CrowdHEALTH

Collective Wisdom Driving Public Health Policies

Del. no. – D3.19. Reliable Information Provision in Healthcare: Design & Open Specification v.1

Project Deliverable

This project has received funding from the European Union's Horizon 2020 Programme (H2020-SC1-2016-CNECT) under Grant Agreement No. 727560

D3.19. Reliable Information Provision in Healthcare: Design & Open Specification v.1

Work Package:		WP3
Due Date:		31/10/2018
Submission Date:		30/11/2017[1]
Start Date of Project:		01/03/2017
Duration of Project:		36 Months
Partner Responsible of Deliverable:		SiLo[2]
Version:		1.1
Status:	☒ Final ☐ Draft ☐ Ready for internal Review ☐ Task Leader Accepted ☐ WP leader accepted ☒ Project Coordinator accepted	
Author name(s):	Dimitris Miltiadou (SiLo), Konstantinos Perakis (SiLo), Argyro Mavrogiorgou (UPRC)	
Reviewer(s):	R. Fernandez (UPM)	U. Wajid (ICE)
Nature:	☒ R – Report ☐ D – Demonstrator	
Dissemination level:	☒ PU – Public ☐ CO – Confidential ☐ RE – Restricted	

REVISION HISTORY			
Version	Date	Author(s)	Changes made
0.1	03/10/2017	Konstantinos Perakis (SiLo)	Draft - Index
0.2	09/10/2017	Dimitris Miltiadou & Konstantinos Perakis (SiLo)	Contribution to section 2
0.3	13/10/2017	Dimitris Miltiadou & Konstantinos Perakis (SiLo) Argyro Mavrogiorgou (UPRC)	Contribution to section 2
0.4	20/10/2017	Dimitris Miltiadou & Konstantinos Perakis (SiLo)	Contribution to sections 2, 3
0.5	23/10/2017	Dimitris Miltiadou & Konstantinos Perakis (SiLo) Argyro Mavrogiorgou (UPRC)	Contribution to section 3
0.6	25/10/2017	Dimitris Miltiadou & Konstantinos Perakis (SiLo)	Contribution to section 3, 4
0.7	3/11/2017	Dimitris Miltiadou & Konstantinos Perakis (SiLo) Argyro Mavrogiorgou (UPRC)	Revisions to sections 2, 3
0.8	8/11/2017	Dimitris Miltiadou &	Contribution to section 3, 4

[1] Please note that the current deliverable was contractually scheduled to be delivered on 31/10/2018. Nevertheless, the consortium deemed it was more appropriate to bring this deliverable forward and deliver the first version along with the first version of the rest of the WP3 components.
[2] Please note that due to the PM allocation within the task, the leadership of the deliverable changed from UPRC to SiLo, which will officially be communicated to the EC with a DoA amendment.

		Konstantinos Perakis (SiLo) Argyro Mavrogiorgou (UPRC)	
0.9	16/11/2017	Dimitris Miltiadou (SiLo)	Minor modifications
0.95	20/11/2017	Konstantinos Perakis (SiLo)	Pre-Final version – Review Ready
0.95_UPM	27/11/2017	M. Patiño (UPM)	UPM Internal Review
0.95_ICE	27/11/2017	U. Wajid, C. Orton (ICE)	ICE Internal Review
0.98	29/11/2017	Dimitris Miltiadou & Konstantinos Perakis (SiLo)	Addressed user comments
0.99	30/11/2017	Argyro Mavrogiorgou (UPRC)	Addressed user comments
1.0	30/11/2017	Konstantinos Perakis (SiLo)	Final version
1.1	11/12/2017	ATOS	Quality review and submission to EC

List of acronyms

CSV	Comma Separated Value
HHR	Holistic Health Record
JSON	JavaScript Object Notation
k-NN	k-Nearest Neighbours
LOCF	Last Observation Carried Forward
LNZ	Last Non Zero
UTC	Coordinated Universal Time
XML	eXtensible Markup Language

Contents

Table of Figures

Table of Tables

Executive Summary

The purpose of D3.19 is to document the preliminary efforts undertaken within the context of Task 3.5 Data Cleaning including Sources Reliability Assessment. Towards this end, the scope of the current deliverable is to document the architecture and design of the Data Cleaner & Sources Verifier component and the mechanisms that will be used in order to address the volatility of the information provision, as well as the reliability of the data sources, within the context of CrowdHEALTH. This report defines the initial version of the Data Cleaner & Sources Verifier component design and component specifications which will drive the software implementation of the first version. As the project evolves, the forthcoming version of this deliverable will contain updates or refinements on the design and the specifications of the Data Cleaner & Sources Verifier component, in order to address the additional requirements coming from the evaluation of the component and the feedback received. Thus, the second and final version of this deliverable will contain the updated or refined version of the Data Cleaner & Sources Verifier component.

1. Introduction

Deliverable D3.19 is a report documenting the initial version of the Data Cleaner & Sources Verifier component's design and specification. Deliverable D3.19 receives as input the Reference Architecture as elaborated in Deliverable D2.4 - Conceptual Model and Reference Architecture v1 [2], and also the technical requirements that were analysed and presented in Deliverable D2.1 – State of the Art and Requirements Analysis v1 [1].

The current deliverable is organized in the following sections:

1. The first chapter introduces the deliverable, defining its scope and relation to the other project tasks and deliverables.
2. In the second chapter, the component's design (i.e. Data Cleaner & Sources Verifier) is presented. The sub-sections are focusing on explaining the scope of the sub-components (i.e. Data Cleaner, and Sources Verifier) of the main component and their positioning within the project, each one followed by a detailed component description. In the component description, the internal architecture of the component is presented, accompanied with the explanation of the internal components of the sub-component, which are thoroughly explained and documented.
3. The third chapter analyses both the Data Cleaner's and the Sources Verifier's specifications. In the sub-sections the specification of each of the internal subcomponents is documented in details.
4. In the final section the conclusions of the current deliverable are presented along with references on the future developments of the main component.

2. Data Cleaner & Sources Verifier Component Conceptual Design

2.1. Data Cleaner Component Design

2.1.1. Component Scope

The scope of the Data Cleaner component is to deliver the software implementation that will provide the assurance that the provided datasets coming from several heterogeneous data information sources will be clean and complete, to the extent possible. The Data Cleaner component is aiming at providing the processes that will detect and correct (or remove) inaccurate or corrupted datasets containing incomplete, incorrect, inaccurate or irrelevant data elements with the purpose of replacing, modifying or deleting these data elements, also known as “dirty” data. The Data Cleaner component will play a significant role in the course of providing accurate HHRs (Holistic Health Records).

Figure 1 illustrates where the Data Cleaner is located in the context of the data handling architecture diagram (being part of the holistic CrowdHEALTH platform architecture).

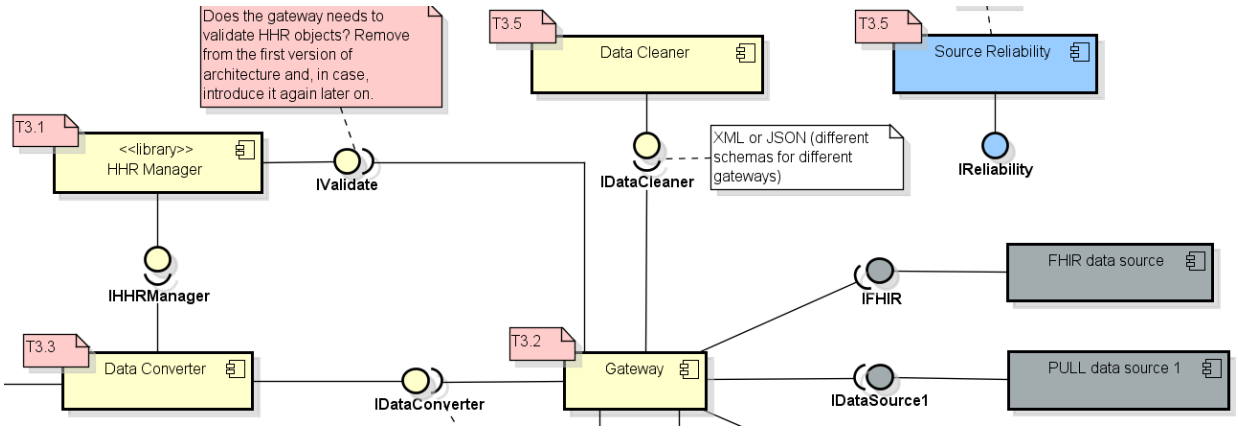

Figure 1: Data Cleaning workflow

The Data Cleaner component will implement the data cleaning workflow providing all the necessary actions towards the aim of consistent datasets across the CrowdHEALTH platform. The data cleaning workflow comprises of the following steps as illustrated also in Figure 2:

1. **Validate Data:** The process of ensuring that a program operates on clean, correct and useful data. Errors should be reported for data element not complying on the specified rules.
2. **De-Cleanse Data:** The process of correcting or removing the data elements for which validation errors were raised.
3. **Check Data Completeness:** The process of ensuring the existence of the required/mandatory data elements on the dataset.

4. **Verify Data**: The process in which the data elements of a dataset are checked for accuracy and inconsistencies after steps like validation, de-cleansing and data completeness are done.
5. **Consolidate & Store Errors' logs:** The process of providing history records of the errors identified and the executed corrective actions.

Figure 2: Data Cleaning workflow

In the initial version of the Data Cleaner component, all steps of the data cleaning workflow will be covered following state of the art techniques (as described in Deliverable D2.4 and also in the context of handling missing data in section 2.1.2.4.1) with the aim of providing a complete and concrete implementation that addresses the volatility of the incoming information towards the aim of providing accurate, consistent and useful data to the CrowdHEALTH platform. Furthermore, this initial version will prepare the ground for the second version of the Data Cleaner component where the list of provided functionalities will be further enriched with more advanced techniques and processes.

2.1.2. Component Description

The **Data Cleaner** component is composed by several internal sub-components, each one providing an interface to support the interconnection and collaboration with the rest of the internal sub-components or other internal components of the CrowdHEALTH platform. Figure 3 illustrates the internal architecture of the Data Cleaner component, displaying all internal sub-components along with their interfaces.

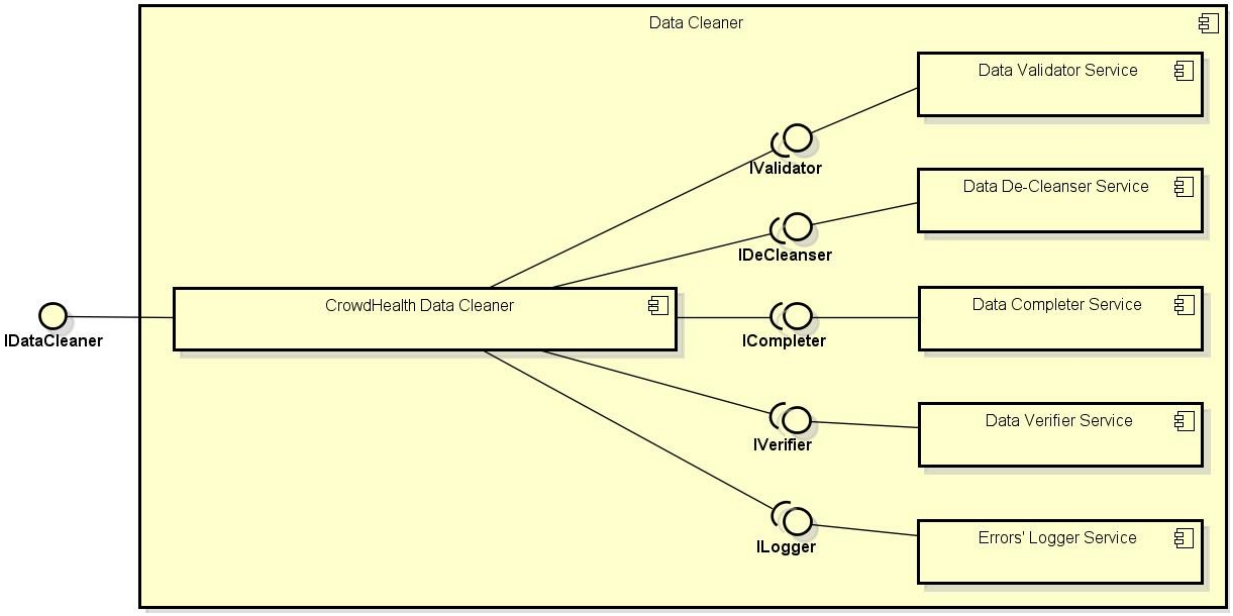

Figure 3: Data Cleaner Component design

As depicted in the internal architecture diagram, the CrowdHEALTH Data Cleaner Service comprises of the following main services:

1. Data Validator Service
2. Data De-Cleanser Service
3. Data Completer Service
4. Data Verifier Service
5. Errors' Logger Service

The **Data Validator** service will perform data validation of the incoming information data with the purpose of identifying errors associated with the conformance to specific set of constraints. This service acts as a safeguard that the data measures compare to defined business rules or constraints set. Indicative examples are provided in subsection 2.1.2.2.

The **Data De-Cleanser** service will perform the de-cleansing step of the data cleaning workflow. In particular, this service performs the necessary corrections or removals of errors identified by the Data Validator Service, and depending on the nature of the error, automated cleansing of the information will be performed based on a predefined set of rules, or a moderator (should such a role eventually be supported) is notified to take further actions.

The **Data Completer** service will safeguard the appropriateness and completeness of the incoming information data. This service safeguards conformance to mandatory fields and required attributes of the dataset based again on a predefined set of rules or the desired configuration. On these rules and configuration parameters the actions taken by the service are determined for either automated filling of the missing values by interpolation or

extrapolation techniques or notification of a moderator (should such a role eventually be supported) to take the necessary actions.

The **Data Verifier** service will ensure that all the corrective actions performed by the Data De-Cleanser and the Data Completer service will be executed in compliance to the Holistic Health Record design of the CrowdHEALTH platform. This service ensures that the data will accurately be corrected or completed and the dataset will eventually be error free.

The **Errors' Logger** Service will undertake the responsibility of keeping in records the reported error logs from the rest of the services and the actions taken towards the data cleaning of the incoming information data. Due to the fuzzy nature of the data cleansing workflow it is mandatory that all error events thrown or actions performed during the processing of the incoming information data are held in records. The CrowdHEALTH moderator (should such a user role eventually be supported) should be able to check over the log entries created, the results of the data cleansing workflow along with detailed information on the error or the action that occurred.

Each of the services mentioned above was carefully conceptualised taking into account the requirements that need to be addressed during the execution of every corresponding step of the data cleaning workflow as described in section 2.1.1.

Figure 4 illustrates the data cleaning workflow along with the procedures and processes supported by the CrowdHEALTH Data Cleaner. Every step of the sequence flow is elaborated in the forthcoming paragraphs.

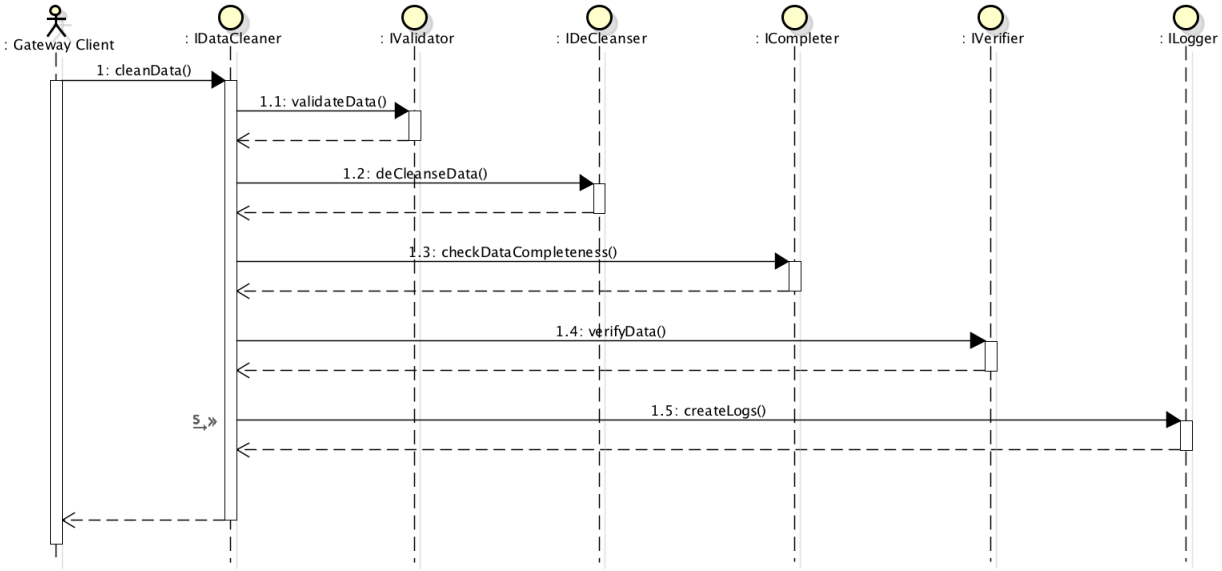

Figure 4: Data Cleaner Component Sequence Diagram

As depicted in Figure 4, the Data Cleaner component is exposing one interface to the rest of the components of the CrowdHEALTH platform, the Data Cleaner interface (IDataCleaner). Through this interface, the Data Gateway component will initiate the data cleaning workflow by pushing the incoming information data received to the Data Cleaner component. Once the Data Cleaner receives the incoming information data it will initiate the first step of the workflow which is the data validation step. Through the IValidator internal interface, the Data Validator service is invoked to perform the error detection due to lack of conformance to the specified set of constraints and in response the list of errors is returned.

Following the data validation step, the Data De-Cleanser service is invoked via the IDeCleanser internal interface. The Data De-Cleanser service is receiving the incoming information data along with the list of errors as identified by the Data Validator service. The Data De-Cleanser service will perform the necessary actions to correct or remove the data elements of the incoming information data marked with errors. As a result, the updated incoming information data is returned along with the list of actions performed on each data element. In the next step, the Data Completer service is invoked by the ICompleter internal interface. The Data Completer service will perform automated filling of missing values again according to the rules set, and will return the updated incoming information data and again the list of actions performed on the corresponding data elements.

Upon performing the data completion step, the Data Verifier service is triggered via the IVerfiier internal interface. The Data Verifier service will ensure that updated incoming information is now error free, accurate and consistent as required by the CrowdHEALTH platform. The results of the verification are returned in response to the verification request.

Finally, once all previous steps have been executed towards the completion of the data cleaning workflow the Errors' Logger Service is triggered via the ILogger internal interface. The Errors' Logger services stores the information in records. These records will give the moderator (should such a role eventually be supported) the opportunity to see exactly when the error was raised and the severity of the error, and finally what were the actions performed by the rest of the internal services on the incoming information data. Once the record for the incoming information data is returned, the Data Cleaner component will return the updated (cleaned) information data to the Data Gateway component as a response to the invocation of the IDataCleaner interface.

2.1.2.1. CrowdHEALTH Data Cleaner Service

The CrowdHEALTH Data Cleaner Service is the mediator service between all the internal components of the Data Cleaner component and it also responsible for the only exposed interface to the other components of the CrowdHEALTH platform, the IDataCleaner.

As already described in sections 2.1 and 2.2, the Data Cleaner component is designed with the aim of executing the data cleaning workflow of CrowdHEALTH platform which includes several steps, each one executed by one of the internal components of Data Cleaner. The

Data Cleaner service is the service responsible for the orchestration of these internal components utilizing their internal interfaces, and for monitoring the execution and assuring the data cleaning workflow is executed successfully.

The CrowdHEALTH Data Cleaner Service provides the single exposed interface which supports the following main function:

- cleanData: The cleanData function is triggered externally (by the Data Gateway which pushes the aggregated information to the Data Cleaner so as to be cleaned), and it constitutes the invocation of the data cleaning workflow in order to perform data cleaning on the incoming data. This function requires as input the incoming data against which the data cleaning workflow will be executed and returns the updated cleaned data. The incoming data will comprise HHR compliant information sources in JSON or XML format. The interface to be exposed by the CrowdHEALTH Data Cleaner Service is in design process and will be documented within the context of D3.21.

2.1.2.2. Data Validator Service

The CrowdHEALTH Data Validator service is performing the data validation processing of the incoming information data. Towards this end, the Data Validator service implements the various data validity checks with the aim to provide guarantees for the accuracy and the consistency of the incoming information data, ensuring the conformance to specific set of constraints. The validation rules are specified by the CrowdHEALTH moderator (should such a role be eventually supported within the context of the project) or by the CrowdHEALTH data scientists, and are integrated in the business logic of the service. More specific, these constraints include:

- Conformance to specific data types (integer, string, etc.).
- Conformance to value representation (e.g. for text formatting "123-45-6789" or "123456780" or "123 45 6789").
- Conformance to uniformity (e.g. all times are provided in UTC, all weight values in KGs, all height values in cm. etc.).
- Conformance to range constraints (minimum and maximum values).
- Conformance to pre-defined values (e.g. values selected from a drop-down list)
- Conformance to regular expression patterns (data that has a certain pattern in the way it is displayed, such as phone numbers).
- Conformance to separation of values (e.g. complete address in free form field without any indication where street ends and city begins.)
- Conformance to cross-field validity (e.g. the sum of the parts of data must equal to a whole).
- Conformance to correct placement of values into attributes (value of ZIP code appears in phone number attribute).

- Conformance to uniqueness (data that cannot be repeated and require unique values (e.g. social security numbers)).
- Identification of duplications.

The Data Validator service provides the internal interface IValidator which supports the following main function:

- validateData: The validateData function is triggered internally and is executed when data validation is requested. This function requires as input the incoming information data against which the data validity will be performed and returns the list of errors associated with conformance to the constraints explained above.

2.1.2.3. Data De-Cleanser Service

The CrowdHEALTH Data De-Cleanser service is performing the de-cleansing processing of the incoming information data. The Data De-Cleanser service undertakes the necessary actions to eliminate the list of errors as provided by the Data Validator service by applying corrections or removals on the incoming information data. The de-cleansing can be performed either in an automated way based on a predefined set of rules, set for example by data scientists within the context of the project, or in a “manual” way, by providing corresponding notifications to the CrowdHEALTH moderator, should such a role eventually be defined and supported in the context of the project.

Examples of data de-cleansing could include:

- Rejection of (or notification about) an arbitrary value when a value from a closed list only is allowed.
- Rejection of (or notification about) a value or a set of values not conforming to cross field validity (e.g. sum of values not adding up correctly).
- Identification of logical errors and notification to the CrowdHEALTH moderator (e.g. female individual having prostate cancer medications prescribed).

The Data De-Cleanser service exposes the IDeCleanser internal interface which supports the following main function:

- deCleanseData: The deCleanseData function is triggered internally and is executed when data de-cleansing is requested. This function requires as input the incoming information data and the list of errors identified and returns the updated incoming information data and the list of actions performed on each data element.

2.1.2.4. Data Completer Service

The CrowdHEALTH Data Completer service is responsible for safeguarding that the incoming information data provided has the required completeness. This is mainly translated in

conformance to mandatory fields and required attributes which cannot be empty. The list of mandatory fields and required attributes is determined based on a predefined set of rules and configuration files. Depending on the case of the missing value the Data Completer will perform automatically filling in information based on interpolation or extrapolation techniques or notify the CrowdHEALTH moderator for manual intervention, should manual intervention be specified and supported. Section 2.1.2.4.1 elaborates more on the indicative list of methods and algorithms that will be utilized to address missing values in the scope Data Completer.

The Data Completer service exposes the ICompleter internal interface which supports the following main function:

- checkDataCompleteness: The checkDataCompleteness function is internally triggered when the data completeness evaluation is required. This function requires as input the incoming information data and will return the updated information data and the list of actions performed on each data element.

2.1.2.4.1. Methods to Address Missing Values

As already noted, missing values on a dataset introduce an element of ambiguity into data analysis. For this reason, missing data is always a challenge and to overcome this significant barrier several methods and algorithms have been developed or adapted. In the scope of Data Cleaner service a variety of methods has been selected to handle missing data, including some relatively simple approaches that can often yield reasonable results. The following is an indicative list of methods and algorithms that the consortium partners aim to investigate and integrate in the data cleaning service.

The first method is the Last Observation Carried Forward (LOCF) method which is also known as LNZ (Last Non-Zero) values[3]. In the Last Observation Carried Forward method, a missing follow-up visit value is replaced by (imputed as) that element's previously observed value, however this is suitable mostly in cases where the variability of the estimated value is low.

Another well-known and popular method is the Moving Average method[4]. In this method, the missing value is filled in by calculating the average of different subsets of the full data set, most commonly by calculating the average of the non-zero values of a subset before and after the missing value.

Linear Regression is yet another popular and widely used[5]. Linear Regression establishes a relationship between dependent variable (Y) and one or more independent variables (X) using a best fit straight line (also known as regression line) and it used to predict the value of target variable based on given predictor variable(s).

[3] http://www.statisticshowto.com/last-non-zero/
[4] http://www.statisticshowto.com/moving-average/
[5] http://www.statisticssolutions.com/what-is-linear-regression/

Alternative methods for automatically filling missing values use basic statistics for measuring central tendencies such as the mean method. Mean consists of replacing the missing data for a given variable by the mean of all known values of that variable. Another variation is the Median which is the middle value for a set of data that has been arranged in order of magnitude. The purpose of using median is that it is less affected by outliers and skewed data and is a value present in the dataset. For categorical values, the mode method is used where the most frequent value is used as the default to fill in the missing values.

Multiple imputation is a method where the missing value is filled by several randomly selected values from a distribution which also reflects sampling variability[6]. This produces a number of plausible alternative versions of the complete data. Each of the alternative datasets is analysed by a standard data completion method like the ones we have already described. The results are later consolidated into one result by calculating the mean, variance, and confidence interval of the variable of concern.

Another method for handling missing data is through the use of the *k*-Nearest Neighbours algorithm (*k*-NN)[7]. It is a non-parametric method and one of the simplest of all machine learning algorithms, employed mainly for classification and regression problems. The input consists of the *k* closest training examples in the feature space, while the output denotes either a class membership (for classification task) or a property value (for regression task). For every observation with a value to be imputed, it identifies 'k' closest observations based on the euclidean distance and computes the weighted average (weighted based on distance) of these 'k' observations.

Finally, an alternative method for automatically filling missing values is using decision trees and especially the C4.5 algorithm[8]. Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). In particular, C4.5 handles both continuous and categorical attributes to construct a decision tree by using Gain Ratio as an attribute selection measure.

It should be noted that, at this point, this list is indicative and not binding, while whether the selection of the missing data handling method will be automated or will include manual intervention is still under discussion. In general, the data cleaning science requires data scientists being involved throughout the data cleaning process. Towards this end, from a technical perspective, several algorithms will be supported, giving the opportunity to a data scientist (or to the CrowdHEALTH moderator, should such a role be eventually supported) to experiment with the data at hand and apply different algorithms customising the thresholds.

[6] https://en.wikipedia.org/wiki/Imputation_(statistics)#Multiple_imputation
[7] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
[8] https://en.wikipedia.org/wiki/C4.5_algorithm

2.1.2.5. Data Verifier Service

The Data Verifier service is responsible for determining and evaluating if the corrective actions taken by the Data De-Cleanser and the Data Completer service were towards the increase of accuracy and consistency of the incoming information data. The Data Verifier service will perform the necessary checks to ensure that the incoming information does not contain any errors and is compliant with the rules of the Holistic Health Records of the CrowdHEALTH.

The Data Verifier service exposes the IVerifier internal interface which supports the following main function:

- verifyData: The verifyData function is triggered internally when the updated incoming information data coming for data de-cleansing and data completeness steps must be verified for accuracy and inconsistencies. The function requires as input the incoming information data and will return the results of the verification.

2.1.2.6. Errors' Logger Service

The CrowdHEALTH Errors' Logger service is responsible for consolidating and storing the error logs as reported and the corrective actions as performed by the rest of the internal services of the Data Cleaner component. The Errors' Logger service will create a new error log record containing all the errors events raised and the corrective actions performed by the rest of the internal services of the Data Cleaner component during the data cleaning workflow execution making sure that all the adequate information is included in the record.

These records are consisted with the information of when the error or cleaning action occurred (date), where the error or cleaning action occurred (dataset) and by who (service) so that the moderator can retrieve a report of the all errors and actions taken within the data cleaning workflow execution.

The Errors' Logger service exposes the ILogger internal interface which supports the following main function:

- createLogs: The createLogs function is triggered internally and is executed when all data cleaning steps are completed in order to create the record of all errors raised and of all corrective actions that took place. This function requires as input the list of errors raised by the rest of the internal services along with the list of corrective actions and returns the new records holding all the necessary information for storage and traceability.

2.2. Sources Verifier Component Design

2.2.1. Component Scope

The role of the Sources Verifier sub-component is to dynamically categorize both known and unknown data sources to specific “levels of trustfulness” (i.e. data reliability, provided information type, data availability), according to a given threshold. As a result, all the data sources that do not meet the “trustfulness” criteria will be rejected from being integrated into the CrowdHEALTH platform, ensuring not only the origination of the data sources’ incoming data, but also the adaptive selection of all these available data sources in order to be connected into the CrowdHEALTH platform.

The Sources Verifier sub-component will implement the sources verification workflow providing all the necessary actions for choosing and finally keeping connected to the CrowdHEALTH platform only the trustful and reliable data sources. In more details, the sources verification workflow consists of the following steps (Figure 5):

1. **Recognize Sources:** The process of identifying and categorizing the available data sources that want to connect to the CrowdHEALTH platform into either known or unknown data sources.
2. **Classify Specifications:** The process of classifying the unknown identified data sources based on the specifications of the identified known data sources.
3. **Map to levels of Trustfulness:** The process of mapping the classified data sources into specific levels of trustfulness.

Figure 5: Sources verification workflow

2.2.2. Component Description

In the context of the Sources Verifier sub-component, an innovative predictive selection mechanism is proposed for achieving data reliability and availability during runtime, concerning both known and unknown data sources that have been identified in order to be connected into the CrowdHEALTH platform. In more details, the Sources Verifier sub-component provides a mechanism that makes use of the aforementioned services, being able to classify both known and unknown data sources to specific “levels of trustfulness”, based on the data sources’ data reliability, provided information type, as well as data availability in terms of the frequency of the data transmission. Hence, the proposed mechanism will consist of three (3) different stages (Figure 6), one stage for each service that is provided.

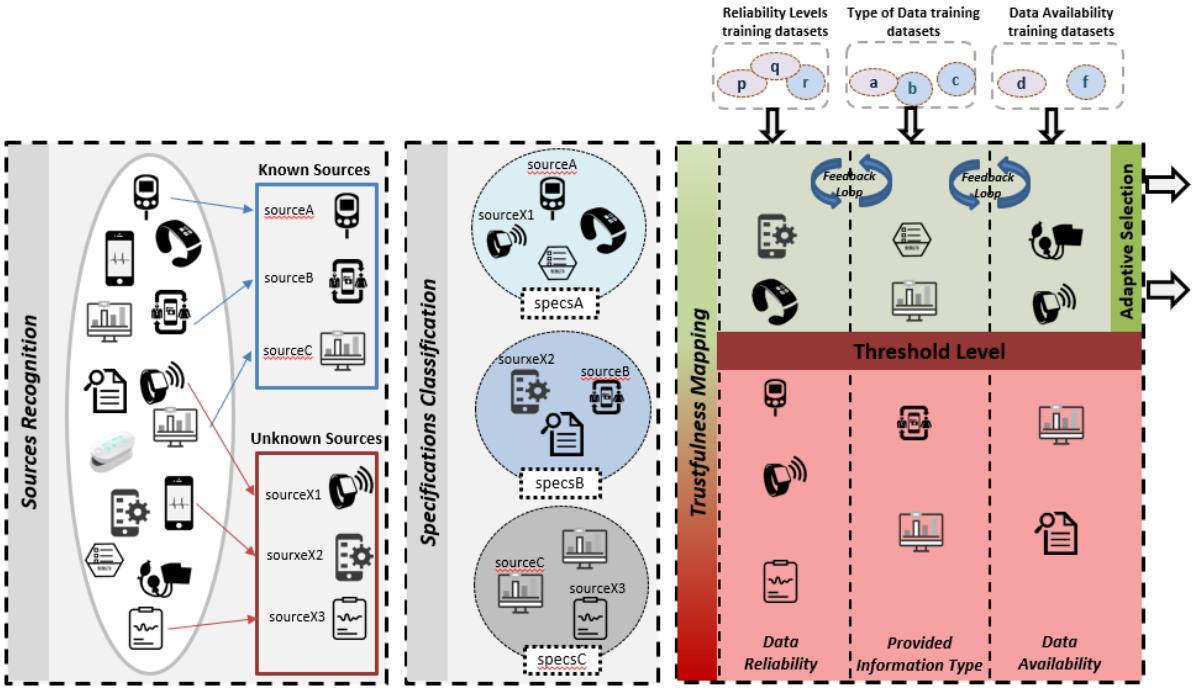

Figure 6: Sources reliability mechanism

In the first stage the identification of both known and unknown data sources' specifications (i.e. software and hardware specifications) takes place. Afterwards, in the second stage a classification mechanism is proposed, that is responsible for classifying data sources' specifications based on the known data sources' specifications, whilst in the third stage the mapping of the data sources (that are from-now-on known) to different levels of trustfulness takes place. It should be noted that the proposed mechanism requires prior knowledge of the used data sources, regarding their specifications (i.e. hardware and software specifications), which will contain the same semantics in terms of specifications' descriptions and measurement units.

Therefore, based on the proposed stages of the Sources Verifier mechanism, this is composed by several internal sub-components, each one providing an interface to support the interconnection and collaboration with the rest of the internal sub-components or other internal components of the CrowdHEALTH platform. Figure 7 illustrates the Sources Verifier component design diagram, displaying all its internal sub-components along with their interfaces.

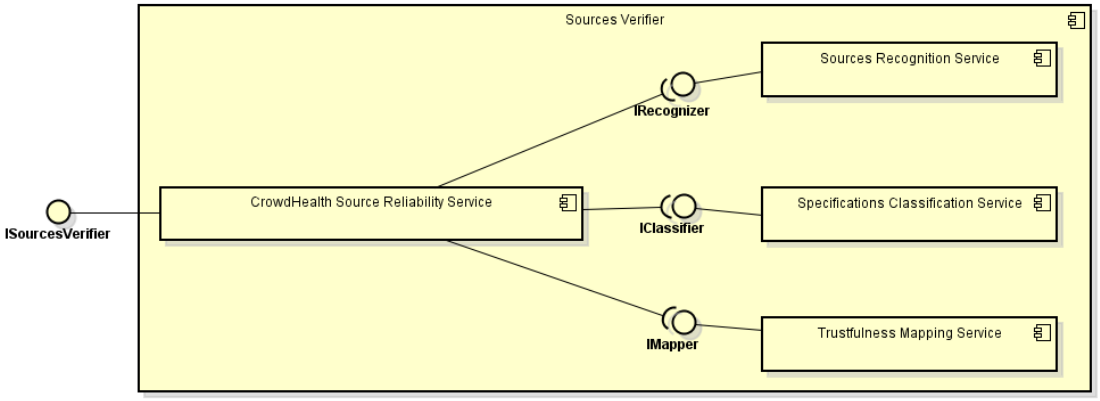

Figure 7: Sources Verifier Component design

More specifically, as depicted in the Sources Verifier component design diagram, the CrowdHEALTH Sources Reliability Service consists of the following three (3) main services:

1. Sources Recognition Service.
2. Specifications Classification Service.
3. Trustfulness Mapping Service.

2.2.2.1. Sources Recognition Service

In the first service of the mechanism (i.e. Sources Recognition Service), the identification of the various available data sources (e.g. sensors, (mobile) applications, tablets, watches, etc.) that need to be connected to the CrowdHEALTH platform takes place. Initially, the various data sources are discovered and categorized into either known or previously unknown data sources. Afterwards, all these data sources are connected via the interface that is provided by the existing network layer to the CrowdHEALTH platform. It is assumed that this network layer is providing an interface to the various data sources, thus being responsible for collecting general information from them, concerning data sources' specifications (i.e. hardware and software). Therefore, information is gathered about both known and unknown data sources' specifications.

2.2.2.2. Specifications Classification Service

In sequel, as soon as the data sources' specifications have been collected, the classification of them occurs through the second service of the mechanism (i.e. Specifications Classification Service). More particularly, knowing all the data sources' (i.e. both known and unknown) specifications (e.g. brand, model, memory, etc.), these are classified according to the similar specifications they have. To this end, two (2) potential scenarios arise: (i) The available known and unknown data sources may have exactly the same specifications, while after classifying them, it is observed that each device is classified into a single classification group (one-to-one scenario), and (ii) The available known and unknown data sources may have partial specifications in common, whilst after classifying them, it is observed that some data sources

may belong to one or more classification groups, having common specifications with one group of data sources, while also having common specifications with a different group of data sources (one-to-many scenario).

2.2.2.3. Trustfulness Mapping Service

Finally, the third service of the mechanism occurs (i.e. Trustfulness Mapping Service), in which the mapping of the data sources to levels of trustfulness takes place. In order for the mechanism to decide whether each data source is considered as reliable or not, a threshold level is being set for each different “trustfulness” criterion (i.e. data reliability, provided information type, data availability). Afterwards, the mechanism aggregates the results of each data source, calculating the average of the pre-defined criteria, deciding whether the data source is considered reliable or not, thus resulting in the adaptive selection of only the reliable data sources (i.e. the data sources that overcome the given threshold).

3. Data Cleaner & Sources Verifier Component Specification

3.1. Data Cleaner Component Design

3.1.1. Data Validator Service

In the following table (Table 1) the specifications with the required information fields for Data Validator services are listed.

Required Field	Field Attribute
Incoming data	In the form of XML or JSON format

Table 1: Data Validator Service Specification

3.1.2. Data De-Cleanser Service

In the following table (Table 2) the specifications with the required information fields for Data De-Cleanser service are listed.

Required Field	Field Attribute
Incoming data	In the form of XML or JSON format
List of identified errors	The list of identified errors per data element in the form of XML or JSON format

Table 2: Data De-Cleanser Service Specification

3.1.3. Data Completer Service

In the following table (Table 3) the specifications with the required information fields for Data Completer service are listed.

Required Field	Field Attribute
Incoming data	In the form of XML or JSON format

Table 3: Data Completer Service Specification

3.1.4. Data Verifier Service

In the following table (Table 4) the specifications with the required information fields for Data Verifier service are listed.

Required Field	Field Attribute
Incoming data	In the form of XML or JSON format

Table 4: Data Verifier Service Specification

3.1.5. Errors' Logger Service

In the following table (Table 5) the specifications with the required information fields for Errors' Logger service are listed.

Required Field	Field Attribute
List of errors and corrective actions	In JSON format.

Table 5: Errors' Logger Service Specification

3.1. Sources Verifier Component Design

3.1.1. Sources Recognition Service

In the following table (Table 6) the specifications with the required information fields for Sources Recognition service are listed.

Required Field	Field Attribute
List of requested specifications	The list of requested specifications per data source in the form of text format (e.g. .txt, .xls, .csv, .xml file)

Table 6: Sources Recognition Service Specification

3.1.2. Specifications Classification Service

In the following table (Table 7) the specifications with the required information fields for Specifications Classification service are listed.

Required Field	Field Attribute
Incoming data (in terms of data sources' specifications)	In the form of text format (e.g. .txt, .xls, .csv, .xml file)

Table 7: Specifications Classification Service Specification

3.1.3. Trustfulness Mapping Service

In the following table (Table 8) the specifications with the required information fields for Trustfulness Mapping service are listed.

Required Field	Field Attribute
Groups of classified data sources	In the form of text format (e.g. .txt, .xls, .csv file)
Reliability levels training datasets	In the form of text format (e.g. .txt, .xls, .csv file)
Type of data training datasets	In the form of text format (e.g. .txt, .xls, .csv file)
Data availability training datasets	In the form of text format (e.g. .txt, .xls, .csv file)

Table 8: Trustfulness Mapping Service Specification

4. Conclusions

The objective of this deliverable was to deliver the initial version of the architecture and design of the Data Cleaner & Sources Verifier component and describe the mechanisms that will be used in order to address the volatility of the information provision, as well as the reliability of the data sources, within the context of CrowdHEALTH. The list of requirements as provided by the deliverable D2.1 – State of the Art and Requirements Analysis v1 [1] and the Reference Architecture as elaborated in the deliverable D2.4 - Conceptual Model and Reference Architecture v1 [2], was utilized as input for the comprehensive analysis that was performed in order to provide the design and specifications of the Data Cleaner & Sources Verifier component, that consists of the Data Cleaner, and the Sources Verifier sub-components.

In the second section, the component's design was presented, describing the scope of the component and providing a detailed component description containing the designed internal architecture and elaboration of the sub-components of the Data Cleaner & Sources Verifier component and their role towards the execution of the envisioned data cleaning, and sources reliability workflow. More particularly, the CrowdHEALTH Data Cleaner sub-component comprises of five (5) main services: 1) The Data Validator Service, 2) The Data De-Cleanser Service, 3) The Data Completer Service, 4) The Data Verifier Service, and 5) The Errors' Logger Service, whereas the Sources Verifier sub-component consists of three (3) main services: 1) The Sources Recognition Service, 2) The Specifications Classification Service, and 3) The Trustfulness Mapping Service.

In the third section the component's specification was analysed, documenting the specifications of the internal sub-components of the Data Cleaner & Sources Verifier component. This analysis will guide the implementation of the internal sub-components.

It should be stressed at this point that the current deliverable comprises the first version of the Data Cleaner & Sources Verifier component and that the forthcoming second and final version will contain the updates or refinements of the design and specifications that will address the identified additional requirements.

5. References

[1] CrowdHEALTH: Collective Wisdom Driving Public Health Policies (2017), Deliverable 2.1 - State of the Art and Requirements Analysis v1

[2] CrowdHEALTH: Collective Wisdom Driving Public Health Policies (2017), Deliverable 2.4 - Conceptual Model and Reference Architecture v1