Linked2Safety

FP7-288328

A Next-Generation, Secure Linked Data Medical Information Space for Semantically-Interconnecting Electronic Health Records and Clinical Trials Systems Advancing Patients Safety in Clinical Research

---

Deliverable D3.1

Interoperable EHR Data Space Design

Editor(s):	David Tian (UNIMAN), Ann Gledson (UNIMAN), John Keane (UNIMAN), Goran Nenadic (UNIMAN), Xiao-jun Zeng (UNIMAN), Hasapis Panagiotis (INTRASOFT), Dimitris Ntalaperas (UBITECH), Konstantinos Perakis (UBITECH), Aristos Aristodemou (UCY), Athos Antoniades (UCY), Cristi Potlog (SIVECO), Constantin Rex (LUH)
Responsible Partner:	UNIMAN
Status-Version:	Final-V1.0
Date:	31/12/2012
EC Distribution:	Public
Project Number:	FP7-288328
Project Title:	Linked2Safety

Title of Deliverable:	D3.1 - Interoperable EHR Data Space Design
Date of Delivery to the EC:	31/12/2012

Workpackage responsible for the Deliverable:	WP3 – Linked2Safety Interoperable EHR Data Space
Contributor(s):	UNIMAN, INTRASOFT, SIVECO, UBITECH, UCY, LUH
Reviewer(s):	INTRASOFT
Approved by:	All Partners

Abstract:	The objective of this deliverable is to present the detailed specifications of Interoperable EHR Data Space software components and their interaction.
Keyword List:	Software Components, Common EHR Schema, Semantic EHR Model, Data Privacy Framework, Non-identifiable Data Cubes, Semantic Annotation/Enrichment, Quality Control, RDFization, Activity Diagrams, Class Diagrams, Sequence Diagrams

Document Description

Document Revision History

Version	*Date*	*Modifications Introduced*	
		Modification Reason	*Modified by*
V0.1	14/12/2012	Typos, Figures 21, 25 updated, requirements analysis added	INTRASOFT, UBITECH, CERTH, LUH
V0.2	21/12/2012	Typos	INTRASOFT, UCY
V0.3	23/12/2012	Typos	UNIMAN
V1.0	28/12/2012	Formatting	UNIMAN

Contents

List of Figures

List of Tables

Definitions, Acronyms and Abbreviations

Table 1: **Definitions, Acronyms and Abbreviations**

Acronym	Title
AOM	Archetype Object Model
BFO	IFOMIS Basic Formal Ontology
COG	Content Oriented Guidelines
EDC	Electronic Data Capture
EHR	Electronic Health Record
OBO	Open Biological and Biomedical Ontologies
OLAP	Online Analytical Processing
OWL	Web Ontology Language
RDF	Resource Description Framework
SDMX	Statistical Data and Metadata Exchange
SNP	Single Nucleotide Polymorphism
SUMO	Suggested Upper Merged Ontology
UML	Unified Modelling Language
URI	Universal Resource Identifier

1. Executive Summary

The present document is Deliverable D3.1 “Interoperable Electronic Health Record (EHR) Data Space Design” of the Linked2Safety project. The Interoperable EHR Data Space is responsible for transforming data of various formats from distributed heterogeneous EHR and EDC (Electronic Data Capture) sources into a common format so that the data can be linked together in the Linked Medical Data Space as defined in WP4.

This deliverable presents the design of the Interoperable EHR Data Space in terms of software components which are specified using the Unified Modelling Language (UML), a widely-used object-oriented software modelling language. The design will be used to implement the components of the Interoperable EHR Data Space in tasks 3.2 and 3.3. The design methodology adopted takes into account the Common EHR Schema (outcome of Task 1.3 “Linked2Safety Common EHR Schema Definition”), which defines the Reference Model and Archetypes of EHRs; the Semantic EHR Model (outcome of Task 1.4 “Linked2Safety Semantic EHR Model”), which is a domain ontology specifically designed for clinical trials; and the Data Privacy Framework (outcome of Task 2.3 “Linked2Safety Data Privacy Framework and Consent Forms”), which defines non-identifiable data cubes to ensure anonymity of patients selected for clinical trials and to govern patients’ data as stated in patients’ consent forms. The outputs of the Interoperable EHR Data Space are non-identifiable semantically-enriched RDF data cubes that are used as input to the Linked Medical Data Space (WP4) and can be linked together in that Space.

Firstly, the outcomes of Tasks 1.3, 1.3 and 2.3 are reviewed. Secondly, components are identified based on the Linked2Safety Reference Architecture presented in the D1.2 “Reference Architecture”. Thirdly, each component is designed using activity diagrams, class diagrams and sequence diagrams. Activity diagrams specify the main activities (operations) of components and their inputs and outputs; class diagrams specify the properties of components; sequence diagrams specify the run-time behaviours of components. Finally, the components are integrated to compose the Interoperable EHR Data Space using activity diagrams, class diagrams and sequence diagrams.

2. Introduction

2.1 Document Scope

The present document is Deliverable D3.1 "Interoperable EHR Data Space Design". This deliverable presents the specifications of the software components of the Interoperable EHR Data Space in terms of activity diagrams, class diagrams and sequence diagrams [36, 37]. This deliverable is derived from the outcomes of Task 1.3 "Linked2Safety Common EHR Schema" [3], Task 1.4 "Linked2Safety Semantic EHR Model" [4] and Task 2.3 "Linked2Safety Data Privacy Framework and Consent Forms" [5]. The outcome of this deliverable will in turn be used to implement the Interoperable EHR Data Space in tasks 3.2 and 3.3.

2.2 Data Homogenization and Semantic Interoperability

A goal of the Linked2Safety project is to facilitate efficient and homogenized access to the increasing wealth of medical information contained in heterogeneous EHR and EDC systems deployed and maintained at regional and/or national level across Europe, by establishing the concepts and tools that will enable scalable, standardized, secure sharing and reuse of statistical medical data as a foundation for policy prediction, planning and adjustments.

The Linked2Safety consortium has adopted the data cube approach as described in D1.1 "Requirements Analysis" [34] to address the ethical requirements of handling sensitive patient data, namely: respecting patient anonymity, data ownership and privacy, as well as compliance with legislative, regulatory and ethical requirements.

The data cube approach also enables the use of only non-identifiable, small, truncated statistical healthcare data aligned to the D1.3 "Common EHR Schema" [3] for online computation and analysis. In this way, the healthcare data maintained by data providers remain in their secure, on-site, off-line physical environment, and only selected, homogenized and anonymized data cubes will leave the clinical data providers' premises and enter the Linked2Safety platform (see also Section 2.3).

The Linked2Safety project is consequently designed to provide homogenized aggregate information that can be used to analyze the contents of clinical data from the clinical partners. For example, the homogenized and semantically enriched data cubes created in the Interoperable EHR Data Space will be linked together by the Linked Medical Data Space (WP4) and pre-processed and input to the Data Analysis Space (WP5) to perform various analyses and data computations.

Another goal of the Linked2Safety project is to enable semantic interoperability and reuse of clinical data without loss of semantics, in particular for data coming from different data providers, thus enabling retention of specific variations in the way data was gathered or differences in the clinical diagnosis approach throughout the homogenized data analysis process.

Semantic interoperability [6-11] has been considered in the healthcare domain for the integration of data from heterogeneous sources through semantic mediation. Ontologies [12-21], pillars of the

Semantic Web framework, are used to annotate message definitions to assign formal understanding and enable mediation between them. In this respect, ontologies play three major roles:

- To provide a source of precisely (or formally defined) terms to share domain artifacts and preserve the consistency of the terms across applications;
- To enable alignment and interlinking of semantically similar or related domain artifacts;
- To allow reasoning about the domain concepts and their interlinking relationships.

Semantic mediation can be used to convert healthcare messages defined in one standard into another. Such transformation requires the lifting of syntactically defined messages into a semantic format, and then the lowering of the semantic format into the syntactic counterpart. During transformation, semantic differences are reconciled through ontology alignment mechanisms.

2.3 Data Privacy Framework and Relation to the Interoperable EHR Data Space

Given sensitive patient data, the Linked2Safety project takes into consideration various aspects of data privacy. Several criteria and requirements emerged during requirements gathering including:

- Patients' anonymity and privacy must be respected;
- All tasks and analyses performed within the Linked2Safety platform will comply with legislative, regulatory and ethical requirements (European and national) as identified in the Deliverable D2.1 "Legal and Ethical Requirements" [35];
- No raw healthcare data will leave data providers' premises. Only non-identifiable, relatively small, truncated healthcare data, named Linked2Safety data, will be used by the Linked Medical Data Space in the Linked2Safety platform;
- Linked2Safety data used in Linked2Safety will be available in a non-personal and non-identifiable form and will not lead (with reasonable effort) to the identification of a person's identity (either directly or indirectly by backtracking).

One aspect of all these criteria is related to legal documents (Consent Forms and Patient Information Sheets) and agreements (Data Transfer Agreement, User Agreement, Platform Manager Agreement) set up in the Deliverable D2.2.a "Linked2Safety Data Privacy Framework and Consent Forms". All these documents and agreements need to be signed by the relevant parties participating in Linked2Safety in accordance with local and European laws. All processing and analysis on medical data should be done on aggregated data from each site. This is in agreement with the best practice, as aggregated results from the analysis of EHRs are published in the majority of medical research publications that involve medical studies (in the form of contingency tables [2]).

In Linked2Safety, an approach is needed that will enable future analysis of the aggregated data using methodologies or ways of analysis that may have not been considered at the time of aggregating the data. Thus the concept of a "closed-world" room is introduced: it is a room located within a data provider's premises, featuring the required hardware infrastructure to process EHRs isolated from any kind of network connections. Physical access to the machines within the room is allowed only to specific personnel of the corresponding data provider and the machines are off line to the outside world. The data provider's staff will execute a program on the computers located in the “closed-world” room that will aggregate the patient data generating the data cubes. This program will offer the option to the data provider to limit the way in which the data will be aggregated so that any legal and ethical issues identified in the Deliverable D2.1 “Legal and Ethical Requirements” [35] that may relate to the type of analyses that may be performed on the data can be addressed.

The Interoperable EHR Data Space actually realises the entire technological facet of the above. The cornerstone of this space is the production and usage of data units called data cubes that contain aggregated data from the analysis of patient data from different EHRs. It is comprised of several components, the most important being data homogenization (while adhering to a single vocabulary) and the creation of an aggregated data cube based on those aligned instance data.

2.4 Requirements Analysis

The design of the ”Interoperable EHR Data Space” covers the following requirements of the Interoperable EHR Data Space, as derived from Deliverable D1.2 “Reference Architecture” [2]:

- NF1. Respect patients’ anonymity, Linked2Safety data’s ownership and privacy;
- NF2. All tasks and analyses performed within the platform will comply with legislative, regulatory and ethical requirements (European and national);
- NF3. Linked2Safety data used in Linked2Safety will be available in a non-personal and non-identifiable form, and should not lead to the identification of a person’s identity (either directly or indirectly e.g. via backtracking);
- NF4. No healthcare data will leave data providers’ premises. Only non-identifiable, small, truncated healthcare data (referred to as Linked2Safety data) will be used by the Linked Medical Data Space;
- NF5. Computation and analysis of healthcare data maintained by data providers will be carried out in their secure, on-site, off-line physical environment.

2.5 Methodology

The Interoperable EHR Data Space has been designed using the following methodology:

1. **Review of the Outcomes of tasks 1.3, 1.4 and 2.3:** As mentioned in the Description of Work [38], the Interoperable EHR Data Space is designed taking into account the outcomes of Task 1.3 "Linked2Safety Common EHR Schema", Task 1.4 "Linked2Safety Semantic EHR Model" and Task 2.3 "Linked2Safety Data Privacy Framework and Consent Forms". Therefore, the outcomes of tasks 1.3, 1.4 and 2.3 are reviewed;

2. **Identification of Components:** The software components are identified in a top-down fashion based on the reference architecture of the Linked2Safety platform presented in D1.2 "Reference Architecture" [2];

3. **Design of Components:** Each component is designed using UML, an object-oriented software design approach, in terms of activity diagrams, class diagrams and sequence diagrams. During the design of the components, the outcomes of tasks 1.3, 1.4 and 2.3 are taken into account;

4. **Integration of Components:** After completion of the design of each component, all components are integrated using UML activity diagrams, class diagrams and sequence diagrams in a bottom-up fashion to compose the Interoperable EHR Data Space.

Activity diagrams [36,37] describe the operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. A class diagram [36,37] is a static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. A sequence diagram [36,37] is an interaction diagram that shows how processes operate with one another in a given order. It is a construct of a message sequence chart and shows object interactions arranged in time. It depicts the objects and classes involved in a scenario and the sequence of messages exchanged between the objects, needed to carry out the functionality of the scenario.

2.6 Document Structure

Section 3 reviews the outcomes of tasks 1.3, 1.4 and 2.3 which are the cornerstones of the Interoperable EHR Data Space. Section 4 presents component identification, component design and component integration. Finally, Section 5 concludes the document.

3. Outcomes of Tasks 1.3, 1.4 and 2.3

3.1 Outcomes of Task 1.3 "Linked2Safety Common EHR Schema Definition"

The outcomes of Task 1.3 [3], the Linked2Safety Common EHR Schema and the RDF Data Cube schema, are summarized here.

3.1.1 Linked2Safety Common EHR Schema

The Linked2Safety Common EHR Schema [3] is a subset of the *open*EHR Schema, pertaining to patient oriented clinical trials data from medical data providers. The schema allows the creation of locally aligned and maintained EHR repositories (with medical and healthcare data) and EDC databases (with clinical trials system information), as well as the alignment, transformation and standardized bridging of the existing EHR and EDC repositories to a common-reference EHR schema – addressing syntactic and technical interoperability between EHR and EDC repositories. It is composed of the Reference Model and the Archetype Model, which are summarized here.

- The Reference Model consists of the following schema files in OWL2:
  - *BasicTypes.owl* – types from rm.data_types and rm.support.identification packages;
  - *Structure.owl* – types from rm.data_structures and rm.common.generic packages;
  - *Resource.owl* – types from the rm.common.resource package;
  - *Content.owl* – item, Section, Entry and all subtypes;
  - *Composition.owl* – rm.composition;
  - *Version.owl* – the Version classes from rm.common.change_control;
  - *Semantics.owl* – annotation properties used semantic mappings of EHR concepts.

- The Archetype Model is composed of the following schema files in OWL2:
  - *Archetype.owl* – types from the Archetype Object Model (AOM);
  - *Profile.owl* – types from the *open*EHR Archetype Profile package;
  - *Clinical.owl* – types used for representing Linked2Safety clinical trials data from medical partners.

Figure 1 depicts the top-down referential hierarchy of the above OWL files which compose the Common EHR Schema.

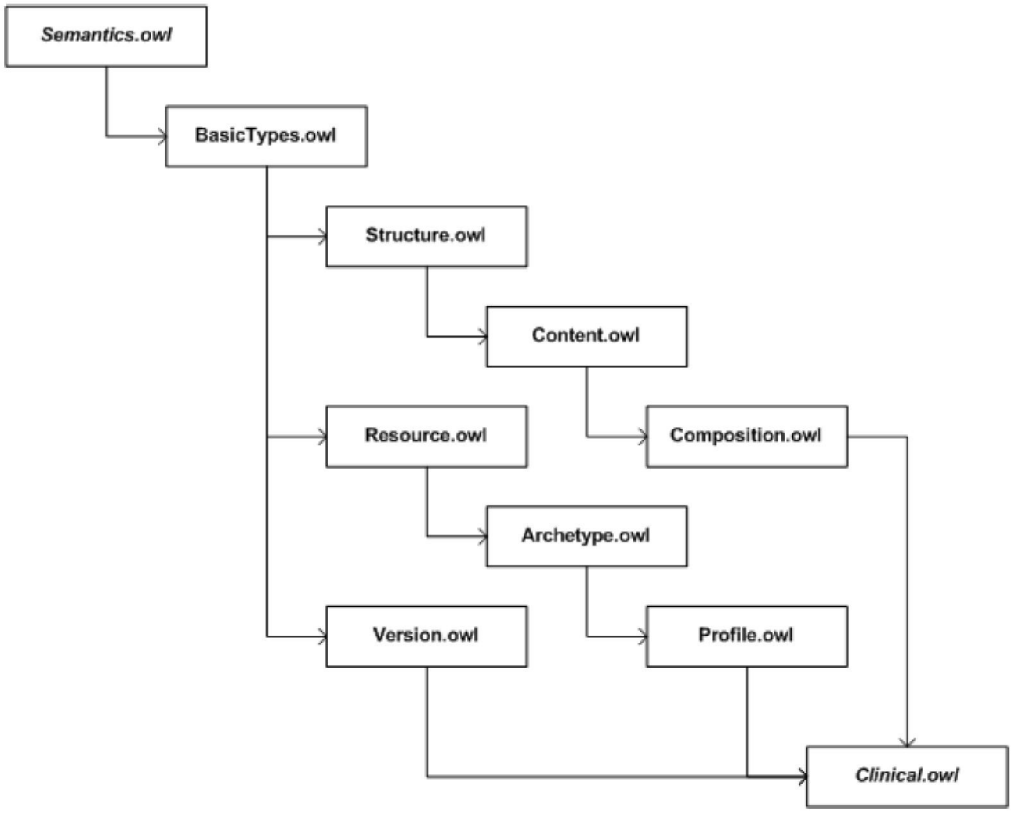

Figure 1: Linked2Safety Common EHR Schema: from basic types (high-levels) to complicated ones (lower-levels).

3.1.2 RDF Data Cube Schema

The RDF data cube schema is composed of the DataCube profile and the RDF DataCube Vocabulary. The *DataCube* profile is based on the *DataWarehouse* profile [3], which contains a set of stereotypes (e.g. *Fact* and *Dimension*). Each stereotype represents a single multidimensional OLAP (Online Analytical Processing) concept by extending the specific UML *metaclass (*considered as the most semantically close to the OLAP concept). The *DataCube* profile includes the following five stereotypes:

- ***Cell*** summarizes a set of *Facts* of a given cube;
- ***Axis*** shows each *Base* (a component of group-by clause) of a given cube;
- ***CellMember*** represents each *Measure;*
- ***CellAxis*** represents each *Descriptor;*
- ***Slice*** represents a predicate formulated on *CellMembers* and/or *AxisMembers.*

The RDF Data-Cube Vocabulary [22] is focused on the publication of multidimensional data on the web. The Linked2Safety project uses the RDF Data-Cube Vocabulary to represent clinical trials data as aggregated RDF Data Cubes, thus enabling data anonymization by omitting personal information.

3.2 Outcome of Task 1.4 "Linked2Safety Semantic EHR Model"

In Task 1.4 [4], the Linked2Safety Semantic EHR Model is designed, implemented and evaluated. The Semantic EHR Model is a domain ontology specifically designed for clinical trials. It incorporates upper-layer and domain ontologies [22-33]. An upper ontology (also known as a top-level ontology) is an ontology (in the sense used in information science) which describes very general concepts that are the same across all knowledge domains. An important function of an upper ontology is to support very broad semantic interoperability between a large number of ontologies which are accessible "under" this upper ontology. It is usually a hierarchy of entities and associated rules that attempt to describe those general entities that do not belong to a specific problem domain.

The Semantic EHR (SEHR) Model is based upon an upper ontology: the IFOMIS Basic Formal Ontology (BFO) [22]. BFO grows out of a philosophical orientation which overlaps with the work done in Suggested Upper Merged Ontology (SUMO). However, it is narrowly focused on the task of providing a genuine upper ontology which can be used in support of domain ontologies developed for scientific research, as for example in biomedicine within the framework of the OBO Foundry. Thus BFO does not contain physical, chemical, biological or other terms which would properly fall within the specialist sciences domains. The other upper-layer ontology we use is the Open Biological and Biomedical Ontologies (OBO) [23]. The OBO Foundry is a collaborative project involving developers of science-based ontologies. Their aim is to establish a set of principles for ontology development with the goal of creating a suite of orthogonal interoperable reference ontologies in the biomedical domain.

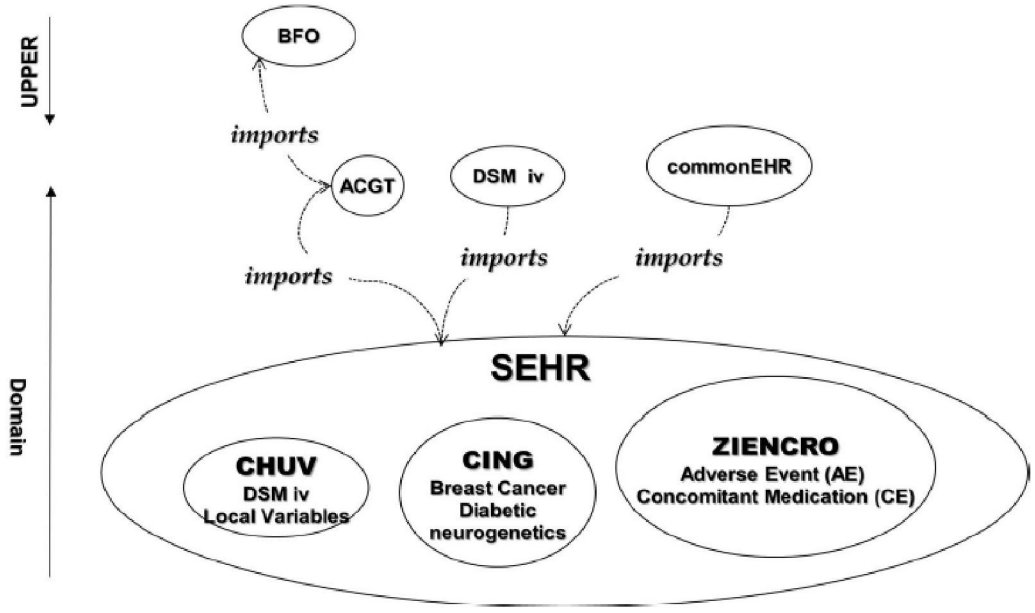

Figure 2: Importing structure of the Semantic EHR Model (SEHR), DSM-IV, ACGT, common EHR, and BFO ontologies

The Linked2Safety Semantic EHR Model imports the ACGT ontology which imports the BFO. For psychiatric cases we have used DSM-iv, which is used in conjunction with the Semantic EHR Model for annotating mental disorder related resources. Figure 2 shows the importing structure of the upper-level ontology BFO, domain ontologies ACGT and DSM-IV, the Common EHR schema (developed in Task 1.3) and the Semantic EHR Model. The Semantic EHR Model incorporates the

best aspects of the upper layer ontologies, providing a way to connect data expressed using these concepts with other external ontologies.

3.3 Outcome of Task 2.3 "Linked2Safety Data Privacy Framework and Consent Forms"

The proposed Linked2Safety data privacy framework [5], which will be under refinement until month 18 of the project, consists of the following approaches:

- Non-identification of patients behind the data achieved through:
  - Data cubes of aggregated data i.e. count of subjects;
  - Data perturbation of aggregated data;
  - Cell suppression with a certain threshold.
- A data governance model (consisting of general terms and conditions) accompanied by legal documents (Data Transfer Agreement, Platform Manager Agreement, User Agreement; all agreements to be signed by the parties involved and Consent Forms with patient information sheets) to govern that patient data is only used as stated in the associated consent forms.

4. Interoperable EHR Data Space Design

4.1 Identification of Components

Based on the Linked2Safety Reference Architecture (D2.1 "Reference Architecture"), the following seven components of the Interoperable EHR Data Space have been identified:

1. Schema Mapping and Alignment component;
2. Raw Data Quality Control component;
3. Transformation to the common EHR Schema component;
4. Non-identifiable Data Cubes Creation component;
5. RDFizer component;
6. Information Loss Quality Control component;
7. RDF Cubes Semantic Annotation, Enrichment and Storage component.

Figure 3 below shows the interactions between the components and which components take into account the outcomes of tasks 1.3, 1.4 and 2.3. The Schema Mapping and Alignment component is designed to take into account the common EHR Schema (the outcome of Task 1.3 "Linked2Safety Common EHR Schema Definition"). The Non-identifiable Data Cubes Creation component is designed by taking account of the outcome of Task 1.3 "Linked2Safety Common EHR Schema Definition" and conforming to the data privacy framework, the outcome of Task 2.3 "Linked2Safety Data Privacy Framework and Consent Forms". The RDF Cubes Semantic Enrichment, Annotation and Storage component is designed by taking account of the Semantic EHR Model, the outcome of Task 1.4 "Linked2Safety Semantic EHR Model".

The Interoperable EHR Data Space is designed using UML. The following subsections present the component design using activity diagrams, class diagrams and sequence diagrams. The final section integrates the components.

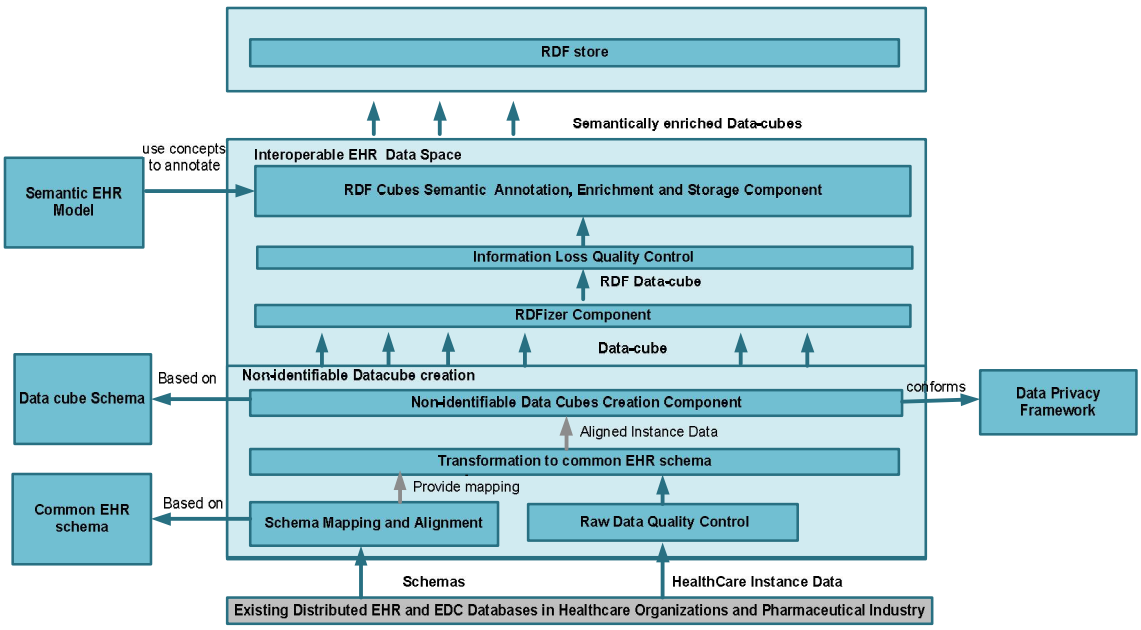

Figure 3: Interaction of Components composing the Interoperable EHR Data Space

4.2 Schema Mapping and Alignment Component

The Schema Mapping and Alignment component incorporates the common mapping and alignment engines required for the automatic transformation of EHRs and EDCs to the common EHR schema, the outcome of Task 1.3 "Linked2Safety Common EHR Schema Definition".

The Common EHR Schema is used as the baseline schema against which all medical data variables from the clinical partners are mapped. The mapping process is performed by expert users from the

clinical partners, which allows them to align their medical data variables to the Linked2Safety Common EHR Schema, thus enabling extensive processing of health data coming from heterogeneous sources, regardless of their origin.

This component outputs a mapping file that will be used by the transformation component to automatically transform data to the Common EHR Schema representation that can be stored in the Linked2Safety repository.

4.2.1 Activity Diagram

Figure 4 shows the activity diagram of the Schema Mapping and Alignment component. This component is mainly a graphical component, presenting a rich user interface that allows clinical data experts to align medical information to the Linked2Safety Common EHR Schema.

The component exposes three major functionalities to the users from a given date specified:

- Creating a new mapping;
- Saving a mapping to a file;
- Editing an existing mapping previously saved to a file.

The Schema Mapping and Alignment component presents a user interface displaying two attribute lists, one on the left and the other on the right of the screen. The first one lists the attributes from the clinical data providers' schema, and the second one lists attributes from the Common EHR Schema. The component allows the user to draw mapping lines from one clinical data attribute to another common schema attribute, thus establishing data relationships between clinical variables and common schema attributes. As the Common EHR Schema is fixed and does not change during the execution of the application, it could remain persistent on screen throughout.

The workflow starts with the application displaying a start page presenting user options. The user can choose to create a new mapping between a clinical data schema and a Common EHR Schema entity, or to edit an existing mapping, previously saved. At any time the user can save the work in progress, either as a draft that needs review or as a final mapping ready for usage.

All mappings are related to the clinical data schema that was used for its creation, and should be used only for transforming files that adhere to that given schema.

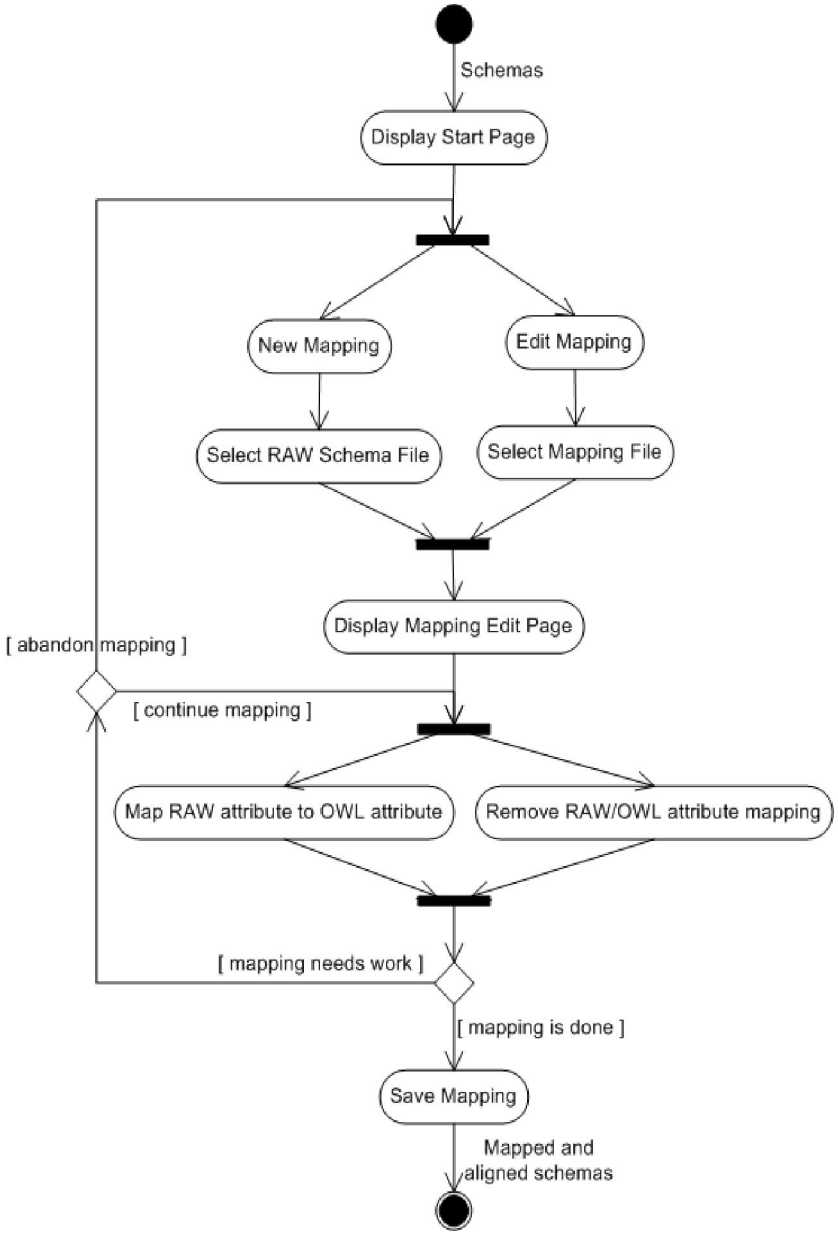

Figure 4: Schema Mapping and Alignment Component Activity Diagram

4.2.2 Class Diagrams

Figure 5 depicts the class diagram of the Schema Mapping and Alignment component. There are four main classes that will be used by this component:

- ***SchemaMappingAndAlignementApplication***, which represents the core class of the component, holding a reference to the mapping file that is being edited, and exposing methods for visually displaying schema attributes and mapping relationships, and for reading schema information and reading and writing mapping files;

- ***MappingFile***, which represents an actual mapping file, exposing its attributes, and allowing the application to read or write its contents;
- ***RAWSchemaFile***, which represents a clinical schema file, exposing its attributes, and allowing the application to read its contents;
- ***OWLSchemaFile***, which represents the Common EHR Schema file, exposing its attributes, and allowing the application to read its contents.

During the life time of the software component, there will be one instance of class SchemaMappingAndAlignmentApplication, and many instances of mapping files, together with their related RAW Schema and OWL Schema files. Note though that OWL Schema files relate to the same Common EHR Schema, against which all clinical (raw) data schemas should be mapped and aligned. Table 2,

Table 3, Table 4 and Table 5 describe the classes SchemaMappingAndAlignmentApplication, MappingFile, RAWSchemaFile and OWLSchemaFile respectively.

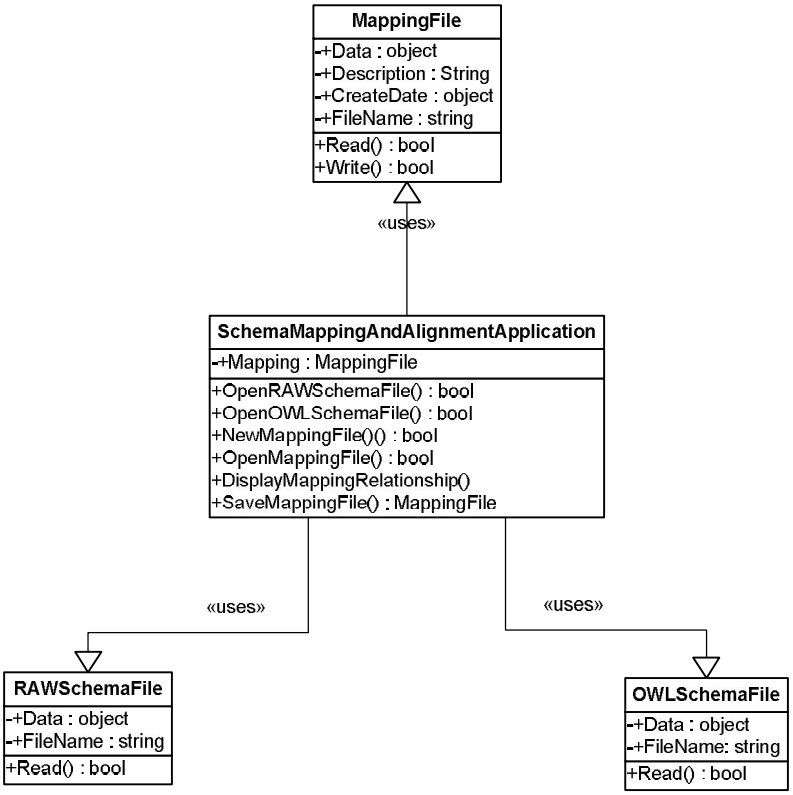

Figure 5: Schema Mapping and Alignment Component Class Diagram

Table 2: SchemaMappingAndAlignmentApplication Class

Class: SchemaMappingAndAlignementApplication	
Attribute	**Definition**
Mapping	A reference to the current mapping file.
Method	**Definition**
OpenRawSchemaFile()	Opens and reads the selected clinical data (RAW) schema file.
OpenOwlSchemaFile()	Opens and reads the Common EHR Schema (OWL) files.
NewMappingFile()	Creates a new mapping based on the selected clinical data schema file.
OpenMappingFile	Creates an existing mapping file, previously saved.
DisplayMapping Relationships()	Displays a graphic representation of the mapping relationships established by the expert user.
SaveMappingFile()	Saves the mapping file containing clinical data schema fields and related common schema attributes.

Table 3: MappingFile Class

Class: MappingFile	
Attribute	**Definition**
Data	The contents of the mapping file stream (XML).
Description	A description of the mapping, as stated by the expert user.
CreateDate	The date when the mapping file was created
FileName	The full name of the mapping file.
Method	**Definition**
Read()	Reads the contents (data) of the file from the given file name.
Write()	Writes the contents (data) of the file to the given file name.

Table 4: RAWSchemaFile Class

Class: RAWSchemaFile	
Attribute	**Definition**
Data	The contents of the schema file stream (XLS, XSD).
FileName	The full name of the schema file.
Method	**Definition**
Read()	Reads the contents (data) of the file from the given file name.

Table 5: OWLSchemaFile Class

Class: OWLSchemaFile	
Attribute	**Definition**
Data	The contents of the schema file stream (XLS, XML).
FileName	The full name of the schema file.
Method	**Definition**
Read()	Reads the contents (data) of the file from the given file name.

4.2.3 Sequence Diagram

Figure 6 depicts the sequence diagram of the Schema Mapping and Alignment component. The sequence starts when the user runs the application. The application will display a start page, allowing the user to choose whether the desired action is to create a new mapping or to edit an existing one. In either case, the application will display an “open file” dialog, for browsing and choosing the raw schema file or the existing mapping file. Following that, the application will display a visual representation of the mappings, as illustrated by the mock-up screen presented.

From that point on, the user actions enter a loop where he/she adds, edits and removes associations between clinical data (RAW) attributes and common schema (OWL) attributes, by drawing lines on the screen or removing lines.

When the user decides that the mapping is complete, he/she saves the mapping to a file that can later be used for transforming clinical data files to aligned data files. At the same time, the user can choose to save the mapping temporarily, so that he/she can later continue work on the mapping. The mock-up user interface of Schema Mapping and Alignment Component is illustrated in Figure 7.

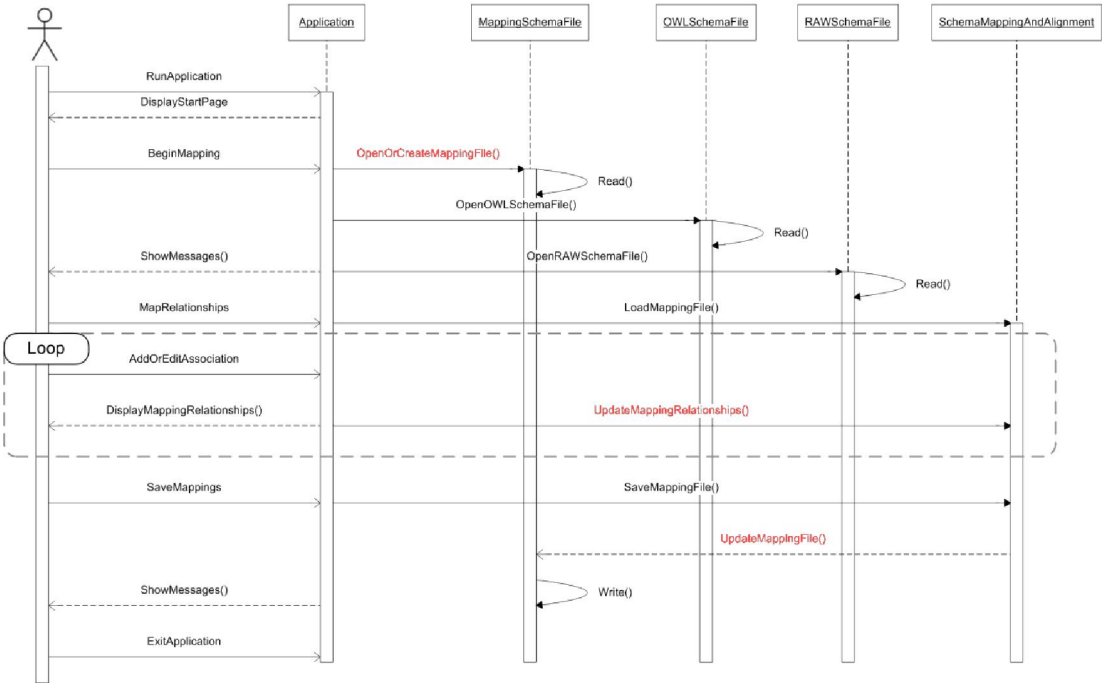

Figure 6: Schema Mapping and Alignment component Sequence Diagram

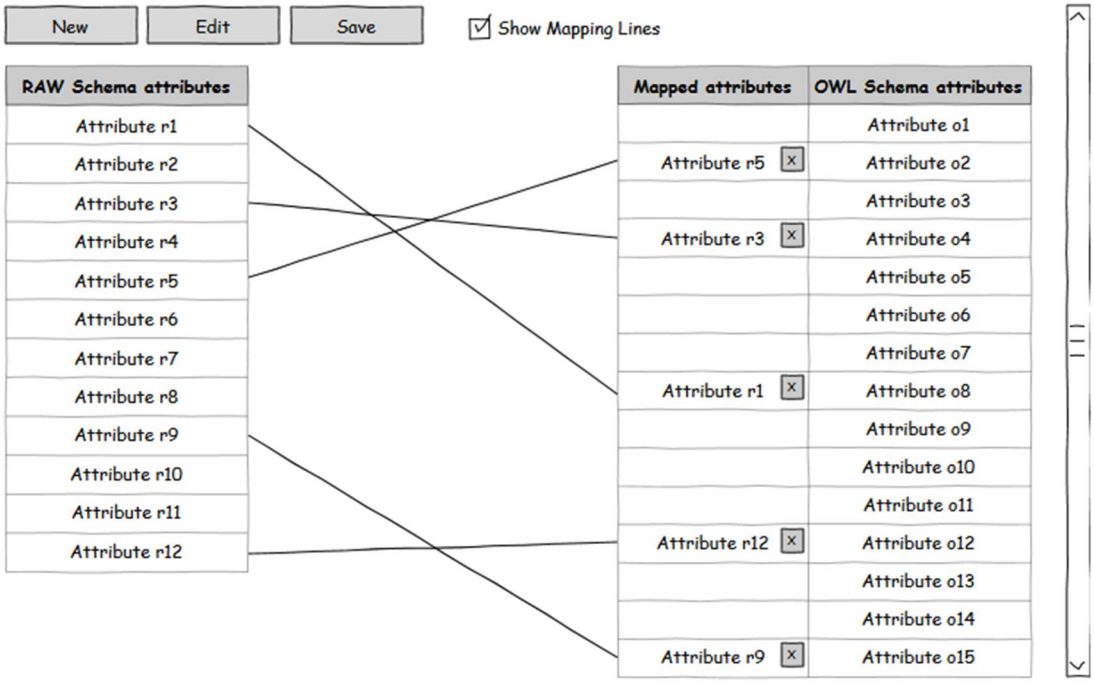

Figure 7: Schema Mapping and Alignment Component Mock-up Screen

4.3 Raw Data Quality Control Component

The Raw Data Quality Control component improves the quality of raw data by removing noise and handling missing values.

4.3.1 Activity Diagram

Figure 8 depicts the activity diagram of the Raw Data Quality Control component. Raw data are initially loaded and then four quality control tests are performed on them. Initially, the data subject gender test is performed so that any subjects with erroneous gender values are removed from the dataset. Then, the missing data test is performed on the dataset so that any subjects whose number of missing values exceeds the predefined threshold are removed and then any variables whose number of missing values exceeds the predefined threshold are also removed. The output of this test is then used in the Hardy-Weinberg Equilibrium test so that any Single Nucleotide Polymorphisms (SNPs) that do not conform to it are removed from the dataset. Finally, the output dataset from the previous test is used in the raw data Allele frequency test, where any SNPs with a minor Allele frequency below a predefined threshold are removed.

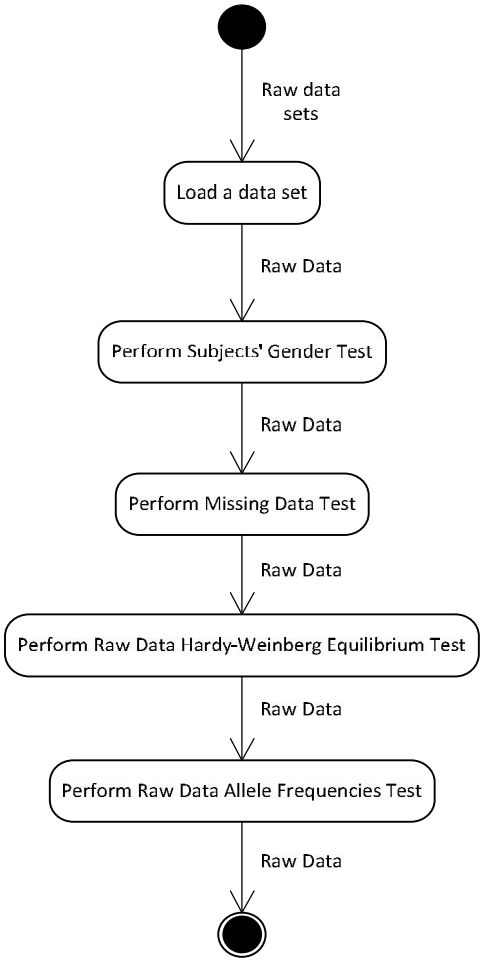

Figure 8: Activity diagram for the Raw Data Quality Control component

4.3.2 Class Diagram

The class diagrams for the the Raw Data Quality Control component are shown in Figure 9. There are two classes: Rawdata and RawdataQC. The Rawdata class contains the information regarding the raw data that will be used by the quality control component. It contains a method for loading the data using the filename and a method for storing the data using the specified filename. There are two variables: “varNames” (contains the names of the variables) and “values” (contains the values of each variable for each subject).

The RawdataQC class uses the Rawdata class to load the data via its getData method and store the data in its storeData method. The rest of the methods of this class perform the quality control tests on the data. The first three variables of the class have the thresholds that are used in the methods that perform the quality control tests, whereas the fourth variable “filename” is used for loading and storing the raw data.

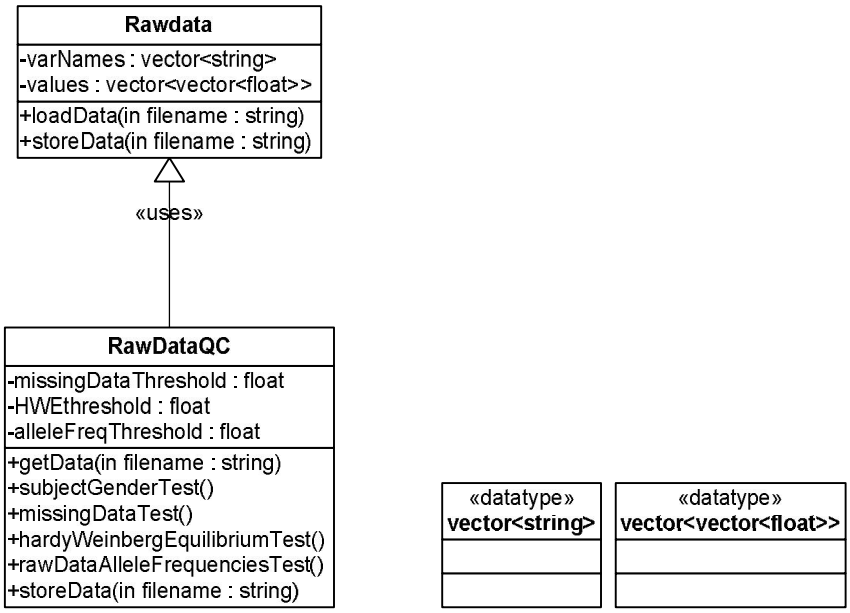

Figure 9: Class Diagrams for the Raw Data Quality Control component

4.3.3 Sequence Diagram

Figure 10 represents the sequence diagram of the Raw Data Quality Control component. Initially, the data are loaded using the method getData of the RawdataQC instance, which uses the method loadData of the Rawdata instance. Once the data is loaded, the methods that perform the quality control tests are executed in the order specified in the activity diagram. Once all these methods are executed, the final data are stored using the method storeData of the RawdataQC instance. The method then uses the storeData method of the Rawdata instance to store the data in the specified filename.

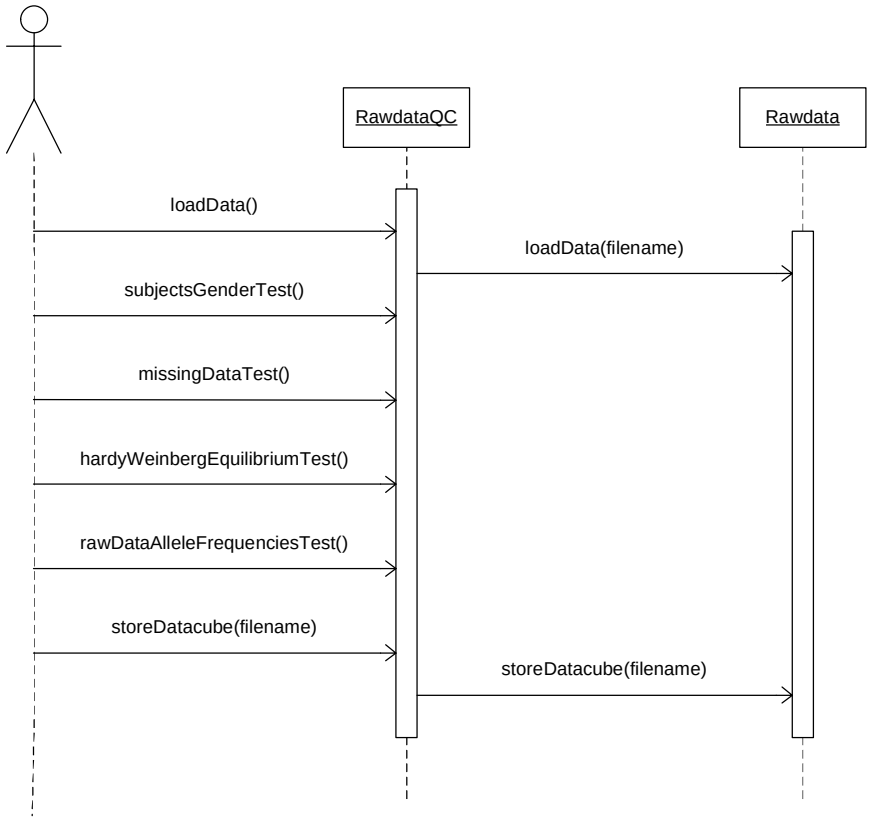

Figure 10: Sequence diagram for the Raw Data Quality Control component

4.4 Transformation to Common EHR Schema Component

The Transformation to Common EHR Schema component performs automatic transformation of EHRs and EDCs to the reference Common EHR Schema. Output instance data is created using the Common EHR Schema, based on the *openEHR* standard.

This component enables automatic transformation of medical information mapped and aligned to the Linked2Safety Common EHR Schema using the Schema Mapping and Alignment component, by employing custom developed software tools to ease the task of translating between metadata standards, thus bridging the gap between analysis and execution.

This component uses input data from the clinical partners in association with a given mapping file, and outputs data transformed using the given mappings to the Common EHR Schema representation that can be stored in the Linked2Safety repository.

4.4.1 Activity Diagrams

Figure 11 depicts the activity diagram of the Transformation to the Common EHR Schema component. The component starts by presenting a start page where the user can select a clinical

data (RAW) file and a mapping file created with the mapping component. The transformation component validates the files as a mapping is tightly related to a clinical data file by its originating structure. If the clinical data file can be mapped using the given mapping file, then the actual automated transformation process begins, where each record from the clinical file is transformed to the common aligned schema, while displaying the progress. At the end, the output file is saved at a given location.

Figure 12 illustrates an activity diagram of the automated mapping process, where each field of a given record of the clinical data (RAW) file is validated against the mapping definition file, identifying if the field is mapped to the common schema. Data is then recorded in the corresponding field in the aligned schema if correctly identified, record by record.

In the process of transforming clinical data files to aligned data files, many errors and exceptions may arise. It is critical for the transformation tool to record all such errors without blocking the transformation process, thus allowing for later inspection of the error log by an expert user, who can interpret the results and decide if the transformation was successful or not.

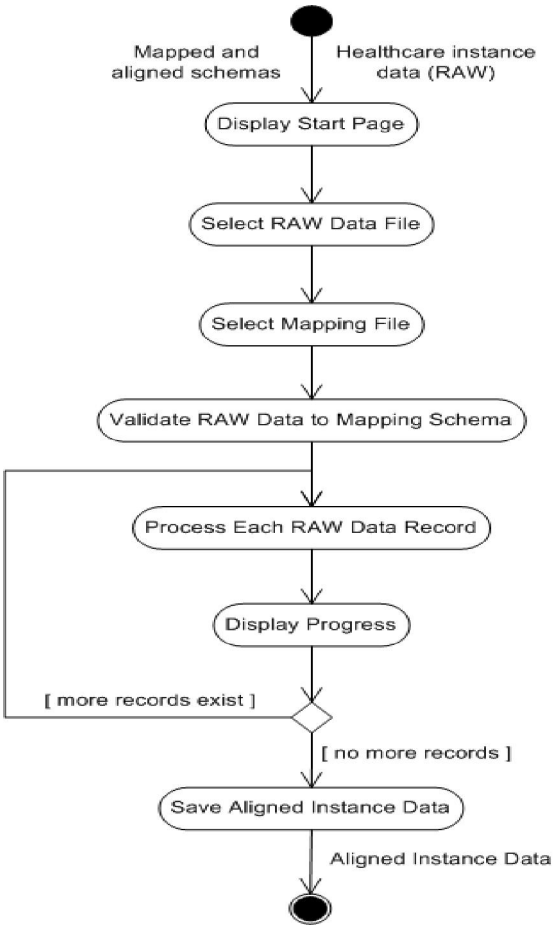

Figure 11: Transformation to Common EHR Schema component Activity Diagram

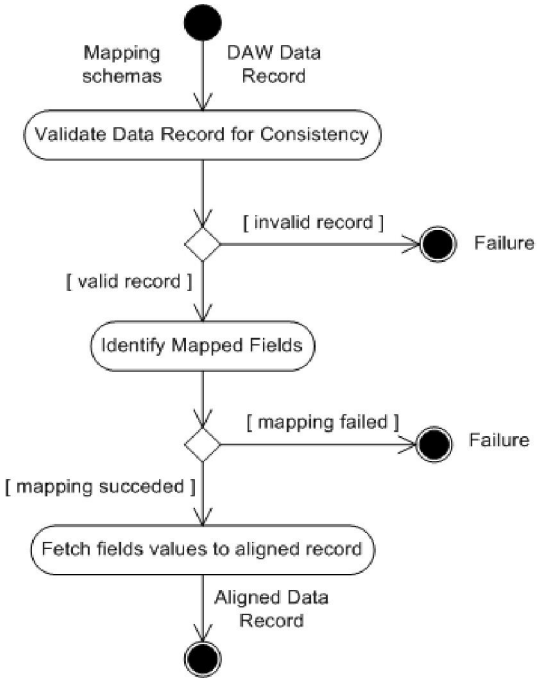

Figure 12: Process Each Raw Data Record Activity Diagram

4.4.2 Class Diagrams

Figure 13 depicts the class diagram of the Transformation to Common EHR Schema component.

There are four main classes that will be used by this component:

- ***TransformationToCommonSchemaApplication***, which represents the core class of the component, holding a reference to the clinical data file that is being processed, as well as the mapping file and the output aligned data file, and exposing methods for visually displaying process progress and for reading the raw data file and mapping information (Table 6);
- ***AlignedDataFile***, which represents an output aligned data file, exposing its attributes, and methods that allow the application to write its contents (Table 7);
- ***RAWDataFile*****,** which represents a clinical schema file, exposing its attributes, and methods that allow the application to read its contents (Table 8);
- ***MappingFile***, which represents the Common EHR Schema file, exposing its attributes, and methods that allow the application to read its contents (Table 9).

During the life time of the component, there will be one instance of class TransformationToCommonEHRSchemaApplication and possibly many instances of aligned data files, together with their related raw data and mapping files.

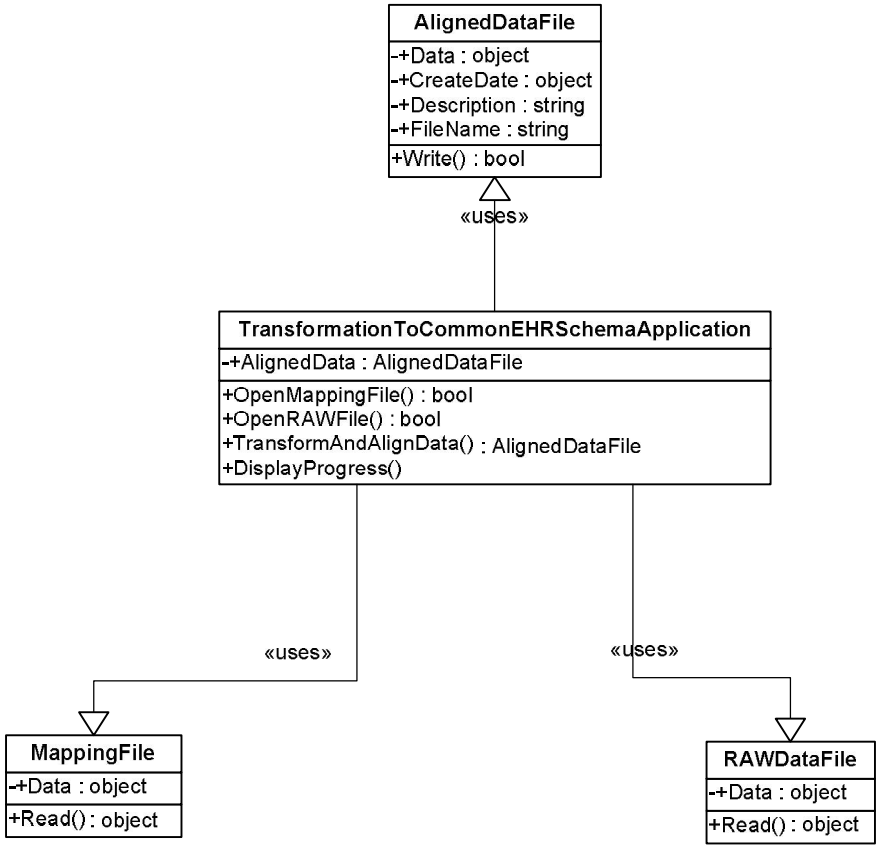

Figure 13: Transformation to Common EHR Schema component Class Diagram

Table 6: TransformationToCommonSchemaApplication Class

Class: *TransformationToCommonSchemaApplication*	
Attribute	**Definition**
AlignedData	A reference to the current aligned data file, while processing output results.
Method	**Definition**
OpenMapping File()	Opens and reads the contents (data) of the mapping file.
OpenRAWFile()	Opens and reads the contents (data) of the clinical instance data file.
TransformAnd AlignData()	Performs automated transformation and alignment of instance data to the common schema format of L2S.
DisplayProgress()	Displays progress of the transformation process on the user interface.

Table 7: AlignedDataFile Class

Class: *AlignedDataFile*	
Attribute	**Definition**
Data	The contents of the mapping file stream (RDF/XML).
Description	A description of the mapping, as stated by the expert user.
CreateDate	The date when the mapping file was created
FileName	The full name of the mapping file.
Method	**Definition**
Write()	Writes the contents (data) of the file to the given file name.

Table 8: RAWDataFile Class

Class: *RAWDataFile*	
Attribute	**Definition**
Data	The contents of the schema file stream (XLS, XSD).
Method	**Definition**
Read()	Reads the contents (data) of the file from the given file name.

Table 9: MappingFile Class

Class: *MappingFile*	
Attribute	**Definition**
Data	The contents of the schema file stream (XLS, XSD).
Method	**Definition**
Read()	Reads the contents (data) of the file from the given file name.

4.4.3 Sequence Diagrams

Figure 14 depicts the sequence diagram of the Transformation to Common EHR Schema component. The sequence starts when the user runs the application. The application will display a start page, allowing the user to choose a clinical data file and a related mapping file and to initiate the transformation process. The application displays the progress of the process while iterating through records from the source clinical data file. It matches each record to the common schema fields and adds data to the aligned output file stream.

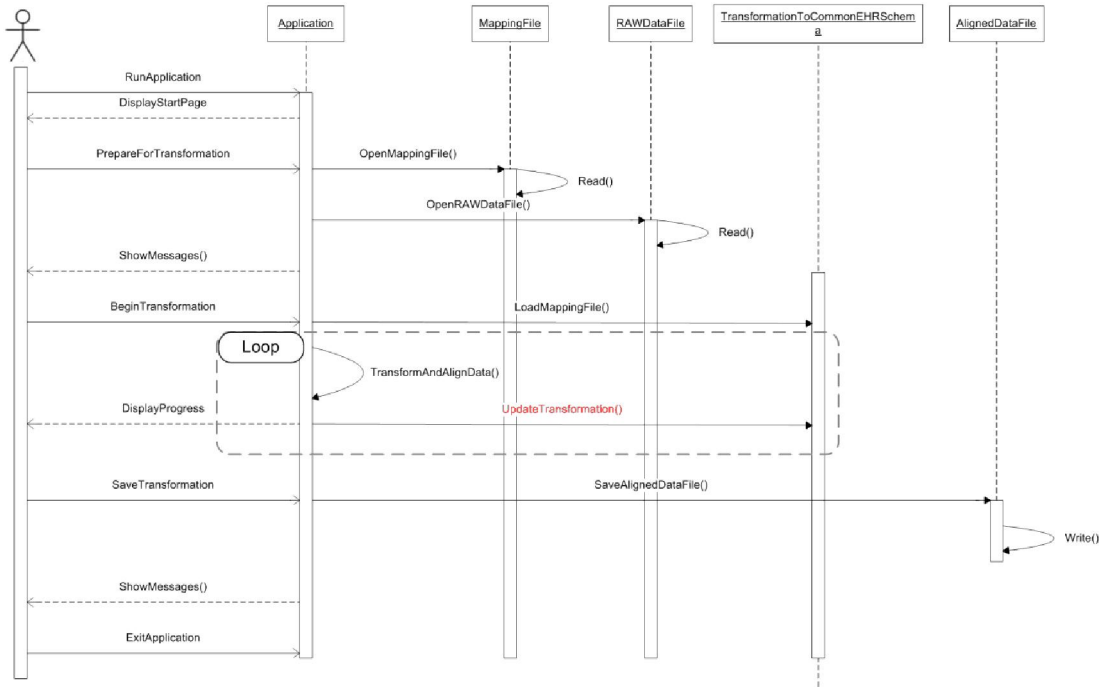

Figure 14: Transformation to Common EHR Schema Component Sequence Diagram

Figure 15 is a mock-up user interface of the Transformation to Common EHR Schema component.

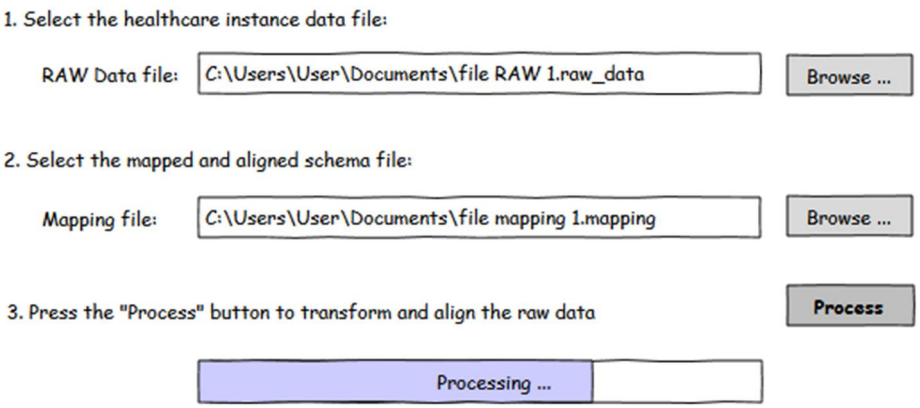

Figure 15: Transformation to Common EHR Schema Component Mock-up Screen

4.5 Non-identifiable Data Cubes Creation Component

This component creates non-identifiable data cubes from aligned instance data output by the Transformation to Common EHR Schema component.

4.5.1 Activity Diagram

Data cubes are created from records described by the Common EHR Schema. In general, a set of records is given as input to the component along with a set of attributes. The attributes are used to define the dimensions (or axis) of a data cube. If the values of an attribute are continuous, the user can define the ranges according to which the values of the attributes are to be discretized. The steps of the algorithm (Figure 16) in detail are the following:

- In the first step, the component gathers the set of records from files in the Common EHR Schema format. Data from these files are aligned in order to be efficiently parsed by subsequent steps of the algorithm;
- Secondly, a set of attributes are given as input from the user to the component. If an attribute has a continuous domain, the user must specify ranges of interest for each attribute. Ranges of interest can also be defined for categorical variables, in case the user wishes distinct values to be merged in the final data cube;
- For each attribute the component forms one-dimensional slices of the data cube. The data for each slice comes from aggregating the count of patients that have specific values for the corresponding values. For example, if *var1* has discrete values from the set {10, 15, 20}, a slice will have elements of the form:

  dim[10] = |p1|
  dim[15] = |p2|
  dim[20] = |p3|

  Here, |p1| is the number of patients that had the value *10* for the attribute discussed, and accordingly for |p2| and |p3|;
- The above process is repeated for each attribute. In the end, an $n_x m$ dimensional cube will be created (*n* being the number of attributes and *m* being the combination of each attribute's values);
- After the cube is created, a perturbation phase follows, during which random noise is introduced to each cell. The purpose of perturbation is to limit the possibility of reverse engineering the data cube in order to identify the patients from the aggregate data. To decide the maximum amount of noise that can be added based on the expected statistical noise introduced, a procedure as described in [1] can be followed;

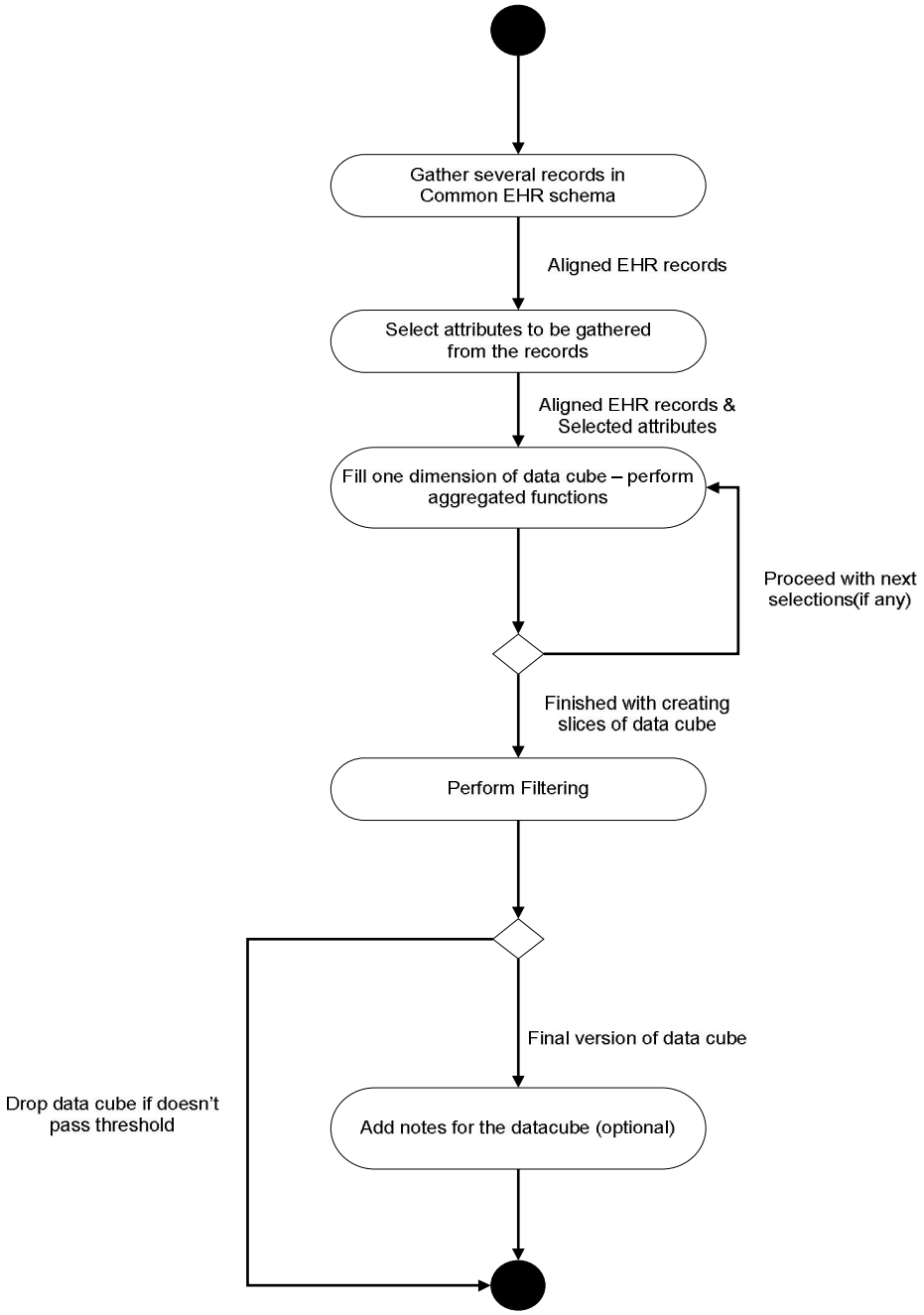

Figure 16: Activity Diagram for Non-identifiable Data Cubes Creation Component

- The next phase filters out areas of the cube that contain little or no information. As *m* can be quite large and as most combinations may have zero or a small count, the cube created in the above step can be sparse. As a result, large chunks of memory may be used to store useless information. Moreover, combinations of values that correspond to small counts may be used to reverse engineer and identify of patients from the set [1]. In order to prevent this, the algorithm performs cell suppression. With cell suppression, cells of the cube that contain values below a certain threshold, are filtered out from the final cube;

- After filtering, the total count of patients will be reduced. If too many cells with small values were filtered in the above step, the data cube may end up with data corresponding to a small count of patients compared with the initial set. In this stage the user will decide whether the count loss that has occurred is acceptable or not. If the count loss is unacceptable, the process finishes and a cube is not created. The user may start the process again by giving another set of attributes, threshold and variable ranges to create another data cube. This is the Information Loss Quality Control component described in Section 4.7.

- If the patient count loss is acceptable the process asks the user to optionally add notes to the resulting data cube. These notes regard the data cube structure as a whole.

4.5.2 Class Diagrams

The UML diagram depicted in Figure 17 shows the class composition of the Non-identifiable Data Cube Creation component. The core classes of the component are the "DataCubeCreator" and the "EHRParser" classes. The "EHRParser" is responsible for parsing data from Common EHR Record files. It is used by the "DataCubeCreator" class which uses the data retrieved by "EHRParser" to create the cubes.

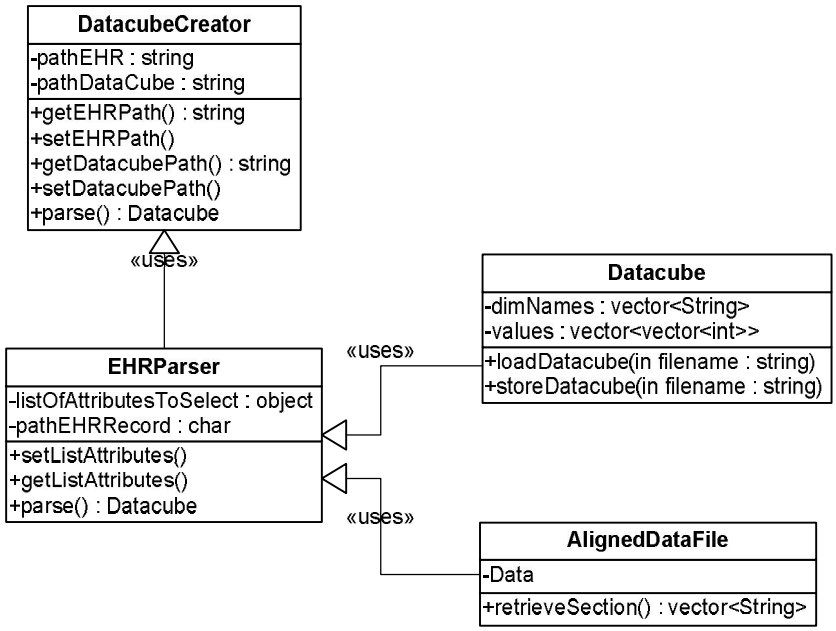

Figure 17: Class Diagram for Non-identifiable Data Cubes Creation Component

A more detailed description of the classes and their methods is given in tables 10 and 11.

Table 10: DataCubeCreator Class

Class: *DatacubeCreator*	
Attribute	**Definition**
pathEHR	The path of the common EHR file
pathDataCube	The path of the created Data Cube
DataCube	Multidimensional array containing DataCube values
Method	**Definition**
getEHRPath()	Getter Method for pathEHR
setEHRPath()	Setter Method for pathEHR
getDataCubePath()	Getter Method for pathDataCube
setDataCubePath ()	Setter Method for pathDataCube
parse()	Parses the data retrieved from the EHR according to the steps defined in 4.5.1. The final Data Cube, if created, is stored in the directory pointed to by *pathDataCube*

Table 11: EHRParser Class

Class: *EHRParser*	
Attribute	**Definition**
listOfAttributesToSelect	Array that defines the set of attributes that the user selects
pathEHRRecord	The path of the EHR file
attributeStepMap	Associative array that stores the discretization step for each attribute.
Method	**Definition**
setListAttributes()	Setter Method for listOfAttributesToSelect
getListAttributes()	Getter Method for listOfAttributesToSelect
parse()	Parses the EHR file and stores the data in intermediate data structures. These structures are needed for the DatacubeCreator objects.

4.5.3 Sequence Diagram

The sequence diagram of the Non-identifiable Data Cube Creation component is depicted in Figure 18. The common EHR file that is produced by the Transformation to Common EHR Schema component is given as input to the Data Cube Creation component. The Data Cube Creation component constructs the cubes, which are then passed in CSV format to the RDFizer component.

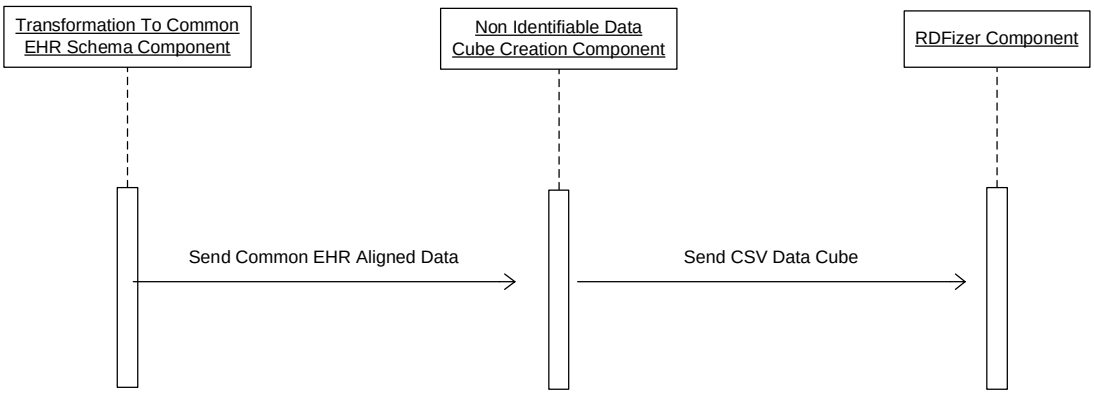

Figure 18: Sequence Diagram for Non-identifiable Data Cubes Creation component

4.6 RDFizer Component

The RDFizer component is responsible for converting the aggregated data in the form of a data cube into an RDF version of the same data adhering to the RDF Data Cube Vocabulary[1], as described in Deliverable D1.3 “Linked2Safety Common EHR Schema”.

The process of the RDFization of a data cube is straightforward. The core element of the RDF Data Cube Vocabulary is the dataset. This class describes the structure of a data cube as a collection of “observations”. The latter class holds the description of an occurrence number, the dimensions coordination for that number along with the semantics (based on Concepts) of those dimensions.

The RDF Data Cube Vocabulary introduces a reusable set of concepts and components based on SDMX (Statistical Data and Metadata Exchange). The SDMX standard includes a set of *content oriented guidelines* (COG) which define a set of common statistical concepts and associated code lists that are intended to be reusable across data sets. In Deliverable D1.3 “Linked2Safety Common EHR Schema”, as part of the data cube work, we have created RDF analogues to the COG. These include:

- *sdmx-concept*: SKOS Concepts for each COG defined concept;
- *sdmx-code*: SKOS Concepts and Concept Schemes for each COG defined code list;

[1] http://www.w3.org/TR/vocab-data-cube/

- *sdmx-dimension*: component properties corresponding to each COG concept that can be used as a dimension;
- *sdmx-attribute*: component properties corresponding to each COG concept that can be used as a attribute;
- *sdmx-measure*: component properties.

4.6.1 Activity Diagrams

The first step of the RDFizer component is to receive the data cube in a readable format (Figure 19). We plan to allow the description of the data cube to be either in plain CSV with a header or an XML file.

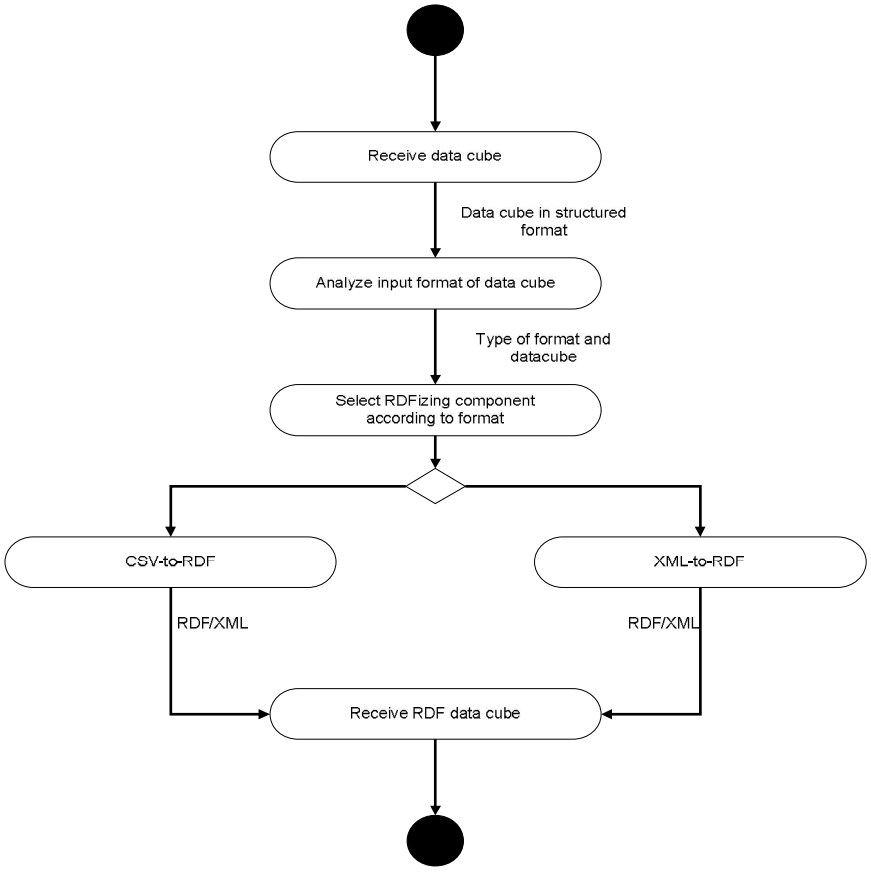

Figure 19: Activity Diagram for the RDFizer component

After recognizing the format of the supplied input file, we convert it to RDF/XML or RDF/Turtle format, as can be seen in the example in Table 12.

Table 12: Example of CSV data cube dimension (partial) data converted to RDF/Turtle

```
Diabetes,Weight,number
1,0,50
1,1,30
2,0,100
2,1,90
3,0,10
3,1,230
[ ... ]
```

```
<http://linked2safety-project.eu/dataset/data-cube/diabetes/0> qb:dataSet
<http://linked2safety-project.eu/dataset/data-cube/diabetes/>;

    5.          sdmx-dimension:Diabetes   <http://linked2safety-project.eu/dataset/data-
        cube/diabetes/1>;

    6.          sdmx-dimension:Weight     <http://linked2safety-project.eu/dataset/data-
        cube/diabetes/0>;
        sdmx-measure:Cases "50"^^xsd:long;
        a qb:Observation.
<http://linked2safety-project.eu/dataset/data-cube/diabetes/1> qb:dataSet
<http://linked2safety-project.eu/dataset/data-cube/diabetes/>;

    7.          sdmx-dimension:Diabetes   <http://linked2safety-project.eu/dataset/data-
        cube/diabetes/1>;

    8.          sdmx-dimension:Weight     <http://linked2safety-project.eu/dataset/data-
        cube/diabetes/1>;
        sdmx-measure:Cases "30"^^xsd:long;
        a qb:Observation.
<http://linked2safety-project.eu/dataset/data-cube/diabetes/2> qb:dataSet
<http://linked2safety-project.eu/dataset/data-cube/diabetes/>;

    9.          sdmx-dimension:Diabetes   <http://linked2safety-project.eu/dataset/data-
        cube/diabetes/2>;

    10.         sdmx-dimension:Weight     <http://linked2safety-project.eu/dataset/data-
        cube/diabetes/0>;
        sdmx-measure:Cases "100"^^xsd:long;
        a qb:Observation.
[ .... ]
```

4.6.2 Class Diagrams

The most important aspect is the core element that shows how we go from a raw description of a data cube, to a version that can be more easily semantically extended. Figure 20 illustrates the class diagram of RDFizer component.

Tables 13-15 describe the classes which compose the RDFizer component. The classes XMLParser and CSVParser implement the Parser Interface, and have the same structure.

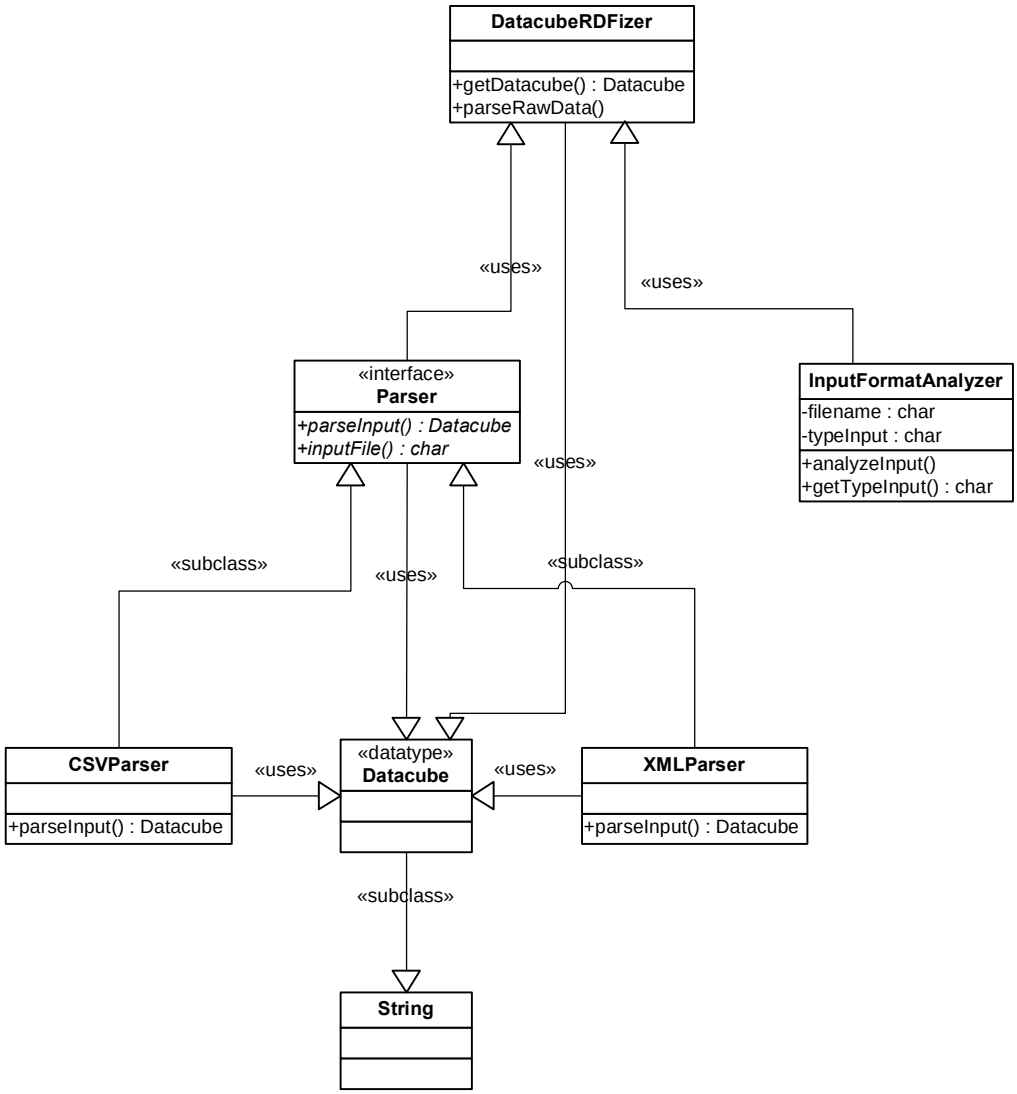

Figure 20: Class Diagram for the RDFizer component

Table 13: DatacubeRDFizer Class

Class*: DatacubeRDFizer*	
Attribute	**Definition**
getDatacube()	Method for retrieving the data cube triples produced after the parsing of the raw data
parseRawData()	Method for parsing the raw data that have been provided for input

Table 14: XMLParser Class

Class: *XMLParser*	
Attribute	**Definition**
filename	The name of the file to be provided as input to the component
Method	**Definition**
parseInput()	Method for parsing the raw data that have been provided for input
inputFile()	Method for setting the path of the input file(s) to be converted

Table 15: Parser Interface

Interface: *Parser*	
Attribute	**Definition**
filename	The name of the file to be provided as input to the component
typeInput	The type of the input file (“CSV” or “XML”) after file “filename” was analysed
Method	**Definition**
analyzeInput()	Method for analyzed the format of the input file “filename”
getInputType()	Method for retrieving the input type

4.6.3 Sequence Diagrams

Figure 21 presents the sequence of actions taken to RDFize a data cube.

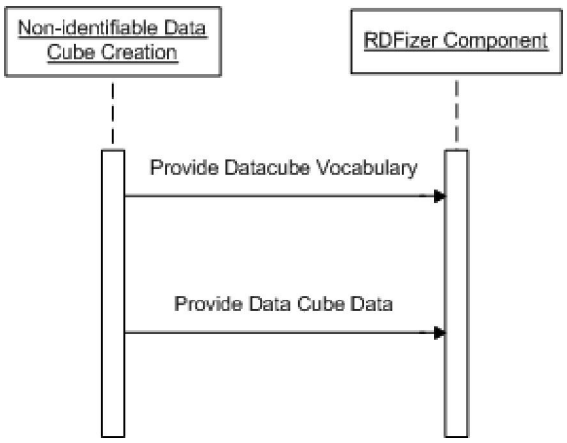

Figure 21: Sequence Diagram for the RDFizer component

4.7 Information Loss Quality Control Component

This component decides whether the data cube is to be later used for data analysis based on its quality.

4.7.1 Activity Diagram

As seen in Figure 22, initially the data cube that needs to be processed is loaded and then the information loss test is performed on the specified data cube. The information loss test calculates the information that is lost in the specified data cube and that value is compared with the predefined threshold for the information loss value. If the information loss value of a data cube is above the accepted threshold then the data cube is discarded, otherwise the data cube is stored.

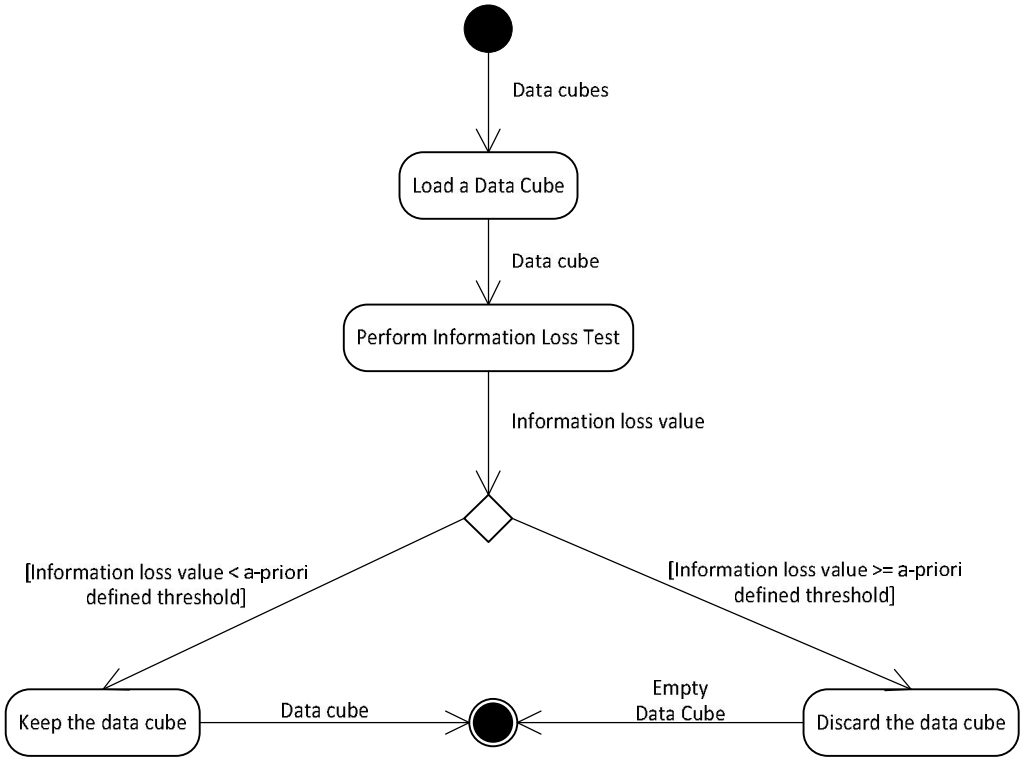

Figure 22: Activity diagram for the data cube Information Loss Quality Control component

4.7.2 Class Diagrams

The class diagram of the Information Loss Quality Control component is presented in Figure 23. The class Datacube contains the information of a data cube. Specifically, the variable "dimNames" contains the names of each data cube dimension. The variable "values" contains different combinations of the values of the dimensions and the total number of subjects that have the

respective combination. The method loadDatacube is used to load a data cube using its filename and the storeDatacube method is used to store the loaded data cube.

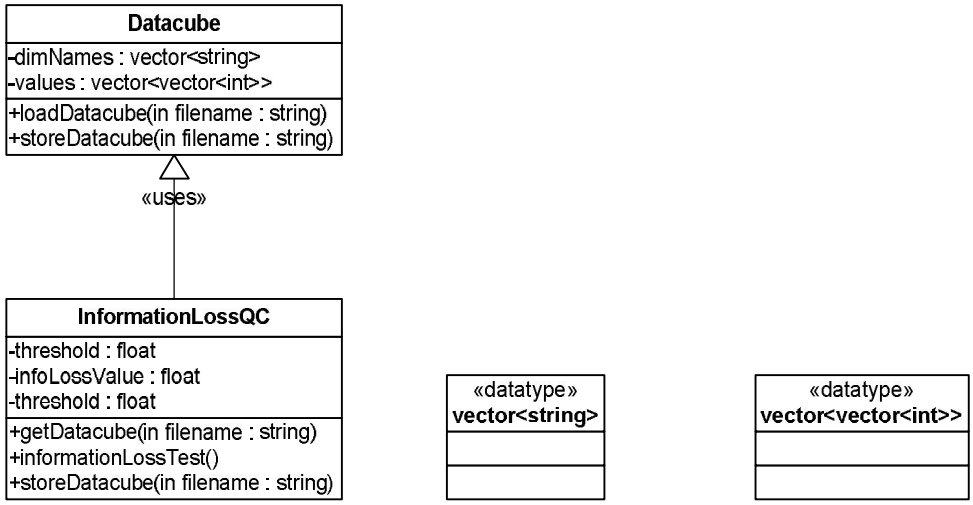

Figure 23: Class diagram for the Information Loss Quality Control component

The InformationLossQC class uses the Datacube class to load a data cube and to store it. The variable threshold has the predefined acceptance threshold of the information loss of a data cube, whereas the variable infoLossValue has the calculated information loss of a data cube. The infoLossValue is calculated in the method informationLossTest. The storeDatacube method stores a data cube if the information loss value of a data cube is below the predefined acceptance threshold.

4.7.3 Sequence Diagram

In Figure 24, the sequence diagram of the component is illustrated. There is an instance of the InformationLossQC class and an instance of the Datacube class. Initially, the user calls the getDatacube method of the InformationLossQC instance, which uses the loadDatacube method of the Datacube instance. Once the data cube is loaded, the informationLossTest method is executed. Finally, the storeDatacube method of the InformationLossQC instance is called. If the information loss of the data cube is below the predefined threshold, the method calls the storeDatacube method of the Datacube instance and the data cube is stored.

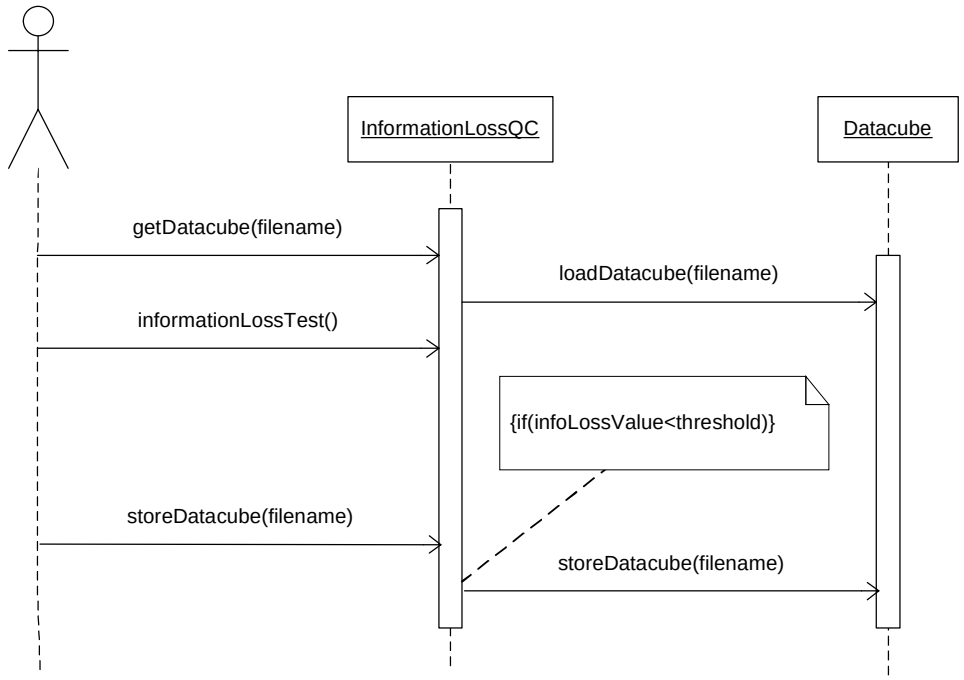

Figure 24: Class diagram for Information Loss Quality Control component

4.8 RDF Cubes Semantic Annotation, Enrichment and Storage Component

The RDF Cubes Semantic Enrichment, Annotation and Storage component implements the semantic linking and the enrichment of annotations with links to relevant medical and clinical resources on the Linked Data Web.

On the one hand, annotations provide additional explanations which may help with interpretation and understanding of data items. On the other hand, annotations act as sources of additional metadata which can be exploited to improve search and retrieval services, in particular for non-expert users who may be unfamiliar with domain-specific terminology. Lastly, the annotations improve the reusability aspect and strengthen the applicability of a model enabling it to interoperate with other existing vocabularies.

The RDF Cubes Semantic Enrichment, Annotation and Storage component enables the semantic annotation and enrichment of the referenced (RDFized) data cubes with the use of the Linked2Safety Semantic EHR Model as well as other globally available clinical and medical ontologies and vocabularies. The produced annotations are incorporated in the "Linked2Safety Semantic EHR Model" by enriching its existing entities (e.g. with the use of the owl:sameAs property).

In particular, the component enables the user to align an entity or instance directly with terms/entities or instances of the same or other medical ontologies, by doing an exact string comparison (direct matching) between the entity and term names, synonyms, and ids. The main input of this component is a set of ontologies (e.g. Linked2Safety Semantic EHR Model and other existing medical ontologies) while the output is an enriched RDF data cube.

4.8.1 Activity Diagram

The first step of this task is to semantically enrich the RDF data cube. The user (who is going to be part of a clinical partner's personnel) will load the data cube. Next, s/he will select the data cube component (slice, observation, dimension, measure, attribute etc.) that is of interest. At this point the user can perform one of the following selections:

- **Add a simple text label**: This is of great interest for a clinical partner as this string can lead to a better understanding of the data (by other scientists);
- **Semantic annotation using a semantic model**: this allows the data cube (or parts of the cube) to be linked with a class (i.e. concept or property) in order to include semantic knowledge. The semantic linking can be done with concepts/properties derived from either the Linked2Safety Semantic EHR Model or other existing semantic models and vocabularies in a systematic way, e.g. the user can choose to annotate a whole cube containing aggregated information of several clinical trials related to AIDS infection with the class entity http://hcls.deri.ie/l2s/sehr/chuv/1.0#AIDS (from SEHR). Indeed, for the annotation we need only the URI of the annotating concept/property;
- **Semantic annotation using Linked Data instances:** Here, the annotation is performed at the instance level. The user can choose to annotate a data cube's component with a Linked Data URI of the RDF description of another piece of knowledge, i.e. the user might want to manually specify that the cube be created (with URI_a) is the same as another data cube (with URI_b);

The process of annotating components of a data cube can be repeated many times. After the user has finished, the data cube is stored in the RDF storage (see Figure 25).

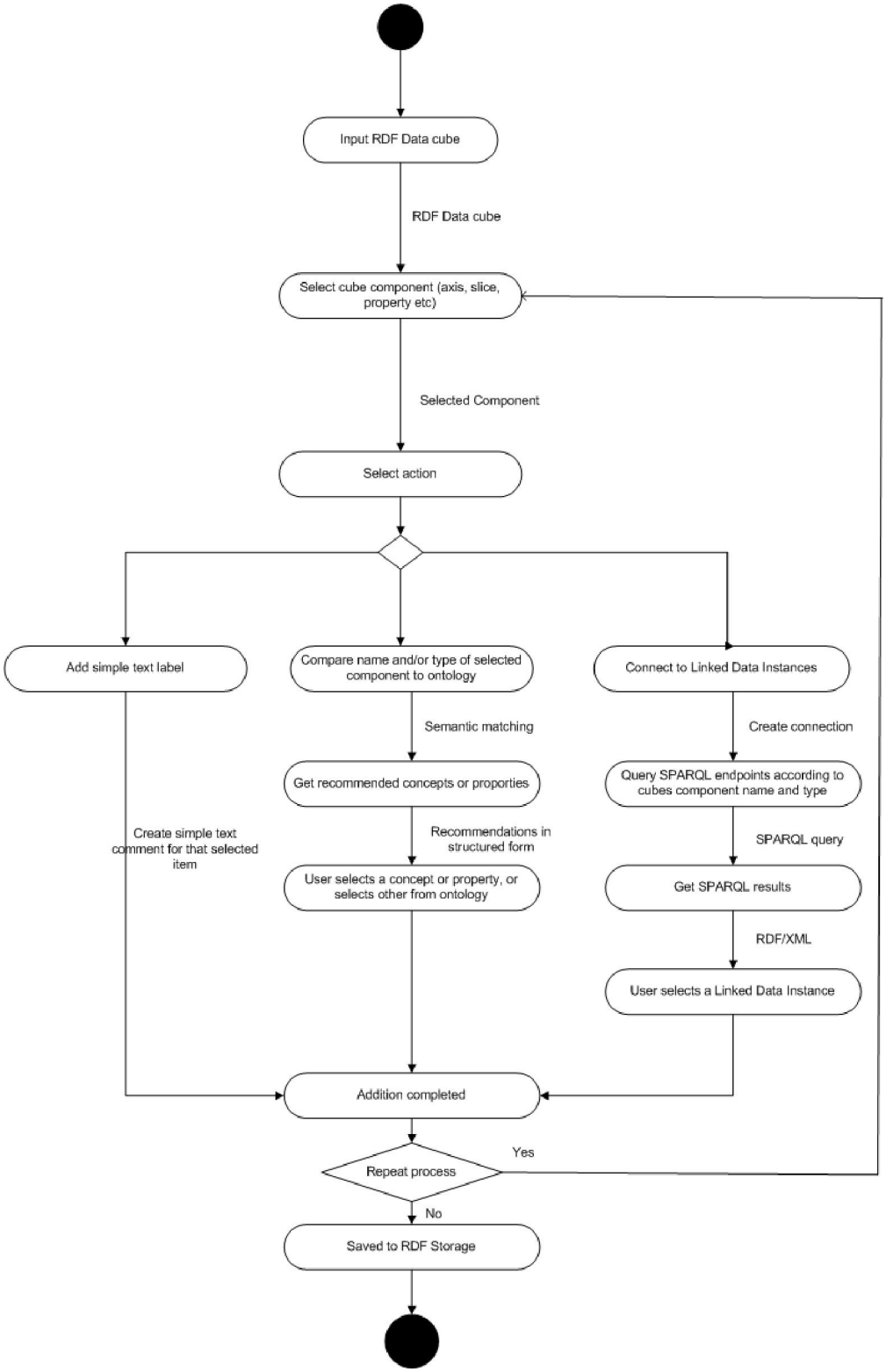

Figure 25: Activity diagram for RDF Cubes Semantic Annotation, Enrichment and Storage component

4.8.2 Class Diagrams

In the following we analyze the annotation process using a class diagram (see Figure 26). Specifically, the annotation scheme is applied to RDF data cubes including the data cube's dimension, hierarchy, slice, etc. The annotation includes the addition of a label, the connection to a concept/property (concept level) of the Linked2Safety database or a different database (ontology) and the connection to an instance (Linked Data instance level) of the Linked2Safety database or another database.

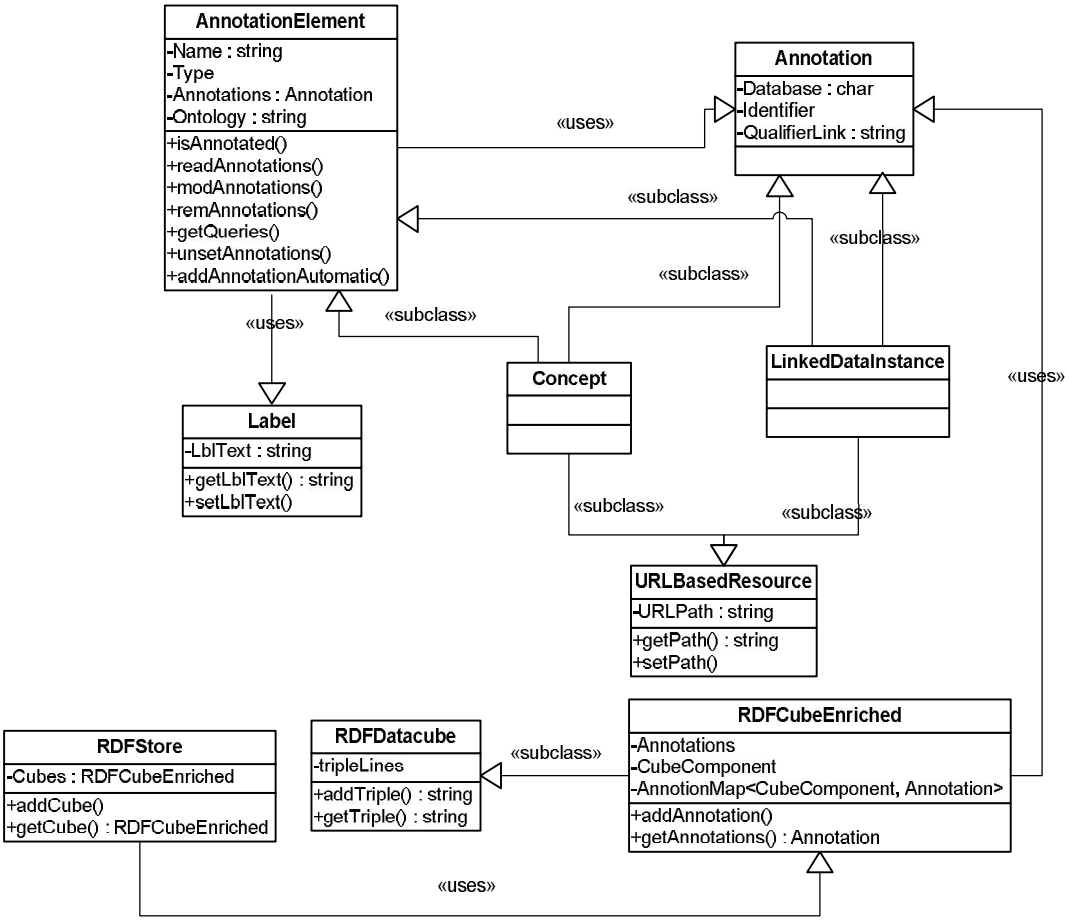

Figure 26: Class Diagram for RDF Cubes Semantic Annotation, Enrichment and Storage Component

We distinguish two core classes: the AnnotationElement (Table 16) and the Annotation (Table 17). The Annotation class represents a single annotation, while the AnnotationsElement class represents the annotatable object in the Linked2Safety model which is the container for the Annotation class instances.

An Annotation Element can be a Linked Data instance or a concept (concept/property) of the semantic model used for the description of the data cube of interest. An Annotation is added to an Annotation Element linking it with another concept or Linked Data instance respectively.

Table 16: AnnotationElement Class

Class*: AnnotationElement*	
Attribute	**Definition**
Name	The name of the (annotated) element
Type	The type of the element
Annotations	Existing annotations of the element
Ontology	The name of the ontology based on which the element has been semantically described
Method	**Definition**
isAnnotated()	Method for evaluating where an data cube element has been annotated
readAnnotations()	Method for returning the annotations found in a data cube component
modAnnotations()	Method for modifying an annotation found in a data cube component
remAnnotations()	Method for removing an annotation found in a data cube component and resynchronized the internal list of annotations.
getQueries()	Method for returning a list of query strings that can be used to query annotation
unsetAnnotations()	Method for deleting all annotations found in a data cube component
addAnnotationAutomatically()	Method for suggesting a collection of annotations for a particular data cube component

Table 17: Annotation Class

Class*: Annotation*	
Attribute	**Definition**
Database	Database describes the database of the annotation
Identifier	Identifier describes the entity of the annotation
QualifierLink	Qualifier variable describes the relation of the annotation with the entities

Another type of enrichment is the addition of a label to an Annotation Element using the Label class (Table 18).

Table 18: Label Class

Class: *Label*	
Attribute	**Definition**
LblText	The label's text
Method	**Definition**
getLblText()	Method for getting the text of a label
setLblText()	Method for setting the text of a label

Both Linked Data instances and concepts are denoted by a URI. Table 19 describes class URIBasedResource.

Table 19: URIBasedResource Class

Class: *URIBasedResource*	
Attribute	**Definition**
URLPath	The URL path of the specific resource
Method	**Definition**
getPath()	Method for getting the URL path of a resource
setPath()	Method for setting the URL path

An enriched RDF Cube contains the data cube components, the annotations and the mapping between the data cube components and the annotations. Table 20 describes RDFCubeEnriched.

As mentioned earlier, the enriched RDF Cube (instance of class RDFCubeEnriched) constitutes part of an RDF data cube and thus it is a subclass of RDF Data Cube class. Table 21 describes the RDFDatacube class.

Finally, the enriched RDF Cube is stored in the RDF store. Table 22 describes the RDFStore class.

Table 20: RDFCubeEnriched Class

Class*: RDFCubeEnriched*	
Attribute	**Definition**
Annotations	The annotations of a data cube
CubeComponent	The components of the data cube that are used as annotation elements
AnnotationMap	A mapping contains a linking between a CubeComponent and an annotation
Method	**Definition**
addAnnotation()	Method for adding a specific annotation to a data cube component
GetAnnotation()	Method for returning the annotations found in a data cube

Table 21: RDFDatacube Class

Class*: RDFDatacube*	
Attribute	**Definition**
tripleLines	The triple of an RDF data cube
Method	**Definition**
addTriple()	Method for adding a triple that describes a dimension in a cube
getTriple()	Method for returning the triple that describes a dimension in a cube

Table 22: RDFStore Class

Class*: RDFStore*	
Attribute	**Definition**
Cubes	The data cube that will be stored in the RDF store
Method	**Definition**
addCube()	Method for adding an enriched RDF cube in the RDF store
getCube()	Method for returning the enriched RDF cubes stored in the RDF store

4.8.3 Sequence Diagrams

This section presents the sequence of actions taken to enrich a data cube via annotations, before storing it in an RDF store (see Figure 27).

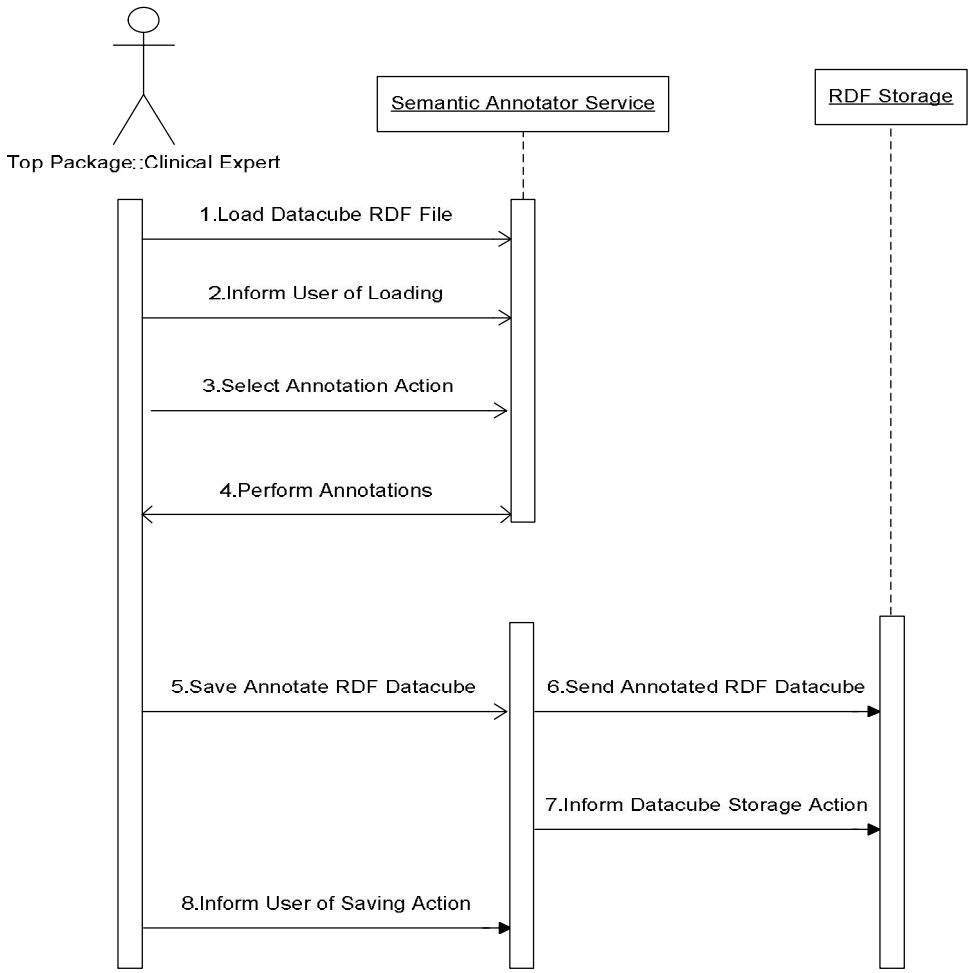

Figure 27: Sequence Diagram for RDF Cubes Semantic Annotation, Enrichment and Storage component

4.9 Integration of Components

After all the components above have been designed, the components are integrated together to compose the Interoperable EHR Data Space.

4.9.1 Activity Diagram

The activity diagram is shown in Figure 28. Each component performs a unique operation and is a top-level activity of the Interoperable EHR Data Space.

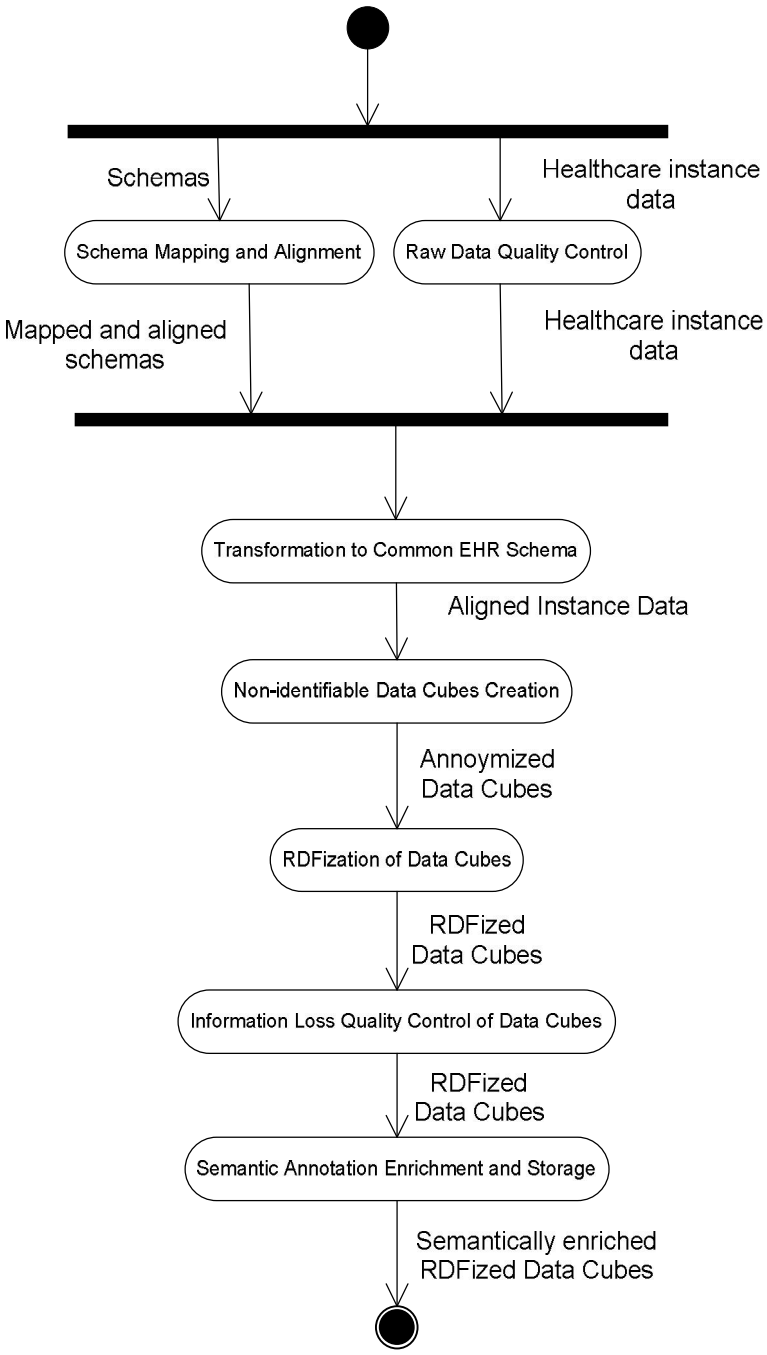

Figure 28: Top-level Activity Diagram of the Interoperable EHR Data Space

4.9.2 Class Diagram

The InteroperableEHRDataSpace class (see Figure 29) is the main program of the Interoperable EHR Data Space. The main method of InteroperableEHRDataSpace takes schemas and healthcare instance data as inputs from existing distributed EHR and EDC databases and creates instances of the seven components: SchemaMappingAndAlignmentApplication, TransformationToCommon-EHRSchemaApplication, DatacubeCreator, DatacubeRDFizer, InformationLossQC, RawDataQC and RDFCubeEnriched.

The main method calls the methods of the objects in order to output semantically enriched RDFized data cubes.

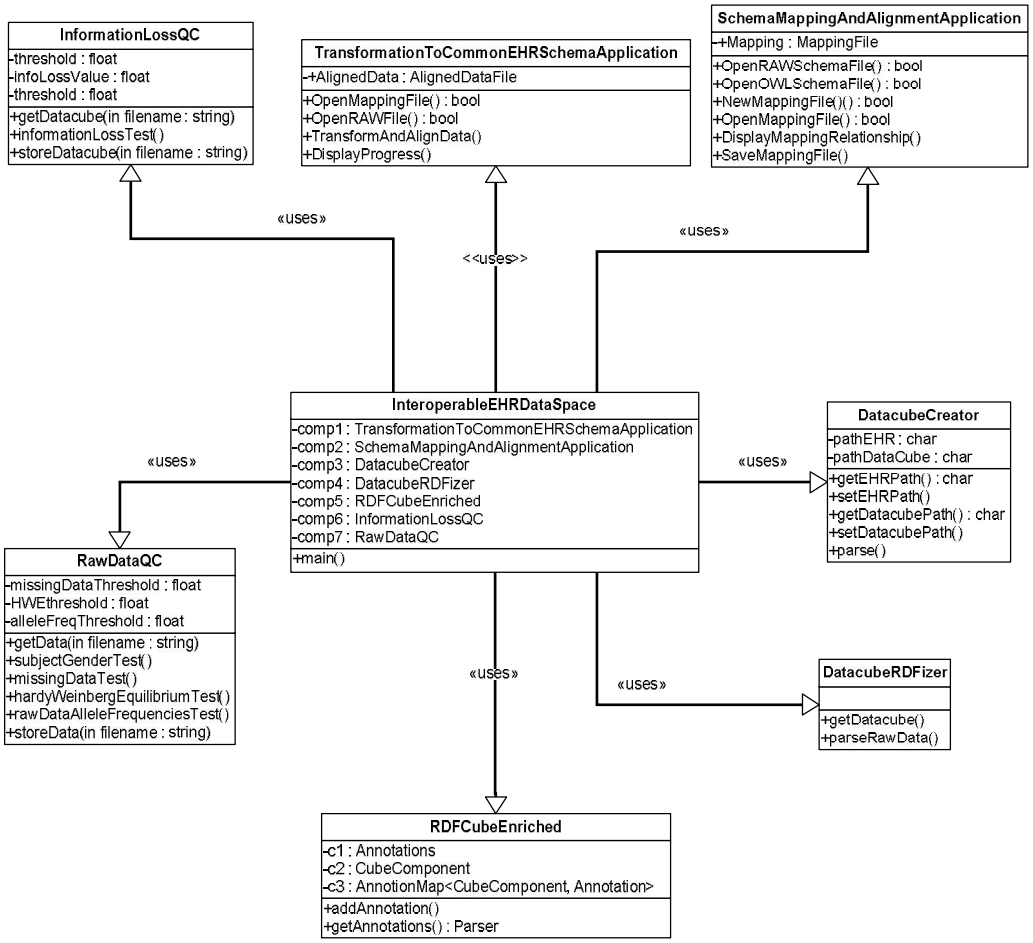

Figure 29: Top-level class diagram of the Interoperable EHR Data Space

4.9.3 Sequence Diagram

The top-level sequence diagram of the Interoperable EHR Data Space is illustrated in Figure 30. The InteroperableEHRDataSpace object sends messages to the following objects: SchemaMappingAndAlignment, RawDataQC, TransformationToCommonEHRSchema, DataCubeCreator, DataCubeRDFizer, InformationLossQC and RDFCubeEnriched; the objects return

outputs to the InteroperableEHRDataSpace. The outputs of the InteroperableEHRDataSpace object are semantically enriched RDFized data cubes.

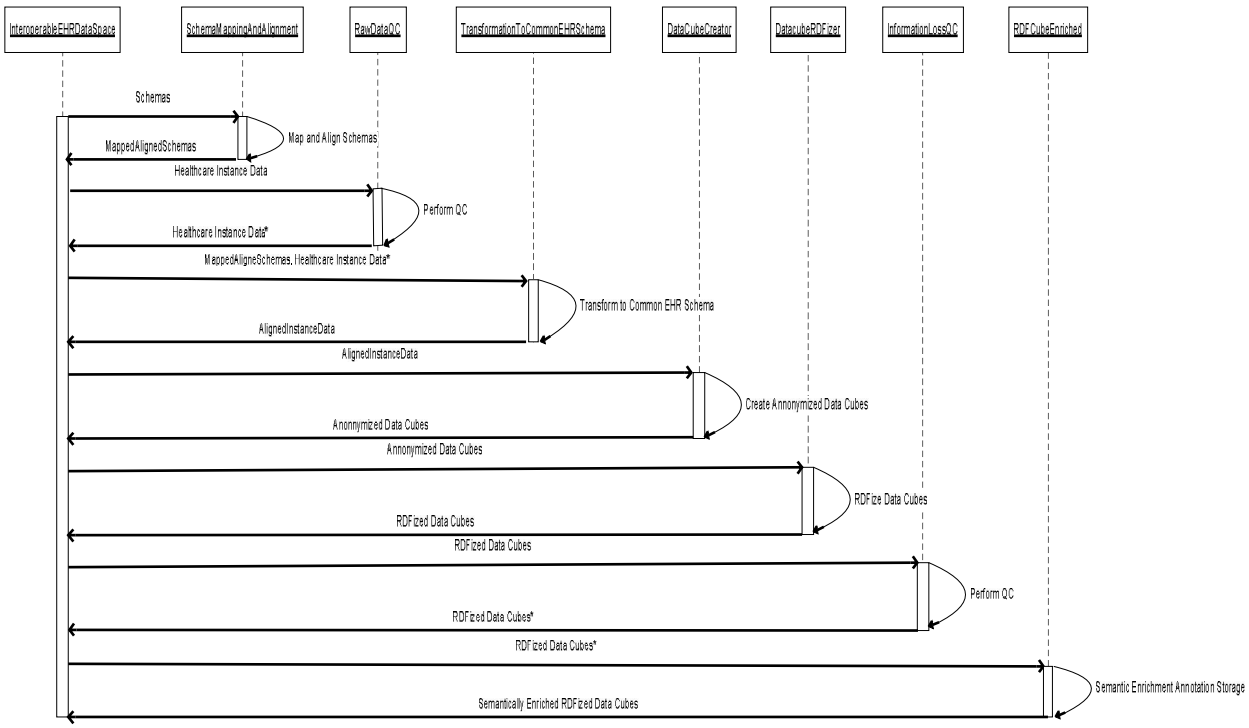

Figure 30: Top-level Sequence diagram of the Interoperable EHR Data Space

5. Conclusion

The Interoperable EHR Data Space has been designed to consist of the following software components:

1. Schema Mapping and Alignment;
2. Raw Data Quality Control;
3. Transformation to common EHR Schema;
4. Non-identifiable Data Cubes Creation component;
5. RDFizer component;
6. Information Loss Quality Control;
7. RDF Cubes Semantic Annotation, Enrichment and Storage component.

The inputs to the Interoperable EHR Data Space are the schemas and the healthcare instance data of some distributed heterogeneous EHR and EDC databases. The outputs of the Interoperable EHR Data Space are non-identifiable semantically-enriched RDF data cubes which can be linked into the Linked Medical Data Space (WP4).

The requirements of the Interoperable EHR Data Space have been satisfied. The outcomes of tasks 1.3, 1.4 and 2.3 have been taken into account in the design of the Interoperable EHR Data Space as follows. The Schema Mapping and Alignment component maps and aligns the schemas of the heterogeneous EHR and EDC databases to the Common EHR Schema (outcomes of Task 1.3 "Linked2Safety Common EHR Schema Definition"). The Non-identifiable Data Cube Creation component creates non-identifiable data cubes based on the data cube schema defined in the Common EHR Schema conforming to the Data Privacy Framework (outcome of Task 2.3 "Linked2Safety Data Privacy Framework and Consent Forms"). The RDF Cubes Semantic Annotation, Enrichment and Storage component semantically enriches RDFized data cubes based on the Semantic EHR Model (outcome of Task 1.4 "Linked2Safety Semantic EHR Model").

The design presented here will be used to implement the components of the Interoperable EHR Data Space in tasks 3.2 and 3.3.

6. References

[1] A. Antoniades et al, The effects of applying cell-suppression and perturbation to aggregated genetic data, Proceedings of the 2012 IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE), Larnaca, Cyprus, 11-13 November 2012

[2] Deliverable D1.2 Linked2Safety Reference Architecture

[3] Deliverable D1.3 Linked2Safety Common EHR Schema

[4] Deliverable D1.4 Linked2Safety Semantic EHR Model

[5] Deliverable D2.2.a Linked2Safety Data Privacy Framework and Consent Forms

[6] P. Waegemann, "EHR vs. CPR vs. EMR," Healthcare Informatics Online, http://www.healthcareinformatics.com/issues/2003/05_03/cover_ehr.htm 2003

[7] Health Information and Management Systems Society, "EHR: electronic health record," Available at: http://www.himss.org/ASP/topics_ehr.asp. Accessed February 15, 2011.

[8] I. Iakovidis, "Towards personal health record: current situation, obstacles and trends in implementation of electronic healthcare record in Europe," International Journal of Medical Informatics, vol. 52, pp. 105-117, 1998.

[9] A. Dobrev, K. A. Stroetmann, V. N. Stroetmann, J. Artmann, T. Jones, and R. Hammerschmidt, "Report on the conceptual framework of interoperable electronic health record and ePrescribing systems," EHR Impact project, Deliverable D1.2, 2008.

[10] E. Edwards, "Electronic Health Records: Essential IT Functions and Supporting Infrastructure," Gartner Research, 2007.

[11] "Health Informatics – Electronic Health Record – Definition, scope and context," ISO/TR 20514:2005.

[12] R. Dick, E. Steen, and D. Detmer, "The Computer-Based Patient Record: An Essential Technology for Healthcare. Revised Edition of original 1991 report," US National Academy of Sciences, Institute of Medicine, Washington , DC. 1997.

[13] Committee European Normalization CEN/TC 251 Health Informatics Technical Committee, "ENV 13606-1: 2000, Health Informatics - Electronic healthcare record communication - Part 1: Extended architecture."

[14] D. Garets and M. Davis, "Electronic Medical Records vs. Electronic Health Records: Yes, there is a difference: A HIMSS Analitics White Paper," 2006.

[15] "Key Capabilities of an Electronic Health Record System" in The National Academies Press Washington, Institute of Medicine of the National Academies, DC, 2003.

[16] HL7 EHR Clinical Research Functional Profile, ANSI/HL7 EHR CRFP, R1-2009, Release 1, 2009.

[17] "HL7 EHR System Functional Model: A major development towards consensus on Electronic Health Record System Functionality -A White Paper," Health Level 7, 2004.

[18] S. Hoffman and A. Podgurski, "Finding a Cure: The Case for Regulation and Oversight of Electronic Health Record Systems," *Harvard Journal of Law and Technology,* vol.22, pp. 107-165, 2008.

[19] "Electronic Health Records Overview," NIH National Center for Research Resources, The MITRE Corporation, 2006.

[20] J. A. Handler, C. F. Feied, K. Coonan, J. Vozenilek, M. Gillam, P. R. Peacock, R. Sinert, and M. S. Smith, "Computerized Physician Order Entry and Online Decision Support," Academic Emergency Medicine, vol. 11, 2004.

[21] Iakovidis, A. Dogac, O. Purcarea, G. Comyn, and G. B. Laleci, "Interoperability of eHealth Systems - Selection of Recent EU's Research Programme Developments," in Proceedings of (CeHR) International Conference 2007, eHealth: Combining Health Telematics, Telemedicine, Biomedical Engineering and Bioinformatics to the Edge, Regensburg, Germany, 2007.

[22] P. Szolovits, "Artificial Intelligence in Medicine," Boulder, CO, USA: Westview Press, 1982.

[23] "IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries,"New York: 1990.

[24] "Coming to terms: Scoping Interoperability for Health Care," Health Level 7 EHR Interoperability Work Group, 2007.

[25] C. Rinner, "Electronic health records (EHRs): Data export of health information systems based on the Entity-Attribute-Value model as CEN prEN 13606 compliant EHR extracts by means of Archetypes," in *Section of Medical Information and Retrieval Systems* Vienna: University of Technology, 2007.

[26] A. Begoyan, "An overview of interoperability standards for Electronic Health Records," Proceedings of the 10th International Conference on Integrated Design and Process Technology (IDPT 2007), 2007.

[27] European Commission Information Society and Media, "SemanticHealth Report-Semantic Interoperability for Better Health and Safer Healthcare," in *Available at:* http://ec.europa.eu/information_society/activities/health/docs/publications/2009/2009semantic-health-report.pdf, 2009.

[28] HealthIT, "Meanigful Use Matrix," in *Available at:* http://healthit.hhs.gov/portal/server.pt/gateway/PTARGS_0_11113_872719_0_0_18/Meaningful_Use_Matrix.pdf, 2009.

[29] G. W. Beeler, J. H. Duteau, G. Grieve, L. McKenzie, and R. Natarajan, "HL7 Version 3 Standard," *Version 3 Ballot,* 2011, Available at: http://www.hl7.org/v3ballot2011may/html/welcome/environment/index.html.

[30] T. Beale, "Archetypes: Constraint-based Domain Models for Future-proof InformationSystems," OOPSLA 2002 workshop on behavioural semantics, 2002.

[31] P. Schloeffel, T. Beale, G. Hayworth, S. Heard, and H. Leslie, "The relationship between CEN 13606, HL7, and openEHR," *Health Informatics Conference,* 2006.

[32] E. Hovenga and S. Garde, "Electronic Health Records, Semantic Interoperability and Politics," Electronic Journal of Health Informatics, vol. 5, p. e1, 2010.

[33] S. Garde, P. Knau, E. Hovenga, and S. Heard, "Towards Semantic Interoperability for Electronic Health Records: Domain Knowledge Governance for openEHR Archetypes," *Methods of Information in Medicine,* vol. 43, pp. 332–343, 2007

[34] Deliverable D1.1 Requirements Analysis

[35] Deliverable D2.1 Legal and Ethical Requirements

[36] M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modelling Language, Addison-Wesley Professional, 3rd Edition, 2003

[37] R. Pooley and P. Stevens, Using UML Software Engineering with Objects and Components, Addison Wesley, 1999

[38] Description of Work of Linked2Safety Project (288328) 2011-08-02