

Patient Monitoring Service (PMS)

Part 2: Architecture evaluation

Description of the initial architecture

Contents

| | | |
|----------|------------------------------------|-----------|
| A | Client-Server View | 3 |
| B | Decomposition View | 4 |
| C | Deployment view | 6 |
| D | Process View | 8 |
| E | Element catalog | 12 |
| E.1 | Components | 12 |
| E.1.1 | ClinicalJobCreator | 12 |
| E.1.2 | ClinicalModelCache | 12 |
| E.1.3 | ClinicalModelDB | 12 |
| E.1.4 | eHealth Platform | 12 |
| E.1.5 | KVStorageService | 13 |
| E.1.6 | MLaaS | 13 |
| E.1.7 | MLModelManager | 13 |
| E.1.8 | OtherFunctionality | 13 |
| E.1.9 | RiskEstimationCombiner | 14 |
| E.1.10 | RiskEstimationProcessor | 14 |
| E.1.11 | RiskEstimationScheduler | 14 |
| E.2 | Modules | 14 |
| E.2.1 | JobMLModelSelector | 14 |
| E.2.2 | JobProcessor | 15 |
| E.2.3 | MLaaSLibrary | 15 |
| E.2.4 | MLLibrary | 15 |
| E.2.5 | MLModelRepository | 15 |
| E.2.6 | MLModelRetrieval | 15 |
| E.2.7 | MLModelStorageManager | 16 |
| E.2.8 | MLModelUpdateProcessor | 16 |
| E.2.9 | ModelUpdater | 16 |
| E.2.10 | SensorDataRiskAssessment | 16 |
| E.3 | Interfaces | 16 |
| E.3.1 | ApplyClinicalModel | 16 |
| E.3.2 | ClinicalJobMgmt | 17 |
| E.3.3 | ClinicalModelCacheMgmt | 17 |
| E.3.4 | ClinicalModelStorage | 17 |
| E.3.5 | FetchClinicalModels | 17 |
| E.3.6 | FetchJobs | 18 |
| E.3.7 | FetchMLModel | 18 |
| E.3.8 | JobMgmt | 18 |
| E.3.9 | KVStore | 18 |
| E.3.10 | LaunchRiskEstimation | 18 |
| E.3.11 | MLaaSAPI | 19 |
| E.3.12 | MLaaSService | 19 |
| E.3.13 | MLAPI | 20 |
| E.3.14 | MLModelMgmt | 20 |
| E.3.15 | MLModelStorage | 21 |
| E.3.16 | MLModelSync | 21 |
| E.3.17 | MLModelUpdateMgmt | 21 |
| E.3.18 | OtherDataMgmt | 21 |
| E.3.19 | PatientRecordMgmt | 22 |
| E.3.20 | Results | 22 |

| | | |
|--------|---|----|
| E.3.21 | SensorDataAnalysis | 22 |
| E.3.22 | SensorDataMgmt | 22 |
| E.4 | Nodes | 23 |
| E.4.1 | eHealthPlatform | 23 |
| E.4.2 | MLaaS Conn (Pilot Deployment) | 23 |
| E.4.3 | MLaaS Service (Dev Deployment) | 23 |
| E.4.4 | MLaaS Service (Pilot Deployment) | 23 |
| E.4.5 | MLaaSCloud | 23 |
| E.4.6 | MLaaSCollectorNode | 23 |
| E.4.7 | RiskEstimationManagementNode (Pilot Deployment) | 23 |
| E.4.8 | RiskEstimationManagementNode (Dev Deployment) | 23 |
| E.4.9 | RiskEstimationMgmtNode | 24 |
| E.4.10 | RiskEstimationProcessorNode | 24 |
| E.4.11 | RiskEstimationProcessorNode (Dev Deployment) | 24 |
| E.4.12 | RiskEstimationProcessorNode (Pilot Deployment) | 24 |
| E.4.13 | TODO Node (Pilot Deployment) | 24 |
| E.4.14 | TODONode | 24 |
| E.5 | Exceptions | 24 |
| E.6 | Data types | 25 |
| E.7 | Unresolved issues | 26 |

A. Client-Server View

Figures

| | |
|--|---|
| A.1 Context diagram for the client-server view | 3 |
| A.2 Primary diagram of the client-server view | 3 |

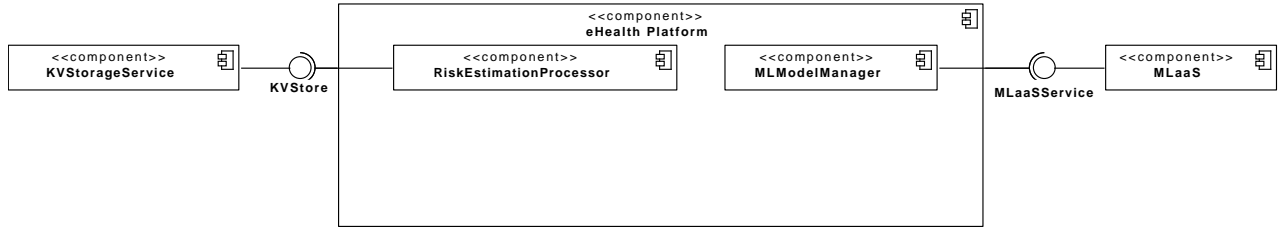


Figure A.1: Context diagram for the client-server view

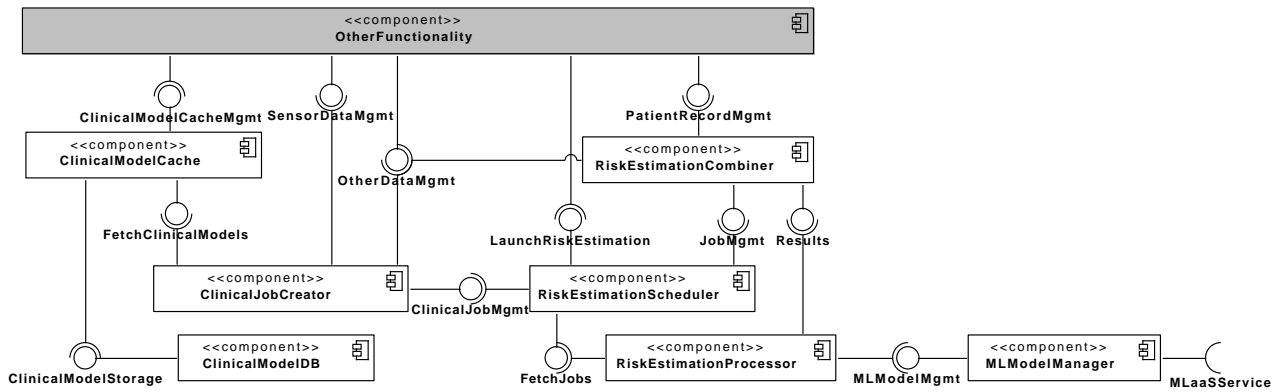


Figure A.2: Primary diagram of the client-server view

This architecture is not yet complete. Functionality that still has to be worked out in more detail is captured by the **OtherFunctionality** component in gray.

B. Decomposition View

Figures

| | | |
|-----|--|---|
| B.1 | Context diagram for the decomposition view | 4 |
| B.2 | Decomposition of the MLModelManager | 4 |
| B.3 | Decomposition of the RiskEstimationProcessor | 5 |

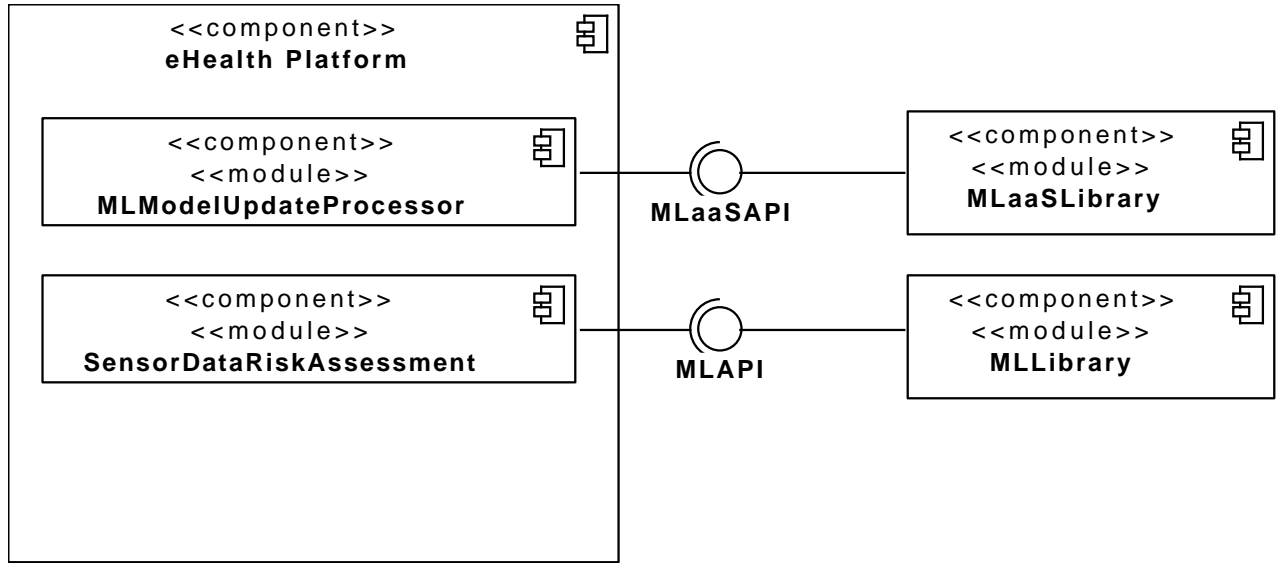


Figure B.1: Context diagram for the decomposition view

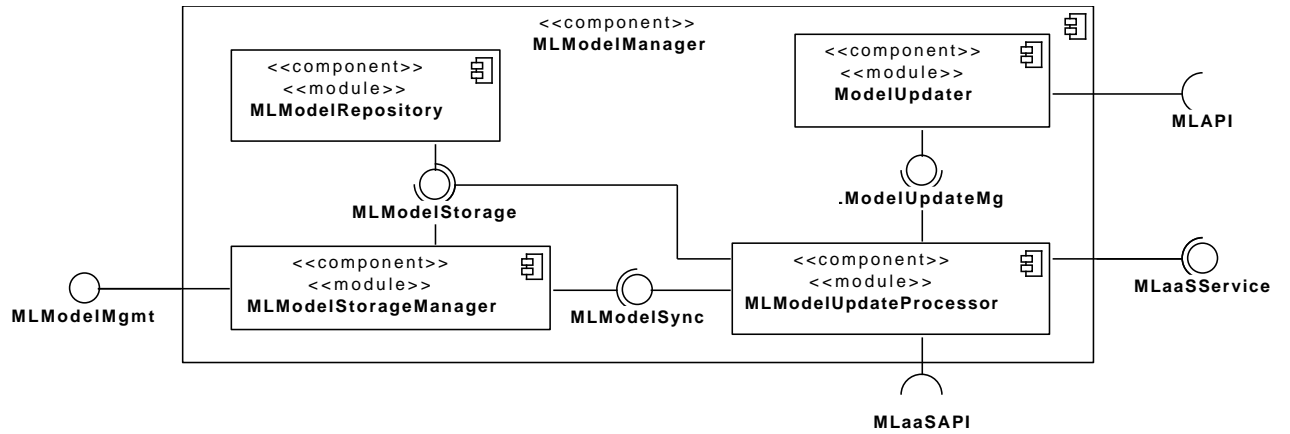


Figure B.2: Decomposition of the MLModelManager

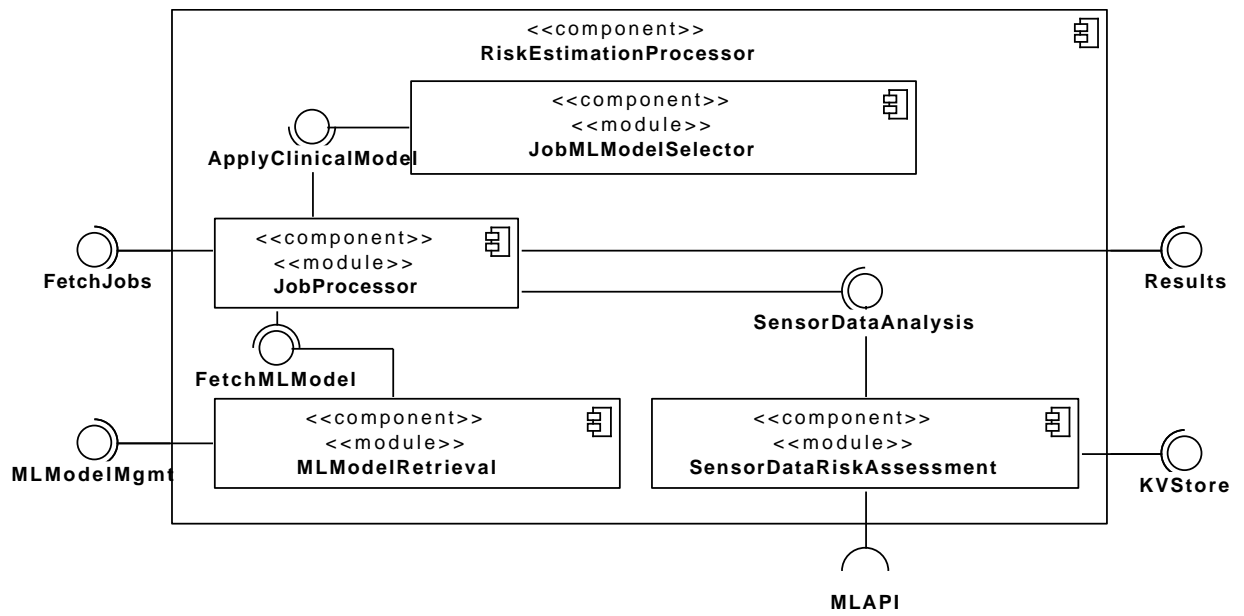


Figure B.3: Decomposition of the RiskEstimationProcessor

C. Deployment view

Figures

| | | |
|-----|--|---|
| C.1 | Context diagram for the deployment view | 6 |
| C.2 | Primary deployment diagram | 6 |
| C.3 | Pilot Deployment (200 patients, 5 clin models) | 7 |
| C.4 | Development Test Deployment (20 patients, 3 clin models) | 7 |

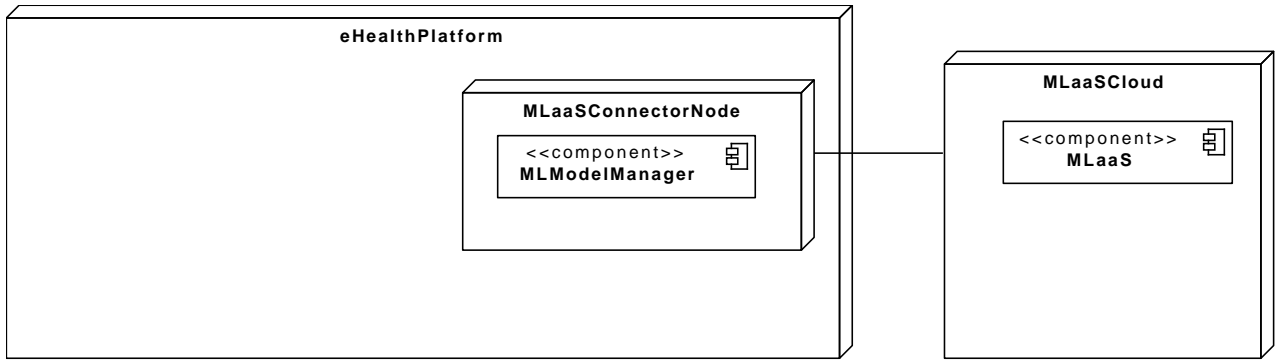


Figure C.1: Context diagram for the deployment view

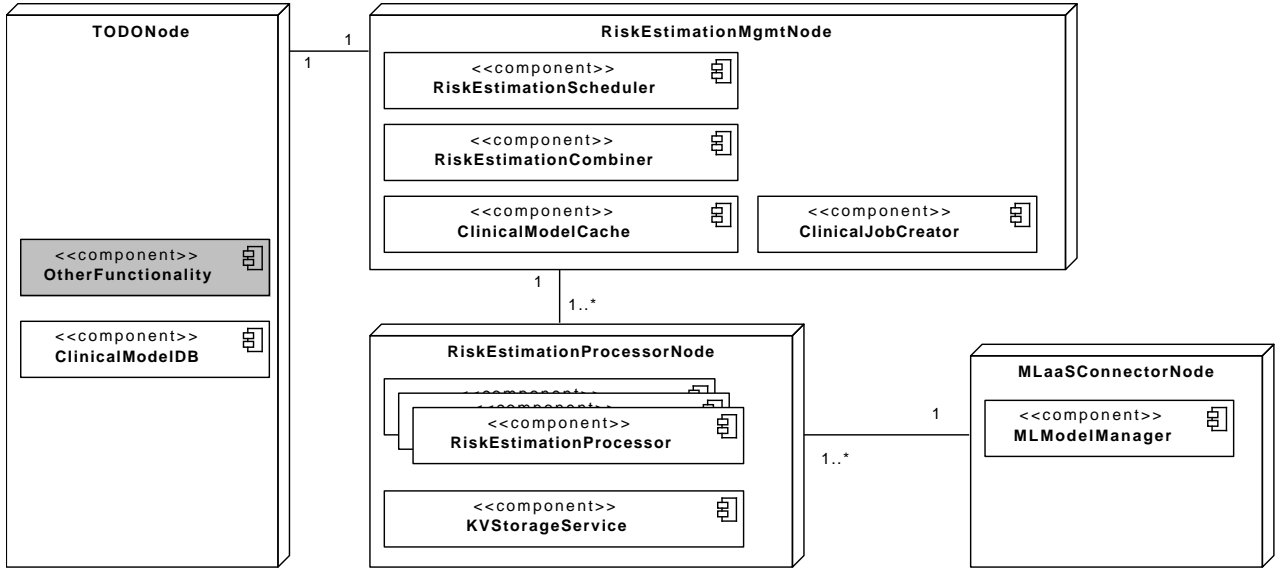


Figure C.2: Primary deployment diagram

This deployment is necessarily incomplete as the architecture is incomplete. The functionality that is not worked out yet is captured by **OtherFunctionality** component, currently deployed on **TODONode**. This **TODONode** will also have to be replaced with the final nodes are required by the replacement of **OtherFunctionality**.

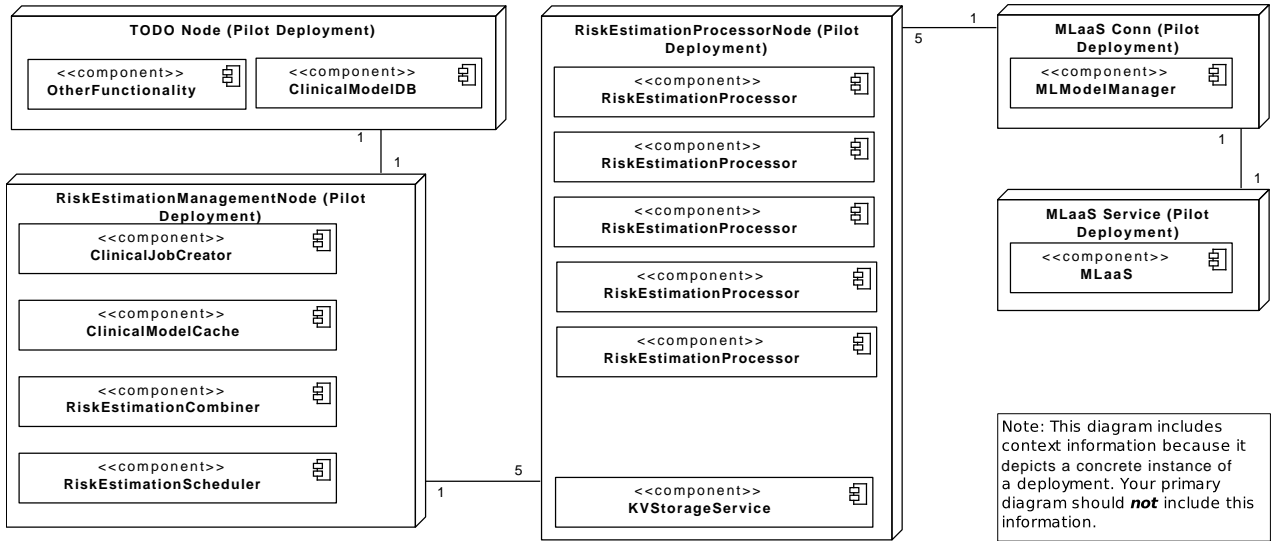


Figure C.3: Pilot Deployment (200 patients, 5 clin models)

This diagram shows a concrete initial pilot deployment for 200 patients with an average of 5 clinical models per patient. This diagram is a concrete instance of a deployment for the specified number of patients and clinical models, therefore it also includes information from the context diagram. Note that it is not correct to include this type of information in your primary deployment diagram.

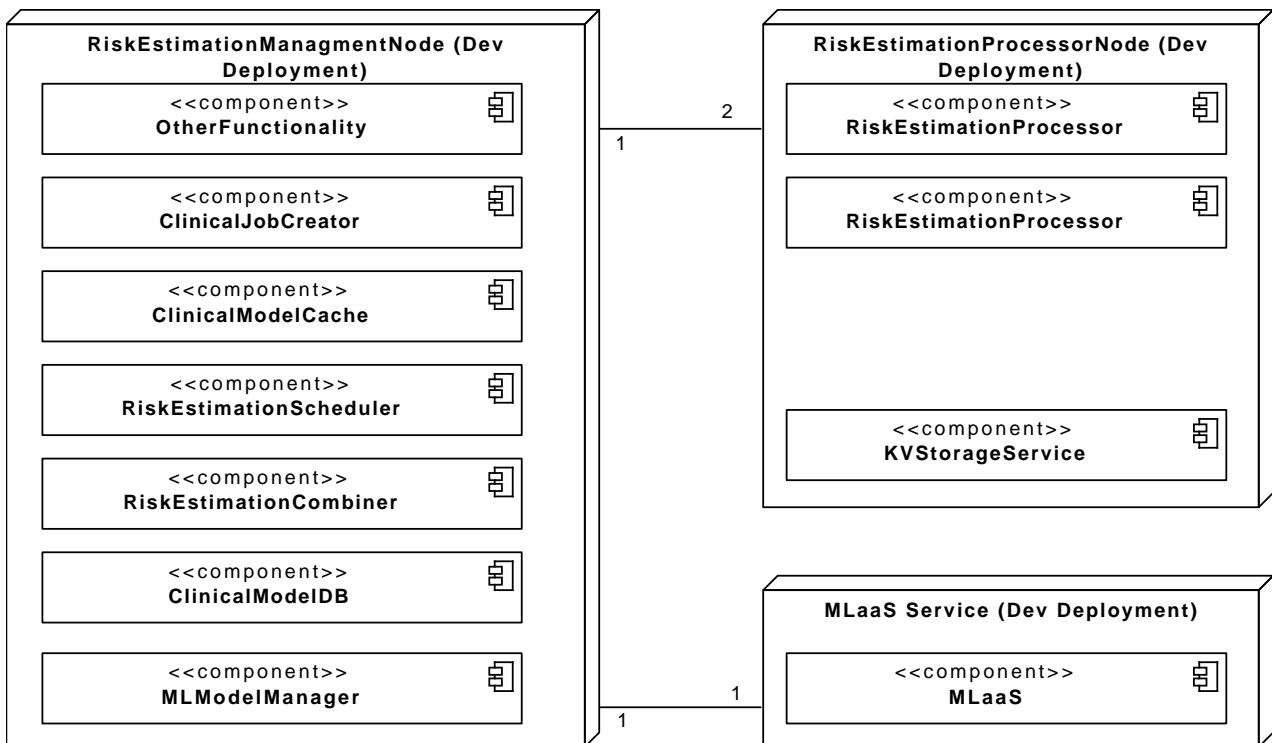


Figure C.4: Development Test Deployment (20 patients, 3 clin models)

This diagram shows a simplified deployment for local development. This diagram is a concrete instance of a deployment for the specified number of patients and clinical models, therefore it also includes information from the context diagram. Note that it is not correct to include this type of information in your primary deployment diagram.

D. Process View

Figures

| | |
|--|----|
| D.1 Risk estimation process | 8 |
| D.2 Compute clinical model result | 9 |
| D.3 MLModel Synchronization | 9 |
| D.4 Processing MLModel updates | 10 |
| D.5 Incoming sensor data | 10 |
| D.6 Combining ClinicalModelJob results | 11 |

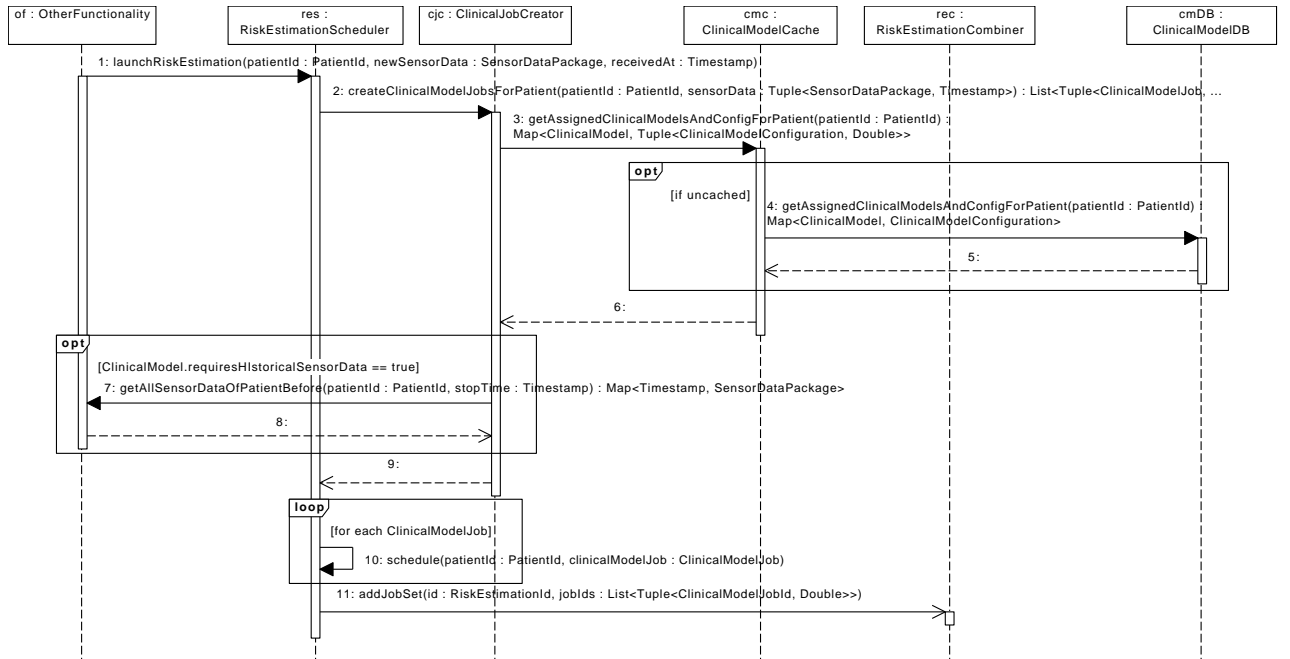


Figure D.1: Risk estimation process

The scheduling and execution of a risk estimation triggered by the arrival of new sensor data. Note that if one or more clinical models in the **ClinicalModelCache** have been invalidated, all relevant clinical models are renewed.

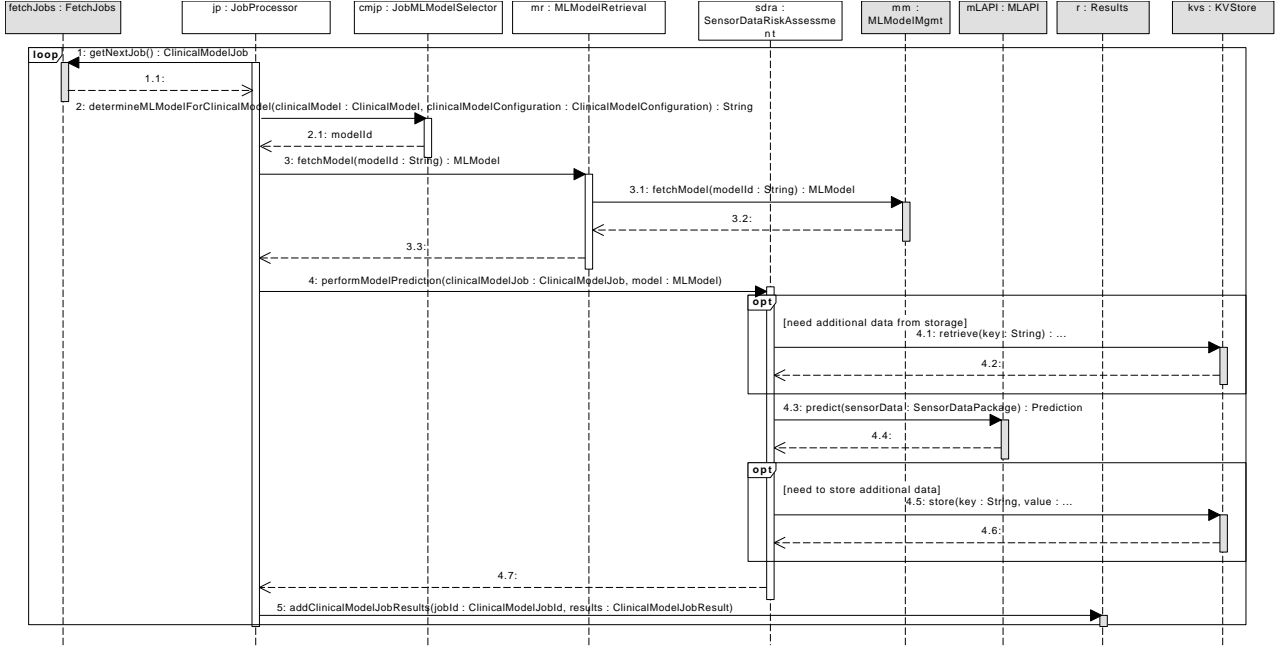


Figure D.2: Compute clinical model result

This sequence diagram shows the detailed application and computation of the clinical model. The first step involves the retrieval of the appropriate **MLModel** (specified by the **ClinicalModel**). The second step involves the calculation of the prediction using the retrieved **MLModels**. Note that, as this diagram depicts the internal flow between modules within the **RiskEstimationProcessor**, the gray lifelines depict its (external) interfaces.

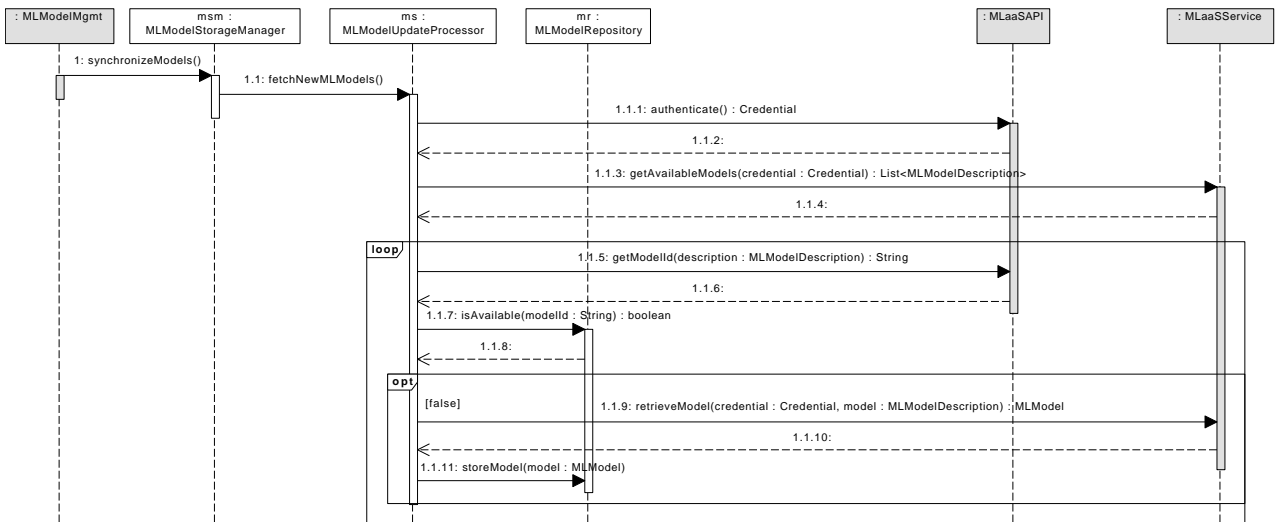


Figure D.3: MLModel Synchronization

As the system operations new **MLModels** may become available for use by the **ClinicalModels**. This diagram illustrates the synchronization logic to fetch any unavailable **MLModels** and to store a local copy of the **MLModels** in the **MLModelRepository** for faster retrieval and use in the sensor data risk assessment. Note that, as this diagram depicts the internal flow between modules within the **MLModelManager**, the gray lifelines depict its (external) interfaces.

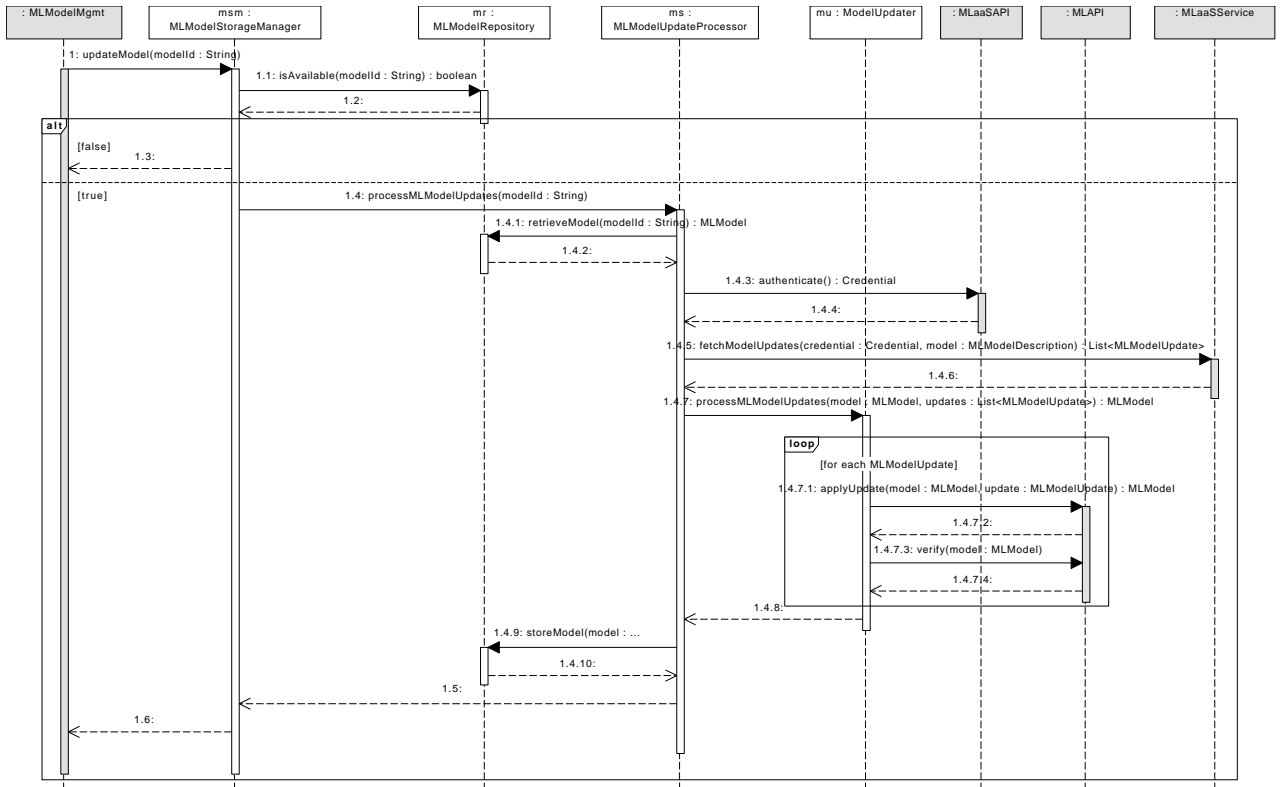


Figure D.4: Processing MLModel updates

MLModelUpdates are made available to improve the existing **MLModels**. This diagram illustrates the retrieval of these **MLModelUpdates** and their application on the local **MLModel**. As updates to these models influence the risk assessments, they need to be triggered for each individual **MLModel** that requires an update. Note that, as this diagram depicts the internal flow between modules within the **MLModelManager**, the gray lifelines depict its (external) interfaces.

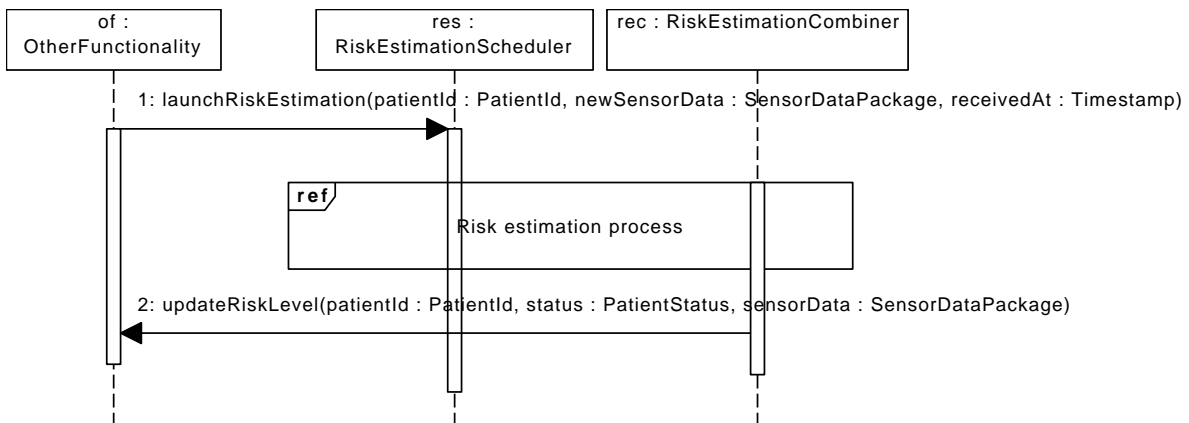


Figure D.5: Incoming sensor data

This diagram shows a small part of the processing of incoming sensor data. It does not yet depict the arrival of the sensor data, storage in a database, or any other processing, as this functionality is not yet worked out.

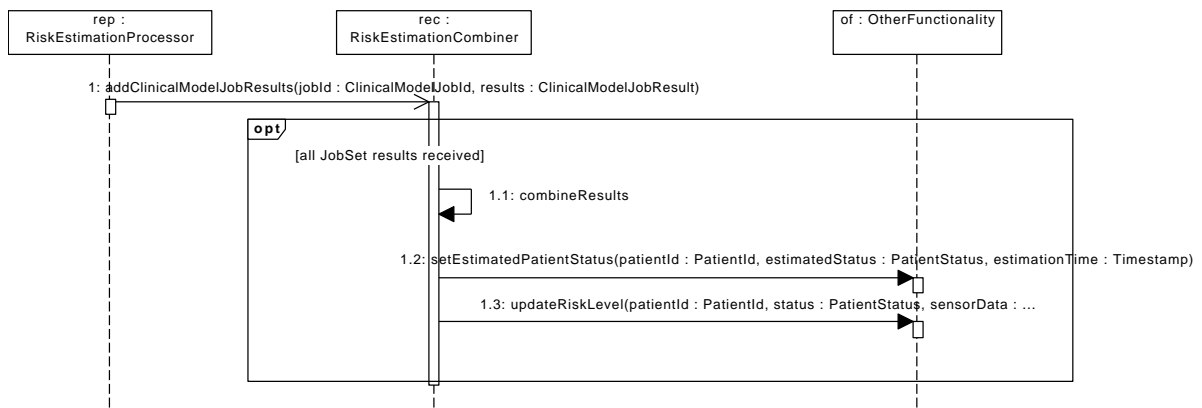



Figure D.6: Combining ClinicalModelJob results

E. Element catalog


E.1 Components




E.1.1 ClinicalJobCreator




Responsibility: The `ClinicalJobCreator` is responsible for constructing the **ClinicalModelJob** objects and populating them with the necessary patient health data for execution by the `RiskEstimationProcessors`.

Super-components:  eHealth Platform

Sub-components: None

Provided interfaces:  `ClinicalJobMgmt`


Required interfaces:  `FetchClinicalModels`,  `OtherDataMgmt`,  `SensorDataMgmt`

Deployed on:  `RiskEstimationManagementNode` (Pilot Deployment),  `RiskEstimationManagementNode` (Dev Deployment),  `RiskEstimationMgmtNode`



Visible on diagrams: figs. A.2, C.2, C.3, C.4 and D.1

E.1.2 ClinicalModelCache




Responsibility: The `ClinicalModelCache` is a read-through cache responsible for caching the **ClinicalModels** that should be evaluated for a certain patient and their configurations. This cache is located close to the `RiskEstimationProcessor` and `RiskEstimationCombiner` in order to improve the latency of the whole risk estimation flow. The items in the `ClinicalModelCache` do not expire over time, but should be invalidated explicitly if needed. Cached models remain available for the `RiskEstimationProcessor` while updates are made in the `ClinicalModelDB`.

Super-components:  eHealth Platform

Sub-components: None

Provided interfaces:  `ClinicalModelCacheMgmt`,  `FetchClinicalModels`


Required interfaces:  `ClinicalModelStorage`,  `OtherDataMgmt`

Deployed on:  `RiskEstimationManagementNode` (Pilot Deployment),  `RiskEstimationManagementNode` (Dev Deployment),  `RiskEstimationMgmtNode`


Visible on diagrams: figs. A.2, C.2, C.3, C.4 and D.1

E.1.3 ClinicalModelDB




Responsibility: The `ClinicalModelDB` stores all the **ClinicalModels** and the configurations of these **ClinicalModels** for the different patients separately from other data. It only allows appending new **ClinicalModels**.

Super-components:  eHealth Platform

Sub-components: None

Provided interfaces:  `ClinicalModelStorage`

Required interfaces: None









Deployed on:  `TODO Node` (Pilot Deployment),  `RiskEstimationManagementNode` (Dev Deployment),  `TODO Node`

Visible on diagrams: figs. A.2, C.2, C.3, C.4 and D.1

E.1.4 eHealth Platform

Responsibility: The parent `eHealth Platform` component that represents the context boundary in the client-server view.

Super-components: None

Sub-components:  `OtherFunctionality`,  `MLModelManager`,  `ClinicalModelCache`,  `RiskEstimationCombiner`,  `RiskEstimationProcessor`,  `RiskEstimationScheduler`,  `ClinicalJobCreator`,  `ClinicalModelDB`

Provided interfaces: None

Required interfaces: < KVStore, < MLaaSService

Deployed on: ☐ TODO Node (Pilot Deployment), ☐ RiskEstimationManagementNode (Pilot Deployment), ☐ RiskEstimationProcessorNode (Pilot Deployment), ☐ MLaaS Conn (Pilot Deployment), ☐ RiskEstimationManagmentNode (Dev Deployment), ☐ RiskEstimationProcessorNode (Dev Deployment), ☐ MLaaSConnectorNode, ☐ RiskEstimationMgmtNode, ☐ RiskEstimationProcessorNode, ☐ TODONode

Visible on diagrams: figs. A.1 and B.1

E.1.5 KVStorageService

Responsibility: Service for key-value storage of data.

Super-components: None

Sub-components: None

Provided interfaces: ~ KVStore

Required interfaces: None

Deployed on: ☐ RiskEstimationProcessorNode (Pilot Deployment), ☐ RiskEstimationProcessorNode (Dev Deployment), ☐ RiskEstimationProcessorNode

Visible on diagrams: figs. A.1, C.2, C.3 and C.4

E.1.6 MLaaS

Responsibility: MachineLearning-as-a-Service cloud provider. This cloud service provides APIs for managing different **MLModels** and exchanging **MLModel** updates with other parties.

Super-components: None

Sub-components: None

Provided interfaces: ~ MLaaSService

Required interfaces: None

Deployed on: ☐ MLaaSCloud, ☐ MLaaS Service (Pilot Deployment), ☐ MLaaS Service (Dev Deployment)

Visible on diagrams: figs. A.1, C.1, C.3 and C.4

E.1.7 MLModelManager

Responsibility: This component is responsible for managing the local **MLModels** for making predictions for malignant events based on the incoming sensor data. It keeps track of the available **MLModels** that can be used and supports exchanging **MLModel** updates with other systems using the APIs of MLaaS providers.

Super-components: ☒ eHealth Platform

Sub-components: None

Provided interfaces: ~ MLModelMgmt

Required interfaces: < MLaaSService

Deployed on: ☐ MLaaS Conn (Pilot Deployment), ☐ RiskEstimationManagmentNode (Dev Deployment), ☐ MLaaSConnectorNode

Visible on diagrams: figs. A.1, A.2, B.2, C.1, C.2, C.3 and C.4

E.1.8 OtherFunctionality

Responsibility: Other functionality that still has to be worked out.

Super-components: ☒ eHealth Platform

Sub-components: None

Provided interfaces: ~ OtherDataMgmt, ~ PatientRecordMgmt, ~ SensorDataMgmt


Required interfaces: < ClinicalModelCacheMgmt, < LaunchRiskEstimation

Deployed on: ☐ TODO Node (Pilot Deployment), ☐ RiskEstimationManagmentNode (Dev Deployment), ☐ TODONode

Visible on diagrams: figs. A.2, C.2, C.3, C.4, D.1, D.5 and D.6

E.1.9 RiskEstimationCombiner




Responsibility: The **RiskEstimationCombiner** is responsible for combining the results of the **ClinicalModelJobs** belonging to a patient risk estimation. More precisely, the **RiskEstimationScheduler** passes the set of the scheduled jobs for a risk estimation to the **RiskEstimationCombiner** before scheduling them. The **RiskEstimationCombiner** then waits for all results to arrive, combines them, and propagates the new sensor data and the results of the risk estimation to the patient record if needed.

Super-components:  eHealth Platform

Sub-components: None

Provided interfaces: \rightarrow JobMgmt, \rightarrow Results


Required interfaces: \leftarrow OtherDataMgmt, \leftarrow PatientRecordMgmt

Deployed on:  RiskEstimationManagementNode (Pilot Deployment),  RiskEstimationManagementNode (Dev Deployment),  RiskEstimationMgmtNode

Visible on diagrams: figs. A.2, C.2, C.3, C.4, D.1, D.5 and D.6

E.1.10 RiskEstimationProcessor




Responsibility: The **RiskEstimationProcessor** is responsible for computing **ClinicalModelJobs**. The **RiskEstimationProcessor** fetches new **ClinicalModelJobs** from the **RiskEstimationScheduler**, uses the specified **MLModel** to calculate the risk prediction, and passes the result to the **RiskEstimationCombiner**. Multiple instances of the **RiskEstimationProcessor** run in parallel to improve the throughput.

Super-components:  eHealth Platform

Sub-components: None

Provided interfaces: None


Required interfaces: \leftarrow FetchJobs, \leftarrow KVStore, \leftarrow MLModelMgmt, \leftarrow Results

Deployed on:  RiskEstimationProcessorNode (Pilot Deployment),  RiskEstimationProcessorNode (Dev Deployment),  RiskEstimationProcessorNode

Visible on diagrams: figs. A.1, A.2, B.3, C.2, C.3, C.4 and D.6

E.1.11 RiskEstimationScheduler




Responsibility: The **RiskEstimationScheduler** is responsible for taking in new and scheduling requests for risk estimations. It keeps track of the throughput of incoming jobs and changes the scheduling from first-in/first-out to dynamic priority earliest deadline first when going into overload modus.

Super-components:  eHealth Platform

Sub-components: None

Provided interfaces: \rightarrow FetchJobs, \rightarrow LaunchRiskEstimation

Required interfaces: \leftarrow ClinicalJobMgmt, \leftarrow FetchClinicalModels, \leftarrow JobMgmt, \leftarrow OtherDataMgmt, \leftarrow SensorDataMgmt



Deployed on:  RiskEstimationManagementNode (Pilot Deployment),  RiskEstimationManagementNode (Dev Deployment),  RiskEstimationMgmtNode

Visible on diagrams: figs. A.2, C.2, C.3, C.4, D.1 and D.5

E.2 Modules

E.2.1 JobMLModelSelector

Responsibility: This module is responsible for determining the appropriate **MLModel** to use for making the prediction on the provided sensor data, as there could be multiple **MLModel** candidates for making predictions.

Super-components:  RiskEstimationProcessor \triangleright  eHealth Platform

Super-modules: None

Sub-modules: None



Provided interfaces: \rightarrow ApplyClinicalModel

Required interfaces: None

Visible on diagrams: figs. B.3 and D.2

E.2.2 JobProcessor






Responsibility: The JobProcessor module coordinates the execution of the **ClinicalModelJobs** to ensure that the required **MLModels** are fetched, and the prediction is performed with the appropriate **MLModel**.

Super-components:  RiskEstimationProcessor  eHealth Platform

Super-modules: None

Sub-modules: None

Provided interfaces: None

Required interfaces:  ApplyClinicalModel,  FetchJobs,  FetchMLModel,  Results,  SensorData-Analysis

Visible on diagrams: figs. B.3 and D.2

E.2.3 MLaaSLibrary

Responsibility: MLaaS development library for working with the MLaaS providers. It supports storage & retrieval of **MLModels**, and exchanging **MLModelUpdates** with MLaaS providers.

Super-components: None

Super-modules: None

Sub-modules: None

Provided interfaces:  MLaaSAPI

Required interfaces: None

Visible on diagrams: fig. B.1


E.2.4 MLLibrary

Responsibility: Library for working with **MLModels** locally and using them to make predictions based on the incoming sensor data. The library can also process external updates to apply them to a specific **MLModel**.

Super-components: None

Super-modules: None

Sub-modules: None



Provided interfaces:  MLAPI

Required interfaces: None

Visible on diagrams: fig. B.1


E.2.5 MLModelRepository

Responsibility: This module stores **MLModels** locally and makes them available for use by the RiskEstimationProcessor when processing **ClinicalModelJobs**.

Super-components:  eHealth Platform  MLModelManager

Super-modules: None

Sub-modules: None



Provided interfaces:  MLModelStorage

Required interfaces: None

Visible on diagrams: figs. B.2, D.3 and D.4


E.2.6 MLModelRetrieval


Responsibility: This module is responsible for retrieving the requested **MLModels** from the MLModelManager.

Super-components:  RiskEstimationProcessor  eHealth Platform

Super-modules: None

Sub-modules: None



Provided interfaces:  FetchMLModel

Required interfaces:  MLModelMgmt

Visible on diagrams: figs. B.3 and D.2


E.2.7 MLModelStorageManager



Responsibility: This module is responsible for managing the locally stored **MLModels**, triggering the synchronization and retrieval of new **MLModels**, and coordinating the updates to the existing **MLModels**.

Super-components:  eHealth Platform  MLModelManager

Super-modules: None

Sub-modules: None



Provided interfaces:  MLModelMgmt

Required interfaces:  MLModelStorage,  MLModelSync

Visible on diagrams: figs. B.2, D.3 and D.4


E.2.8 MLModelUpdateProcessor





Responsibility: The MLModelUpdater processes and applies updates to the local **MLModels** to prevent updates to these models from requiring to retrieve the full **MLModels**.

Super-components:  eHealth Platform  MLModelManager

Super-modules: None

Sub-modules: None



Provided interfaces:  MLModelSync

Required interfaces:  MLaaSAPI,  MLaaSService,  MLModelStorage,  MLModelUpdateMgmt

Visible on diagrams: figs. B.1, B.2, D.3 and D.4


E.2.9 ModelUpdater


Responsibility: This module is responsible for processing **MLModelUpdates** and applying them to the **MLModel**.

Super-components:  eHealth Platform  MLModelManager

Super-modules: None

Sub-modules: None



Provided interfaces:  MLModelUpdateMgmt

Required interfaces:  MLAPI

Visible on diagrams: figs. B.2 and D.4


E.2.10 SensorDataRiskAssessment



Responsibility: This module performs the actual risk assessment by using the **MLModel** to make a prediction on the sensor data.

Super-components:  RiskEstimationProcessor  eHealth Platform

Super-modules: None

Sub-modules: None

Provided interfaces:  SensorDataAnalysis


Required interfaces:  KVStore,  MLAPI

Visible on diagrams: figs. B.1, B.3 and D.2

E.3 Interfaces

E.3.1 ApplyClinicalModel

Provided by:  JobMLModelSelector


Required by:  JobProcessor


Operations:

- String determineMLModelForClinicalModel(**ClinicalModel** clinicalModel, **ClinicalModelConfiguration** clinicalModelConfiguration)
 - Effect: Apply the **ClinicalModel** to obtain the most appropriate **MLModel** to use for the calculation.
 - Returns: Identifier of the **MLModel** to use.
 - Sequence Diagrams: fig. D.2

Diagrams: None

E.3.2 ClinicalJobMgmt

Provided by:  ClinicalJobCreator


Required by:  RiskEstimationScheduler

Operations:

- List<Tuple<**ClinicalModelJob**, Double>> createClinicalModelJobsForPatient(**PatientId** patientId, Tuple<**SensorDataPackage**, **Timestamp**> sensorData)
 - Effect: Construct the list of **ClinicalModelJobs** for a specified patient. These jobs are populated with the necessary data for their execution by the **RiskEstimationProcessor**.
 - Returns: A list of tuples containing the **ClinicalModelJob** and its weight.
 - Sequence Diagrams: fig. D.1

Diagrams: None

E.3.3 ClinicalModelCacheMgmt

Provided by:  ClinicalModelCache

Required by:  OtherFunctionality


Operations:

- void invalidateCacheEntries(**PatientId** patientId)
 - Effect: The **ClinicalModelCache** will invalidate (i.e., remove) all items in its cache for the patient identified by patientId. If the cache does not contain any items for this patient, nothing is changed. After invalidating the cached items for a certain patient, the next request for them will lead to fetching them from the database and storing them in the cache again.
 - Sequence Diagrams: None

Diagrams: None

E.3.4 ClinicalModelStorage

Provided by:  ClinicalModelDB


Required by:  ClinicalModelCache



Operations:

- Map<**ClinicalModel**, **ClinicalModelConfiguration**> getAssignedClinicalModelsAndConfigForPatient(**PatientId** patientId) throws *NoSuchPatientException*
 - Effect: Fetch and return the ids of the **ClinicalModels** assigned to the patient identified by patientId and their configurations for this patient. The configurations are returned as a **ClinicalModelConfiguration** containing a map of configuration parameters and their value.
 - Returns: A map with for every applicable **ClinicalModel** the corresponding **ClinicalModelConfiguration**.
 - Sequence Diagrams: fig. D.1

Diagrams: None

E.3.5 FetchClinicalModels

Provided by:  ClinicalModelCache

Required by:  ClinicalJobCreator,  RiskEstimationScheduler

Operations:

- Map<**ClinicalModel**, Tuple<**ClinicalModelConfiguration**, Double>> getAssignedClinicalModelsAndConfigForPatient(**PatientId** patientId)
 - Effect: The **ClinicalModelCache** will return the clinical models assigned to the patient identified by patientId and their configurations for this patient. The configurations are a map of configuration parameters and their value. If the cache contains data for the patient with given id, the data from the cache is returned. If not, the **ClinicalModelCache** will fetch all clinical models assigned to the patient with given patientId and their configurations for this patient, store these in the cache and return them.
 - Returns: A map with for every applicable **ClinicalModel**, a tuple containing the corresponding **ClinicalModelConfiguration** for the specified patient and the corresponding weight factors used for combining the results.

- Sequence Diagrams: fig. D.1

Diagrams: None

E.3.6 FetchJobs

Provided by: ¶ RiskEstimationScheduler

Required by: ¶ JobProcessor, ¶ RiskEstimationProcessor

Operations:

- **ClinicalModelJob** getNextJob()
 - Effect: Returns the next clinical model computation job that must be performed (i.e., the first job in the queue).
 - Returns: The next clinical model computation job that must be performed.
 - Sequence Diagrams: fig. D.2

Diagrams: None

E.3.7 FetchMLModel

Provided by: ¶ MLModelRetrieval

Required by: ¶ JobProcessor

Operations:

- **MLModel** fetchModel(String modelId) throws *NoSuchMLModelException*
 - Effect: Fetch the **MLModel** corresponding with the provided modelId.
 - Returns: The requested **MLModel**.
 - Sequence Diagrams: fig. D.2

Diagrams: None

E.3.8 JobMgmt

Provided by: ¶ RiskEstimationCombiner

Required by: ¶ RiskEstimationScheduler

Operations:

- void addJobSet(**RiskEstimationId** id, List<Tuple<**ClinicalModelJobId**, Double>> jobIds)
 - Effect: Adds a set of identifiers for jobs belonging to a single risk estimation identified by id. The partial results of each clinical model computation job identified by an element in jobIds have to be combined in order to find the final result of the risk estimation as a whole.
 - Sequence Diagrams: fig. D.1

Diagrams: None

E.3.9 KVStore

Provided by: ¶ KVStorageService

Required by: ¶ RiskEstimationProcessor, ¶ SensorDataRiskAssessment, ¶ eHealth Platform

Operations:

- **Object** retrieve(String key) throws *IOException*
 - Effect: Retrieve the object with the provided key from the Key-Value storage.
 - Returns: The object retrieved from storage.
 - Sequence Diagrams: fig. D.2
- void store(String key, **Object** value) throws *IOException*
 - Effect: Store the provided object in the Key-Value storage under the provided key.
 - Sequence Diagrams: fig. D.2

Diagrams: None

E.3.10 LaunchRiskEstimation

Provided by: ¶ RiskEstimationScheduler

Required by: ¶ OtherFunctionality

Operations:


- void launchRiskEstimation(**PatientId** patientId, **SensorDataPackage** newSensorData, **Timestamp** receivedAt)

- Effect: The **RiskEstimationScheduler** will fetch the clinical models and their configurations associated to the patient identified by **patientId** from storage using the **ClinicalModelCache**, notify the **RiskEstimationCombiner** of the different jobs that will be performed for a single risk estimation and schedule the individual jobs in its queue. In normal modus, queued jobs are returned in FIFO order. In overload modus, the system switches to dynamic priority: earliest deadline first and enqueues jobs of patient with a high risk level with an earlier deadline (2 instead of 5 minutes) to prioritize them over patients with lower risk levels. The **SensorDataPackage** **newSensorData** is passed because its arrival triggered the risk estimation. A risk level is estimated based on the computation of these clinical models. For the computation of the clinical models, the given sensor data **newSensorData** (which is the new sensor data that was received) is used and other required data is fetched from the respective databases if needed. The given time-stamp **receivedAt** is used in order to avoid fetching the new sensor data from the database. This time-stamp represents the time at which the new sensor data was received.
- Sequence Diagrams: figs. D.1 and D.5

Diagrams: None

E.3.11 MLaaSAPI

Provided by:  MLaaSLibrary

Required by:  MLModelUpdateProcessor

Description : This API provides a number of operations for managing the models deployed on the MLaaS cloud.

Operations:

- **Credential** **authenticate()** throws *InvalidAPIKeyException*
 - Effect: Construct a **Credential** object for authenticating requests with the MLaaS service provider. This operation uses the previously configured API key to construct the **Credential** object. If such an API key is not configured, an exception will be thrown.
 - Returns: Authentication token for subsequent requests.
 - Sequence Diagrams: figs. D.3 and D.4
- **void delete(MLModel model, Credential token)** throws *AuthorizationException*
 - Effect: Delete a model from the online MLaaS service.
 - Sequence Diagrams: None
- **MLModelDescription getModelDescription(List<MLModelDescription> descriptions, String modelId)**
 - Effect: This method retrieves the correct **MLModelDescription** that corresponds with the **MLModel** identified by the provided **modelId**.
 - Returns: The **MLModelDescription** corresponding with the **MLModel** identified by the **modelId**.
 - Sequence Diagrams: None
- **String getModelId(MLModelDescription description)**
 - Effect: Extract the **modelId** from the **MLModelDescription**.
 - Returns: String with the **modelId** of the **MLModel** specified in the **MLModelDescription**.
 - Sequence Diagrams: fig. D.3
- **void register(MLModel model, Credential token)** throws *AuthorizationException*
 - Effect: Register a new model with the MLaaS service.
 - Sequence Diagrams: None
- **void setAPIKey(String apiKey)** throws *AuthenticationException*
 - Effect: Set the credentials for authenticating with the ML-as-a-Service provider.
 - Sequence Diagrams: None

Diagrams: None

E.3.12 MLaaSService

Provided by:  MLaaS

Required by:  MLModelManager,  MLModelUpdateProcessor,  eHealth Platform

Description : This interface offers operations for interacting with MLaaS service providers.

Operations:

- **List<MLModelUpdate> fetchModelUpdates(Credential credential, MLModelDescription model)**
 - Effect: Fetch a list of model updates to apply to the local **MLModel**.
 - Returns: The list of **MLModelUpdates** to apply locally.

- Sequence Diagrams: fig. D.4
- **List<MLModelDescription> getAvailableModels(Credential credential)**
 - Effect: Get the list of available ML models to use for making predictions on incoming sensor data.
 - Returns: The list of available MLModels.
 - Sequence Diagrams: fig. D.3
- **void pushModelUpdates(Credential credential, MLModelDescription model, List<MLModelUpdate> updates)**
 - Effect: Send a list of MLModelUpdates to the online service.
 - Sequence Diagrams: None
- **MLModel retrieveModel(Credential credential, MLModelDescription model)**
 - Effect: Retrieve a specific **MLModel** corresponding with the provided **MLModelDescription**.
 - Returns: The requested **MLModel**.
 - Sequence Diagrams: fig. D.3

Diagrams: None

E.3.13 MLAPI

Provided by: MLLibrary

Required by: ModelUpdater, SensorDataRiskAssessment

Description : This is the API provided by the MLLibrary to use **MLModels** locally for making predictions and processing updates to these models.

Operations:

- **MLModel applyUpdate(MLModel model, MLModelUpdate update)**
 - Effect: Applies the provided **MLModelUpdate** on the provided **MLModel** and returns the resulting **MLModel**.
 - Returns: The resulting **MLModel** after applying the requested **MLModelUpdate**.
 - Sequence Diagrams: fig. D.4
- **Prediction predict(SensorDataPackage sensorData)**
 - Effect: Make a prediction with the active **MLModel** on the provided sensordata.
 - Returns: The prediction based on the provided sensor data.
 - Sequence Diagrams: fig. D.2
- **void verify(MLModel model) throws InvalidMLModelException**
 - Effect: Verifies that a provided **MLModel** is working correctly after applying **MLModelUpdates**.
 - Sequence Diagrams: fig. D.4

Diagrams: None

E.3.14 MLModelMgmt

Provided by: MLModelManager, MLModelStorageManager

Required by: MLModelRetrieval, RiskEstimationProcessor

Description : The interface provides operations for managing different **MLModels** locally, querying available models for making predictions on newly incoming sensor data.

Operations:

- **MLModel fetchModel(String modelId) throws NoSuchMLModelException**
 - Effect: Fetch the **MLModel** corresponding with the provided modelId.
 - Returns: The requested **MLModel**.
 - Sequence Diagrams: fig. D.2
- **List<String> getAvailableModels()**
 - Effect: Get the list of available **MLModels** that can be used for making **Predictions** locally.
 - Returns: The list of identifiers of the available **MLModels** that can be used.
 - Sequence Diagrams: None
- **void synchronizeModels()**
 - Effect: Synchronize the list of locally-stored **MLModels** by checking for newly-available **MLModels** that are not stored in the local MLModelRepository and retrieving these **MLModels** from the MLaaS provider.
 - Sequence Diagrams: fig. D.3
- **void updateModel(String modelId) throws NoSuchMLModelException**

- Effect: Update the model with the specified id by fetching **MLModelUpdates** and applying them to the model.
- Sequence Diagrams: fig. D.4

Diagrams: None

E.3.15 MLModelStorage

Provided by: ¶ MLModelRepository

Required by: ¶ MLModelStorageManager, ¶ MLModelUpdateProcessor

Operations:

- boolean **isAvailable**(String modelId)
 - Effect: Check whether an **MLModel** (identified by the modelId) is available in the local MLModelRepository.
 - Returns: Boolean indicating if the specified model is present in the repository.
 - Sequence Diagrams: figs. D.3 and D.4
- **MLModel** **retrieveModel**(String modelId)
 - Effect: Retrieve the **MLModel** with the specified id.
 - Returns: The requested **MLModel**.
 - Sequence Diagrams: fig. D.4
- void **storeModel**(**MLModel** model)
 - Effect: Store an **MLModel** in the local MLModelRepository
 - Sequence Diagrams: figs. D.3 and D.4

Diagrams: None

E.3.16 MLModelSync

Provided by: ¶ MLModelUpdateProcessor

Required by: ¶ MLModelStorageManager

Operations:

- void **fetchNewMLModels**()
 - Effect: Fetches newly available **MLModels** and store them in the MLModelRepository.
 - Sequence Diagrams: fig. D.3
- void **processMLModelUpdates**(String modelId) throws *NoSuchMLModelException*
 - Effect: Retrieve and process the updates for the specified **MLModel** and store the updated **MLModel** in the MLModelRepository.
 - Sequence Diagrams: fig. D.4

Diagrams: None

E.3.17 MLModelUpdateMgmt

Provided by: ¶ ModelUpdater

Required by: ¶ MLModelUpdateProcessor

Operations:

- **MLModel** **processMLModelUpdates**(**MLModel** model, List<**MLModelUpdate**> updates)
 - Effect: Requests the ModelUpdater to apply the list of **MLModelUpdates** on the provided **MLModel**.
 - Returns: The **MLModel** resulting from the application of the list of **MLModelUpdates** on the initial **MLModel**.
 - Sequence Diagrams: fig. D.4

Diagrams: None

E.3.18 OtherDataMgmt

Provided by: ¶ OtherFunctionality

Required by: ¶ ClinicalJobCreator, ¶ ClinicalModelCache, ¶ RiskEstimationCombiner, ¶ RiskEstimationScheduler


Operations:


- **PatientStatus** **getPatientStatus**(**PatientId** patientId) throws *NoSuchPatientException*

- Effect: Fetch and return the status of the patient identified by the patientId.
- Returns: The status of the patient.
- Sequence Diagrams: None
- void setEstimatedPatientStatus(**PatientId** patientId, **PatientStatus** estimatedStatus, **Timestamp** estimationTime) throws *NoSuchPatientException*
 - Effect: Update the patient status estimation of the patient identified by patientId to the given value estimatedStatus and update the time of estimation to estimationTime. The time of estimation is the time at which the corresponding estimation job for this patient was completed.
 - Sequence Diagrams: fig. D.6

Diagrams: None

E.3.19 PatientRecordMgmt

Provided by:  OtherFunctionality


Required by:  RiskEstimationCombiner



Operations:

- **PatientRecord** getPatientRecord(**PatientId** patientId) throws *NoSuchPatientRecordException*
 - Effect: This will fetch the EHR record of the patient with given id from the HIS and return it. If the HIS is not available, an older (cached) copy of the patient record is returned if possible.
 - Returns: The requested patient record.
 - Sequence Diagrams: None
- void updateRiskLevel(**PatientId** patientId, **PatientStatus** status, **SensorDataPackage** sensorData) throws *NoSuchPatientException*
 - Effect: This will update the risk level of the patient identified by patientId through the HIS. If sensorData is given, it will be stored in the patient record as well.
 - Sequence Diagrams: figs. D.5 and D.6

Diagrams: None

E.3.20 Results

Provided by:  RiskEstimationCombiner


Required by:  JobProcessor,  RiskEstimationProcessor


Operations:

- void addClinicalModelJobResults(**ClinicalModelJobId** jobId, **ClinicalModelJobResult** results)
 - Effect: Sends the result of the performed clinical model computation (identified by the jobId) to the **RiskEstimationCombiner** for combining with the other partial results belonging to the same risk estimation.
 - Sequence Diagrams: figs. D.2 and D.6

Diagrams: None

E.3.21 SensorDataAnalysis

Provided by:  SensorDataRiskAssessment

Required by:  JobProcessor



Operations:

- void performModelPrediction(**ClinicalModelJob** clinicalModelJob, **MLModel** model)
 - Effect: Perform a prediction using the provided **MLModel** and the data provided in the clinicalModelJob definition. Afterwards the results are submitted for combination by the **RiskEstimationCombiner**.
 - Sequence Diagrams: fig. D.2

Diagrams: None

E.3.22 SensorDataMgmt

Provided by:  OtherFunctionality

Required by:  ClinicalJobCreator,  RiskEstimationScheduler

Operations:

- void addSensorData(**PatientId** patientId, **SensorDataPackage** sensorData, **Timestamp** receivedAt)

- Effect: Store the given sensor data and meta-data.
- Sequence Diagrams: None
- Map<**Timestamp**, **SensorDataPackage**> getAllSensorDataOfPatient(**PatientId** patientId)
 - Effect: This will fetch and return all sensor data belonging to the patient identified by patientId
 - Returns: The sensor data and the timestamp of their arrival in the system.
 - Sequence Diagrams: None
- Map<**Timestamp**, **SensorDataPackage**> getAllSensorDataOfPatientBefore(**PatientId** patientId, **Timestamp** stopTime)
 - Effect: Fetch and return all sensor data belonging to the patient identified by patientId which was received before the specified time stopTime.
 - Returns: The sensor data and the timestamp of their arrival in the system.
 - Sequence Diagrams: fig. D.1

Diagrams: None

E.4 Nodes

E.4.1 eHealthPlatform

Responsibility: Container node representing the deployment context boundary for the eHealth Platform.

Visible on diagrams: fig. C.1

E.4.2 MLaaS Conn (Pilot Deployment)

Responsibility: This node is responsible for the connection with the MLaaS Service.

Visible on diagrams: fig. C.3

E.4.3 MLaaS Service (Dev Deployment)

Responsibility: This represents the MLaaS cloud service.

Visible on diagrams: fig. C.4

E.4.4 MLaaS Service (Pilot Deployment)

Responsibility: This represents the cloud MLaaS service.

Visible on diagrams: fig. C.3

E.4.5 MLaaS Cloud

Responsibility: Cloud infrastructure of the MLaaS Service provider.

Visible on diagrams: fig. C.1

E.4.6 MLaaSConnectorNode

Responsibility: Node responsible for handling connections with external MLaaS providers.

Visible on diagrams: figs. C.1 and C.2

E.4.7 RiskEstimationManagementNode (Pilot Deployment)

Responsibility: The RiskEstimationManagement node hosts the creation, scheduling and combining of the clinical jobs.

Visible on diagrams: fig. C.3

E.4.8 RiskEstimationManagmentNode (Dev Deployment)

Responsibility: The development deployment is a single RiskEstimationManagement node that contains all other functionality, job creation, scheduling, and combination.

Visible on diagrams: fig. C.4

E.4.9 RiskEstimationMgmtNode

Responsibility: The RiskEstimationMgmtNode keeps track of the risk estimation jobs, and schedules and combines the results.

Visible on diagrams: fig. C.2

E.4.10 RiskEstimationProcessorNode

Responsibility: Multiple RiskEstimationProcessor nodes enable the processing of multiple sensorData inputs in parallel.

Visible on diagrams: fig. C.2

E.4.11 RiskEstimationProcessorNode (Dev Deployment)

Responsibility: The development deployment only uses two RiskEstimationProcessorNodes to run **ClinicalModelJobs** in parallel.

Visible on diagrams: fig. C.4

E.4.12 RiskEstimationProcessorNode (Pilot Deployment)

Responsibility: There are five instances of the RiskEstimationProcessorNode to enable the processing of multiple **ClinicalModelJobs** in parallel.

Visible on diagrams: fig. C.3

E.4.13 TODO Node (Pilot Deployment)

Responsibility: This TODO node for the pilot deployment contains the **OtherFunctionality** of the system that has not been worked out in detail yet.

Visible on diagrams: fig. C.3

E.4.14 TODONode

Responsibility: Temporary node that requires further decomposition.

Visible on diagrams: fig. C.2

E.5 Exceptions

- *AuthenticationException* (Δ Exception):
Signals a problem with the credentials used for authentication. These may be invalid or expired.
- *AuthorizationException* (Δ Exception):
Signals that the principal has insufficient access rights to perform the operation.
- *Exception*:
Subtypes: Δ AuthenticationException, Δ AuthorizationException, Δ InvalidAPIKeyException, Δ InvalidMLModelException, Δ IOException, Δ NoSuchMLModelException, Δ NoSuchPatientException, Δ NoSuchPatientRecordException
Base exception class.
- *InvalidAPIKeyException* (Δ Exception):
Thrown when no correctly formatted API key is configured for constructing authentication Credential[s].
- *InvalidMLModelException* (Δ Exception):
Thrown when a provided MLModel is invalid.
- *IOException* (Δ Exception):
Thrown when an IO operation fails.
- *NoSuchMLModelException* (Δ Exception):
Thrown when the requested MLModel cannot be found.
- *NoSuchPatientException* (Δ Exception):
Thrown if no patient with the specified identifier exists.

- *NoSuchPatientRecordException* (Δ *Exception*):
Thrown if no patient record for the patient with the given identifier exists in the HIS.

E.6 Data types

- **ClinicalModel:**
Attributes: String id, String name, String description, List<String> requiredParameters, List<String> optionalParameters, List<String> applicableMLModelIds, boolean requiresHistoricalSensorData
A clinical model. This model has an id, name, description. It contains a list of parameters for using the model and a set of MLModelIds to specify which types of MLModels can be used in the context of the ClinicalModel.
- **ClinicalModelConfiguration:**
Attributes: Map<String, String> modelParameters
The configuration for a clinical model. Represented as a set key/value pair for the parameters of the clinical model.
- **ClinicalModelJob:**
Attributes: **ClinicalModel** clinicalModel, **ClinicalModelJobId** jobId, **SensorDataPackage** sensorData, **PatientId** patientId, **ClinicalModelConfiguration** clinicalModelConfiguration, **PatientRecord** patientRecord, Map<**Timestamp**, **SensorDataPackage**> historicalSensorData
A single job for the computation of a clinical model. This contains a ClinicalModelJobId, ClinicalModelId and PatientId respectively identifying the job itself, the corresponding clinical model and the patient. If the corresponding risk estimation the job belongs to is triggered by the arrival of a sensor data package, this sensor data is also contained.
- **ClinicalModelJobId:**
A piece of data uniquely identifying a certain clinical model job in the system. This architecture does not specify the exact format of this identifier, but possibilities are a long integer, a string, a URI etc.
- **ClinicalModelJobResult:**
The result of the computation of a clinical model containing the estimated risk level for the patient. The resulting risk score for ClinicalModels are on the same unit scale so the the results of multiple ClinicalModels can be combined.
- **Credential:**
Authentication object for making authenticated requests to the MLaaS service provider.
- **MLModel:**
Attributes: String modelId, **MLModelDescription** modelDescription
Machine learning model for making predictions based on sensordata. The actual internal encoding of the model itself is left open.
- **MLModelDescription:**
Attributes: String uuid, String name, String description, String providerName
Object providing the MLModel meta data. It does not contain the model itself.
- **MLModelUpdate:**
Attributes: **MLModelDescription** modelDescription
Represents a model update that can be applied on an MLModel. These model updates enable exchanging changes to the MLModel between different parties without sending the actual MLModel to the other parties.
- **Object:**
Root class for objects that can be stored in the Key-Value storage service. Data types that need to be saved need to inherit from this class.
- **PatientId:**
A piece of data uniquely identifying a patient in the system. This architecture does not specify the exact format of this identifier, but possibilities are a long integer, a string, a URI, etc.
- **PatientRecord:**
The structured contents of the EHR record of a certain patient, including medication history, recent treatments, allergies, etc.
- **PatientStatus:**
The status of a patient, i.e., their risk level.

- **Prediction:**
A prediction from the MLModel.
- **RiskEstimationId:**
A piece of data uniquely identifying a risk estimation performed by the PMS. This architecture does not specify the exact format of this identifier, but possibilities are a long integer, a string, a URL etc.
- **SensorDataElement:**
Attributes: String sensorId, String sensorType, String measurementType, String measurementUnit, **BigDecimal** measurementValue, String measurementValueString
Each sub-package lists the id of the sensor, the type of the sensor, the type of the measurements and the measurements itself. These measurements range from a single value (e.g., in case of the blood pressure) to a complex data structure (e.g., in case of an ECG). Non-numerical types of measurements can be captured in the measurementValueString attribute.
- **SensorDataPackage:**
Attributes: List<**SensorDataElement**> sensorDataElements
A package of sensor data, i.e., a list of sensorDataElements.
- **Timestamp:**
The representation of a time (i.e., date and time of the day) in the system.

E.7 Unresolved issues

SA Plugin v6.0.8 (VP OpenAPI v16.3)

No failed checks