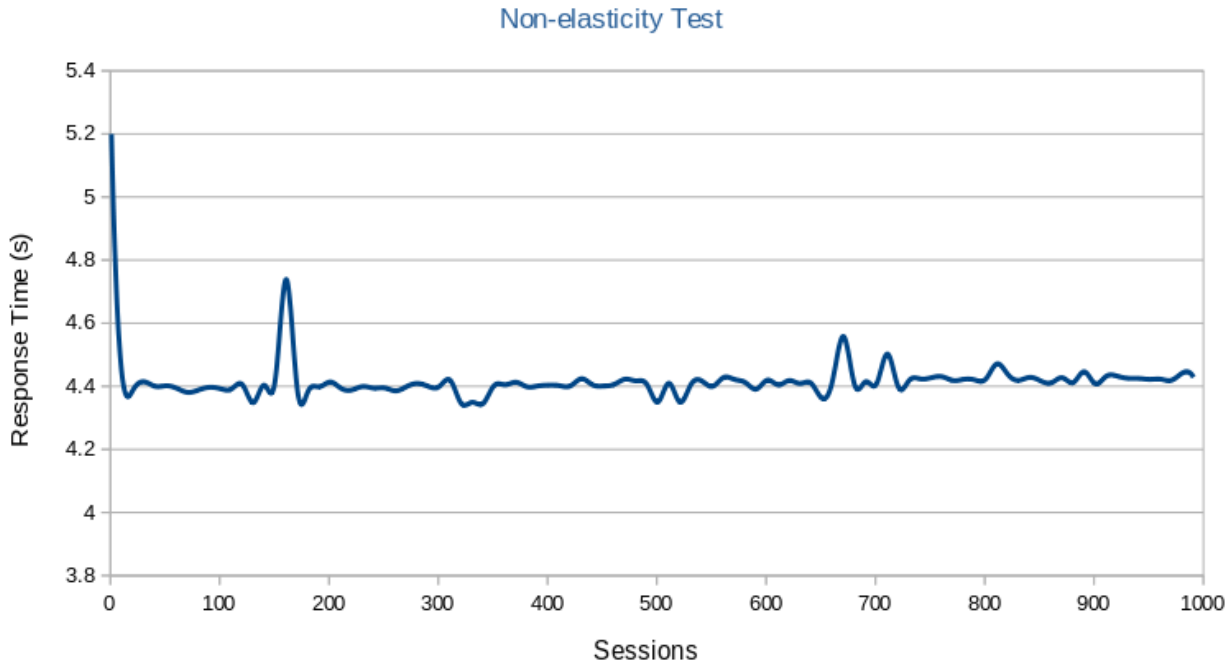
Again, the trend of the graph is linear. This is due to the fact that, although the development system's queue continues to grow, its service time remains constant.

The following non-responsiveness test has been performed again for the development phase. In this case each dataset used to train the classifier has been increased by 30 sessions with respect to the previous one. The evaluated metric is the time between the sending of the first data and the production of the i-th classifier.



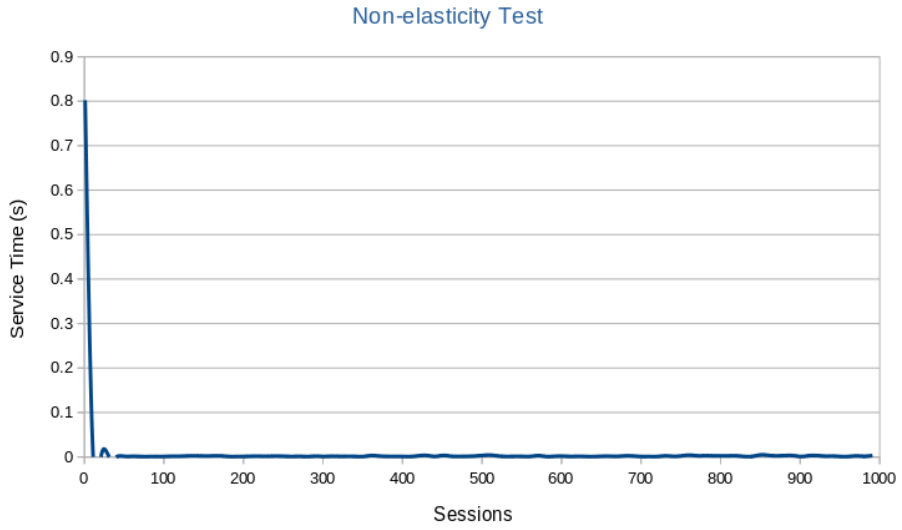
It is possible to see from the graph a nonlinear trend due to the increase in the time required to develop the classifiers as the size of the dataset grows.

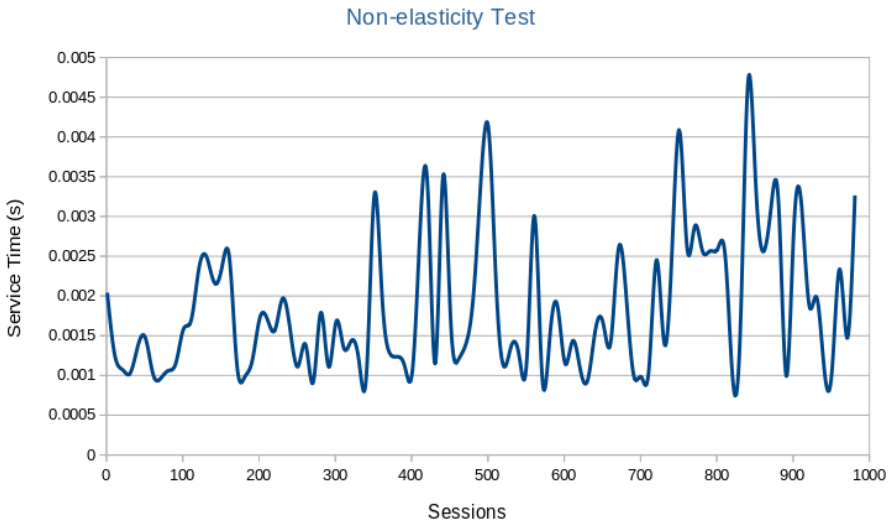
The non-elasticity test has been performed for the execution phase. It represents the time from the moment when the ingestion system produces a session to the moment when the execution system generates the command.



The results obtained show a constant response time of 4 seconds, which is not acceptable in a real case. This delay is due to the communication time between the modules. Therefore, an embedded solution would be preferable rather than a cloud one.

The following graphs refer to the same test as the previous one, but the metric taken into account is service time. The latter refers to the time it takes the classifier to produce a command.





In the graph on the left it is possible to see a complete overview of the test results. Instead, the graph on the right shows the results in more detail without considering the first two outliers. The outliers are due to the initial loading of the classifier into memory.

In conclusion, the service time is acceptable in that it remains below 5 milliseconds.

Testing (Non-Resiliency)

Ingestion System

ID	Input Error	Consequence	Score
I1	Missing records (calendar, environment) Missing label in development or monitoring phase	The raw session is not complete so it is discarded.	4
I2	Missing headset EEG data (under threshold)	The number of missing channels satisfies the threshold then the session is marked and sent to the Preparation System (solved by the Preparation System).	1
I3	Missing headset EEG data (over threshold)	The number of missing channels does not satisfy the threshold so the session is discarded	4
I4	Duplicated record (same uuid and same values)	The record is overwritten with the same values.	1
I5	Receipt of records with same uuid and different values	The record is overwritten with values that belongs to another session.	5
I6	Receipt of a label during the execution mode	The label is inserted into the raw session object and sent to the Preparation System (solved by the Preparation System).	1
I7	Wrong record content	Record discarded (validation fails).	4
I8	Channel with missing time series values	Missing values are not detected and they are propagated to the Preparation System.	5
I9	Receipt of records related to different sessions	The sessions are discarded because wrongly considered as incomplete. The error is not propagated.	5
I10	Receipt of records related to the same session in random order	The records are synchronized during the building of the raw session in the data store.	1
TOTAL			**31**

Preparation System

ID	Input Error	Consequence	Score
P1	Wrong raw session structure	Validation failure	4
I2.1	Missing channel in the headset field (in range)	The channel is recovered by interpolation	1
I2.2	Missing channel in the headset field (out of range)	The raw session is discarded because the interpolation is not feasible	4
I5	Receipt of raw session with wrong values	Not detected, the prepared session with incorrect features is sent to Segregation System (development phase) or Execution System (execution phase)	5
I6	Receipt of a label during the execution mode	The label is not used	1
I8	Missing or wrong values inside a channel	Not detected, the prepared session with incorrect features is sent to Segregation System (development phase) or Execution System (execution phase)	5
TOTAL			**20**

Segregation System

ID	Input Error	Consequence	Score
S1	Wrong prepared session structure	Validation failure	4
I5	Receipt of prepared session with wrong features (caused by values belonging to another session)	Not detected and sent to development system in the next dataset	5
I8	Receipt of prepared session with wrong features (caused by missing values in time series)	Not detected and sent to development system in the next dataset	5
TOTAL			**14**

Development System

ID	Input Error	Consequence	Score
D1	Wrong dataset structure	Validation failure	4
I5	Receipt of dataset containing sessions with wrong features (caused by values belonging to another session)	Not detected, the classifier will be trained with wrong features	5
I8	Receipt of dataset containing sessions with wrong features (caused by missing values in time series)	Not detected, the classifier will be trained with wrong features	5
TOTAL			**14**

Execution System

ID	Input Error	Consequence	Score
E1	Wrong classifier json structure (development phase)	The validation fails	4
E2	Wrong prepared session json structure (execution phase)	The validation fails	4
I5.1	Receipt of classifier trained with wrong features, caused by values belonging to another session (development phase)	Not detected, the bad trained classifier will be executed	5
I5.2	Receipt of prepared session with wrong features, caused by values belonging to another session (execution phase)	Not detected, a label will be produced considering the prepared session with wrong features	5
I8.1	Receipt of classifier trained with wrong features, caused by missing values in time series (development phase)	Not detected, the bad trained classifier will be executed	5
I8.2	Receipt of prepared session with wrong features, caused by missing values in time series (execution phase)	Not detected, a label will be produced considering the prepared session with wrong features	5
TOTALE			**28**

Monitoring System

ID	Input Error	Consequence	Score
M1	Wrong session label structure	The validation fails	4
I5.1	Receipt of session labels with same uuid and different values from the same source (Ingestion System and Execution System)	Not detected, the threshold is not exceeded and a system deadlock occurs	5
I5.2	Missing session label (not received from Ingestion System and Execution System)	Not detected, the threshold is not exceeded and a system deadlock occurs	5
I8.1	Receipt of session label generated by a bad classifier	Not detected, will be generated a wrong accuracy report	5
I8.2	Receipt of session label generated by prepared session with wrong features	Not detected, will be generated a wrong accuracy report	5
TOTAL			**24**

Testing (Non-Interoperability)

Feature	Ingestion	Preparation	Segregation	Development	Execution	Monitoring	Score
Mixed records of different sessions	n	y	y	y	y	y	1
Concurrent sessions of different users	n	y	n	y	y	n	1
Accuracy evaluated by environment (monitoring phase)	y	y	-	-	n	n	1
Classifiers using calendar as features (development phase)	y	n	y	n	y	-	1
TOTAL							**4**

Testing (Non-Automation)

Ingestion System

Configure Ingestion System (Data Administrator)

Use case steps	Cognitive Cost (1-4)	Normalized Salary Cost	Occurency %	Subtotal
1. The Data Administrator edits the “ip” and “port” parameters in the configuration file.	2	1.4	1	2.8
2. The Data Administrator edits the “execution_window” and “monitoring_window” parameters in the configuration file	1	1.4	1	1.4
3. The Data Administrator edits the “missing_sample_threshold” parameter in the configuration file	3	1.4	1	4.2
4. The Data Administrator starts the Ingestion System.	1	1.4	1	1.4
TOTAL				**9.8**

Preparation System

Configure Preparation System (System Administrator)

Use case steps	Cognitive Cost (1-4)	Normalized Salary Cost	Occurency %	Subtotal
1. The System Administrator edits the “ip” and “port” parameters in the configuration file	2	1	1	2
2. The System Administrator edits the “max_eeg” and “min_eeg” parameters in the configuration file	2	1	1	2
3. The System Administrator edits the “features” parameters in the configuration file	2	1	1	2
2. The System Administrator starts the Preparation System.	1	1	1	1
TOTAL				**7**

Segregation System

Configure Segregation System (Data Analyst)

Use case steps	Cognitive Cost (1-4)	Normalized Salary Cost	Occurency %	Subtotal
1. The Data Analyst edits the “ip” and “port” parameters in the configuration file	2	1.4	1	2.8
2. The Data Analyst edits the “collecting_threshold” parameter in the configuration file	4	1.4	1	5.6
3. The Data Analyst edits the “training_set_size”, “validation_set_size” and “testing_set_size” parameters in the configuration file	3	1.4	1	4.2
4. Runs the Segregation System	1	1.4	1	1.4
TOTAL				**14**

Evaluate the Balance Bar Chart (Data Analyst)

Use case steps	Cognitive Cost (1-4)	Normalized Salary Cost	Occurency %	Subtotal
1. Analyze the Balance Bar Chart with an image viewer	3	1.4	1	4.2
2. Analyze the Balancing Report with a text editor	3	1.4	1	4.2
3. Writes 'balanced' in the Balancing Report with a text editor	1	1.4	0.8	1.12
4. Writes 'not balanced' in the Balancing Report with a text editor	1	1.4	0.2	0.28
5. Starts the Segregation System	1	1.4	1	1.4
TOTAL				**11.2**

Evaluate the Radar Diagram (Data Analyst)

Use case steps	Cognitive Cost (1-4)	Normalized Salary Cost	Occurency %	Subtotal
1. Analyze the Radar Diagram with a image viewer	3	1.4	1	4.2
3. Writes 'good quality' in the Quality Report with a text editor	1	1.4	0.8	1.12
4. Writes 'bad quality' in the Quality Report with a text editor	1	1.4	0.2	0.28
5. Starts the Segregation System	1	1.4	1	1.4
TOTAL				**7**

Development System

Configure Development System

Use case steps	Cognitive cost	Normalized salary cost	Occurrency	Subtotal
1. The ML Engineer edits the “ip” and “port” parameters in the configuration file.	2	2	1	4
2. The ML Engineer edits the “initial_number_of_generations”, “number_of_hidden_layers_range” and “number_of_hidden_neurons_range” parameters in the configuration file.	2	2	1	4
3. The ML Engineer edits the “validation_error_threshold” and “test_error_threshold” parameters in the configuration file.	2	2	1	4
4. The ML Engineer starts the Development System.	1	2	1	2
TOTAL				**14**

Evaluate Training Report

Use case steps	Cognitive cost	Normalized salary cost	Occurrency	Subtotal
1. The ML Engineer analyzes the “gradient_descent_plot.png” and the “early_training_report.json”	3	2	1	6
2. The ML Engineer sets the valid_generations parameters as "true" with a text editor in early_training_report.json.	1	2	0.4	0.8
3. The ML Engineer sets the valid_generations parameters as "false" with a text editor in early_training_report.json	1	2	0.6	1.2
4. The ML Engineer edits the Number of Generations	1	2	0.6	1.2
5. The ML Engineer restarts the system.	1	2	1	2
TOTAL				**11.2**

Evaluate Top Classifiers

Use case steps	Cognitive cost	Normalized salary cost	Occurrency	Subtotal
1. The ML Engineer analyzes the top_five_classifier_report.json	2	2	5*1	40
2. The ML Engineer sets the paramenter actual_best as "true" with a text editor in top_five_classifiers_report.json	1	2	5*0.2	2
3. The ML Engineer sets the paramenter actual_best of the previous classifiers as "false" with a text editor in top_five_classifiers_report.json.	1	2	5*0.8	8
4. The ML Engineer edits the Number of Generations	1	2	0.2	0.4
5. The ML Engineer restarts the system.	1	2	1	2
TOTAL				**52.4**

Evaluate Test Results

Use case steps	Cognitive cost	Normalized salary cost	Occurrency	Subtotal
1. The ML Engineer analyzes the test_best_classifier_report.json.	2	2	1	4
2. The ML Engineer sets the valid_classifier parameters as "true" with a text editor in test_best_classifier_report.json.	1	2	0.8	1.6
3. The ML Engineer restarts the system.	1	2	0.8	1.6
4. The ML Engineer sets the valid_classifier parameters as "false" with a text editor in test_best_classifier_report.json.	1	2	0.2	0.4
5. The ML Engineer requests a Reconfiguration of the systems to the Messaging System	1	2	0.2	0.4
6. The ML Engineer restarts the system.	1	2	0.2	0.4
TOTAL				**8.4**

Execution System

Configure Execution System (System Administrator)

Use case steps	Cognitive Cost (1-4)	Normalized Salary Cost	Occurency %	Subtotal
1. The System Administrator edits the “execution_window_value” and “monitoring_window_value” parameters in the configuration.	1	1	1	1
2. The System Administrator edits the “ip” and “port” parameters in the configuration file	2	1	1	2
3. The System Administrator launches the Execution System.	1	1	1	1
TOTAL				**4**

Monitoring System

Configure Monitoring System Use Case (ML Engineer)

Use case steps	Cognitive Cost (1-4)	Normalized Salary Cost	Occurency %	Subtotal
1. The ML Engineer edits the “max_errors_tolerated” parameter in the configuration file	4	2	1	8
2. The ML Engineer edits the “labels_threshold” parameter in the configuration file (decision of execution window and monitoring window)	4	2	1	8
3. The ML Engineer starts the Development System.	1	2	1	2
TOTAL				**18**

Analyze Accuracy Report Use Case (ML Engineer)

Use case steps	Cognitive Cost (1-4)	Normalized Salary Cost	Occurency %	Subtotal
1. The ML Engineer analyzes the “accuracy_report.json”	2	2	1	4
2. The ML Engineer marks the Classifier as “Accepted”	1	2	0.7	1.4
3. The ML Engineer marks the Classifier as “Not Accepted”	1	2	0.3	0.6
TOTAL				**6**