Personalised Health Monitoring and Decision Support Based on Artificial Intelligence and Holistic Health Records

D3.8 – Standardisation and Quality Assurance of Heterogenous Data II

WP3 Personalised Holistic Health Records

Dissemination Level: Public
Document type: Report
Version: 1.0
Date: August 31, 2022

The project iHelp has received funding from the European Union's Horizon 2020 Programme for research, technological development, and demonstration under grant agreement no 101017441.

Document Details

Project Number	101017441
Project Title	iHelp - Personalised Health Monitoring and Decision Support Based on Artificial Intelligence and Holistic Health Records
Title of deliverable	Standardisation and Quality Assurance of Heterogenous Data II
Work package	WP3 Personalised Holistic Health Records
Due Date	August 31, 2022
Submission Date	August 31, 2022
Start Date of Project	January 1, 2021
Duration of project	36 months
Main Responsible Partner	UPRC
Deliverable nature	Report
Authors' names	George Manias, Eleftheria Kouremenou (UPRC), Fabio Melillo (ENG), Ainhoa Azqueta (UPM)
Reviewers' names	Athanasios Dalianis (ATC), Aristodemos Pnevmatikakis (iSPRINT)

Document Revision History

Version History			
Version	**Date**	**Author(s)**	**Changes made**
0.1	2022-06-30	George Manias (UPRC)	Initial version and Table of Contents
0.2	2022-07-04	George Manias (UPRC)	Input in Sections 3 and 5
0.3	2022-07-15	Fabio Melillo (ENG)	Input provided in Section 6
0.4	2022-07-18	Ainhoa Azqueta (UPM)	Input provided in Sections 4
0.5	2022-07-21	Eleftheria Kouremenou (UPRC)	Input Update
0.6	2022-07-25	George Manias, Eleftheria Kouremenou (UPRC)	Input Update in Section 8 and Section 1.3 added
0.7	2022-08-01	George Manias (UPRC)	Internal Review and input update
0.8	2022-08-09	Athanasios Dalianis (ATC)	1st Internal Review
0.9	2022-08-14	Aristodemos Pnevmatikakis (iSPRINT)	2nd Internal Review
0.9.1	2022-08-25	George Manias (UPRC)	Update and revision of the document based on

			comments and feedback from Internal Reviewers
0.9.2	2022-08-29	Pavlos Kranas (LXS)	Quality Review
1.0	2022-08-31	Dimosthenis Kyriazis (UPRC)	Final version

Table of Contents

Table of Figures

Executive summary

This deliverable (titled “Standardisation and Quality Assurance of Heterogenous Data II”) describes the initial implementations and first prototypes of the iHelp Standardisation and Quality Assurance Mechanism. As introduced in D3.7 – “Standardisation and Quality Assurance of Heterogenous Data I”, the Standardisation and Quality Assurance Mechanism is a unified and integrated mechanism consisting of three (3) core sub-components, the Data Cleaner, the Data Qualifier, the Data Harmonizer, and two (2) integrated sub-components: the Primary Data Mapper and the Secondary Data Mapper, which are responsible for providing the mapping operations between the raw data resources and the Holistic Health Records (HHRs) resources. This holistic mechanism is the core component that seeks to provide various cleaning, pre-processing, harmonization, and mapping functionalities and services on the incoming raw data. Specifically, it provides to the wider research and healthcare community a wide range of innovative solutions for the cleaning, qualification, transformation, harmonization, and mapping of raw healthcare-related data.

The current document builds upon the initial design and specifications of D3.7 – “Standardisation and Quality Assurance of Heterogenous Data I”, aiming to provide the initial implementation, the first prototypes and a concrete overview of how the proposed mechanism integrates with the overall architecture of the iHelp platform and other components in the Data Ingestion pipeline, and specifically (i) how to retrieve the incoming data from the Data Gateways; (ii) how to interexchange data with the already identified project’s message bus; and (iii) how to send the final processed, transformed and HHR aligned data to iHelp’s Big Data Platform.

To support the aforementioned functionalities, the iHelp Standardisation and Quality Assurance Mechanism specifications exemplify the respective sub-components, the overall data pipeline and workflow, the internal functionalities supported by each sub-component, the interaction points with different components as well as the technical details that drive the implementation and deployment of this holistic mechanism. Finally, this document - in different subsections, e.g., in Section 5.2, - seeks to review the current state of the art in order to identify the baseline technologies and approaches for the realization of the implemented Standardisation and Quality Assurance Mechanism.

1 Introduction

iHelp aims to develop standardisation and quality procedures to ensure that the data modelling, transformation, and management operations facilitate data sharing and analytics, not only in the context of the iHelp project, but also in the wider healthcare ecosystem. Given the challenge that in modern societies healthcare-related data are being obtained from various sources and in divergent formats, this deliverable aims to provide technologies for dealing with this issue through the harmonization and transformation of the raw collected health data into the project's common HHR format, through finding common links or similarities between primary and secondary data types and available HHR resources. Moreover, these raw data also include missing values, non-matching words and partially overlapping concepts, hence state-of-the-art cleaning approaches and techniques are utilized to provide cleaned and qualified data.

In summary, the Standardisation and Quality Assurance Mechanism consists of five (5) sub-components with many different internal architectures and functionalities to cover all the different requirements of the project's stakeholders and the overall data processing and data mapping procedures that are implemented. In the following Sections of this document, the implementation and utilization details of the iHelp Standardisation and Quality Assurance Mechanism are analysed in detail.

1.1 Objective of the Deliverable

The main objective of this deliverable is to provide the software demonstrations for the subcomponents of the iHelp Standardisation and Quality Assurance Mechanism and to report the work that has been conducted in the context of task T3.4 ("Standardisation and Quality Assurance of Heterogeneous Data") at this phase of the project. On top of this, it introduces and analyses the internal architecture and data workflow, the initial design and the specifications of this holistic mechanism, and the first prototypes that have been implemented based on them. In addition, the deliverable outlines the main sub-components and their internal functionalities and clarifies the reason for their existence and particular implementation. Moreover, the internal integration between these sub-components, as well as the external integration with other components of the iHelp Data Ingestion pipeline, are analysed. Finally, this deliverable seeks to describe the overall positioning of the Standardisation and Quality Assurance Mechanism into the iHelp platform.

1.2 Structure of the Deliverable

This document is structured as follows: Section 1 introduces the deliverable, its main objectives and the updates on top of the first iteration of this series of deliverables (i.e., D3.7 – "Standardisation and Quality Assurance of Heterogenous Data I"), while Section 2 introduces the architecture, the objectives and the overview of the Standardisation and Quality Assurance Mechanism, discussing its scope, and the basic concepts and key features of its design. The subsequent sections introduce the five (5) main sub-components of the Standardisation and Quality Assurance Mechanism and analyse the implementation of the first prototypes implementing them, and the integration activities. Specifically, Section 3 covers the data cleaning, data verification and data validation aspects focusing on the Data Cleaner sub-component, Section 4 focuses on the data qualification functions of the Data Qualifier sub-component, while Section 5 focuses on transformation and harmonisation solutions to support interoperability and transformation of incoming data through the analysis and description of the Data Harmonizer sub-component. On top of this,

Section 6 focus on the primary data mapping and the secondary data mapping functionalities respectively, hence the sub-mechanisms of the Primary Data Mapper and the Secondary Data Mapper, that are incorporated and strongly integrated with the Data Harmonizer sub-component, are further analysed. Finally, Section 7 concludes the document and states any future work and deliverable concerning the task T3.4 ("Standardisation and Quality Assurance of Heterogeneous Data").

1.3 Updates since D3.7

This document is consistent with D3.7 – "Standardisation and Quality Assurance of Heterogenous Data I" which provided the overall architecture and initial design and specifications of Standardisation and Quality Assurance Mechanism and its internal sub-components. It provides more technical details of the implementation of the software components and the first prototypes that were developed in accordance with the initial design and specifications. In this context, an overview of the first prototypes and the ways to further exploit and utilize them are introduced in this deliverable. What is worth to mention is that a major update and revision in the internal architecture and functionality of the Data Harmonizer has been identified and introduced in this deliverable. More specifically, the Automated Machine Translation subfunction has been dropped, as it is not required by any pilot scenario, since the primary and secondary data that are fetched and populated into the iHelp platform are already translated into the English language. Moreover, an Ontology-based Domain Terminology Mapping mechanism has been introduced to integrate and map different medical terminology standards into the iHelp project. An additional revision is also introduced in the context of the Data Qualifier. The "Reliability Window" subfunction has updated to "Datasource Qualifier", hence offering a better description and explainability of the overall functionality and goal of this subfunction. In addition, the interfaces introduced in D3.7 have been further updated and revised in this deliverable, as a result of the integration phase and the overall implementation of the end-to-end Data Ingestion pipeline of the iHelp project. Finally, it should be noted that, as the project progresses, the Standardisation and Quality Assurance Mechanism will be further evaluated and perhaps revised. The final possible set of updates and enhancements will be reported in D3.9 – "Standardisation and Quality Assurance of Heterogenous Data III" that is the last iteration of this series of deliverables.

2 Standardisation and Quality Assurance Mechanism

This section describes the scope and the overall architecture of the iHelp Standardisation and Quality Assurance Mechanism, while it also places this holistic mechanism into the iHelp data ingestion pipeline and further analyses its correlation and integration with other technical components and mechanisms of the project.

2.1 Overview

Nowadays, the healthcare domain faces various challenges related to the diversity and variety of data, their huge volume, and their distribution, thus processing and analysis of these data become more and more complex and challenging. Hence, approaches, applications, and solutions to address issues that derive from the wealth of healthcare Big Data are vitally important. The collection, the quality estimation, as well as the interpretation and the harmonization of the data, that derive from the existing huge amounts of heterogeneous medical devices and data sources, face a dramatic increase of interest in the healthcare domain. To this end, to address all these issues this specific task of the iHelp project has four objectives and targets. On one hand it seeks to assure the incoming data accuracy, integrity, and quality, while on the other hand it seeks to perform different types of transformation, interoperability, and integration operations on the raw data. On top of this, this task aims – through the utilization of specific measures and rules – to ensure data quality and to provide a predictive selection mechanism for achieving data sources' reliability during runtime and for providing the decision whether a connected data source is considered as reliable or not. Finally, advanced data mappers are introduced and implemented to provide an automated structure mapping mechanism between data resources and widely known and approved in the healthcare domain HHR resources and data model (K., M., M., + 19) following the mapping approaches that are identified under the scopes of task T3.1 ("Data Modelling and Integrated Health Records") of the iHelp project.

2.2 Internal Architecture

As presented in the above sub-section the Standardisation and Quality Assurance Mechanism seeks to enhance the quality, interoperability and harmonization of the incoming data, and the extraction of valuable information and knowledge out of them. To this end, three (3) core sub-components have been identified and are incorporated into the internal workflow and architecture of the Standardisation and Quality Assurance Mechanism. These specific three (3) subcomponents are being depicted in Figure 1 and are the Data Cleaner, the Data Qualifier, and the Data Harmonizer. The implementation and utilization of the above-mentioned components aim to enhance the value of the incoming/raw data.

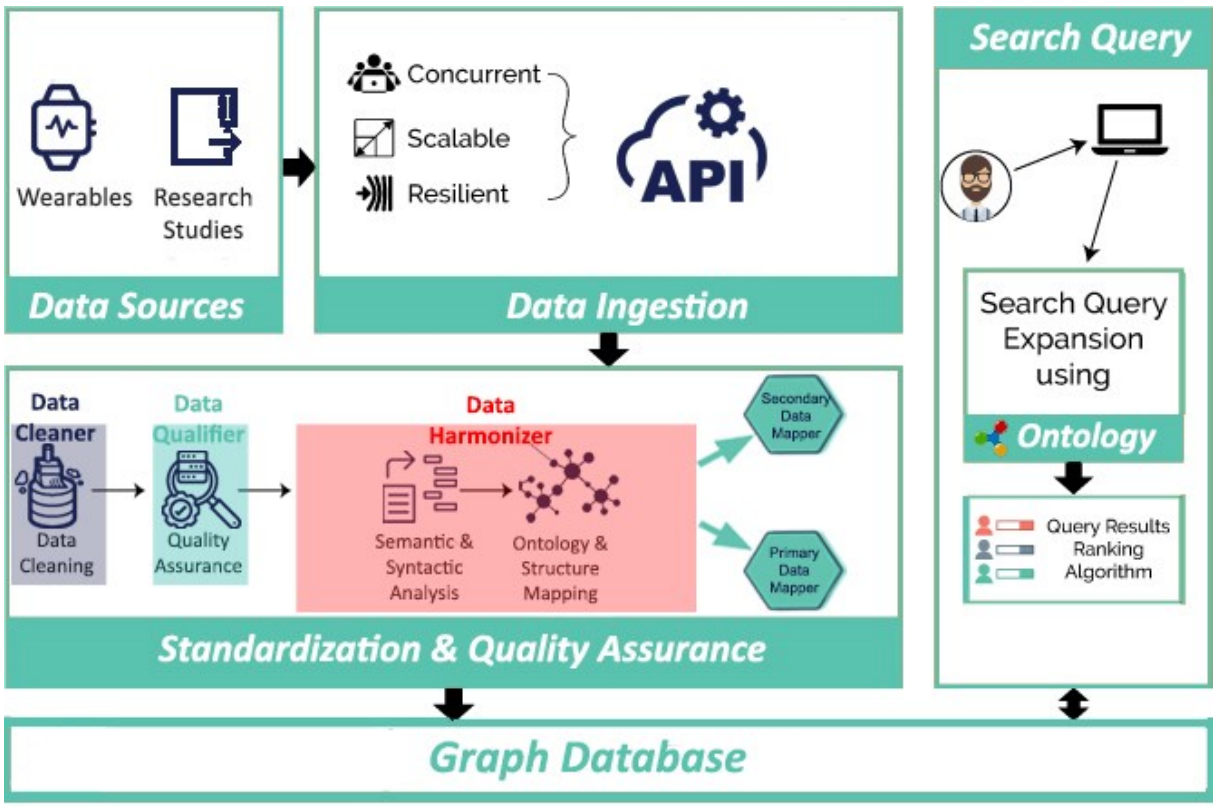

Figure 1: Standardisation & Quality Assurance mechanism.

Following the Data Ingestion step, one of the preliminary actions of this mechanism is to deal with data derived from data sources of different reliability, that may have produced uncleaned and faulty data. Thus, from the very beginning of the overall processing pipeline it aims to clean all the collected data and to measure and evaluate the quality of both the connected data sources and their produced data. To successfully achieve that, the mechanism exploits two (2) separate modules, the Data Cleaner sub-component and the Data Qualifier sub-component. Sequentially, in the harmonization phase, the interpretation and transformation of the collected cleaned and reliable data takes place through the implementation and utilization of the Data Harmonizer. The latter incorporates two (2) sub-mechanisms, the Semantic and Syntactic Analysis mechanism and the Ontology and Structure Mapping Mechanism, in order to transform the cleaned and reliable data and to provide interoperable and harmonized data. Finally, the transformed and interoperable data are mapped to the common HHR format through the utilization of the Primary and Secondary Data Mapper, external sub-components and closely integrated with the Data Harmonizer, as also depicted in Figure 1. The specifications, internal architectures and implementation procedures of all the above introduced sub-components are further explained and detailed in the next sections of this deliverable. At this point, it should be noted that this deliverable includes only a brief introduction and description regarding the Secondary Data Mapper sub-components, as this sub-component will start being designed and implemented after the final modelling and establishment of the common HHR model and format that will be followed across the iHelp project and platform and that will be introduced in the context of task T3.1. Moreover, as the project progresses and in collaboration and alignment with the parallel work that is being implemented under the scopes of T3.1, more details about the implementation and architecture of these two sub-components will be outlined.

2.3 Overall Objectives

Data have long been a critical asset for organizations, businesses, and governments and their analysis is of major importance for every stakeholder in order to be able to handle and extract value and knowledge out of them. The advances in the fields of IoT, cloud computing, edge computing and mobile computing have led to the rapidly increasing volume and complexity of data, thus the concept and term of Big Data have experienced enormous interest and use over the last decade. The spectacular growth in the creation, storage, sharing and consumption of data during the last decade indicates the need for modern organizations to fuse advanced analytical techniques with Big Data in order to deal with them and to get significant value from them. Hence, Big Data and their analysis facilitate personalised healthcare and risk assessment, therefore clinicians are able to identify patients who are eligible for appropriate treatments, which results in savings of time and cost. On top of this, R&D on personalised healthcare makes diagnostics smarter and more targeted, like in the case of Pancreatic Cancer, while early identification and personalised treatments can help in the design of improved screening programs and can also allow people to live longer, healthier, and more productive lives. The ability to identify which preventive measure and intervention is delivering the desired impact can massively help in the development of new diagnostic and treatment regimes. Especially when it comes to the healthcare domain, the successful exploration and interpretation of all these data play a vital role (J., F. + 20). On top of this, healthcare data are available in different formats (e.g., images, signals and wavelengths) and may derive from different healthcare stakeholders (i.e. patients themselves, healthcare professionals, etc.). Hence, many healthcare organizations find themselves overwhelmed with data, but lacking truly valuable information. At the same time, due to the improvement in the automatic collection of data from medical devices and systems, researchers and analysts can monitor data or information that can be accessed in electronic configuration (Pan. 16).

Moreover, the term *Big Data* defines a two-fold meaning in these data. On one hand, it describes a change in the quality and type of data that modern healthcare organizations possess, which has potential impacts throughout the entire healthcare domain and stakeholders. On the other hand, it describes a massive volume of both structured and unstructured data that is huge and complicated to be processed using traditional database and software techniques (C., P. 14). On top of this, unstructured data can be defined as data that do not comfort in predefined data models and traditional structures that can be stored in relational databases. Data generated by medical reports, medical advice, texts in questionnaires or even posts on modern social networks are such types of unstructured data and their main characteristic is that they include information that is not arranged according to a predefined data model or schema. Therefore, these types of data are usually difficult to be managed, and as a result analysing, aggregating, and correlating them in order to extract valuable information and knowledge is a challenging task. Hence, deriving value and knowledge from this type of data based on the analysis of their semantics, meanings and syntactic is of major importance (M., B. + 10). The latter demonstrates the need for the modern stakeholders in the healthcare domain to implement techniques, mechanisms, and applications that focus their operations on the concept of providing cleaned, qualified and interoperable data, for offering more precise and personalised prevention & intervention measures, higher experience for patients' health monitoring, risk assessment and personalised decision support, hence actionable and valuable knowledge as depicted in Figure 2.

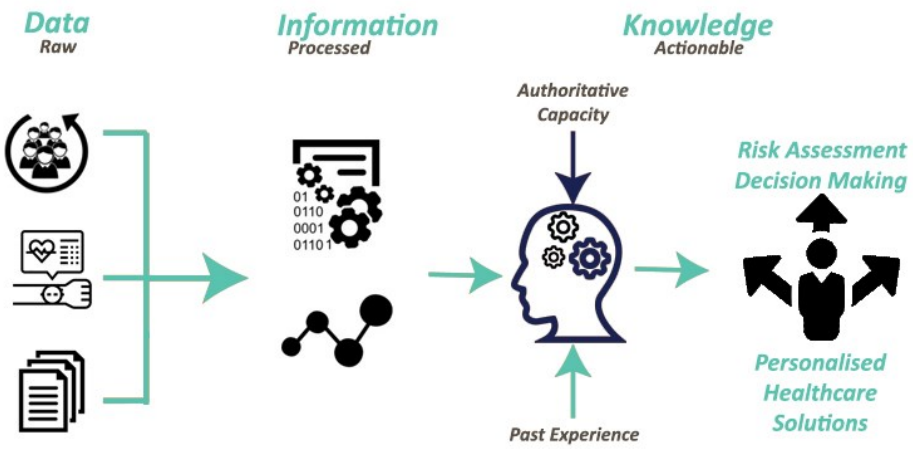

Figure 2: Data to Knowledge path.

To this end, the Standardisation & Quality Assurance mechanism is the key mechanism for addressing all the above issues and challenges and for deriving actionable knowledge from the raw data. The initial design of this holistic mechanism is based on the following basic objectives, to be addressed through the implementation of the different sub-functions that have been introduced in previous sub-sections.

- *Data Cleaner* aims to assure the incoming data's accuracy, integrity, and quality.
- *Data Qualifier* aims to provide a decision whether a connected data source will be considered as reliable or not.
- *Data Harmonizer* aims to:
  - support data coming from divergent sources in order to deal with different formats and to enhance the interoperability of data.
  - provide automated health data transformation to the identified HHR format.
  - consolidate data physically and virtually into knowledge graphs.
- *Primary Data Mapper* aims to provide a structure mapping mechanism between primary data resources and HHR data format.
- *Secondary Data Mapper* aims to provide a structure mapping mechanism between secondary data resources and HHR data format.

2.4 Positioning and Relations with other components

As presented in Section 6 of the D2.4 – "Conceptual model and reference architecture I", the iHelp platform consists of four main big blocks that represent core functionalities and solutions that the iHelp project aims to provide. One of these main blocks is the Data Ingestion block, which represents all those components that are used to fetch, process and store data derived from heterogeneous sources. Processed data will comply with the common HHR data model and will be stored into the project's central Data Storage.

Moreover, as it has been introduced by the above sub-sections of this deliverable, the Standardisation and Quality Assurance Mechanism has strong integration and dependencies with various components that are part of the overall data ingestion pipeline, which is presented in Figure 3. At first, raw data from heterogeneous data sources should be fetched into the iHelp platform. Once the data are profiled and encoded into the appropriate data schema, they are ingested either as batches or through a streaming process. The overall ingestion process is being accomplished through the utilization of Data Connectors and

the Data Gateway of the project that will be designed and implemented under the scopes of task T3.2 “Primary Data Capture and Ingestion”. Once data are extracted from source systems, their structure or format need to be adjusted, therefore these raw data should be further cleaned, processed and harmonized. To this end, the Standardisation and Quality Assurance Mechanism and its sub-components will be utilized. Then, quality assured, cleaned and harmonized data should be mapped and transformed into the project’s common format and model, the HHR. Hence, the Primary and Secondary Mapper sub-functions that will also be implemented under the scopes of this task, will rely on the specification of the common data model, the HHR, that will be defined under the scopes of T3.1 (“Data Modelling and Integrated Health Records”). Finally, the HHR Importer mechanism, which will be provided by the T4.4 (“Big Data Platform and Knowledge Management System”), is responsible to store data into the relational data schema of the iHelp’s Big Data Platform. To summarize, the Standardisation and Quality Assurance Mechanism can be utilized every time a sample of data or a new dataset is ingested into the iHelp platform. Once, the data are ingested and become available into the whole iHelp platform by the Data Gateway component, then this holistic mechanism is responsible for the final processing, cleaning, qualification, harmonization, and transformation to the needed HHR format of these data.

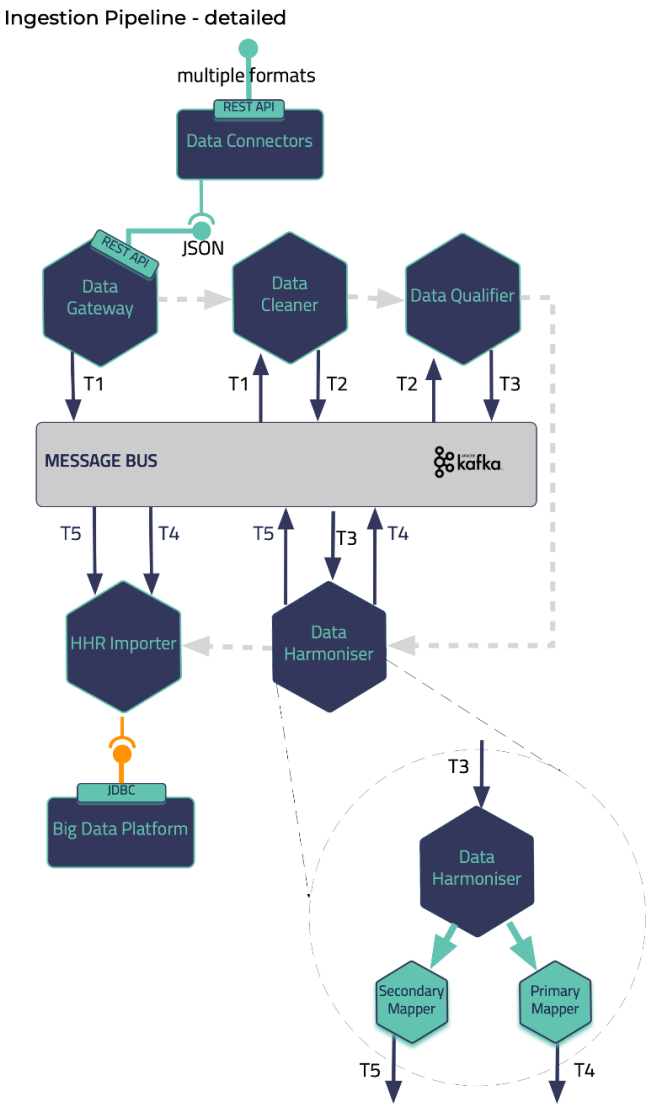

Figure 3: iHelp Data Ingestion pipeline.

Futhermore, as depicted in Figure 3, all the components that are being incorporated into the Data Ingestion pipeline of the iHelp platform will inter-exchange their input and output data through the utilization of specific Kafka queues. Kafka[1] has been identified as the project's internal message bus and data inter-exchange system. The latter implies that each of these sub-components involved into the Data Ingestion pipeline will consume data from one Kafka topic and will produce the output into another Kafka topic, so that the next involved sub-component can retrieve the processed data.

At this point, it should be noted that as the initial software prototypes of the Standardisation and Quality Assurance Mechanism and its sub-components have already been released and are further described in the context of this deliverable in the next sections. Furthermore, the overall functionality and performance of this mechanism have been validated and evaluated in project's different pilot scenarios and the input and output parameters, the internal workflows and functionalities have been revised, updated, and extended. As the project progresses further updates will be reported at the next and final version of this deliverable, due in M32.

[1] https://kafka.apache.org/

3 Data Cleaner

The Data Cleaner sub-component is utilized as an integrated sub-component in the overall iHelp Data Ingestion pipeline, and its main objective is to deliver the software implementation that provides the assurance that the provided data coming from several heterogeneous data sources are clean and complete, to the extent possible. This sub-component is designed to minimize and filter the non-important data, thus improving the data quality and importance. To address a portion of these challenges, referring mainly to reducing the complexity and facilitating the analysis of large datasets, data cleaning procedures attempt to improve the data quality and to enhance the analytical outcomes, since wrong data can drive an organization to wrong decisions, and poor conclusions.

To this end, this sub-component seeks to assure the incoming data's accuracy, integrity, and quality. The Data Cleaner sub-component is utilized for every new incoming dataset, as well as new data samples in the case of prospective data, in the platform since it seeks to detect and correct (or remove) inaccurate or corrupted data from the datasets. The input to this sub-component is provided by the shared message bus which is utilized in the scopes of the iHelp project (i.e., Kafka). The topic from which data in the format of messages are consumed is set dynamically, as a parameter whenever the sub-component is called. The input message is batches of data samples that are consumed by the appropriate Kafka topic, along with a set of cleaning rules identified by the stakeholder or the data provider. The latter allows the sub-component to provide all the necessary cleaning actions and to produce consistent and cleaned data and datasets. Finally, the Data Cleaner acts also as a producer providing cleaned data to another Kafka topic, in order cleaned data to be passed to the rest sub-components of the data ingestion pipeline.

Specifically, the Data Cleaner workflow comprises of three discrete, but integrated steps, each one of them being provided as an individual sub-function. These sub-functions are being depicted in Figure 4 and are analyzed in the following sub-sections.

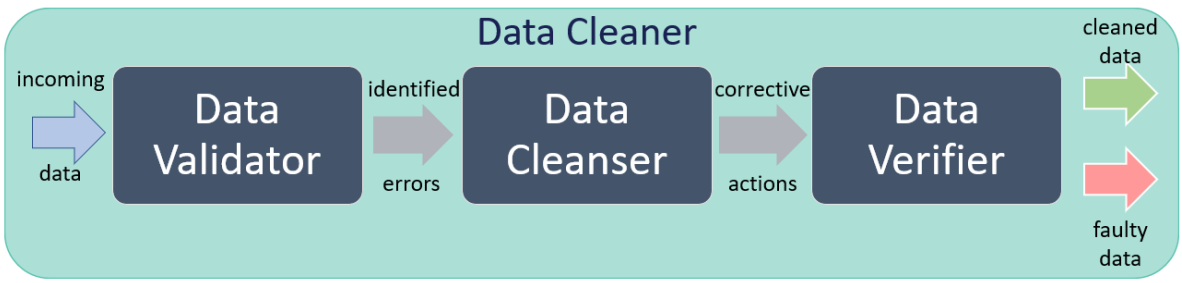

Figure 4: Data Cleaner internal workflow.

The architecture and design of the Data Cleaner sub-component seeks to address the volatility of the incoming data information towards the aim of providing data accuracy, consistency, and completeness to the iHelp platform. Thus, the Data Cleaner sub-component implements all the processes that identify inaccurate or corrupted datasets that may contain incorrect, incomplete, or irrelevant data elements and consequently replace, modify, or delete these data elements safeguarding the reliability and appropriateness of the incoming data information. The software prototypes of the Data Cleaner sub-component are driven by this specification. Moreover, in order to facilitate the overall cleaning functions and procedures and to collect and identify specific pilot needs concerning the data schemas, data

constraints and cleaning actions of the different pilots' datasets, a document has been circulated to each pilot. This document, which was introduced during one of the iHelp consortium meetings, is a live document, and is being introduced in the Annex A – Cleaning Action and Constraints where a corresponding example template for cleaning actions and constraints concerning Pilot#1 of the iHelp project in being presented.

To this context, the Data Cleaner component is composed by one main internal function, namely the *DataCleaningService*, which in turn consists of three internal services utilized by each sub-component: the *ValidationService*, the *CleaningService*, and the *VerificationService*. The main service, *DataCleaningService*, handles all the incoming and outgoing traffic of the Data Cleaner sub-component and is the only service exposed to the rest of the platform components. The ***DataCleaningService*** is the main service of the Data Cleaner sub-function, which is in charge of executing the data cleaning workflow of iHelp platform. Since the data cleaning workflow comprises of several sequential steps, each one implemented by one of the internal services of the component, the *DataCleaningService* is responsible for the orchestration of these internal services as well as for monitoring their execution, and thus providing the execution results to the requestor. Contrary to the main service, the three internal services are not exposed to the rest of the platform components, whereas the *DataCleaningService* is interacting with these services internally to realize the overall data cleaning workflow. The main service of the Data Cleaner is introduced below, while the provided functionalities of the services are discussed in the following sub-sections.

initiateCleaning ("datasourceID": string, " datasetID": string, "schema": json, "schemaKey": json, "confParameters"."cleaning": array_of_rules, "values": array_of_values_for_each_record, "batchSize": number_of_batches, "currentBatchStart": start_of_batch, "currentBatchEnd": end_of_batch). This is the main function that initiates the ***DataCleaningService***. It receives as input:

- the **datasourceID** and the **datasetID**, to further identify the source and to forward this information also in the other components of the data ingestion pipeline and especially to the Data Qualifier to perform the final qualification of both the datasource and the dataset itself,
- the **data schema** and the **schemaKey** of the dataset, to recognize and analyze the schema and structure of the data sample, as well as its primary key,
- the cleaning rules to be performed as **confParameters*,***
- the **values** of each record to further orchestrate the rest of the internal services towards the execution of the Data Cleaner workflow,
- the **batchSize,** which indicates the number of data samples to be processed and consumed as message from the respective Kafka topic,
- the **currentBatchStart** and the **currentBatchEnd** parameters that indicate the exact numbering of records that the current batch of data samples starts and ends.

To better describe and analyze the different parameters the below example follows based on the incoming data from the HDM pilot of the iHelp project.

- *"**datasourceID**": "HDM"*, the ID of the data source or the data provider,
- *"**datasetID**": "ConditionOccurrence",* the ID of the processed dataset,
- *"**schema**": {"name": "conditionOccurrence", "namespace": "eu.ihelp.hdm", "type": "record", "fields": [{"name": "person_id", "type": "int"}, {"name": "condition_concept_id", "type": "int"}, {"name": "condition_start_date",*

"type": {"logicalType": "timestamp-millis", "type": "long"}}, {"name": "condition_start_datetime", "type": {"logicalType": "timestamp-millis", "type": "long"}}, {"name": "condition_end_date", "type": {"logicalType": "timestamp-millis", "type": "long"}}, {"name": "condition_end_datetime", "type": {"logicalType": "timestamp-millis", "type": "long"}}, {"name": "condition_type_concept_id", "type": "int"}, {"name": "stop_reason", "type": "string"}, {"name": "provider_id", "type": "int"}, {"name": "visit_occurrence_id", "type": "int"}, {"name": "condition_source_concept_id", "type": "int"}]}, which indicates the exact schema of the data samples as provided by the Data Gateway component of the iHelp project,

- ***"schemaKey"**: {"name": "conditionOccurrencePK", "namespace": "eu.ihelp.hdm", "type": "record", "fields": [{"name": "condition_occurrence_id", "type": "int"}]}* that describes and is provided also by the Data Gateway component,
- ***"values"**: [[1799773, 12367, 4112752, 1542841200000, null, null, null, 32817, null, null, 18772122.0, null], [236218, 12367, 443597, 1309298400000, null, null, null, 32817, null, 6.0, 3568304.0, null]],* the set of values for each record of data,
- *"**confParameters**": {"**cleaning**": [{"visit_occurrence_id": ["int", "not_null", "max_12_digits"]}, {"person_id": ["int", "not_null", "max_12_digits"]}, {"condition_concept_id": ["int", "not_null", "max_12_digits"]}, {"condition_start_datetime": ["date", "not_null"]}, {"condition_start_date": ["simple_date", "not_null"]}]*, where different cleaning rules are introduced as a parameter. For instance, the "visit_occurrence_id" should exist in each record (not_null), should be integer (int) and of maximum 12 digits (max_12_digits). Thus, any record that does not follow this rule is considered as an erroneous record from the Data Validator and the Data Cleanser performs the needed cleaning actions on this.
- *"**batchSize**": 10*, that indicates the total number of records to be processed,
- *"**currentBatchStart**": 11*, the first record of the processed batch,
- *"**currentBatchEnd**": 20,* the last record of the processed batch.

3.1 Data Validator

The Data Validator sub-function performs the data validation functionality with the purpose of identifying errors associated with the conformance to a specific set of constraints and schemas. Therefore, the Data Validation service performs the validation of the incoming information data with the purpose of identifying errors based on conformance to a specific set of constraints. As depicted in Figure 5 this sub-function incorporates two specific steps. At first, it receives as input the data along with the needed data schema (Data Load & Data Schema), in order then to be able to review the conformance of the provided data to their schema and provide the corresponding validation reports and identify possible errors (Review Data, Validation Reports).

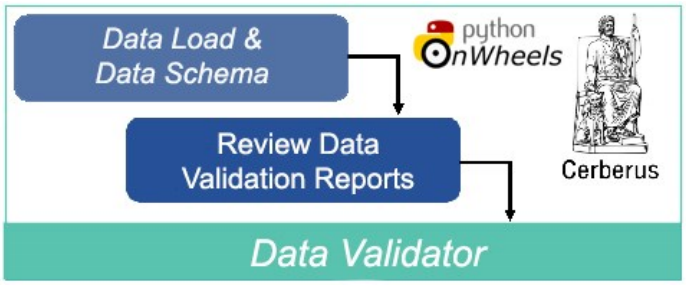

Figure 5: Data Validator Conceptual Diagram.

The ***ValidationService*** is the internal service responsible for the data validation processing of the incoming data. The *ValidationService* performs a series of validation checks in order to evaluate the conformance to a set of constraints currently integrated in the business logic of the service. The current list of validation rules includes the following:

- Conformance to specific data type.
- Conformance to mandatory fields.
- Conformance to specific value length.
- Conformance to specific value representation.
- Conformance to specific value range.
- Identification of duplicate values for the data elements.
- Identification of duplicate data elements.

At this point, it should be noted that the list of the validation rules will be furtherly enriched as the project evolves.

As for its provided functions, the ***ValidationService*** implements the following sub-functions:

- validateData(dataset: File, dataschema: Pandas_schema): List: This is the main function that initiates the ValidationService. It receives as an input the provided dataset (in csv format), as well as its corresponding data schema, being responsible for identifying and returning the list of errors based on the evaluation of the set of the constraints that have been set. In order to properly work, validateData exploits the following sub-functions:
  - checkString(i: String): Boolean: This is an internal function validating the conformance to the specific data type.
  - checkDecimal(d: Decimal): Boolean: This is an internal function validating the conformance to the specific data type.
  - checkInt(i: Integer): Boolean: This is an internal function validating the conformance to the specific data type.
  - checkMaxDigits(i: Numerical): Boolean: This is an internal function validating the conformance to the specific value length.
  - checkRange(i: Numerical, j: Numerical): Boolean: This is an internal function validating the conformance to the specific value range.

3.2 Data Cleanser

The Data Cleanser sub-function seeks to correct or remove all the data elements for which validation errors were raised, considering missing, irregular, unnecessary, and inconsistent data. This sub-function entails

the main sub mechanism of the Data Cleaner sub-function and its main goal is to correct or remove all the data elements for which validation errors were raised, considering missing, irregular, unnecessary, and inconsistent data. Thus, the Data Cleanser sub mechanism performs the necessary corrections or removal of errors identified by the *ValidationService*. Under this scope, several steps of the overall cleaning process are implemented, such as Parsing, Collection, Standardizing, Matching and Consolidation that are also depicted in Figure 6. To successfully perform the above-mentioned steps, the Data Cleanser service exploits the several python tools and libraries, such as Pandas[2], Keras[3], NumPy[4], and Scikit-learn[5].

Figure 6: Data Cleanser Conceptual Diagram.

The ***CleaningService*** is the internal service responsible for the cleaning of the incoming data. The *CleaningService* eliminates the list of errors identified by the *ValidationService* by applying all the necessary corrective actions on the data elements marked with errors. Cleaning is performed in an automated way based on a set of actions currently integrated in the business logic of the component. The current list of cleaning actions includes the following:

- Deletion (drop) of the complete record (row).
- Replacement of data element's value with the mean value.
- Replacement of data element's value with the maximum value.
- Replacement of data element's value with the minimum value.
- Replacement of data element's value with the most frequent value.

At this point, it should be noted that the list of the cleaning actions will be further enriched as the project evolves.

As for its provided functions, the ***CleaningService*** implements the following functions:

- cleanData (errors: List, dataset: File): This is the main function that initiates the CleaningService. It receives as an input the provided dataset (in csv format), as well as the identified list of errors produced by the ValidationService, being responsible for constructing and returning the cleaned dataset. In order to properly work, cleanData exploits the following internal functions:
  - dropRow (row_num: Integer): Void: This is an internal function rejecting the complete record (row).
  - replaceWithMean (column: String, row_num: Integer): Void: This is an internal function replacing the data element's value with the mean value.

[2] https://pandas.pydata.org/
[3] https://keras.io/
[4] https://numpy.org/
[5] https://scikit-learn.org/stable/

- replaceWithMax (column: String, row_num: Integer): Void: This is an internal function replacing the data element's value with the maximum value.
- replaceWithMinimum (column: String, row_num: Integer): Void: This is an internal function replacing the data element's value with the minimum value.
- replaceWithMostFrequent (column: String, row_num: Integer): Void: This is an internal function replacing the data element's value with the most frequent value.

3.3 Data Verifier

The Data Verifier sub-function aims to check the data elements of a dataset for accuracy and inconsistencies and to verify the compliance to the identified iHelp's HHR data models, exploiting the libraries of Pandas and Scikit-learn. The main objective of this sub-function is to check the data elements of a dataset for accuracy and inconsistencies after the steps of data validation and cleaning are performed. To this end, it ensures that all the corrective actions performed by the *CleaningService* are executed in compliance with the data models design of the iHelp platform. To this end, this service, conceptually comprising the Accuracy and Consistency step as depicted in Figure 7, seeks to ensure that data are accurately corrected or completed, and that the dataset is eventually error free.

Figure 7: Data Verifier Conceptual Diagram.

The ***VerificationService*** is the internal service responsible for the verification and evaluation of the corrective actions undertaken by the *CleaningService*, aiming to ensure the accuracy and the consistency of the updated incoming data according to the iHelp platform requirements.

Based on the aforementioned, the *VerificationService* checks and confirms that the *CleaningService* has successfully performed all the needed cleaning actions, returning a *null* list since no further corrective actions are needed to take place.

- ***LoggingService***: It is the internal service responsible for keeping the records that contain all the identified errors and the corrective actions undertaken, in order to address these errors during the data cleaning workflow execution by the rest of the internal services of the Data Cleaner component. This information is also communicated to the Data Qualifier to further calculate the level of quality and trust of both the dataset and the data source. For each execution of the data cleaning workflow a unique record is created and stored in the list of the records kept by the *LoggingService*. In the current implementation the list of the records is kept in a csv file in the local file system where the LoggingService is running. In order to achieve that, the LoggingService implements the following main function:
  - createLog(errors: List): Void: This is the main function that initiates the LoggingService. It receives as an input the identified list of errors, being responsible for creating a new record

(i.e. log file) based on the information provided as input. This new record is appended as a new csv file.

3.4 Interface

The Data Cleaner component is integrated asynchronous with the Kafka message bus of the iHelp project. The execution of the Data Cleaner component is initialized by the whole data ingestion process and provides the cleaned and the faulty data as a result of the execution. The Data Cleaner expects the dataset for which the data cleaning workflow is executed, accompanied by its corresponding data schema, the values of each record, as well as the cleaning rules that will be performed. This set of parameters is further analysed and explained in the introduction of Section 3 and documented below:

```
{
  "datasourceID": "data_provider",
  "datasetID": "dataset_name",
  "schema": {
    "name": "dataset_name",
    "namespace": "dataset_ihelp_internal_id",
    "type": "record",
    "fields": [
      {
        "name": "name_of_the_attribute",
        "type": "value_type_of_the_attribute"
      }
      …
    ]
  },
  "schemaKey": {
    "name": "name_of_the_key_attribute",
    "namespace": "dataset_ihelp_internal_id",
    "type": "record",
    "fields": [
      {
        "name": "name_of_the_key_attribute",
        "type": "value_type_of_the_key_attribute"
      }
    ]
  },
  "confParameters": {
    "cleaning": [
      {
        "name_of_the _attribute": ["cleaning_rule1", "cleaning_rule2", …, "cleaning_ruleN"]
      }
      …
    ],
    "harmonizer": {
      "key1": "value1",
      "key2": "value2"
    }
  },
  "values": [
    [
      array_of_values_for_each_record
    ],
    …
  ],
  "batchSize": number_of_batches,
  "currentBatchStart": start_of_batch,
```

```
  "currentBatchEnd": end_of_batch
}
```

The output is a JSON object produced by both the *VerificationService* that contains information about the cleaned and the faulty data. The latter is produced to a Kafka topic from which the Data Qualifier consumes the message to further provide the reliability and qualification measures for both the dataset and the data source.

```
{
  "dataProvider": "data_provider",
  "dataset": "dataset_name",
  "schema": {
    "name": "dataset_name",
    "namespace": "dataset_ihelp_internal_id",
    "type": "record",
    "fields": [
      {
        "name": "name_of_the_attribute",
        "type": "value_type_of_the_attribute"
      }
      …
    ]
  },
  "schemaKey": {
    "name": "name_of_the_key_attribute",
    "namespace": "dataset_ihelp_internal_id",
    "type": "record",
    "fields": [
      {
        "name": "name_of_the_key_attribute",
        "type": "value_type_of_the_key_attribute"
      }
    ]
  },
  "confParameters": {
    "cleaning": [
      {
        "name_of_the _attribute": ["cleaning_rule1", "cleaning_rule2", …, "cleaning_ruleN"]
      }
      …
    ],
    "harmonizer": {
      "key1": "value1",
      "key2": "value2"
    }
  },
  "values": [
    [
      array_of_values_for_each_record
    ],
    …
  ],
  "faultyData": [
      array_of_cleaned_records
  ]
  "batchSize": number_of_batches,
  "currentBatchStart": start_of_batch,
  "currentBatchEnd": end_of_batch
}
```

3.5 First Prototype Overview

The scope of the Data Cleaner sub-component is to undertake all the processes regarding the data validation and data cleaning of all the incoming heterogeneous data, to the extent possible. The Data Cleaner component provides the interface that implements the data cleaning workflow as documented in deliverable D3.7 – "Standardisation and Quality Assurance of Heterogenous Data I" of the project as also in the previous subsection of this deliverable. The Data Cleaner is utilized by the iHelp platform ensuring data accuracy and consistency of the incoming datasets.

The architecture and design of the Data Cleaner sub-component was documented in D3.7 – "Standardisation and Quality Assurance of Heterogenous Data I" with the purpose of addressing the volatility of the incoming data information towards the aim of providing data accuracy, consistency, and completeness to the iHelp platform. Thus, the Data Cleaner component implements all the processes that identify inaccurate or corrupted datasets that may contain inaccurate, incorrect, incomplete, or irrelevant data elements and consequently replace, modify, or delete these data elements safeguarding the reliability and appropriateness of the incoming data information. The software prototype of the Data Cleaner component was driven by this specification.

3.5.1 Baseline Technologies

The Data Cleaner sub-component has started to being developed based on the utilization of Python 3.7 and the Flask[6] python micro web framework. Flask is a powerful framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine that is independent from particular libraries or tools and that supports a large list of extensions for application features. Besides the Flask framework, a list of libraries and tools has been used in the context of the Data Cleaner to support several functionalities of the component and are introduced in the previous sub sections. On top of this, for the data structure handling Pandas[7] library has been selected, while NumPy[8] library is used for all the numerical computations.

3.5.2 Source code

This sub-section offers valuable information with regards to the overall availability and utilization of the code that has been implemented in the context of the Data Cleaner.

3.5.2.1 Availability

The first software prototype of the Data Cleaner is provided in iHelp's GitLab repository and can be found under the URL:

https://gitlab.ihelp-project.eu/ihelp/t34-data-cleaner

3.5.2.2 Installation and Use

The Data Cleaner software prototype is a Python and dockerised project. As a result, in order to be able to run the prototype manually, docker should be properly preinstalled and preconfigured on the system. In order to use the Data Cleaner, it is highly recommended to make use of a docker container. Firstly, the user needs to download the current version of the binaries by executing the following:

[6] https://flask.palletsprojects.com/en/2.0.x/
[7] https://pandas.pydata.org/
[8] https://numpy.org/

```
$ git clone https://gitlab.ihelp-project.eu/ihelp/t34-data-cleaner
```

Then assuming the docker is already installed in the host machine, the user needs to create the corresponding docker image, by executing the following:

```
$ docker build –t gitlab.ihelp-project.eu:5050/ihelp/t34-data-cleaner/cleaner .
```

If the user does not select to clone the whole code of the subcomponent, then he/she can directly pull the corresponding docker image from the project's Gitlab Container Registry. The latter assumes that the user has an account in the project's registry.

```
$ docker pull https://gitlab.ihelp-project.eu:5050/ihelp/t34-data-cleaner/cleaner
```

After building or pulling the image, he/she can check that it is available in the host machine's docker catalogue and finally can run it by executing the following:

```
$ docker run -d gitlab.ihelp-project.eu:5050/ihelp/t34-data-cleaner/cleaner
```

4 Data Qualifier

The goal of the Data Qualifier sub-component is to automatically categorize both known and unknown data sources to specific reliability levels. To this end, the provided microservice seeks to provide a predictive selection mechanism for achieving data source's reliability during runtime providing a trustfulness of the connected data sources.

The Data Qualifier sub-component classifies data sources as reliable or non-reliable both during the primary and secondary data injection. A data source will be classified as reliable when most of the datasets received from this data source have most if the data correct, otherwise it will be considered as non-reliable. This sub-component receives data by subscribing to a Kafka topic, specifically it acquires the cleaned and faulty data produced by the Data Cleaner sub-component. The cleaned data is the dataset with the appropriate changes applied by the *CleaningService* of the Data Cleaner sub-component. The faulty data informs about the values that have been *cleaned* by the *CleaningService*. For instance, if one or more attributes (i.e., column) in a record (row) is cleaned in a data set, the faulty data will contain the array with the original fault values of the row.

The Data Qualifier sub-component is divided into different sub-functions shown in Figure 8. The Dataset Qualifier sub-function processes the cleaned dataset and the faulty data to evaluate the dataset reliability. For that purpose, it calculates the size of the dataset and takes into account the amount of cleaned data in that dataset. The reliability is provided for the whole data set. These values range from 0-1, where 1 is the highest reliability and 0 the lowest. First it calculates the reliability of each column, per column, the reliability is one minus the total number of faulty values divided by the total number of occurrences of the column in the dataset. The Datasource Qualifier sub-function calculates the reliability of a specific device that produces the data. For instance, a smart watch monitors the heartbeats, sleeping time, number of steps, blood pressure among other metrics. If the heartbeat values are considered faulty for a batch of data or period of time, the heartbeat sensor is considered not reliable. Finally, the Data Qualifier sub-component output is published to a Kafka topic. The output is composed by the data source identifier, the reliability of each data set took into account to calculate the data source reliability and data source reliability.

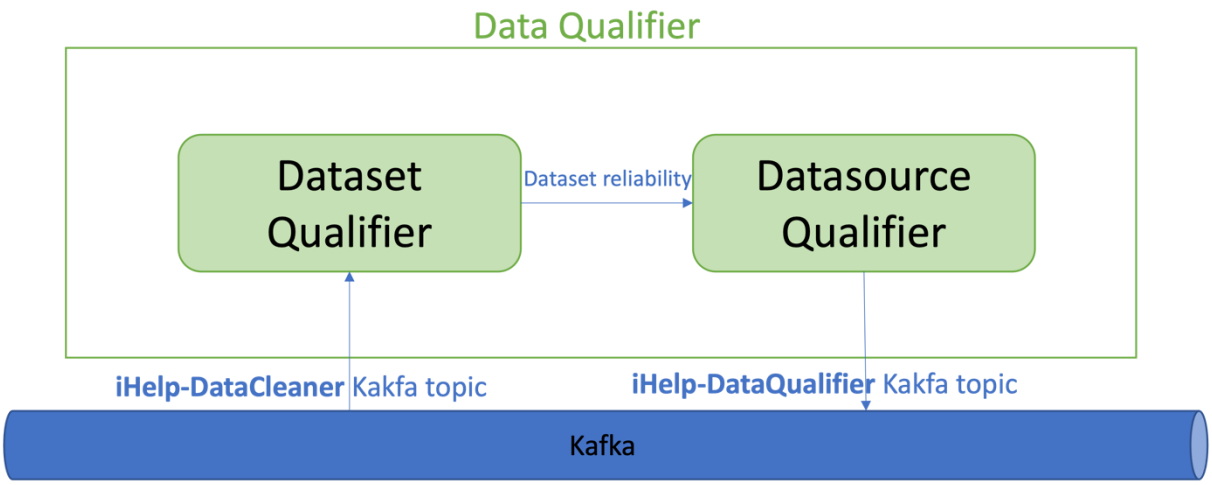

Figure 8: Data Qualifier internal workflow.

4.1 Dataset Qualifier

The Dataset qualifier sub-function receives the data source identifier, the data set identifier, the cleaned dataset and faulty data, among other information produced by the Data Cleaner sub-component and classifies the dataset. This information is obtained from the Kafka topic iHelp-DataCleaner. To do so, the Dataset qualifier calculates the percentage of attributes cleaned per column in relation to the total number of elements for that attribute in the data set. For instance, if 20% of the data in one column was cleaned by the Data Cleaner, the reliability of the column will be 0.8 (1 - 0.2). This is calculated by the Dataset Qualifier for every column. The Dataset reliability is calculated as the average of the reliability of each attribute. This component outputs this information together with the source identifier, data set identifier, i.e.: `HDM, Persons, 0.0909`, where HDM is the datasource identifier, Persons is the dataset identifier and 0.0909 is the dataset reliability.

4.2 Datasource Qualifier

The Datasource Qualifier sub-function receives as input the information from the Dataset Qualifier and periodically calculates the reliability of a data source. The periodicity is a configuration parameter that depends on the application (i.e., how frequent the datasets are generated). The datasets from the same data source may be generated every 5 minutes or every hour or once per day. Based on this frequency the reliability of the data source is calculated for a time period (window) that must also be configured. For instance, the reliability of the data source may be calculated once per day, if the datasets are received every hour. The reliability of a data source can be calculated every hour, if datasets are received every five minutes. This component will output the name of the data source, the reliability of each dataset and the aggregation of the reliabilities of all datasets cleaned during that period (window), i.e.: `HDM, [{Persons, 0.0909,{Persons2, 0.0892},…], 0.09.` Where HDM is the datasource identifier, next it is the array of all datasets reliability processed during that period and 0.09 is the datasource reliability.

4.3 Interface

The Data Qualifier is integrated with the Kafka message bus of the iHelp project. The Data Qualifier is subscribed to the iHelp-DataCleaner Kafka topic from which receives the output generated by the Data Cleaner sub-component and publish the result to the iHelp-DataQualifier Kafka topic. Data arrive to the Data Qualifier sub-component through a JSON object, defined in section 3.4, where the datasourceID field contains the data source identifier, the datasetID contains the dataset identifier, the schema field contains the information related to the non-key columns of the dataset, the schemaKey field contains the information of the columns that takes part in the dataset key, the values field contains the data set with the cleaned values and the faultyData field contains the array of cleaned records. The rest of fields that appear in the DataCleaner JSON object are used by other components of the data ingestion pipeline.

The output is a JSON object produced by the Datasource Qualifier that contains the data source identifier (datasourceID field), the reliability of each dataset considered to calculate the datasource reliability (datasetsQuality field) and the datasource reliability (datasourceQuality field).

```
{
  "dataProvider": "data_provider",
  "datasetQuality": [
    {
```

```
      "dataset": "dataset_name",
      "datasetQuality": dataset_reliability,
    },…
  ],
  "datasourceQuality": datasource_reliability
}
```

4.4 First Prototype Overview

The initial design and specifications of the Data Qualifier was presented in deliverable D3.7 – "Standardisation and Quality Assurance of Heterogenous Data I" of the project. In that deliverable was defined the internal architecture and the first prototype of the Data Qualifier component and the functionality of each sub-function.

The Data Qualifier provides the iHelp project with a system to control the quality of the data being received, allowing it to decide whether the data source that is producing the data is reliable or whether it may be corrupted and sending erroneous data.

4.4.1 Baseline Technologies

The Data Qualifier component is implemented using Kafka Client and Flink[9]. The Data Qualifier consumes data asynchronously from a Kafka topic and directly produces data asynchronously to another Kafka topic. More specifically, this sub-component receives the input message to its system by subscribing to a Kafka topic the JSON object produced by the Data Cleaner sub-component. Finally, the Data Qualifier sub-component output is published to a Kafka topic. The output is a JSON object composed by the reliability of the data source and the reliability of the datasets. Flink is a framework and a distributed processing engine that provides Kakfa connectors to subscribe and to publish to different topics. Flink applications are implemented using a set of stateless and stateful operators that connected among them create a direct acyclic graph (DAG) also called query. A continuous query is deployed and ready to process the JSON objects received from the Kafka topic as soon as they are published by the Data Cleaner component and the result is published using the Kafka connector that allows to publish JSON objects to the Kafka messages bus.

4.4.2 Source code

This sub-section offers valuable information with regards to the overall availability and utilization of the code that has been implemented in the context of the Data Qualifier.

4.4.2.1 Availability

The first data qualifier prototype code is available at the iHelp Gitlab repository:

https://gitlab.ihelp-project.eu/ihelp/t34-data-qualifier

4.4.2.2 Installation and Use

The Data Qualifier is a java-based component that is containerized. Actually, the Data Qualifier is running in the iHelp Kubernetes cluster but can be deployed locally using a docker distribution or within another Kubernetes cluster.

[9] https://flink.apache.org/

Docker containerized deployment

The docker-based distribution allows the user to deploy the data qualifier on their own local machine. To do this, the user needs to have Docker desktop installed to run the data qualifier and a Kafka distribution to consume and publish the JSON objects with the structure indicated in Section 3.4.

The first step is to clone the project from the iHelp git repository:

```
$git clone git@gitlab.ihelp-project.eu:ihelp/t34-data-qualifier.git
```

Next, access the Docker folder inside the project and configure the Data Qualifier with the kafka bootstrapServers endpoints and the period to be considered to classify the data source reliability:

```
$cd t34-data-qualifier/Docker
$vim data-qualifier.properties
```

Once it is configured, save the file and run the following command to create the image:

```
$docker build -t data-qualifier .
```

Finally, run the command to deploy the container and check that the data-qualifier is running accessing the localhost:8081 endpoint, as depicted in Figure 9:

```
$docker run -d -p 0.0.0.0:8081:8081 --name ihelp-data-qualifier ihelp-data-qualifier
```

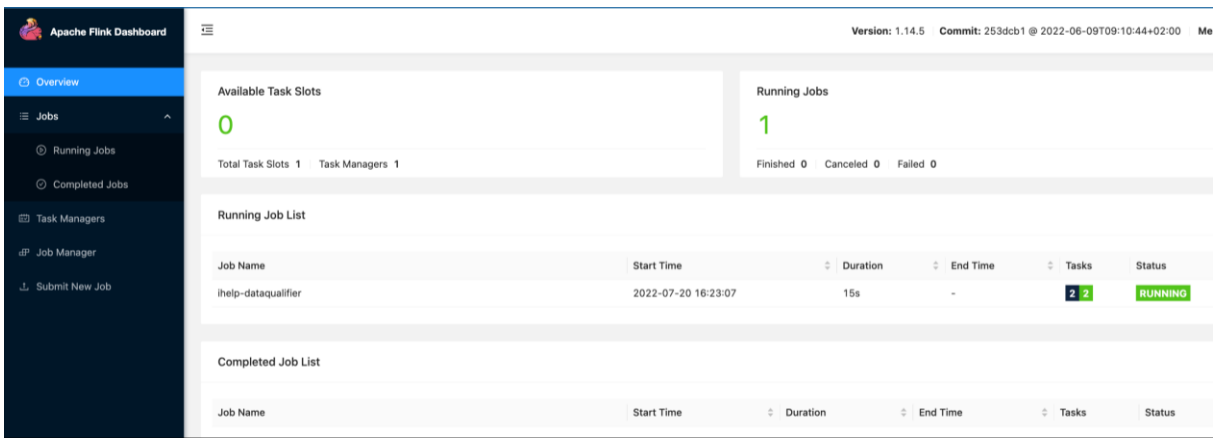

Figure 9: Apache Flink Dashboard.

Kubernetes cluster deployment

To deploy the Data Qualifier within a Kubernetes cluster, the user must clone the repository, configure the Kakfa bootstrapServers endpoints, create the image and push the image into a docker registry. In this examen the iHelp docker registry is being used to push the image:

```
$ git clone git@gitlab.ihelp-project.eu:ihelp/t34-data-qualifier.git
$ cd t34-data-qualifier/Docker
$ vim data-qualifier.properties
$ docker build -t gitlab.ihelp-project.eu:5050/ihelp/t34-data-qualifier/qualifier-
1.0 .
$ docker push gitlab.ihelp-project.eu:5050/ihelp/t34-data-qualifier/qualifier-1.0
```

Once the image is available at the docker registry it is time to deploy it, first the iHelp-DQ-Deployment.yaml has to be configured to set the image endpoint, by default is configured with the iHelp docker registry image endpoint(`gitlab.ihelp-project.eu:5050/ihelp/t34-data-qualifier/qualifier-1.0:latest`). And then, deploy the pod:

```
$ cd ..
$ vim iHelp-DQ-Deployment.yaml
$ kubectl apply -f iHelp-DQ-Deployment.yaml
```

To check that the pod is running execute the following command:

```
$ kubectl get pods -o wide | grep dataqualifier
ihelp-dataqualifier-6c967ff8fd-6fzln    1/1       Running    0                177m
10.42.1.161   k8s-2   <none>           <none>
```

5 Data Harmonizer

The Data Harmonizer sub-component is utilized as an integrated sub-component in the overall iHelp Data Ingestion pipeline, and its main objective is to support and harmonize data coming from heterogeneous sources into a common format. To this end, it seeks to provide annotated health data & harmonize them with the HHR format. The sub-component utilizes its own internal sub-functions in order to correlate data resources to be compliant with the HHR model that are defined in the scope of the Task 3.1 ("Data Modelling and Integrated Health Records").

The Data Harmonizer sub-component is utilized asynchronous and is integrated with the provided message bus mechanism, as it consumes and produces corresponding messages to the provided Kafka queues. The message that is being consumed are cleaned and qualified data, which are further harmonized and transformed. Hence an annotated, transformed and HHR-aligned message is the output of this specific sub-component. To this end, the Data Harmonizer sub-component has been determined to internal integrate closely with the Data Cleaner and Data Qualifier sub-components.

The Data Harmonizer sub-component incorporates the use of two integrated sub-functions, as shown in Figure 10.

Figure 10: Data Harmonizer internal workflow.

This sub-component seeks to support data coming from several sources in order to deal with different formats, thus, to enhance the interoperability of data. In recent years many approaches, standards, ontologies, and vocabularies have been proposed as means of achieving various tasks of interoperability between heterogeneous and independent datasets. More specifically in the healthcare domain, an advanced Semantic Interoperability technique was introduced with emphasis on the utilization of Structural and Ontology Mapping services along with Terminology Linking services in order to transform the clinical information into interoperable and processable data using eHealth standards and terminologies (K., M., M. + 19). The above introduced approach provides the means for common representation of domain specific datasets and the means for achieving interoperability across diverse databases and datasets.

5.1 Semantic & Syntactic Analysis

To exploit what the Data Harmonizer offers, data first needs to be structured and annotated. To this end, in the first subcomponent, the Semantic & Syntactic Analysis, cleaned and qualified data are analyzed, transformed, and annotated with appropriate URI metadata. In next phases and steps, semantic and syntactic URI-annotated data (Unified Resource Identifier) are interlinked through the task of Ontology

Mapping. The main objectives of this first sub-function of the Data Harmonizer are the identification and recognition of entities, which are further used for interconnection and interlinking with the HHR resources and model that have been identified in the context of the T3.1 “Data Modelling and Integrated Health Records”. Moreover, classifying named entities found in data into pre-defined categories, such as persons, places, organizations, dates etc., makes it possible to identify, design and use proper widely used and controlled vocabularies and standards. The overall Data Harmonizer sub-function is further enhanced and completed in the next step by the utilization of Ontology Mapping submechanism, where an Ontology and Structuring Mapping service is utilized in order to interlink not only URI-annotated data with proper ontologies, but also to interlink and correlate datasets among them.

5.2 Ontology & Structure Mapping

The Structural Mapping sub-function takes advantage of well-established ontology alignment approaches to perform the mapping between the schema/model of incoming document with the use-case specific target schema/model in the iHelp platform.

Ontology alignment approaches can be utilized for finding structural mappings between two different data models. A number of tools and frameworks have been developed for aligning Ontologies, which vary in the degree of user intervention required to produce accurate mappings. In typical Ontology alignment approaches, data models or Ontologies are usually converted to a graph representation before being matched. Such graphs can be represented in the Resource Description Framework (RDF) line of languages by triples of the form <*subject, predicate, object*>. In this context, aligning ontologies is sometimes referred to as "ontology matching".

Successful annotation, transformation and mapping of data and corresponding ontologies in terms of semantic and syntactic interoperability of data is one of the key elements of the Data Harmonizer sub-function. To this end, one of the main objectives of the Ontology Mapping subcomponent is to save correlated, annotated and interoperable data in JSON-LD format and as linked ontologies. Hence, it is feasible to store semantic facts and the support of the corresponding data schema models. Moreover, this subcomponent seeks to map concepts, classes, and semantics defined in different ontologies and datasets and to achieve transformation compatibility through extracted metadata. In addition, a data modelling subtask by standard metadata schemas is defined in order to specify the metadata elements that should accompany a dataset within a domain. To this end, semantic models for physical entities (e.g. specific grading features of Pancreatic Cancer) and measures (e.g. specific grading features of Pancreatic Cancer) will be identified. These models are based on a set of transversal and domain-specific ontologies and could provide a foundation for high-level interoperability and rich semantic annotations across the healthcare ecosystem. As shown in Figure 11, there are several levels of structuring before reaching proper ontologies. At the beginning, the annotation and creation of metadata representations through the utilization of JSON-LD technology is a key point. Afterwards, vocabularies and taxonomies expressed by RDFs are created and in the final step they are correlated and interlinked into ontologies with high semantic expressivity through the utilization of OWL technology.

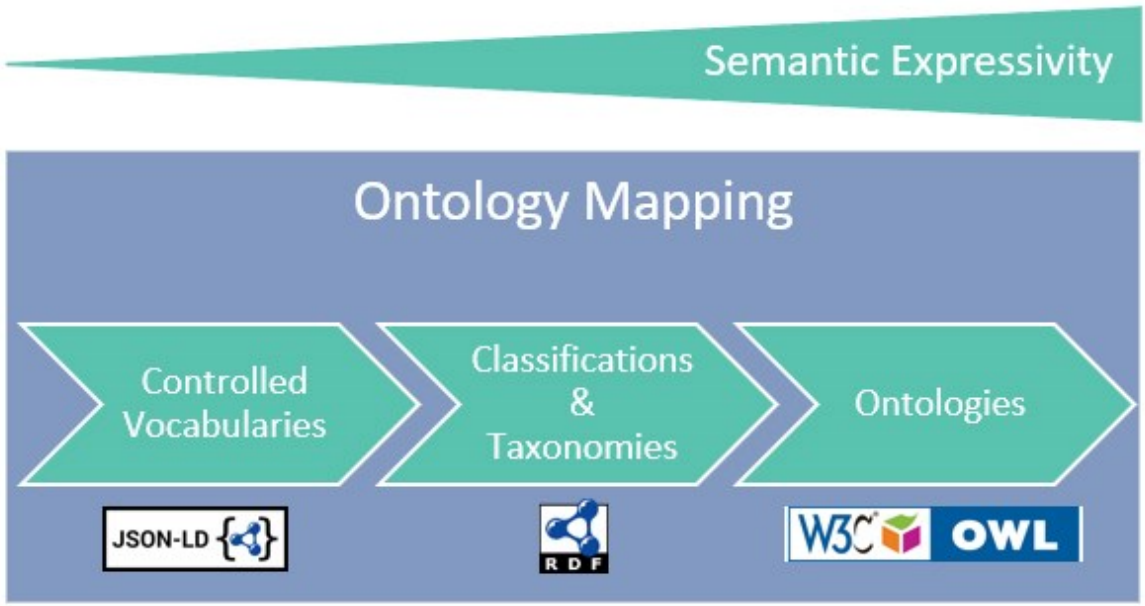

Figure 11: Ontology Mapping Steps.

On top of this, ontologies are central to the Data Harmonizer as they allow applications to agree on the terms that they use when communicating and they enable the correlation of divergent data and datasets from various sources. To this end, the utilization of ontologies under the scope of Data Harmonizer facilitates communication by providing precise notions that can be used to compose messages (queries, statements) about the healthcare domain. In stakeholders and user level, the ontology helps to understand messages by providing the correct interpretation context. Thus, ontologies, if shared among stakeholders, may improve system interoperability in the healthcare ecosystem. The overall approach that is followed brings together techniques in modeling, computation linguistics, and information retrieval in order to provide a semi-automatic mapping method and a prototype mapping system that support the process of Ontology Mapping for the purpose of improving and enhancing interoperability and usage of data during the whole data lifecycle.

The novelty of the proposed Ontology Mapping sub-function is not solely the use of formal application ontologies as an initial mechanism to achieve meaningful interoperability, but moreover the utilization of divergent ontologies to support the formal application ontologies mapping process, integrated into an architectural framework.

5.3 Ontology-based Domain Terminology Mapping

One of the main functionalities of the Ontology Mapping submechanism is the Ontology-based Domain Terminology Mapping. This sub-function provides a set of intelligent services to manage terminology resources and make the data semantically interoperable. In addition, it provides a set of operations on widely used and known medical terminologies used for the coding of medical knowledge, which further enhance the information structures that are provided as outputs from the Data Harmonizer sub-component. To this end, a set of functionalities has been integrated and implemented in the context of the design and initial implementation of this sub-function. More specifically, this sub-function utilizes two different sources of terminologies. In one hand, it utilizes the domain-specific Ontology introduced and provided in the context of T3.1 (“Data Modelling and Integrated Health Records”), while in the other hand

it also utilizes the Unified Medical Language System (UMLS)[10] that integrates and distributes key medical terminologies and coding standards to further facilitates the creation of interoperable solutions (Bod, 04). The internal workflow of the proposed sub-function is depicted in Figure 12.

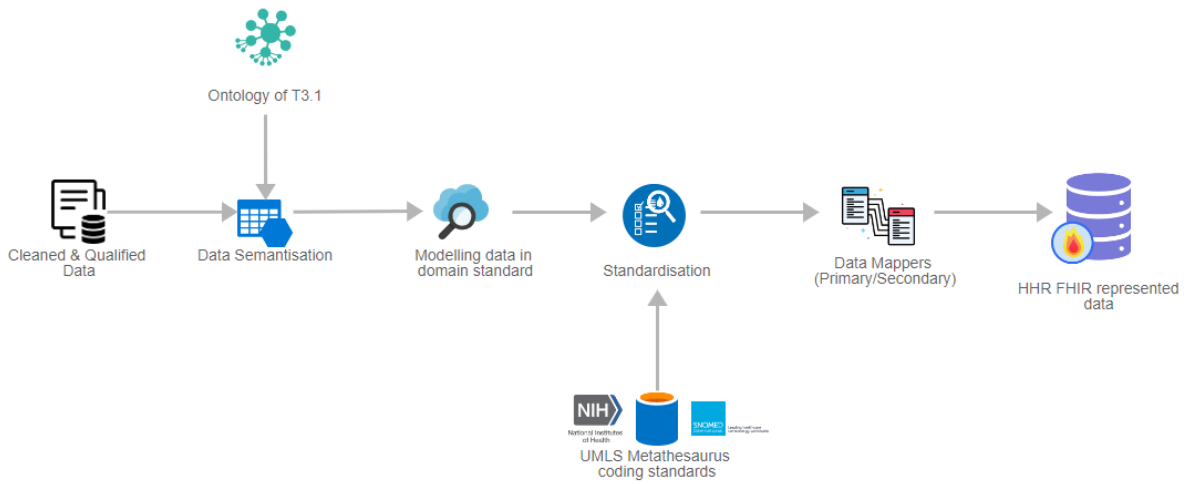

Figure 12: Ontology-based Domain Terminology Mapping internal workflow.

The design and implementation of this sub-function offers the availability to load different terminologies both from widely used standards and vocabularies, as also from the domain-specific ontology that is being created in the context of the iHelp project. In addition, it provides the flexibility to the whole project to utilize new releases of terminologies and to provide mappings or translations between different terminologies and standards. The latter is addressed through the extensible searching and querying functionality for specific elements of the different examined terminologies and standards, such as ICD-10 and SNOMED CT International. This way, it is possible to provide a series of functionalities (semantics) about these information elements found within more complex structures. The overall workflow of the introduced sub-function can be encapsulated through the below steps:

- Cleaned and qualified data are semantically analysed and mapped to concepts and instances of the ontology provided through T3.1 (“Data Modelling and Integrated Health Records”).
- Data are standardized into project’s common data model and domain standard.
- The Unified Medical Language System (UMLS) and the different terminologies, coding standards and vocabularies that are offered through this system are utilized to further transform the medical terms between terminologies in a controlled and supervised manner.
- Finally, standardised and harmonized data are fed into the Primary and Secondary Data Mappers to be transformed into the HHR FHIR compliant format. Both the Primary and Secondary Data Mappers are further described in Section 6.

5.4 Interface

The Data Harmonizer sub-component is utilized in an asynchronous way and integrated with the Kafka message bus of the iHelp project. Hence, no endpoint or API call is available in order to trigger the execution

[10] https://www.nlm.nih.gov/research/umls/index.html

of this specific functionality. The Data Harmonizer is subscribed to the iHelp-DataQualifier Kafka topic form which receives the output generated by the Data Qualifier sub-component and produces the result to the iHelp-DataHarmonizer Kafka topic. Cleaned and qualified Data arrive to the Data Harmonizer sub-component as an annotated JSON object, defined both in sections 3.4 and 264.3. Once, the internal functionalities of the Data Harmonizer are finished, then the standardized data are communicated either to the Primary or to the Secondary Data Mapper, depending on the nature of the data (i.e., primary or secondary). The response of the mappers is the final harmonized and standardized into HHR format data that then are populated to the iHelp-DataHarmonizer Kafka topic to finally stored into the iHelp's Big Data Platform. Thus, the final output is a JSON object produced by the Data Harmonizer that contains the data transformed into the HHR format and based on the different attributes and coding standards into which they have been standardized and mapped. It should be noted that the main concepts are mapped using a unique ID of the concept they represent as also stated in next sections and in the description of the internal functionalities of the iHelp's Mappers. For instance, the below code snippet depicts the final output of the transformed to HHR format data.

```
{
  "dataProvider": "data_provider",
  "dataset": "dataset_name",
  "schema": {
    "name": "dataset_name",
    "namespace": "dataset_ihelp_internal_id",
    "type": "record",
    "fields": [
      {
        "name": "name_of_the_attribute",
        "type": "value_type_of_the_attribute"
      }
      …
    ]
  },
  "schemaKey": {
    "name": "name_of_the_key_attribute",
    "namespace": "dataset_ihelp_internal_id",
    "type": "record",
    "fields": [
      {
        "name": "HHR_name_of_the_key_attribute",
        "type": "HHR_value_type_of_the_key_attribute",
        {
                "id": "HHR_value_id"
        }
      }
    ]
  },
  "confParameters": {
    "cleaning": [
      {
        "name_of_the _attribute": ["cleaning_rule1", "cleaning_rule2", …, "cleaning_ruleN"]
      }
      …
    ],
    "harmonizer": {
      "key1": "value1",
      "key2": "value2"
    }
  },
```

```
  "values": [
    [
      HHR_array_of_values_for_each_record
    ],
    …
  ],
  "batchSize": number_of_batches,
  "currentBatchStart": start_of_batch,
  "currentBatchEnd": end_of_batch
}
```

5.5 First Prototype Overview

The scope of the Data Harmonizer sub-component is to undertake all the processes regarding the final transformation and harmonization of data into common health information structures, and code systems or medical terminologies that enhance the interoperability of the healthcare-related data. The Data Harmonizer sub-component provides the interface that implements the data harmonization workflow as documented in deliverable D3.7 – "Standardisation and Quality Assurance of Heterogenous Data I" of the project as also in the previous subsection of this deliverable. The Data Harmonizer is utilized by the iHelp platform ensuring data standardization and homogeneity of the incoming datasets.

The architecture and design of the Data Harmonizer sub-component was documented in D3.7 – "Standardisation and Quality Assurance of Heterogenous Data I" with the purpose of providing technologies for harmonizing and transforming the collected health data into HL7 FHIR format, through finding common links or similarities between primary and secondary data types and available HL7 FHIR resources. Thus, the Data Harmonizer sub-component implements all the processes that utilize widely used and known coding standards and terminologies, coupled with domain-specific ontologies. The first software prototype of the Data Harmonizer sub-component was driven by these specifications and further facilitates the aggregation of the distributed heterogeneous data coming from multiple health related datasets and provides data mapped into globally recognized standards and a common format.

5.5.1 Baseline Technologies

The overall code has started to be implemented based on Python3.7 programming language. Moreover, SPARQL, a widely used RDF query language, is being utilized in order to identify and interlink standards and resources that have been identified in the context of D3.1 – "Data Modelling and Integrated Health Records: Design and open specification I" with the provided incoming data. The latter is being used to perform queries on Knowledge bases to identify and interlink appropriate entities based on the ones that have already been recognized from the raw data. In addition, the Unified Medical Language System (UMLS) is utilized, as also different python packages and libraries for the utilization of ontologies and the terminology mapping services.

5.5.2 Source code

This sub-section offers valuable information with regards to the overall availability and utilization of the code that has been implemented in the context of the Data Harmonizer.

5.5.2.1 Availability

The first software prototype of the Data Harmonizer is provided in iHelp's GitLab repository and can be found under the URL:

https://gitlab.ihelp-project.eu/ihelp/t34-data-harmonizer

5.5.2.2 Installation and Use

The Data Harmonizer software prototype is a Python and dockerised project. As a result, in order to be able to run the prototype manually docker should be properly preinstalled and preconfigured on the system. In order to use the Data Harmonizer, it is highly recommended to make use of a docker container. Firstly, the user needs to download the current version of the binaries by executing the following:

```
$ git clone https://gitlab.ihelp-project.eu/ihelp/t34-data-harmonizer
```

Then assuming the docker is already installed in the host machine, he or she needs to create the corresponding docker image, by executing the following:

```
$ docker build –t gitlab.ihelp-project.eu:5050/ihelp/t34-data-
harmonizer/harmonizer .
```

If the user does not select to clone the whole code of the subcomponent, then the user can directly pull the corresponding docker image from the project's Gitlab Container Registry. The latter assumes that the user has an account in the project's registry.

```
$ docker pull https://gitlab.ihelp-project.eu:5050/ihelp/t34-data-
harmonizer/harmonizer
```

After building or pulling the image, he/she can check that it is available in the host machine's docker catalogue and finally can run it by executing the following:

```
$ docker run -d gitlab.ihelp-project.eu:5050/ihelp/t34-data-harmonizer/harmonizer
```

6 Data Mappers

The overall functionality of the Data Harmonizer is complemented by the internal utilization and integration with the iHelp Data Mappers sub-mechanisms. These sub- mechanisms perform the mapping procedures between the data elements and the HL7 FHIR resources to finally provide data into HHR FHIR compliant format. The Data mappers are internal integrated with the Data Harmonizer sub-component as depicted in Figure **12**. More specifically, the Primary Mapper sub-component will enable the mapping of primary data, that is the clinical data from the electronic health records, into the common data format that will be used in the iHelp platform (HHR). While the Secondary Mapper sub-component will enable the mapping of secondary data (e.g., from mobile, wearable, and social-media platforms) into the data format (schema, model) used in the iHelp platform (HHR).

6.1 Primary Data Mapper

The Primary Data Mapper sub-component seeks to enable the mapping of primary data, that is the clinical data from the Electronic Health Records (EHRs), into the common data format that are used in the iHelp platform (HHRs). To this end, this specific component provides the necessary transformation functions that are required to map the primary data to the HHRs stored in the iHelp platform. Initial information about the HHRs and the mapping process can be found at Sections 4 and 5 of D3.1 – "Data Modelling and Integrated Health Records Design and open specification I" (K., D., P. + 21).

The Primary Data Mapper receives cleaned, qualified, and harmonized data as part of the overall ingestion pipeline. Sequentially, the harmonized health records that are being received from the Data Harmonizer are mapped to the HHR ontology-based records, through semantic matching and harmonization techniques. Finally, after the necessary mapping/transformation process, the Primary Data Mapper produces and sends the new HHR aligned record, through a Kafka topic to the HHR Importer component.

6.1.1 First Prototype Overview

The scope of the Primary Data Mapper sub-component is to syntactically transform the data coming from a data model to the internal HHR data model defined in D3.1 – "Data Modelling and Integrated Health Records Design and open specification I" (K., D., P., + 21). The first prototype Is based on a specific pilot, HDM (Hospital de Dénia-MarinaSalud). That organization internally uses the standard Ohdsi OMOP data model[11], therefore the module developed in T3.4 Standardisation and Quality Assurance of Heterogeneous Data, converted between two standards, OMOP toward HHR (based on HL7/FHIR)[12].

The Primary Data Mapper sub-component provides a REST API interface that implements that conversion and is publicly available at GitHub[13]. The mapper, in this first version, allows to convert a single OMOP resource or list of them, simply invoking the related endpoint.

The current implementation also allows two different methods for ensuring the translation. Specifically, in OMOP, the main concepts are mapped using a unique ID of the concept they represent. The Ohdsi initiative, mapped many standard dictionaries in a common database, available for consultation at the Athena[14]

[11] https://www.ohdsi.org/data-standardization/the-common-data-model/
[12] https://hl7.org/FHIR/
[13] https://github.com/ihelp/omop2fhir.json
[14] https://athena.ohdsi.org/search-terms/start

website, and every concept present in the OMOP data model should refer those unique IDs. In the current implementation of the iHelp Primary Data Mapper for HDM, some of those IDs were included in the mapper, as a facility, until the whole dataset will be ready (thanks to WP3 and WP6 joint work) and until the final version of the Ontology-based Domain Terminology Mapping that will provide the right concept using the FHIR standard. The dictionaries included in this release are: OMOP, SNOMED CT[15], LOINC[16], and RxNorm[17].

The other method allowed by the module expects the filling of the necessary attributes for generating the whole code, following the FHIR standard, and not only the OMOP concept ID as above.

A simple example is the following: the first option uses the ID of the measurementConcept as the one mapped by OHDSI and refers to the SNOMED CT ontology.

```
{…
"measurementConcept": { "id": 4042059 },
…}
```

In this case, the supported dictionary has to be included in the sub-component container.

The second option allows to invoke the service by passing the details like the following:

```
{ …
"measurementConcept": { "conceptName”: “Serum HDL cholesterol level”, "vocabularyId": "SNOMED",
"conceptCode": "166832000"},
…}
```

In this case, there is no need to include any dictionary, because the mapping will be only syntactically and all the details will come from the Ontology-based Domain Terminology Mapping sub-function developed in iHelp.

Both the invocations will generate the FHIR attribute, and conform to the HHR, like the following:

```
{ …
"code": {
"coding": [
{ "system": "http://snomed.info/sct", "code": "166832000", "display": "Serum HDL cholesterol measurement"
}]
},
…}
```

6.1.1.1 Source code

This sub-section offers valuable information with regards to the overall availability and utilization of the code that has been implemented in the context of the Primary Data Mapper.

6.1.1.1.1 Availability

The first software prototype of iHelp Primary Data Mapper for HDM is provided in iHelp's GitLab repository and can be found under the URL:

https://gitlab.ihelp-project.eu/ihelp/t34-data-harmonizer/

[15] https://www.snomed.org/
[16] https://loinc.org/
[17] https://www.nlm.nih.gov/research/umls/rxnorm/index.html

6.1.1.1.2 Installation and Use

The Primary Data Mapper for HDM software prototype is a SpringBoot Java dockerised project and a prefilled postgress database, including the dictionaries listed above.

As a result, in order to be able to run the prototype manually docker should be properly preinstalled and preconfigured on the system. In order to use the Primary Data Mapper, it is highly recommended to make use of a docker container.

The user can directly pull the corresponding docker image from the project's Gitlab Container Registry. The latter assumes that the user has an account in the project's registry.

```
$ docker pull https://gitlab.ihelp-project.eu:5050/ihelp/t34-data-harmonizer/omop2fhir-ihelp
$ docker pull https://gitlab.ihelp-project.eu:5050/ihelp/t34-data-harmonizer/postgres-ihelp
```

After building or pulling the image, he/she can check that it is available in the host machine's docker catalogue.

To compose the container the docker-compose file should have the following:

```
version: '3.7'
services:
  converter:
    container_name: omop2fhir-ihelp
    image: ${DOCKER_IMAGE_REPO}omop2fhir-ihelp # pushing to ihelp docker registry
    build: ./
    ports:
      - "8080:8080"
    depends_on:
      - dbpostgresql
  dbpostgresql:
    container_name: dbpostgresql-ihelp
    image: ${DOCKER_IMAGE_REPO}postgres-ihelp
    restart: always
    logging:
          options:
            max-size: 10m
            max-file: "3"
    ports:
      - "5432:5432"
    environment:
      - POSTGRES_PASSWORD=omop
      - POSTGRES_USER=omop
      - POSTGRES_DB=omop_db
```

6.2 Secondary Data Mapper

The Secondary Mapper sub-component will enable the mapping of secondary data (e.g. from mobile, wearable and social-media platforms) into the standard data format (HHR schema, model) used in the iHelp platform. The component will provide the necessary transformation functions that are required to map the secondary data from heterogeneous sources to the holistic health records stored in the iHelp platform. The Secondary Mapper will be an integral part of the iHelp platform as it supports the enrichment of typical health records with the secondary (e.g., lifestyle, social etc.) data of the individuals.

Based on the adoption of microservices architecture model in the iHelp project, the Secondary Data Mapper component will be exposed as a microservice in the iHelp platform. As shown in Figure 13, it will offer dedicated converter/mapper functions that will handle data from different secondary data devices or interfaces. Since, during the project, iHelp platform will support secondary data ingestion from a specific number or type of devices (e.g., the mobile application or Fitbit activity tracker), thus the mapper functions can be designed and configured to deal with the data (interfaces, models, formats, syntax) associated with those devices. However, the secondary data mapper services will be extensible in nature, thus allowing the development and integration of mapping functions that deal with other types of devices or data models.

A controller mechanism with the secondary data mapper will be responsible for assigned the incoming data to the relevant mapper functions. The controller will work on the basis of interpreting meta data associated with the incoming data (batch or stream) and forwarding the data to the relevant mapper function.

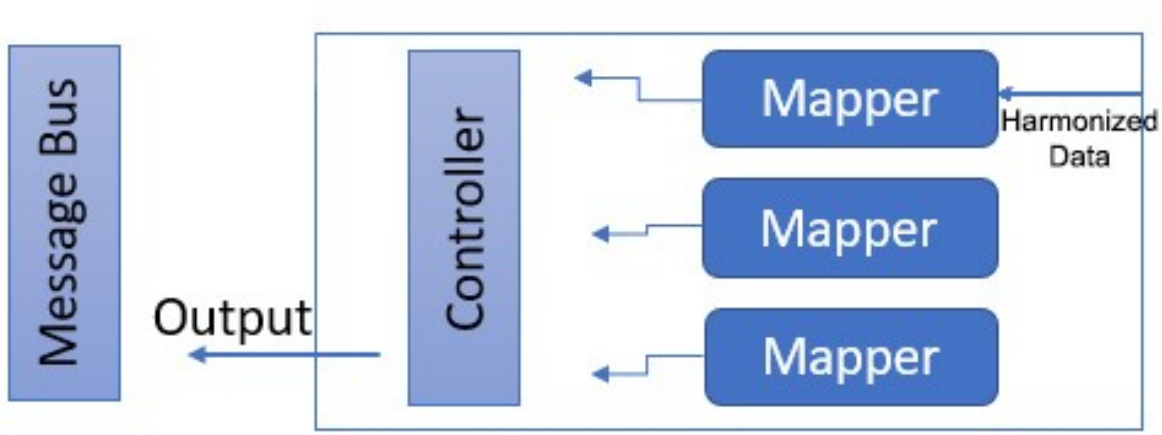

Figure 13: High-level architecture of Secondary Data Mapper.

The Secondary Data Mapper microservice will be utilized either asynchronous or synchronous depending on the deployment status; and it will be integrated with the iHelp message bus mechanism. The interaction with the message bus will allow the microservice to produce secondary data derived from different mobile and wearable devices in the standardised HHR format. To this context, this microservice seeks to apply the relevant mapping function to convert the harmonized data, which are derived from the Data Harmonizer and are the input to its system, according to the standard HHR format that will be defined under the scope of T3.1 (“Data Modelling and Integrated Health Records”) and expose the converted data or the outcome of the mapping function to the provided message bus (Kafka) topic. To this end, the secondary data mapper microservice will serve as the final stage of the data cleaning and standardisation operation, before the data is passed on to the iHelp’s Big Data Platform through the HHR Importer mechanism, which will be provided by T4.4 (“Big Data Platform and Knowledge Management System”). As the recruitment process

from pilots' side has just started and the collection of patients' secondary data will follow the next months, thus the overall implementation of this mapper will be described and analyzed in the next and final version of this series of deliverables, i.e., D3.9 – "Standardisation and Quality Assurance of Heterogenous III" due on M32.

7 Conclusion

This document reports the work that has been currently done in the scope of T3.4 – "Standardisation and Quality Assurance of Heterogeneous Data", whose main objective it is to provide the main data processing components of the project. To this end, this deliverable describes the first software prototype of the Standardisation and Quality Assurance Mechanism. The provision of the early prototype facilitates the evaluation and feedback provision as the project progresses. Five different subcomponents have been explained including the interfaces, the baseline technologies, where they are available and how to install and use them. The prototypes are built upon open-source technologies that have been developed by solid communities.

The specifications that have been introduced the context of D3.7 – "Standardisation and Quality Assurance of Heterogenous Data I" have been utilized and revised for the realization and implementation of this holistic mechanism, encompassing its main functionalities regarding the assurance of the incoming data's accuracy, integrity, and quality, the interoperability of data, the automated data transformation to the HHR model, and their aggregation into unique turn-key offerings. On top of this, this report describes the baseline technologies adopted and the sub-mechanisms developed to ensure effective contributions towards standardisation and quality assurance of healthcare data. The deliverable highlights the functionalities used for various purposes (e.g., for data management, data cleaning, data transformation, data mapping etc) and the approaches/techniques implemented to make sure that the data remains within the quality constraints while being used by different stakeholders and applications in the project.

The next release will include new functionalities that will be explained in the forthcoming deliverable. Moreover, part of the introduced mechanism will also be open-source and offered to the healthcare, research, and business communities to take advantage of the work done in the context of the project and to further enhance the overall exploitation and impact of the iHelp platform. Moreover, as already stated, this deliverable includes the first prototypes of the incorporated sub-components, as also an initial and brief outline concerning the Secondary Data Mapper sub-component. The latter will start being developed and evaluated after the final modelling of the common HHR model and format of the secondary data that will be introduced in the context of task T3.1 ("Data Modelling and Integrated Health Records"). In next deliverable, more details about the implementation and architecture of this sub-component will be outlined.

To conclude, the current document is delivered in M20 (August 2022) and is the second version of a series of documents and reports that are planned to be released under the scopes of task T3.4 ("Standardisation and Quality Assurance of Heterogeneous Data") and throughout the project's lifetime. On M32 (August 2023), a third version is planned to be delivered on M32 (August 2023) to cover remaining aspects and to correct potential erroneous decisions or unnecessary implementation that might have been identified earlier, so that it can drive the final definition and implementation of the Standardisation and Quality Assurance Mechanism.

Bibliography

O. Bodenreider, “The Unified Medical Language System (UMLS): integrating biomedical terminology”, *Nucleic Acids Research,* D267-270, 2004.

Y.E. Bulut, “AI for data science: artificial intelligence frameworks and functionality for deep learning, optimization, and beyond”, *Technics Publications*, 2018.

V. Chavan, and R.N. Phursule, “Survey paper on big data”, *Int. J. Comput. Sci. Inf. Tech-nol*, vol. 5, no. 6, pp. 7932-7939, 2014.

P. P. Jayaraman, A. R. M. Forkan, A. Morshed, P. D. Haghighi, and Y. B. Kang, “Healthcare 4.0: A review of frontiers in digital health”, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 2, p. e1350, 2020.

M. Kalogerini, A. Dalianis, C. Pandolfo, F. Melillo, and G. Manias, “D3.1 - Data Modelling and Integrated Health Records: Design and open specification I”, iHelp, 2021.

A. Kiourtis, A. Mavrogiorgou, A. Menychtas, I. Maglogiannis, and D. Kyriazis, “Structurally mapping healthcare data to HL7 FHIR through ontology alignment”, *Journal of medical systems*, vol. 43, no. 3, pp. 62, 2019.

G. Manias et al., “D3.7 - Standardisation and Quality Assurance of Heterogenous Data I”, iHelp, 2021.

M. Mosley, M. H. Brackett, S. Earley, and D. Henderson, “DAMA guide to the data management body of knowledge”, *Technics Publications*, 2010.

K. Muir et al, “D6.1 - Coordination of pilot scenarios for personalised healthcare - early risk identification, prevention and intervention measures I”, iHelp, 2022.

S. C. Pandey, “Data mining techniques for medical data: a review”, *In 2016 International Conference on Signal Processing, Communication, Power and Embedded System*, pp. 972-982, 2016.

List of Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
CA	Consortium Agreement
CSV	Comma Separated Values
D	Deliverable
DoA	Description of Action
EHRs	Electronic Health Records
EU	European Union
HHRs	Holistic Health Records
JSON-LD	JavaScript Object Notation – Linked Data
M	Month
NER	Named-Entity Recognition
NLP	Natural Language Processing
OWL	Web Ontology Language
R&D	Research and Development
RDF	Resource Description Framework
REST	Representational State Transfer
T	Task
URI	Uniform Resource Identifier

Annex A – Cleaning Action and Constraints

Pilot #1 - UNIMAN

The pilot focuses on Genomics and Epigenomics Markers for Early Risk Assessment of Pancreatic Cancer.

Sample Datasets

Risk_factors_1 Sample Dataset

	A	B	C	D	E	F	G	H	I	J	K
1	ID	Gender	Age group	Current weight	Height (m)	BMI	FH	Smoke	Diabetes	Chronic pancreatitis	FruitVeg
2	10001	0	3	80	1,8	24,7	0	1	0	0	1
3	10002	0	3	90	1,75	29,4	1	0	0	1	1
4	10003	0	2	50	1,6	19,5	1	1	1	1	1
5	10004	1	2	65	1,72	22,0	1	1	1	1	0
6	10005	0	2	55	1,63	20,7	1	1	1	0	0
7	10006	0	3	49	1,55	20,4	1	2	1	0	0
8	10007	1	3	45	1,53	19,2	1	1	1	0	0
9	10008	0	3	77	1,72	26,0	0	0	0	0	1
10	10009	0	4	88	1,81	26,9	1	1	0	1	1
11	10010	1	1	71	1,60	27,7	1	2	1	1	1
12	10011	1	2	89	1,90	24,7	0	0	1	1	0
13	10012	0	4	105	1,79	32,8	1	0	0	1	1

Figure 14: Snapshot of UNIMAN pilot Risk_factors_1 sample dataset.

Dictionary

Variable	Answer	Code
Gender	Female	1
	Male	2
Age group	40-50	1
	51-60	2
	61-70	3
	>=71	4
FH	Yes-Family history of pancreatic cancer (first degree relative)	0
	No-family history of pancreatic cancer in 1st degress relative	1
Smoke	Non-smoker	0
	Smoke >=25 cigarettes/day	1
	Smoke 15-24 cigarettes/day	2
Diabetes	No diabetes type II	0
	Yes-Diabetes type II	1
chronic pancreatitis	No	0
	Yes	1
FruitVeg	No- I do not have 5 portions of fruit and vegtables per day	0
	Yes- I have 5 portions of fruit and vegtables per day	1

Figure 15: Dictionary and description of Risk_factors_1 sample dataset.

Wellbeing Sample Dataset

ID	I've been feeling optimistic about the future	I've been feeling useful	I've been feeling relaxed	I've been feeling interested in other people	I've had energy to spare	I've been dealing with problems well	I've been thinking clearly	I've been feeling good about myself	I've been feeling close to other people	I've been feeling confident	I've been able to make up my own mind about things	I've been feeling loved	I've been interested in new things	I've been feeling cheerful
10001	1	2	2	4	2	2	3	2	4	5	3	3	2	5
10002	3	2	4	5	3	5	2	5	3	4	2	5	2	5
10003	3	4	2	5	1	3	5	4	3	4	5	1	3	5
10004	2	4	5	3	2	4	1	1	2	3	3	2	1	4
10005	4	5	2	3	5	3	2	1	5	2	2	1	3	5
10006	5	3	4	1	2	2	2	4	2	1	3	1	5	5
10007	1	3	2	2	4	4	1	1	1	1	3	3	3	5
10008	5	3	4	5	2	3	5	4	3	5	5	1	5	2
10009	5	4	2	5	3	1	3	3	5	1	5	3	3	5
10010	3	5	5	1	5	2	3	3	4	5	1	3	2	5
10011	1	3	3	5	5	3	3	4	4	5	5	4	5	3
10012	3	5	1	2	1	3	2	4	2	5	3	2	4	1
10013	2	1	2	4	5	2	3	4	3	4	3	4	2	5
10014	1	5	2	4	2	5	3	3	3	1	5	3	2	2

Figure 16: Snapshot of UNIMAN pilot Wellbeing sample dataset.

Dictionary

Table 1: Dictionary and description of Wellbeing sample dataset.

Code	Description
1	None of the time
2	Rarely
3	Some of the time
4	Often
5	All of the time

Food group Sample Dataset

ID	Cereals (grains, beans, legumes)	Vegetables	Fruits (sometimes grouped with vegetables)	White meat	Red meat	Processed meat	Dairy	Confectionery (aka sugary foods)
10001	6	8	4	2	1	1	2	7
10002	4	1	3	9	3	1	3	5
10003	3	4	8	6	6	8	2	5
10004	3	4	2	5	1	8	1	3
10005	6	3	4	5	3	4	5	5
10006	8	-9	4	8	6	6	9	5
10007	1	1	8	3	4	2	4	4
10008	6	3	1	9	4	7	1	2
10009	9	9	3	7	5	3	3	6
10010	4	6	3	1	2	6	6	8
10011	6	6	3	3	5	7	9	5
10012	1	6	8	1	3	9	4	3
10013	4	9	8	1	3	2	2	7
10014	6	6	7	2	5	9	1	7

Figure 17: Snapshot of UNIMAN pilot Food group sample dataset.

Dictionary

Table 2: Dictionary and description of Food group sample dataset.

Code frequency	Description of code

1	Never or less than once a month
2	1-3 times per month
3	once a week
4	2-4 times per week
5	5-6 times perweek
6	once a day
7	2-3 times per day
8	4-5 times per day
9	6+ times per day
-9	missing values

Physical Activity Sample Dataset

	A	B	C
1	ID	Moderate physical activit	Vigorus physical activity
2	10001	1	1
3	10002	0	0
4	10003	1	1
5	10004	1	1
6	10005	1	1
7	10006	0	0
8	10007	1	1
9	10008	0	0
10	10009	0	0
11	10010	1	1
12	10011	1	0
13	10012	0	1
14	10013	0	1
15	10014	0	0

Figure 18: Snapshot of UNIMAN pilot Physical Activity sample dataset.

Dictionary

Table 3: Dictionary and description of Physical Activity sample dataset.

Type of physical activity	Description	Code	Meaning
Moderate physical activity	On average have you undertaken at least 30 minutes of moderate physical activity per day – either at home or at work. (These activities can be made up of many components, for example, moving a table, pushing a vacuum cleaner, bowling or playing golf).	0	No
		1	Yes
Vigorous physical activity	On average have you undertaken 20 minutes or more of energetic activity at least 3 times per week whilst NOT at work. (These include, for example, keep fit, dancing or exercises, swimming or other brisk sport, long walks, jogging or running, hard work in a job at home or in the garden, cycling).	0	No
		1	Yes

Conceptual Diagram

Provide a simple UML (or any other graphic class) diagram representing the names of entities described in the dataset, their relationship and cardinality. Just for reference, the following figure provides an example of a class diagram to be replaced with the actual diagram of the dataset.

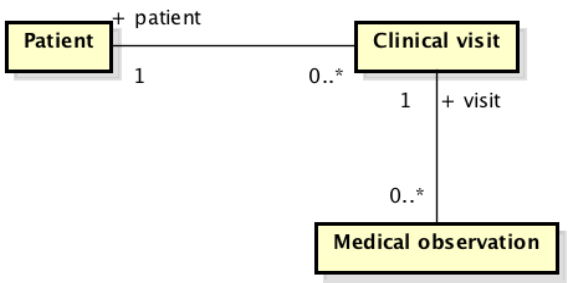

Figure 19: Example of dataset's entities UML Conceptual Diagram

List of Entities

List and describe the entities reported in the conceptual diagram of the previous section using a table as in the following example.

Table 4: Example table listing entities of a sample dataset.

#	Entity name	Description
1	Patient	Demographics and other administrative information about an individual receiving care or other health-related services
2	Clinical visit	An interaction between a patient and healthcare provider for the purpose of providing healthcare services or assessing the health status of a patient

3	Medical Observation	Measurements and simple assertions made about a patient

Constraints – Cleaning Actions

Mandatory Constraints that must be fulfilled for each unique attribute:

- **Specific data type** (e.g. Numeric, String)
- **Mandatory field**
- **Specific value length** (e.g. maximum 20 digits)

Optional Constraints that could be fulfilled for each unique attribute:

- **Specific coding standard** (e.g. LOINC, SNOMED, ICD10)
- **Value representation** (e.g. text formatting "123-45-67" or "1234567" or "123 45 67")
- **Value uniformity** (e.g. all times are provided in UTC, all weight values in KGs, etc.)
- **Value range constraints** (minimum and maximum values)
- **Pre-defined values** (e.g. values selected from a drop-down list)
- **Regular expression patterns** - data that has a certain pattern in the way it is displayed, such as phone numbers)
- **Separation of values** (e.g. complete address in free form field without any indication where street ends, and city begins)
- **Uniqueness** - data that cannot be repeated and require unique values (e.g. social security numbers)
- **Logical Error** (e.g. female individual with prostate cancer medications prescribed)
- …

For the different constraints described, the list of cleaning (corrective) actions should be documented in the table. The following list includes some examples that can be used or combined for the described constraints. Note: This is an indicative and not an exhaustive list. Additional cleaning actions can be introduced and described by the UC partner in case they are not covered in the list below.

- **Deletion of value** that does not conform to a constraint by:
  - Drop whole entity
  - Drop specific attribute
  - …
- **Replacement of value** that does not conform to a constraint through:
  - Transformation of wrong data type value
  - Prediction of erroneous/missing value
  - Prediction of erroneous/missing value based on similar values in the past
  - Creation of a list of features with high percentage of similarity with the same value
  - ……

Risk_factors_1 Sample Dataset

ID (example)

Table 5: Example table for specifying constraint rules and cleaning actions for the ID attribute.

	#	Constraint Type	Constraint Description	Cleaning Action
Mandatory	1	Specific data type	The expected value must be a Numeric value	Replacement of value through transformation of non-numeric value with a numeric one
	2	Mandatory field	The value is mandatory	Deletion of value by dropping the whole entity

	3	Specific value length	Positive integer max 5 digits	Deletion of value by dropping the whole entity
Optional	4	Uniqueness	All the values must be unique	Deletion of value by dropping the duplicate entries and keep only the first one
	5	Value representation	“1234567890”	Replacement of value through transformation to the expected format