Dissertation

Master in Medical Information Systems Management

# *Exporting data from an openEHR repository to standard formats*

**Jorge Daniel Borges Brilhante Almeida**

Leiria, March 2015

Dissertation

Master in Medical Information Systems Management

# *Exporting data from an openEHR repository to standard formats*

**Jorge Daniel Borges Brilhante Almeida**

Masters dissertation performed under supervision of Doutor Ricardo João Cruz-Correia, Professor of School of Management and Technology from Leiria Polytechnic Institute and the Faculty of Medicine of the Oporto University

Leiria, March 2015

*This page was intentionally left blank*

*"Declare the past, diagnose the present, foretell the future."*
***Hippocrates***

*"Arguably the greatest technological triumph of the century has been the public-health system, which is sophisticated preventive and investigative medicine organized around mostly low- and medium-tech equipment; ... fully half of us are alive today because of the improvements."*
***Richard Rhodes***

*This page was intentionally left blank*

# Acknowledgements

To my son, who gave me the strength to endure during the long and lonely nights of writing.
To my friends and family for their patience and support.
To Professor Ricardo Correia and Samuel Frade for their guidance, dedication, collaboration and availability throughout this stage.
And finally, to everyone who have in some way contributed so that this thesis could be concluded.

*This page was intentionally left blank*

# Abstract

With the healthcare sector computerization, a large amount of data is produced from medical encounters, therapeutic outcomes and other aspects of healthcare provider's organizations current activity. Decision support systems cover several methodologies and approaches that may be applied to the healthcare sector and, since that they store and analyze data in a tabular format, it becomes necessary to assure that data sources with different data representations can be used to feed these systems.

The present work focuses on the development of a methodology to export data from an openEHR repository to standard formats through a software tool which adapts itself to different data sources for later exploration in statistical and decision support systems.

From use case and requirements analysis to the efective development of the tool, several steps were performed to document progress and to ground conclusions regarding operational test data.

Obtained results indicate that this data export is feasible, but also highlight the need to define parameters so that the tool may function.

*Keywords: Decision Support Systems; openEHR; Systems Integration*

*This page was intentionally left blank*

# Resumo

Com a informatização do sector da saúde, uma grande quantidade de dados é produzida a partir de encontros médicos, resultados terapêuticos e outros aspectos da actividade corrente dos prestadores de cuidados. Os sistemas de apoio à decisão englobam várias metodologias e abordagens que se podem aplicar ao sector da saúde e, sendo que esses sistemas armazenam e analisam dados em formato tabular, torna-se necessário assegurar que fontes de dados com diferentes representações de informação podem ser utilizadas para alimentar estes sistemas. O presente trabalho debruça-se no desenvolvimento de uma metodologia para a exportação de dados de um repositório openEHR para formatos *standard* através de uma ferramenta de *software* que se adapte às diferentes fontes de dados para posterior análise em sistemas estatísticos e de apoio à decisão.
Desde a análise de casos de uso e requerimentos até ao efectivo desenvolvimento da ferramenta, vários passos foram dados para documentar o progresso e fundamentar conclusões respeitantes aos dados dos testes operacionais.
Os resultados obtidos indicam que esta exportação é exequível, mas evidenciam também a necessidade de definir parâmetros para que a ferramenta possa funcionar.

Palavras-chave: Sistemas de Apoio à Decisão; openEHR; Integração de Sistemas

*This page was intentionally left blank*

# List of figures

*This page was intentionally left blank*

# List of tables

*This page was intentionally left blank*

# Abbreviations and Acronyms

EHR – Electronic Health Record

ETL – Extract, Transform, Load

IEEE – Institute of Electrical and Electronics Engineers

epSOS – European Patients Smart Open Services

DSS – Decision Support System

DW – Data Warehouse

TPS – Transaction Processing Systems

MIS – Management Information System

ES – Expert Systems

EIS – Executive Information Systems

DSA – Data Staging Area

DPA – Data Presentation Area

DAT – Data Access Tools

OLAP – On-Line Analytical Processing

OLTP – On-Line Transaction Processing

CDW – Clinical Data Warehouse

DM – Data Mining

KDD – Knowledge Discovery in Databases

XML – eXtensible Markup Language

BIRCH – Balanced Interactive Reducing and Clustering using Hierarchies

CURE – Clustering Using REpresentatives

DBSCAN – Density-Based Spatial Clustering of Applications with Noise

OPTICS – Ordering Points To Identify the Clustering Structure

CLIQUE – CLustering In QUEst

STING – STatistical INformation Grid

ISO – International Organization for Standardization

IHE – Integrating the Healthcare Enterprise

RM – Reference Model

AM – Archetype Model

IT – Information Technology

SM – Service Model

CEN – European Committee for Standardization

HL7 – Health Level Seven

ADL – Archetype Definition Language

API – Application Programming Language

SNOMED – Systematized Nomenclature Of MEDicine

ICD – International Classification of Diseases

LOINC – Logical Observation Identifiers Names and Codes

ICPC – International Classification of Primary Care

CKM – Clinical Knowledge Manager

GRM – Guideline Representation Model

RDBMS – Relational DataBase Management System

AQL – Archetype Query Language

SQL – Structured Query Language

OQL – Object Query Language

JFC – Java Foundation Classes

GUI – Graphic User Interface

JSON – JavaScript Object Notation

SOAP – Simple Object Access Protocol

REST – REpresentational State Transfer

WSDL – Web Service Definition Language

WADL – Web Application Description Language

# Index

# Achievements

- Article submitted to the HCist 2014 International Conference on Health and Social Care Information Systems and Technologies and published on Procedia Technology

- openEHR Translation Tool

- Project report

*This page was intentionally left blank*

# 1 Introduction

Nowadays, the need for information and the knowledge that it may provide are essential in the health care sector because, in essence, the practice of medicine is an information-driven strive. Whether to resource optimization, service evaluation or improving quality of service, systems generated information have yet unexplored potentialities.

However, the quality of data from where information is extracted is yet below the expected level and, thus leading to a situation in which the obtained knowledge is limited and often irrelevant or redundant. This is one of the main reasons that information systems who support physicians, health professionals, hospitals and primary care units fail to answer due to their base limitation: the quality of fed data, namely at the heath record level.

And what roots the lack of data quality? Usually, two reasons support this fact: non-investment in measures or infrastructures to improve data quality due to the fact that its economical return is not immediately visible and also the fact that data which is considered to be valuable to operational usage may not be suitable for medical research.

openEHR arose as a new instrument available to represent information and knowledge in health. It is a standard that comprises a set of electronic health care architecture specifications and establishes the base to the development of interoperable modular applications. As these specifications define the way clinical information concepts are organized by Information Services, obtaining semantic interoperability amongst different systems becomes possible. The openEHR standard is developed, revised and refined in a collaborative manner, sharing an international high quality and interoperable electronic health record vision.

## 1.1 Knowledge from data

In a time where is almost impossible to give a step without leaving a digital footprint, and if we transpose this idea to medicine, nearly every physician or medical staff activity produces data that is valuable for collection, review and management. The constantly growing volume of information will later give place to medical knowledge which will be used to provide the best patient health care possible.

For medical knowledge to increase, it requires valid data sets resulting from basic clinical-collected data to be used in research. This fact uncovers the utter importance of biomedical informatics, concerned with gathering, treating and optimally using information in healthcare and biomedicine and why is it critical to the current and future practice of medicine by studying healthcare outcomes that result from this practice [1] [2].

Health information technology has become significantly important in medical research by implementing technologies like electronic health records, knowledge-management systems and clinical data storage, thus improving medical research and advancing outcomes research [3]. By outcomes research, we understand the comprehension of one or more health care practices and interventions end results [4]. These end results can influence decision-making processes towards resource management through better indicators such as cost-effectiveness or comparative effectiveness, since medical knowledge not only comprises "*diseases and diagnoses but also about the organization of health care and the procedures followed in the institutions through which health care is delivered*"[5].

To achieve this, data collection must occur so it can be analyzed, treated and then forwarded to report findings. Therefore, patient or patient-related information generated during encounters or performed exams are one of the most important sources of information for knowledge extraction and clinical decision support systems.

In the healthcare environment, we have been observing an increasing progression of information systems which can support clinical research and several tasks, from medical encounters to patient information storage: EHR systems (to collect data in a structured format, reduce redundancy and identify intervention-eligible patients) [6], data mining tools (to predict patterns in patient conditions and their behaviors) [7], decision support systems (to provide clinicians with evaluations and recommendations to support clinical decision making) [8], computerized physician order entry systems (to automate the medication ordering process and implement legible and complete orders in a standardized form) [9], clinical data warehouses (to combine data from several sources to achieve conclusions otherwise impossible to achieve separately by means of business requirements analysis, data and architecture design) [10][11]. Such systems feed clinical information management platforms that have an important role in the improvement of healthcare.

### 1.1.1 Data Quality

As mentioned before, data is generated from the most various sources. However, none of this data is useful if not suitably stored. Data quality has an important part in this process and, contrary to what is often thought, is does not concern only data insertion errors or misspellings. In fact, the consequences of poor data quality can have an enormous impact on business and organizations, leading to large costs due to several causes such as scattered or overlapping data, duplicated registries, inaccurate or unreliable information and incomplete or inconsistent data. Transposing this issue to the health sector, it is easy to understand its importance. Whether it is for research purposes or to resolve a structural question, data quality is vital to achieve proposed objectives and failure to assure it can have disastrous consequences, namely with the loss of human lives. Several tools exist to address these problems and with different application domains. Two examples of these tools which can be applied the health domain are Artkos [12] and ClueMaker [13]. While the first aims at the data integration instance-level conflict resolution and error localization through the application of a proprietary metamodel during the ETL process in a Data Warehouse for health applications and pension data (even though it's an academic project), the latter has evolved for a commercial product and provides object identification and deduplication, removing duplicated copies of repeating data using clues as an input for a matching algorithm by means of rich expressions of the semantics of data [14].

### 1.1.2 Interoperability

According to IEEE, interoperability is the "Ability of a system or a product to work with other systems or products without special effort on the part of the customer. Interoperability is made possible by the implementation of standards." [15]. This definition implies other two notions:

- technical interoperability;
- semantic interoperability.

By technical interoperability, we understand the exchange of data between systems, whichever means are used, and relies solely on the data exchange process. Concerning semantic interoperability, it relies on the ability of the destination system to use the exchanged data in a useful manner and applying the destination system business rules, thus sharing the same domain, interpretation, context, codes and identifiers between sender and destination systems.
When technical and semantic interoperability is implemented, it is possible to move forward to achieve process interoperability, which is the conjunction of the first two with the fact of the human beings across a network can share a common understanding. In the healthcare sector these three types of interdependent interoperability are of particular importance, since information of an EHR in a specific healthcare unit must be accessible by health professionals or organizations in regions other than the one belonging to the patient's residence area. One example of this interoperability is the epSOS project [16], aimed at the integration of several countries health data integration in a way that could be possible to access patient's clinical

information even when outside of national territory. However, different levels of interoperability are found within European Union because of different views and barriers towards technological investment [17], even though it is widely agreed that semantic interoperability is the motor of life-long EHR's [18].

### 1.1.3 Decision Support Systems

It is a known fact that a bad or an uninformed decision can have a disastrous impact in an organization, whatever activity sector it belongs to. Due to high market competition and evolution, one decision can then mean the fall, maintenance or uprising in the rank of major players in a company's market segment. One good example of poor decision making with a devastating result was the Blockbuster videogame and movie rental provider: by failing to observe and to adapt to the customers' needs and market tendencies towards online movie renting rather than DVD or other physical support, it declared bankruptcy in 2010, after almost having the monopoly of movie and game rental in several countries.

Decision support systems (DSS) appeared to fill the gap between operational systems and the need to obtain information suitable for decision makers to opt for one of the several available hypothesis presented in the most grounded way. Since the majority of operational systems are not suited to the information needs, an adequate environment for storing and managing data creation became obligatory. However, to better understand the importance of DSS in an organizational context, some concepts must be apprehended first.

#### 1.1.3.1 The decision-making process

The decision-making process for whatever problem involves several steps towards its solution in a successful manner or not. These steps are:

- Situation identification and analysis;
- Alternatives development;
- Alternatives comparison;
- Classification of each alternative risk;
- Best alternative election;
- Execution and evaluation.

Failing to build a decision based in the previous mentioned steps will lead to a poor decision support and, as a consequence, loss of outcome whether it may be financial or not.

### 1.1.3.2 Decision Characteristics

There are various kinds of decision, depending on the management level in which we are focusing. Figure 1 illustrates the relation between these two:

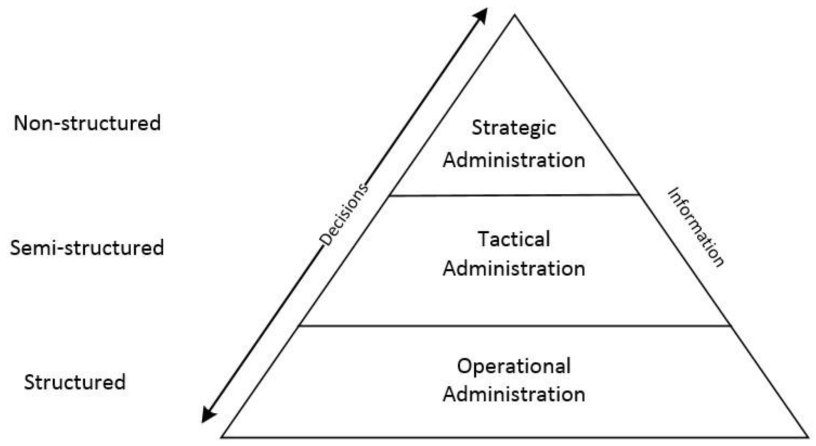


**Figure 1 – Decision characteristics and management level relationship**

As it can be observed, information is generated and used at operational, tactical and strategic administration levels to base every type of decision, which can be structured, semi-structured or unstructured, respectively.

By *structured decisions*, we understand the ones in which is possible to specify the procedure to follow. These decisions are typical of operational administration and an example can be the decision to perform an inventory. Operational systems are located at this level.

*Semi-structured decisions* are made at the tactical administration level and have some impact along the time. Regarding to the procedure to follow, some steps can be previously known, but their influence in the final decision is not relevant. One example can be the decision to create a new institutional website or launching a new marketing campaign. Management information systems are located at this level.

Regarding the *unstructured decisions*, these are made at the strategic administration level and do not take in count the procedure to follow to implement them. These decisions have effect in the organization as a whole and one example can be the budget approval or a long term objective definition. Decision support systems are located at this level.

The organization decision-making is sequential in nature. Decisions are not isolated events and each of them has a relation to some other decision or situation. As it may appear as a ‘snap’, it is made only after long chain of developments and a series of related earlier decisions.

### 1.1.3.3 Decision support systems definition and positioning

A decision support system is a complex system which allows access to the organization's databases, problem modelling and simulation performing, amongst other features, in order to aid the decision maker to organize information and to assist in every step of the decision-making process [19]. It is mainly used in the resolution of complex and poorly structured problems and combines analytical models or techniques with traditional data processing functions, such as access and recovery of information. Figure 2 represents the positioning of decision support systems among other information systems.

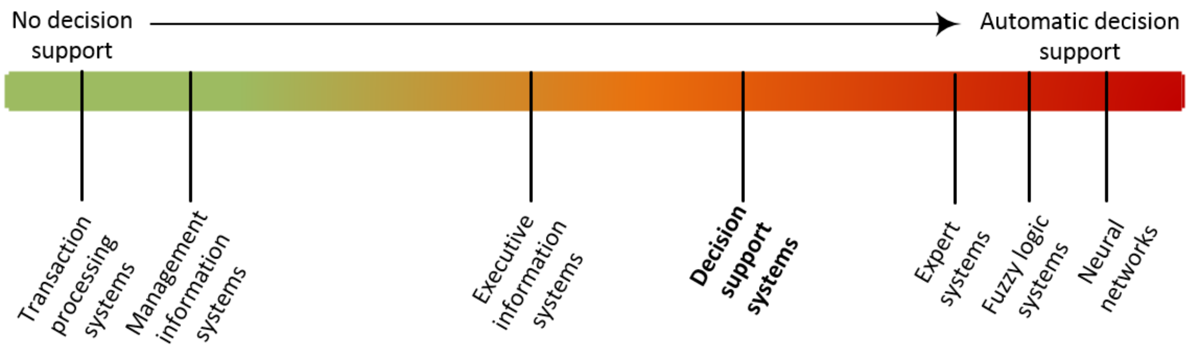

No decision support
Automatic decision support
Transaction processing systems
Management information systems
Executive information systems
Decision support systems
Expert systems
Fuzzy logic systems
Neural networks

**Figure 2 – Decision support systems positioning**

On the far left of the bar are located the transaction processing systems (TPS) and management information systems (MIS). These are projected for producing operational reports, based on routine operations with little or no integration and not oriented for decision support.

The far right side of the bar is where expert systems (ES) can be found. These systems use algorithms and advanced heuristics to replicate human logic to support decisions previously programmed to process, thus being excellent tools, but with a low flexibility level due to its narrow scope.

Somewhere in the middle of the diagram is the DSS and EIS (Executive Information Systems), which possess a high flexibility for data retrieval and analysis, providing a good decision support with specific tools that combine several viewpoints of data and allowing the decision maker to obtain valuable information.

Transposing this diagram to the reality of a company which sells a determined product, TPS and MIS can produce reports which show how many units have been sold and which has been the biggest buyer, the most profitable route of delivery or the lowest raw material cost supplier. However, a decision support system can produce reports combining information to achieve other conclusions, such as the most profitable product given the number of sold units with the lowest raw material cost and with the most optimized route of delivery on a determined season of the year. This information can help the decision maker to whether invest in a big truck to maximize delivery route profit or to buy two smaller trucks to increase delivery flux over the second most profitable route.

Decision support systems can then have a wide range of application and can be found in different sectors of activity. Nevertheless, they are more valuable when used in an environment where dimension and lack of choices combine or simply no suitable data appears to be noticeable at first glance. One good example of this combination is the health sector: along with computerized physician order entry, decision support systems can decrease medication error rates and adverse drug events [20], while improving clinical practice. This is, however, a controversial matter, due to published studies stating that computerized physician order entry systems can also facilitate medication errors [21]. These two antagonistic views, despite of their motivations, uncover the need to improve these systems with everyone’s contribution.

## 1.2 Knowledge extraction systems

As previously stated, every day, an enormous amount of data is generated around the world and with the most diverse origins. Most managers consider that their workplace has too much information and don’t use it in decision making due to the information overload, storing it for later use and believe that the cost of information retrieval is higher than retrieved information real value. This situation leads to a situation where only a small amount of data is used due to its complexity and volume.

Being true that every generating system has its own method to bring out conclusions about its data, usually under the form of reports of listings, it is also true that organizations have more than one system or data source within their network or structure. Each of these data source can then be combined and/or analyzed to infer deductions otherwise impossible to obtain separately, which makes knowledge extraction techniques an extremely important tool for decision-making activities, supplying decision makers with concise and valuable information.

### 1.2.1 Data Warehousing and Data Mining

Data Warehousing and Data Mining concepts are tightly related, due to the fact that data mining algorithms are usually applied to data contained inside a data warehouse. To understand how these concepts relate with each other, it is important to first describe each of them separately.

#### 1.2.1.1 Data Warehousing

A Data Warehouse (DW) belongs to the DSS category and according to Bill Inmon (2005), considered the father of the data warehouse concept, it is "*a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions.*" [22]. In other words, a data warehouse is focused in a subject area (business process), combining cleansed data from different sources throughout a defined period of time in a way where mainly no data is updated or deleted. When data deletion or update is performed in the source systems, a new version is stored in the data warehouse expiring (but not deleting) the old version, thus enabling to find patterns in data that may help future decisions.

This collection of data flows within the data warehouse environment from the operational (production) environment through a complex process of extraction, transformation and loading of data (ETL). In this process, data is extracted from its sources to the Data Staging Area (DSA) where it is transformed (cleansed from incongruences, bad or missing values) using several techniques to minimize the possibility of later problems before being loaded into the Data Presentation Area (DPA), where one or more *data marts* are located. A *data mart* contains data from a single business process which, as we can deduct, will lead to the presence of more than one at this area. Data Access Tools (DAT) will then access these *data marts* to apply modelling techniques, such as data mining or to produce ad-hoc queries and reports [23]. This process is schematized in Figure 3, according to Ralph Kimball (2002).

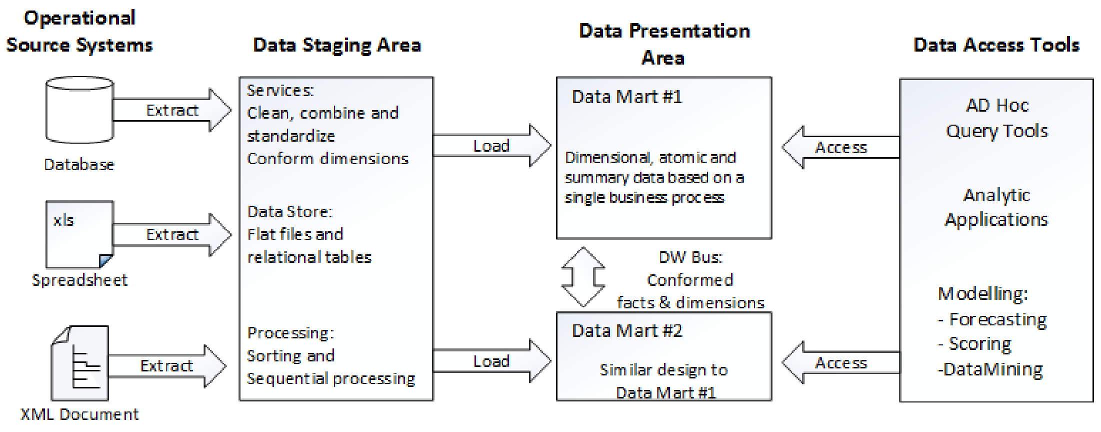


**Figure 3 – Data Warehouse basic elements**

Queries are performed by data access tools against data marts and not in Data Staging Area, since this is where data is prepared to load. We can compare the DSA with a restaurant's kitchen where meals are prepared for its customers so they can be served at the Data Presentation Area. Data Access Tools can range from ad hoc query tools for direct reporting (approx. 80 to 90 percent) to more complex tools for data mining, forecasting or other modelling applications which may even feed their results back to operational systems, thus enriching source data for later extraction back into the data marts.

There are some important concepts within the data warehouse structure:

- Time – time is always present, even if just implicitly, thus implementing the non-volatility characteristic of a data warehouse. So, it may or may not be present in a table form;
- Fact table – This table stores the numerical measurements in an entire data mart, avoiding duplication across it and each row in this table represents a measure. Usually, its primary key is composed of foreign keys and it has a many-to-many relationship;
- Dimension table – Dimensional tables enclose the textual measurements or descriptors and usually have a high number of columns, being related to fact tables;
- Granularity – The level of granularity of a data warehouse is defined by its dimensions, as well as the measurements' scope.

Data warehousing unveils the possibilities of derived data by combining different data sources in a multidimensional representation (data cube), which contains the facts (or cells), measures and attributes, thus implementing On-Line Analytical Processing (OLAP) by opposition to On-Line Transaction Processing (OLTP) present in an operational system [24]. The major differences between these two reside mostly in speed and the used schema for data storage. Table 1 table summarizes these differences by categories.

| | ***OLTP System*** | ***OLAP System*** |
|---|---|---|
| ***Data Source*** | Operational data | Merged data from operational systems |
| ***Data Purpose*** | Support business processes | Support decision making, planning, and forecasting |
| ***Speed*** | Usually fast | Speed is dependable of the stored data amount |
| ***Queries, Inserts and Updates*** | Short and simple standard queries; Short and fast operations | Complex and long tailored queries; Long running batch jobs for data refresh |
| ***Database design*** | Highly normalized, usually 3NF; Many tables | No normalization, usually star schema; Fewer tables |
| ***Space Consumed*** | Relatively small | Large consumption due to aggregates |
| ***Backup and Recovery*** | Crucial for disaster recovery and business continuity | Not important for business continuity |

**Table 1 – Differences between OLTP and OLAP Systems**

Regarding data present in each of these systems within an organization, and due to its purpose, it can be said to be rather distinct. Even though OLAP systems originate in OLTP systems, the whole paradigm changes so it can be obtained a kind of data which can support decision making. Primitive and derived data dissimilarities can then be better observed in Table 2 [25].

| | *Primitive/Operational data* | *Derived/DSS data* |
|---|---|---|
| ***Availability need*** | High, critical for business continuity | Relaxed availability, non-critical for business continuity |
| ***Detail level*** | High detailed and updated data when accessed | Summarized/Calculated and historical data |
| ***Target community*** | Clerical community | Managerial community |
| ***Access rate*** | High | Relatedly low |
| ***Access level*** | One unit at a time | One set at a time |
| ***Operation type*** | Repetitive tasks (Insert/Update/Delete) | Heuristic/Recalculating tasks (Not directly updated) |
| ***Redundancy*** | Low/None | High |
| ***Structure*** | Static | Flexible |

**Table 2 – Operational data and DSS data differences**

When implementing a data warehouse in a clinical context, the main reasons behind this decision are the need for administrators to have a broader view in terms of costs, personnel, bed allocation, billing, admission-discharge-transfer operations and expansion planning of the organization (hospital) or sector (in case of government institution), amongst others. It is known that clinical systems produce a high volume of data documenting patient care and several projects have been executed which have raised some difficulties and identified several problems in source systems to be solved, while developing techniques to optimize the ETL process and leading to successful but never-finished projects [25] [26].

Clinical Data Warehouses (CDW) are a specific implementation of the general data warehouse methodology and are central to evidence-based medicine by allowing access to healthcare providers to an information repository to which all have contributed. However, given the specificities of the health care, building a clinical data warehouse which serves manageability and investigation can be a challenging task [10].

#### 1.2.1.2 Data Mining

Data Warehouses and other OLAP systems do not allow us to identify new groups, associations, rules or patterns in contained data, since OLAP systems are exploration-oriented, while data mining (DM) is knowledge-oriented. This has led to coupling OLAP with data mining to enable a next-generation systems towards KDD (Knowledge Discovery in Databases) with promising results, even though these two started by being considered different fields [27].

Data mining refers to the extraction of important and predictive information from a large amount of data or, as described by Mehmed Kantardzic (2011), *"Data mining is a process of* *discovering various models, summaries, and derived values from a given collection of data."* [28]. It is widely accepted as an important mean to achieve the recognition of patterns or models in analyzed data through several algorithms and techniques such as neural networks, decision trees, genetic algorithms, nearest neighborhood or rule induction towards KDD [29].

Since its introduction in the late 1990's, data mining has been applied in several fields, from finance to commerce or from health to business. With the dissemination and evolution of data mining systems as well as the processing capability power in the early 2000's, new algorithms were introduced to perform outlier detection, web log and multimedia data matchmaking, bioinformatics mining, web mining and text mining, among others.

Integration of systems and the exponentiality of data generation by means of monitoring technologies and data streams, which are virtually impossible to store completely, led to the implementation of data sampling and the use of aggregations and, consequently, the emerging of parallel or distributed data warehousing concept through the growing use of XML technologies such as XML Schema-based data warehouse design, XML OLAP Cube in addition to already used standard data [30].

#### *1.2.1.2.1 Data mining process and methodologies*

Data mining projects are not easy to accomplish and many of them fail essentially due to the lack of vision (not knowing what information to bring out of data), incorrect, outdated or missing data, department bickering, legal restrictions, technical difficulties (several data formats) and poor interpretation. Since data mining is an iterative process, several steps were defined as essential, which are:

- ❖ Present the problem and hypothesis formulation;
- ❖ Data gathering;
- ❖ Data pre-processing (includes sub-tasks, such as data preparation and data reduction);
- ❖ Model construction (using the most appropriate techniques);
- ❖ Model evaluation and interpretation.

For these steps to be accomplished successfully, a methodology must be followed. This ensures that, in every step of the project, it is possible to know what tasks were performed earlier, thus having an updated status regarding the overall project execution. Some data mining methodologies are described in light detail below and are based in previous stated step order, each of them with its changes [31].

**PRMA methodology**

PRMA stands for Preparation, Reduction, Modelization and Analysis. It was followed by Sholom Weiss and Nitin Indurkhya (1995) while describing a machine learning method for predicting the value of a real-valued function, given the values of multiple input variables [32] and essentially presents slight changes from the general steps:

- Data origin business description;
- Data preparation;
- Data reduction;
- Modelization;
- Analysis.

**SCECMR Methodology**

This methodology is a little more fragmented. It was developed by Usama Fayyad and is characterized by promoting two sub-steps of the general model and adding a third step, while renaming some to reflect the change:

- Problem domain comprehension;
- Data Selection;
- Data Cleansing;
- Data Enrichment;
- Data Codification;
- Modelization;
- Reporting;
- Model production implementation.

**SEMMA Methodology**

The SAS Institute (Statistical Analysis System Institute, Inc.) has developed this five-step methodology to conduct a data mining project [33]. Its name, as in with the previously presented methodologies, derives from the first letter of each of its steps designation:

- Sample;
- Explore;
- Modify;
- Model;
- Assess.

## CRISP-DM Methodology (CRoss-Industry Standard Process for Data Mining)

CRISP-DM is supported by the majority of the computational tools and one of the most used methodologies. It is well documented and implements the iterative and incremental features of the standard data mining process while running its own steps in a structured way, allowing for its revision in any stage [34]. An overview of the whole process can be found in Figure 4.

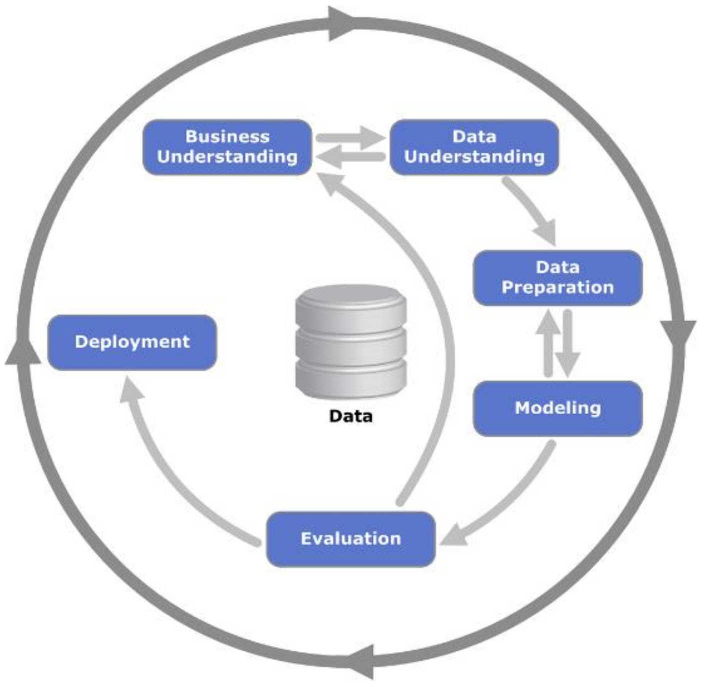


**Figure 4 – The CRISP-DM methodology**

As it can be observed, six stages are comprised in this methodology. The first stage focuses on understanding the project objective and transposing it onto an effective plan. The second stage is centered in understanding the gathered data, while identifying threats to its quality or possible hidden data subsets. The third stage involves all activities related to the construction of the data set which will be modeled. Next, on the fourth stage, several techniques will be applied according to its applicability to the raw dataset in a manner that they can be evaluated on the fifth stage to check for its compliance to the project objectives. If so, the sixth stage is achieved and the model is deployed.

Since it is a well-documented process, for every stage there is one or more sub-tasks resulting in a document containing important information for the next stages. Below, Table 3 presents each stage sub-tasks and resulting documents.

| *Stage* | *Stage sub-tasks* | *Stage resulting document(s)* |
|---|---|---|
| *Business understanding* | Meeting for requisites understanding<br>Meeting for business understanding<br>Data source analysis | Model specification document<br>Data selection process specification |
| *Data understanding* | Data selection process design<br>Data combining<br>Data normalization<br>Data tests | Technical specification regarding data selection process<br>Data quality document |
| *Data preparation* | Data selection for data mining process<br>Data cleansing<br>Data codification | Technical specification about data document update |
| *Modeling* | Algorithm selection<br>Test plan definition<br>Model creation<br>Model evaluation | Model evaluation document |
| *Evaluation* | Real problem model evaluation<br>Obtained results evaluation | Model evaluation document update<br>Data mining process description document |
| *Deployment* | Obtained model deployment planning<br>Process maintenance planning<br>Model deployment | Data mining process description document final update<br>Obtained model results and its importance to business |

**Table 3 – CRISP-DM stage sub-tasks and resulting documents**

For every data mining methodology, there are several common aspects. Whatever methodology is chosen, its process is iterative, giving place to a continuous feed along the process between stages and between iterations and the data mining process continues even after a solution is deployed. The process always starts with the problem/business understanding and all stages are similar between methodologies, while the data preparation are the ones that are more time and effort consuming.

#### *1.2.1.2.2 Data mining techniques*

Data mining methodologies are useful for organizing the whole process, but for each stage or aspect there are known problems and certain techniques to address them. For these and every other unexpected situations, the experience of the data miner is of the uttermost importance, since along the process several questions will be presented and the way they are dealt will dictate the success or failure of the data mining project.

**<u>Data preparation</u>**

One of the most critical step in the data mining process is the initial preparation of initial data

set. This task is many times neglected when taught or in research over prediction methods but, in the real world, what happens is just the opposite.
When dealing with raw data sets, it is common to find missing values (such as measure errors or unavailable values), distortions, insertion errors or inadequate samples (which can generate outliers). So, some objectives at this point go through organizing data in a standard way to be processed by modelization software and choosing the best features (or variables) that lead to a better model performance, which normally involves data transformation.

## Data organization

Computers have the ability to analyze many data in a standard way, but that same ability is limited by the description of original data. During this analysis, some data sets can be standardized, while others can present combinations or text for each feature. For learning or prediction methods to benefit from additional knowledge, it must be introduced at this point and each feature must be numerically coded. This coding can assume categorical values (eye color), discrete (qualitative) values, continuous (quantitative) values, periodical values or numerical values, which have an order or distance relation unlike categorical ones. However, a feature must me defined to prediction objective, thus promoting it to *label*.

## Data transformation

In order to apply data mining techniques, there is always the need to transform original data. Even though transformations are not optimal and are mostly manually made, the human experience and knowledge are fundamental during feature selection and feature composition tasks.
Feature composition is related with data reduction, while feature selection focuses on normalization, smoothing and differences and ratios. By normalizing data, we understand a kind of transformation that reduces values to the same scale (e.g.: decimal), thus maintaining the values between an interval. Smoothing is applied when data detail will not be used by the data mining method, thus being a form of data reduction. Differences and ratios can contribute to learning methods by specifying a difference or ratio, leading to lower number of alternatives, while being possible to use with dependent or independent variables.

## Missing data

Missing data is a problem within a data set and not easy to solve, due to its repercussions depending on the amount of cases detected. If the missing value is the label value, the sample is discarded. However, if many samples have the label value empty, it will undoubtedly derail a whole project. Some solutions can be used, such as using a constant or the feature average, but it will never be the correct value.

**Outliers**

Outliers are sample values that do not fall within the scope of the remaining data model. They can be originated by a simple insertion error or, being plausible, resulting from a spurious data variation.

Determining outliers is a tricky task, because if a sample is considered an outlier and therefore erased, the model performance can be compromised. For instance, in credit card transactions analysis, outliers can be indicators of fraud. However, in the majority of applications this is not the case.

The most used techniques for outlier detection are statistic, distance or deviation-based.

**Data reduction**

We must never forget that when building a model, we are building one to apply to big data sets. So, the objective is to tune it with a sample data set before feeding it the whole one, which raises the question of what data can be selected without compromising result quality. Data reduction must then involve several analysis parameters, such as computational effort, learning speed, learning precision, model representation, model precision or complexity. These parameters are usually incompatible with each other and cannot be simultaneously improved.

Some data reduction can be achieved by feature removal, case removal or value removal. Feature removal will enable higher precision and higher learning speed and can be accomplished by selecting a feature subgroup that has an identic performance to the whole group, by replacing the initial feature set with another one obtained from its composition or by feature ranking based on data consistency, information content, sample distance and feature statistic dependence.

Case removal is performed when the software that will be used or other techniques cannot handle the load. Usually, the most simple and used method is sampling in which the size depends on the feature number and problem complexity. This sampling can be random or systematic.

Value removal is implemented by discretizing feature continuous values. This means that values are binned into discrete symbols. One good example is age categories (child, adolescent, adult, mid-aged and old): depending on the age value, it will belong to one of these discrete symbols. However, border values can be difficult to establish.

**Data exploration**

When exploring data, graphical representation greatly increases its meaning. Results nowadays can be represented in one, two, three, four or more dimensions and, according to Tufte (2007), graphical excellence "*is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space*" [35].

Some methods of visualization can assume the form of pie charts, bar charts, scatter plots or parallel coordinates, each of them with a proper use.

Additionally to graphical exploration, statistical methods can help to understand some aspects of a data set. Such methods include statistical measures for data location (mean, median or mode) and data dispersion (variance or standard deviation). Other measures can relate two or more features to determine their dependency or relation, like covariance or correlation matrices.

**Learning from data**

Data mining approaches are based in human learning capabilities and computational learning is made from data. It consists of two main phases:

- ❖ Estimating unknown dependencies in the system;
- ❖ Using estimated dependencies to predict new outputs for later system data feed.

These phases correspond to the two classical types of inference (induction and deduction), which can be observed in Figure 5.

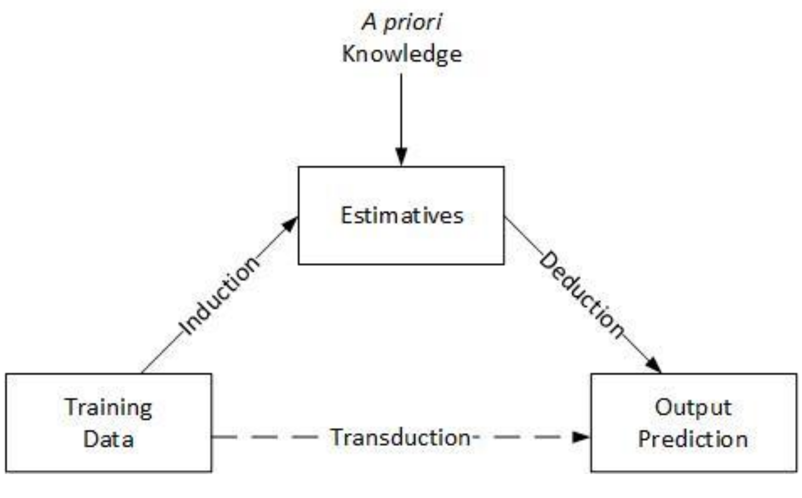


**Figure 5 – Inference types**

In the inference process, a training data set provides induction to the model estimation that, when fed with *a priori* knowledge (usually human-driven) can make deductions and predict outputs. Estimating only one model implies a global function learning for all input values, which generates a problem when the objective is to only deduct estimates for some values. So, when estimating a function without building a global model, transduction is used.

Model estimation process can be described, formalized and implemented using different learning methods. These learning methods are algorithms which estimate an unknown mapping or dependency between entries and outputs of a system from a data set and, when precisely estimated, it can be used to predict outputs given its inputs.

Within the data mining methods, we can catalogue them as supervised or unsupervised. Supervised methods are used to estimate a dependency from known input-output data while in unsupervised methods only the input data are known and the algorithm searches for patterns among all variables. The majority of data mining methods is supervised, which means they preprocess data from a training set so the algorithm can learn the association between the variable values and the predictor variables. However, the model obtained from this training set must be validated so it won't act prejudicially when applied to the data to be classified. This provisional model is applied to a test set where the target variable values were hidden to observe how the model behaves. After the necessary adjustments to decrease error rate, the obtained model is then applied to a validation set again with the target variable values hidden where some more adjustments are made to minimize error rate. The resulting model is then ready to apply to the final data set to be evaluated. The whole view of this process is shown in Figure 6.

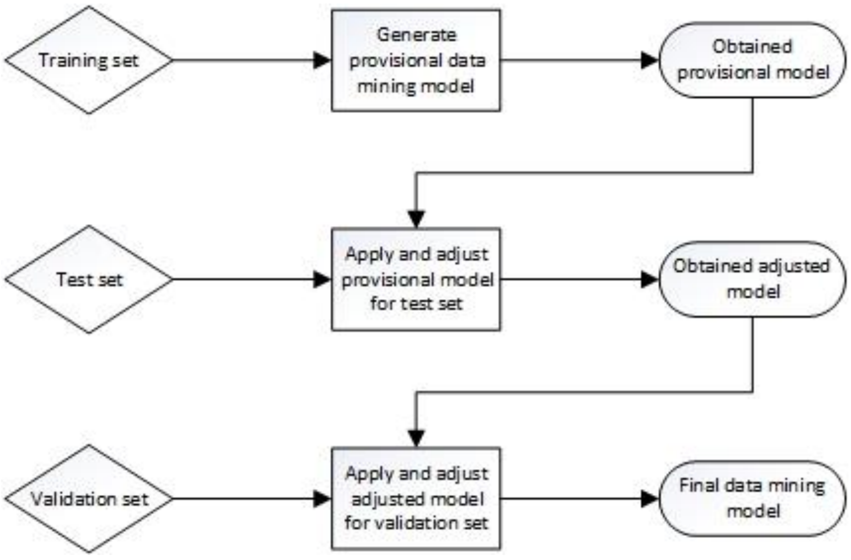

Training set
Generate provisional data mining model
Obtained provisional model
Test set
Apply and adjust provisional model for test set
Obtained adjusted model
Validation set
Apply and adjust adjusted model for validation set
Final data mining model

**Figure 6 – Supervised modeling methodology process**

The accuracy values on the test set and on the validation set will not be as high as the ones on the training set, due to *overfitting*. *Overfitting* is the violation of the principle of parsimony, in which is stated that a model or procedure must contain all that is necessary for the modelling, but nothing more [36]. When *overfitting* occurs, certain idiosyncratic trends or structures are computed, which may lead to erroneous interpretation.

Between model complexity and generalizability, a balance must be achieved in order to obtain a model which can be reliable. When increased, model complexity will result in a more accurate model in the training set, but a higher error rate in provisional and test sets.

There are no guidance lines on how to perform data splitting to obtain the training or test sets, even though it is known that a small training set will result in a low robust model and with a weak generalization capability and that a small test set will lead to a low confidence in the generalization error. However, there are some techniques for addressing these issues, being all the most used ones based in the resampling concept:

- **Resubstitution** – all samples are used for training and test sets. It is the simplest method, but also not common to use in real data mining applications;
- **Holdout** – Half or two thirds of data are used for the training set and the remaining are used for the test set. Because each set samples are independent, the generalization error is pessimist but, repeating the process with different training and test set samples randomly selected will improve the estimated model;
- **Leave-one-out** – the resulting model is obtained from using n-1 samples for training and evaluated with the remaining ones and then repeated n times with different training sets of size n-1. This approach demands high computational resources, since it is necessary to build and compare many models;
- **Cross-validation** – This is a mix of the holdout and leave-one-out methods. Data is divided in P subsets, all of equal size. P-1 subsets are used for training and the remaining ones are used for test sets. This is the most used method;
- **Bootstrap** – In its simplest form, instead of repeatedly analyzing data subsets, random subsampling with complete sample substitution is analyzed. This allows to generate "fake" data sets with the same size of the original data set, which is useful when working with data sets that have a low number of samples.

## Model evaluation

Most obtained models are evaluated on an error criteria basis (training error and generalization error). Such evaluations are performed using metrics like Mean Square Error, Root Mean Square Error or Non-Dimensional Error Index for regression problems or hypothesis evaluation for classification problems. Regression problems are related with the prediction of a result based on two or more variables whilst classification problems point a class where a sample should be placed.

In a classification problem, the error rate is:

$$Error\ rate = \frac{Number\ of\ errors}{number\ of\ cases}$$

This logically means that the accuracy of a model is obtained by:

$$1 - Error\ rate$$

For hypothesis evaluation, each case (or sample) receives a "verdict". In this perspective, it can be found to be a true positive (TP), true negative (TN), false positive (FP) or false negative (FN). As it can be deducted, a true positive or true negative is a sample correctly classified respectively as positive or negative, while a false positive or false negative is a sample classified as positive or negative when in reality it was found to be otherwise.

When measuring a model precision or sensibility, the percentage or correctly predicted items in a given class can be obtained by:

$$P = \frac{TP}{TP + FP}$$

For recall or specificity model measuring, the percentage of correctly classified items in a given class is obtained by:

$$R = \frac{TP}{TP + FN}$$

Model evaluation is of great importance due to the fact that a balanced model will be possible to apply to the real dataset and perform in the most optimized way, whatever its size. It is virtually impossible to build a flawless model, but it is very easy to build one that has a high generalization degree and also a high error rate. The keyword is, therefore, balance.

### *1.2.1.2.3 Data mining algorithms*

To build a model, it is necessary not only to apply the right techniques but also the right algorithms to the data set in a way that information can be extracted. There is a myriad of available algorithms which can be used with this objective, each of them with their advantages and disadvantages. The most used ones are described below, even though there are many more (including variants) which can be applied.

**Bayes Classifier**

The Bayes Classifier has its roots in the Bayes theorem, which can be observed below. This allows for the classifier to add external information in the data analysis process with an *a priori* probability and also with an *a posteriori* probability.

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

The X symbolizes the sample intended to be classified, while H represents the hypothesis to be tested. P(H|X) is the probability of H to be true, given observation X (*a posteriori* probability) and P(H) is the probability of H to be true (*a priori* probability).

In theory, the Bayes Classifier has the lowest classification error when compared with other classification algorithms. However, that not always is true due to uncertainty regarding some assumptions about features and conditional independence of classes.

**Linear Regression**

The objective of the regression analysis is to determine the relation between the output variable (Y) and one or several input variables ($X_1$, $X_2$,…, $X_n$). Since the measurement of the output value can be expensive, it is more affordable to measure de input values and establish the relationship between them in a way that it is possible to control the output.

**Decision Trees**

Decision trees are widely used in the real world, especially in classification problems. The objective is to create a classification model in which is possible to predict the class of a new sample.

A decision tree is composed by nodes, branches and leaves. The nodes represent where attributes are tested, while branches and leaves represent the possible values attributes can take and the classes' tags, respectively. The path between the root and one leave expresses a decision rule. Figure 7 presents an example of a decision tree.

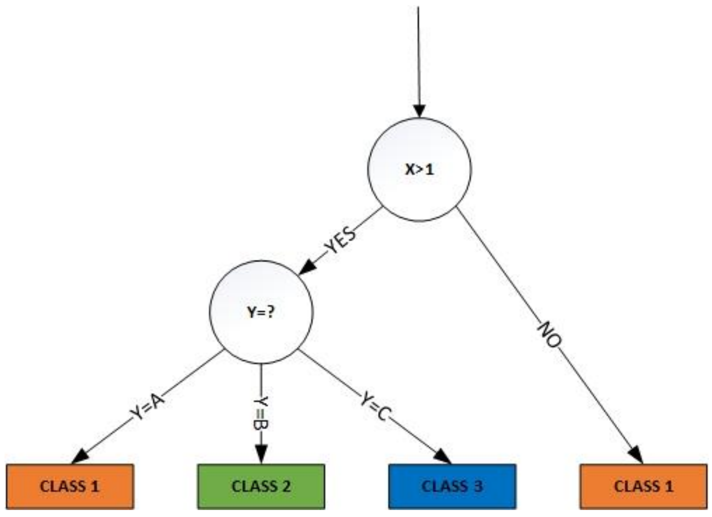


**Figure 7 – Decision tree example**

One of the most used algorithms within decision trees is the ID3 algorithm. It starts by choosing one attribute to partition samples and then applying the same method to the partitioned samples recursively. Each branch of the tree represents one decision rule and the algorithm relies on the concepts of entropy and information gain to generate them. These concepts derive from the Claude Shannon's Information Theory [37].

Entropy can be defined, in information theory, as the measure of uncertainty regarding a random variable (or sample) and the average information lost. The ID3 algorithm aims at the entropy minimization by minimizing the number of tests needed to classify a sample. This can be translated in the following formula:

$$Entropy\ (S) = -\sum_{i=1}^{n} p_{\mathrm{i}} \cdot \log_2 (p_{\mathrm{i}})$$

Being that

$$p_{\mathrm{i}} = \frac{number\ of\ registries\ in\ class\ \mathrm{i}}{total\ number\ of\ registries}$$

The information gain concept, which consists in minimizing the needed information to classify a sample in the resulting sub tree, is applied by the ID3 algorithm recurring to the following formula:

$$Gain\ (S, A) = Entropy(S) - \sum_{i=1}^{m} p_{\mathrm{i}} \cdot Entropy(S_{\mathrm{i}})$$

Attribute selection for a node is based on the assumption that the tree's complexity is related with the amount of information contained in an attribute's value, thus leading to the objective of minimizing the number of tests to classify a sample from a given dataset.
However, the ID3 algorithm has a high sensitivity to features with a high number of different values, such as social security numbers. This situation will lead to low entropy values, unless there is a way to circumvent it. Since ID3 gives preference to high ramification factor, C4.5 algorithm uses the ratio between information gain and entropy from the sample set. This can be represented by

$$GainRatio(p, T) = \frac{Gain(p, T)}{SplitInfo(p, T)}$$

Where ***SplitInfo*** is

$$SplitInfo(p, test) = -\sum_{j=1}^{n} P'\left(\frac{j}{p}\right) \cdot \log\left(P'\left(\frac{j}{p}\right)\right)$$

By applying the gain ratio to the data set being tested for the node, C4.5 chooses the best fitted attribute to split the set based on the normalized information gain. In addition, C4.5 allows to work with missing and numerical value attributes, as well as to perform the tree pruning by discarding branches which does not contribute to error reduction in the classification [38].
There are several advantages in using decision trees such as the possibility to work with continuous values as well as symbolic values, the ability to generate understandable rules, the construction of interpretable models, the clear designation of the most important features contributing to prediction and the fact of being light from the computational point of view, among others. However, there are some disadvantages which restrain its wider use. Namely, the size of the tree has a great influence on its accuracy due to the fact that when having a high number of splitting characteristics, the classification algorithm tends to increase the branch number, rendering the extracted decision rule useless or highly difficult to understand. Other limiting characteristic is that most of the algorithms demand discrete values as target attributes in order to be applied, which excludes most induction algorithms which can be used [39].
In conclusion, decision trees allow to generate decision rules during which generalization is applied and the rule set is compact. This process implies ordering the resulting rules, being

necessary to state a class to allocate the undefined samples by the resulting rules.

**Association Rules**

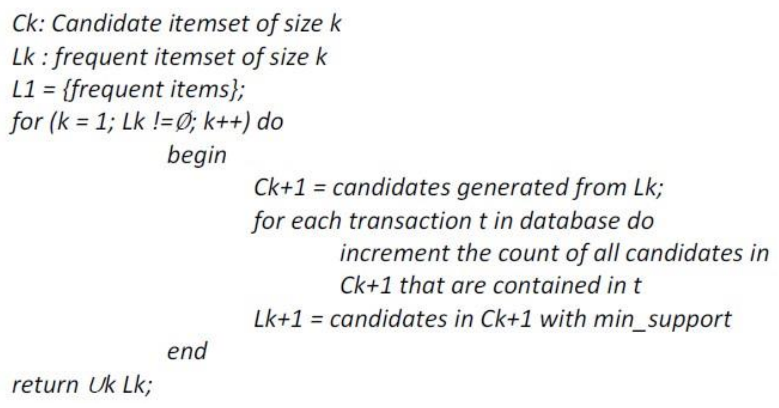
This is the most common way to find local patterns in non-supervised systems with the objective of finding interesting rules which are previously unknown. However, information overload needs to be avoided in order to keep data analysis as smooth and quick as possible [28].

Three types of association rules can be obtained from applying algorithms to a specific dataset: the *useful rule* (which can be explained in an intuitive way), the *trivial rule* (which does not add any value to an empiric experience) and the *inexplicable rule* (which does not have a plausible explanation).

One of the most used examples for association rules is the market basket analysis. In this analysis, the shopping cart (market basket) content is examined in order to find items which are repeated in other shopping carts (*itemsets*) and can contribute to define item localization in stores or to provide client-oriented advertising based on shopping habits. Nevertheless, it can be applied to other domains, such as business or research. For instance, to predict service degradation during utilization peaks in diverse times of the year on a cell network, detecting insurance fraud or even in studying a new drug side effect. Several difficulties arise here, namely the high number of baskets to examine, the exponential frequent itemsets in relation to different items and the need for scalable algorithms in order to increase linear complexity par opposition to exponential complexity due to computational resources limitation.

To find frequent itemsets, its support must be higher than a certain value and by support we understand the percentage of transactions that contain that same itemset. In addition, an itemset that contains x items is called an x-itemset [39].

The Apriori Algorithm is the reference algorithm to discover association rules within data. It was first presented in 1994 by Rakesh Agrawal and Ramakrishnan Srikant and it is composed by two phases. In the first phase, it starts by generating candidates and then it performs the counting and selection of the obtained candidates. In the second phase, the algorithm generates the association rules. The pseudo code for algorithm can be found in Figure 8.



```
Ck: Candidate itemset of size k
Lk : frequent itemset of size k
L1 = {frequent items};
for (k = 1; Lk !=Ø; k++) do
        begin
            Ck+1 = candidates generated from Lk;
            for each transaction t in database do
                increment the count of all candidates in
                Ck+1 that are contained in t
            Lk+1 = candidates in Ck+1 with min_support
        end
return ∪k Lk;
```

**Figure 8 – Apriori Algorithm pseudo code**

Like any other algorithm, Apriori has some advantages and disadvantages. Amongst the advantages, we can list the ease of data interpretation, the ease of understanding and the possibility to produce results without certainty or expectation. However, the exponential growth complexity along with the problem dimension, the poorly defined detail level and the problems arisen with products (items) which have a low degree of appearance (Market Basket Analysis) are the main disadvantages of this algorithm.

## Clustering

When analyzing samples, sometimes it is useful to group them in classes by resemblance. Clustering is a set of methodologies aimed at automating records association by segmenting a whole data set in groups based on similarities for better interpretation and deeper analysis, where patterns that may exist in data are unknown, thus not existing predefined classes.

This record organization is not classification-oriented mainly because there is no target variable to predict or estimate and, unlike association rules process, clustering belongs to the unsupervised learning category. One cluster aggregates several samples with similar characteristics, being that these same samples are distinct from others belonging to other clusters.

Typical applications for clustering are the initial data relation discovery, pre-processing stage for other algorithms, pattern recognition, geographical data analysis, image processing, financial data analysis or document classification in a web context.

Several algorithms can be invoked, each of them with its own approach. For illustrative purposes, only the K-means algorithm will be described, which is one of the most used and is positioning-based. However, there are other algorithms that can be used, depending on the desired objective. Other approaches comprise hierarchy (BIRCH, CURE), density-based

(DBSCAN, OPTICS, CLIQUE), grid-based (STING, WaveCluster) or model-based (statistical approach, AI, Soft Computing).

K-means algorithm follows a five step procedure to achieve its objective. In the first step, the number of clusters (groups) is defined (K). The next step is to randomly calculate the cluster initial center (or seed) so it can be possible to associate each sample to its nearest seed in the third step. In step four, a new center for each cluster is estimated and, in the fifth step, a repetition of steps three and four are performed until the algorithm converges.

It is important to refer that K-means can only be used with numerical (and preferably normalized variables). Also, the partitioning result depends of the prototype initialization, which make it necessary to avoid circumstances where the algorithm can be "stuck" in a local minimum. One possible alternative is to run the algorithm several times and with different initializations in order to achieve an optimal global positioning clustering.

Amongst its advantages, the K-means algorithm is a simple and easy one to understand and its samples are automatically associated to clusters. On the other hand, the need to define the initial cluster number along with the obligation that each sample belongs to a cluster makes this algorithm too sensitive to outliers, which can lead to problems when dealing with different size, density or form clusters [28][40].

## 1.3 openEHR

The electronic health record is one of the root components in health. What started by an initiative to evolve from paper records to digital records soon became a source of information on which bases great part of today's medical research.

It is a known fact that standards exist in many areas, and the healthcare sector is no exception. According to International Organization for Standardization (ISO), a standard can be defined as *"a document that provides requirements, specifications, guidelines or characteristics that can be used consistently to ensure that materials, products, processes and services are fit for their purpose"* [41].

openEHR is an "*open standard specification that describes the management, storage, retrieval and exchange of data in Electronic Health Record*" [42]. It first appeared in 1992 inserted in the GEHR/openEHR European project. Later, Australia invested in the initiative, in the meantime renamed from Good European Health Record to Good Electronic Health Record. Currently maintained by the non-profit openEHR Foundation [43], an international and online community to promote electronic health records of high quality supporting the need of patients and clinicians around the world, openEHR aims to fulfil technical and clinical objectives such as more involvement in real clinical trials, rate of archetype publishing increase, software development growth, openEHR 2.x reference model characterization or service specifications and IHE [44] mappings definition [45].

### 1.3.1 Architecture

The archetype concept was the most significant one presented by openEHR. It designates "*a model defining some domain concept, expressed using constraints on instance structures of an underlying reference model*" and introduces the two-level methodology approach [46]. In this methodology, we have the Reference Model (RM) and the Archetype Model (AM). The Reference Model represents a generic but specific to healthcare domain characteristics of health record components in a way it can be stable over time and to allow interoperability, whilst the Archetype Model constraints the Reference Model information structure by defining the rules in which values are expressed, thus allowing for semantic interoperability [47].

Usually, the Reference Model comprises a few classes such as observation, role, act and participation. In the Archetype Model we can find the medical concepts like lab results, cephalic perimeter or blood pressure constraining a Reference Model class (e.g. observation) to the blood pressure archetype [48].

One of the main advantages of this approach is that domain experts can create, maintain and evolve archetypes without worrying about IT issues, due to the fact that an archetype's semantics lies in the Reference Model. This allows to create virtually an unlimited number of archetypes from the Archetype Model, each of them describing a distinct and complete clinical concept, while having a formal and defined mechanism to permit its intelligent querying [49][50].

There is also a Service Model (SM), which implements the services available in an openEHR system and that are compliant with other standards such as CEN/ISO 13606 and designed to be interoperable with communication standards such as HL7. Figure 9 schematizes an openEHR structure diagram as Thomas Beale and Sam Heard presented in 2008 [51]:

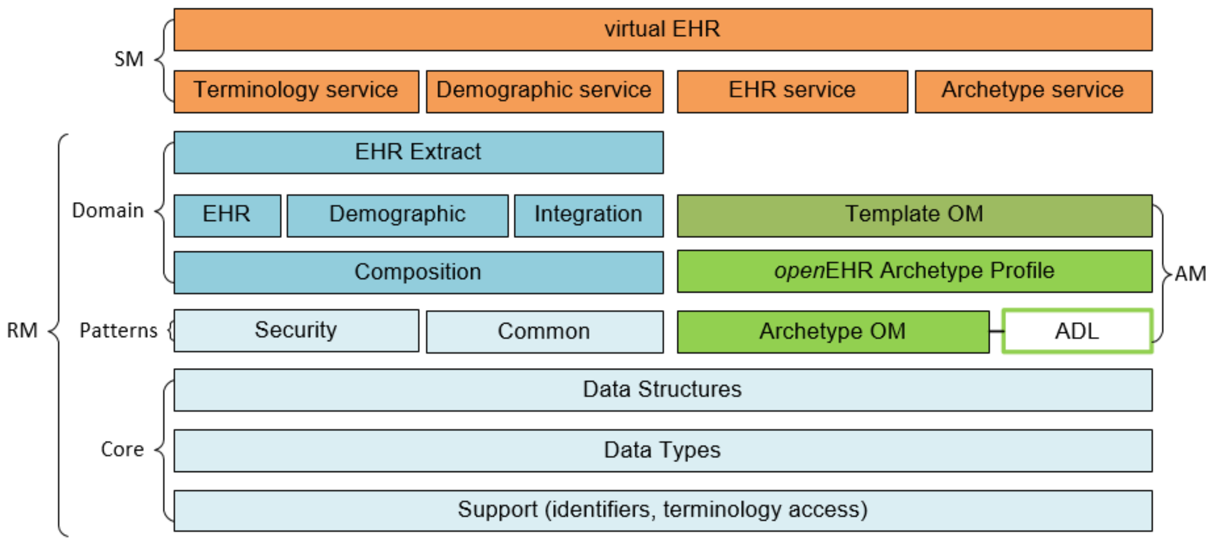


**Figure 9 - openEHR architecture**

Dependency between components is read from bottom to top and each component contains packages to fulfill its functions. As we can see, the Core components group is composed by the *Support* component (terminology, coding system data), the *Data Types* component (text, number, boolean, etc.) and the *Data Structures* component (table, tree, list, etc.). This group is responsible for providing data for other components to process.

The Patterns components group holds the *Security* (packages for access control and security) and the *Common* component (contains common packages and links the Reference and Archetype models).

In the Domain components group we can find the *EHR* and *Composition* components, which define relationships in the EHR system between all other components. As for the *EHR Extract* component, it concerns all methods for building an interoperable EHR system. *Integration* component contains the packages responsible for legacy systems integration by means of integration archetype (a custom converter) use for translation from a message/artifact pair to an openEHR archetype. Regarding *Demographics* component, it holds the packages describing the patient demographics like healthcare providers or other entities involved.

The Archetype Model contains all components related with the use and description of archetypes and templates. Templates are a composition of one or more archetypes, usually in the disposition of a screen form or report. They can have constraints defining different values from the ones defined in one of its composing archetypes as well as including or removing its constraints number. As for the ADL component mentioned in this model, it refers to the Archetype Definition language, which is "*a formal language for expressing archetypes, which* *are constraint-based models of domain entities, or what some might call 'structured business rules'.*" [52]

The top level model in the figure refers to the Service Model layer. Its focus is on the availability of services to other applications and systems and is closely coupled to previously mentioned components. Here, we can find the *Virtual EHR API*, which is responsible for the providing the means for an application to communicate with the EHR and modify partially or completely an existing artifact through its corresponding archetype. Below the Virtual EHR API, are the four component services positioned around the EHR:

- ❖ **Terminology service** – This component is the bridge to all the terminology present in the EHR coding systems, such as SNOMED, ICDx, LOINC, ICPC and others, contributing to ontology and terminology-based knowledge;
- ❖ **Demographic service** – in an EHR system, this component provides communication to the information about different entities participating in care (patients, providers, etc.), enabling the access, storage and modification of data;
- ❖ **EHR service** – Enables the manipulation of data at the server side dealing with coarse-grained composed information to provide a more fine-grained information, such as averages or other criteria-based information;
- ❖ **Archetype service** – Provides the interface to online repositories accessible by applications or the EHR system itself.

Once the set of services is not a static one, it is expected that this layer will grow to accommodate the continuously growing services need [48] [51] [53].

The openEHR EHR information model focuses in fine-level granularity components, such as Data Types and Support components, which will feed the higher level components like Compositions, EHR and Demographic components. Due to the separation between clinical concepts and knowledge layers, data inserted in an openEHR system happens at runtime against the archetype-defined constraints [48]. Inside, it is organized in *Directories*, which are a group of *Compositions* where data committed during an activity is represented. Inside a composition, we can find the *Section*, which acts like a descriptor of each composition and enables search based in these same descriptors.

An *Entry* symbolizes any clinical activity that varies from a radiological exam to a simple drug prescription and is formed by a *Cluster* and an *Item*. The cluster type represents more elaborated entries, whereas item describes its data constraints descriptors (filed name, data type and value). Regarding the EHR component, it contains a root package with the same name. Inside, besides the *Directory* and *Compositions*, the information stored relates with *EHR_Access* (access control, security policy, allowed users definitions), *EHR_Status* (which keeps track of the EHR system) and *Contributions* (EHR system change logs used mainly for versioning) [49] [51] [53].

### 1.3.2 Community participation

Domain Knowledge Governance can be supported by a broad archetype repository which can implement archetypes' defined processes building the foundations for representing that same knowledge through semantically interchangeable clinical content [54]. Additionally to the openEHR Foundation initiatives, and to promote participation, an Archetype Editor has been released for archetype development using the Archetype Definition Language (ADL). After submission and validation, an approved archetype is available in an online repository, accessible through the Clinical Knowledge Manager (CKM). Archetypes can be referred, combined and/or derived to build specialized archetypes, even though submission is needed to achieve approval [49][50][55]. One example of this community participation was the development of a methodology to integrate guideline representation models (GRM), which represent steps or orientations towards reducing costs or preservation of patient safety, with openEHR archetypes, templates and rules through a rule engine for treatment of large B-cell lymphoma [56].

### 1.3.3 openEHR repository data structure

Like any other information system, openEHR-based repositories rely on databases to store data. Besides the Relational Database Management System (RDBMS) there are other options to store data, such as eXtensible Markup Language (XML) databases. RDBMS are dependent of a tabular data representation in which data is dispersed in several tables related between them while in a XML database the data is stored in XML format. Since XML databases are usually associated with document-oriented databases, they are mostly used in an environment where a file holds all information regarding an employee, a customer or a business partner. However, the most common topology of an openEHR repository is based on a hybrid approach in which data is stored on RDBMS with some XML data fields, thus contributing for a myriad of query mechanisms depending of the database type, engine and integration level.

Other possible storage technologies to implement are object oriented databases, which are best suited for complex data (or complex data relationships) or plain text storage (when having sequential records per line), either one with little usage quota [57].

### 1.3.4 openEHR repository data query

When querying in an openEHR repository, several languages can be employed. In addition to this, the multi-level modelling approach must not be forgotten. One example is the use of AQL (Archetype Query Language) which allows for EHR archetype based data access regardless of the core storage mechanism.

Several implementations of openEHR repositories have been identified and categorized leading to a panoply of used database storage types and approaches [57].

For data querying, depending of the database engine, numerous programming languages can be employed. For instance, for SQL Server, MySQL or PostgreSQL databases, SQL language (or dialects) are used to access data. When referring to XML databases, Xpath or XQuery based queries are applied with the same objective. As for object-oriented databases, Object Query Language (OQL), which is a derivation of SQL statements, is one of the possible choices for queries, while in plain text storage databases the use of parsers is needed.

Regarding performance, some studies have already been carried out to compare database agility in a medical context. These studies concluded that XML databases perform better than standard relational databases for a relatively low volume of records. As the number of records grows, queries performance degrades even though these queries are undoubtedly easier to build in cases where document retrieval based on their element order is a requisite. In what concerns database size, XML databases require more storage as the number of records increases several times more than relational databases, particularly in an openEHR repository due to its native verbosity and the openEHR Reference Model, where the tree structure with its codes and terminology description contributes to that same verbosity [58].

So, what is the best solution to this performance problem of native XML databases versus the inability of RDBMS to query directly XML documents and other document-centric data? XML-enabled RDBMS can improve query speed by decomposing a XML document and mapping its structure into the tabular format in a way that is possible to conduct queries, but the performance increase achieved by this method is little or almost inexistent, because the computational needs to rebuild information for output are too high and the hierarchy of the XML documents is lost in this shredding operation. Storing XML data in a RDBMS as a datatype can provide document storage, but implies losing the capability of performing queries inside the document. Other possibility is to provide a kind of native XML storage, where XML data is stored “as is”, therefore maintaining hierarchy and enabling XML querying through XQuery statements, leveling the performance with native XML databases, even though a small part of RDBMS

vendors support this approach [59].
Concerning data extraction, depending on the storage mechanism and purpose of extracted data, it can be an easy or a bit more difficult task. Presuming that data is stored in a native XML database and the objective is to extract it so it can be used in a data analysis system. In this case, and because the majority of these systems use a tabular data format, direct data mapping is not possible due to information structure differences and the need for an intermediate instrument arises. Possible hypothesis are the use of web services or a tool which can connect to the database, extract data and present it in a format that can be imported to the destination system. Direct interaction with the XML database can be implemented for data analysis tools to draw conclusions about the information stored, such as XPath query in order to apply data mining algorithms, for example [60][61].

## 1.4 The OpenObsCare project

The VCObsGynCare is a clinical record software designed to be used by obstetrics and gynecology doctors, anesthesiologists, nurses and administrative staff to register inpatient admission and discharge, childbirth and newborn data, as well as surgical procedures, nursing records and gynecological interventions. It is currently in use at almost all hospitals in the north of Portugal. The version of this software that will be openEHR-based (VCOpenObsCare) is currently in development, and its repository service layer will be the initial data source for this work’s exportation tool.
OpenObsCare project aims to investigate the way the openEHR standard can be implemented in obstetrics medical records. This project studies the dynamic generation of clinical record modules, the development of forms based in clinical concepts (openEHR templates and archetypes) defined by the international openEHR community, and the way that patient data can be stored, optimizing its future reuse in, for example, clinical management and research. The project intends to have, as results, several scientific publications and at least one installation of an openEHR based obstetrics clinical record (VCOpenObsCare) in a health institution and therefore being a proof-of-concept.

**Project tasks**

The project was divided in five tasks, each one sequentially dependent from the previous.

1. **Project management** - this task aims to broadcast the results, other projects, institutions and investigators, establish connections and do project promotion through a web site,

social networks and other media channels.

2. **OpenEHR archetype and template creation** - the objective of this task is the definition of the necessary information to gather within the obstetrics health care scope, so that it is possible to automatically produce forms and data repositories in the following tasks. With this in mind, this task was divided in five sub-tasks:
    a. Current Obstetrics health record concept analysis
    b. Exploration of existing Clinical Knowledge Managers (CKM) for useful archetypes
    c. Creation of new openEHR archetypes and adaptation of the existing ones
    d. Definition of openEHR templates, given the existing and new archetypes
    e. Submission of new archetypes to the international community
3. **Template to web form conversion tool** - The conversion of the obtained templates in the previous task to HTML5 format for web use methodology definition is the goal of this task. The intention is to extend this methodology to any template in a wider form and, therefore, the task was divided in three sub-tasks:
    a. Read and export template information
    b. Define the corresponding matches between form fields and template/archetype elements
    c. Form creation
4. **Design of a Data repository** - the objective is to develop a data repository to store information obtained from the previous task developed forms. During this task, NoSQL and Object Oriented databases investigation will take place, thus giving place to the following sub-tasks:
    a. Data repository operation design preparation
    b. Application persistence layer creation under the form of services according with the openEHR standard service model specification
    c. Data repository creation
    d. Integration of the data repository with the VCOpenObsCare software for later usage in production
5. **Exporting data to standard formats** - The aim of this task is to elaborate a methodology for exporting data from the previous task obtained repository, to standard formats, enabling further form reports and data analysis through Data Mining or Business Intelligence systems. Three sub-tasks were defined to accomplish this

objective:

a. Stored openEHR objects content and clinical concepts mapping tool development
b. Export database definition
c. Comparative test elaboration

# 2 Aim

As described earlier, an openEHR repository may not store information in a tabular form. This presents a major problem when trying to analyze data in knowledge discovery systems or software. The aim of this research is to develop a methodology to extract data from an openEHR repository to standard formats, enabling further form reports and data analysis using other software. The result will be a tool that has the ability to connect to any openEHR repository-based web service and to export selected data into standard formats, such as CSV, R, SQL or SPSS, thus enabling data analysis by third parties to extract knowledge.

*This page was intentionally left blank*

# 3 Use cases and requirements

Having in mind that the purpose of this tool is to allow clinical and possibly non-clinical staff to select and export data from an openEHR repository web service, a use case diagram and requirements table was built in a way that the last project task would be defined.

## 3.1 Use case diagram

The use case diagram for the tool was modeled having in mind the necessary steps to export data from the repository in a simple manner. Figure 10 shows the interaction of the user with the system towards the data export. As it can be observed, to be able to export data from the web service, the user needs to run through six interactions to be able to finally export data:

1. **Choose web service address to connect to** – in this step, the user inserts the web service address to connect. The program will then try to confirm if the web service is online. If true, the program will move to the next step;
2. **Choose method to retrieve available templates** – the user will choose the method to retrieve the available templates from the web service methods list. The method will then be invoked;
3. **Choose template to display its' variables** – Every template has at least one variable. The user will choose the variables to export from the list and the program will retrieve their information;
4. **Choose file location** – The program will then ask the user to choose file location store the exported data;
5. **Choose export file format** – The user must choose a file format to save the exported data from four available formats;
6. **Export data** – After completing the previous steps, the user will then start the process of exporting the data to the desired format.

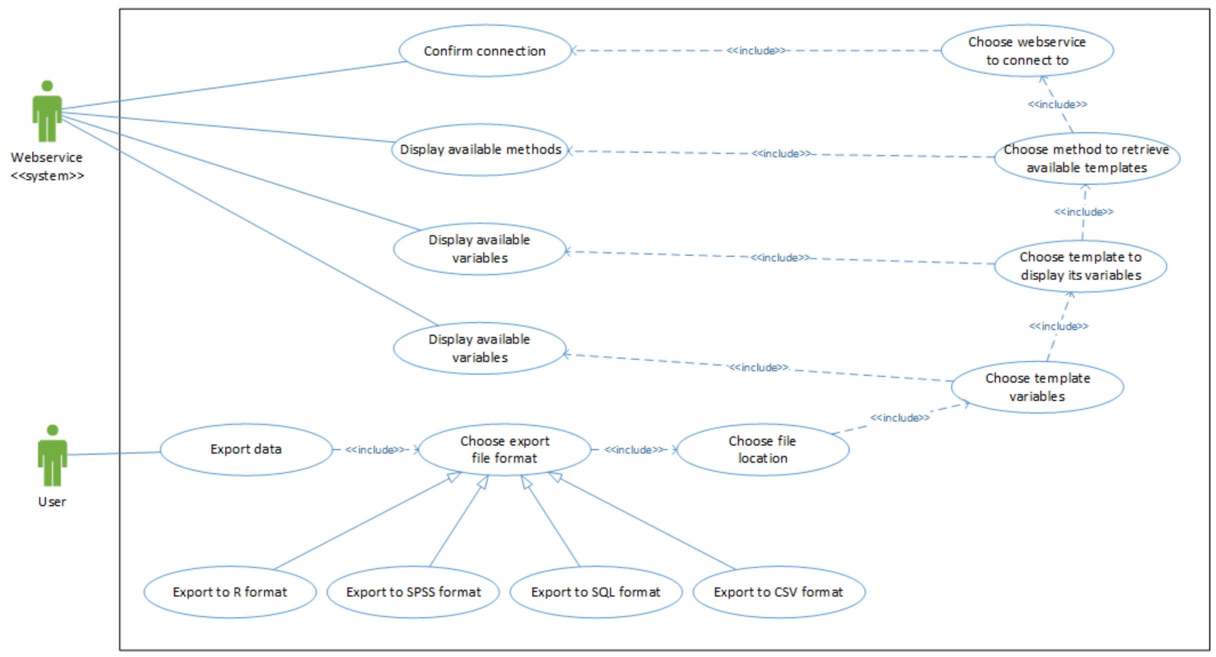

Webservice
<<system>>
User
Confirm connection
<<include>>
Choose webservice to connect to
Display available methods
Choose method to retrieve available templates
Display available variables
Choose template to display its variables
Display available variables
Choose template variables
Export data
Choose export file format
Choose file location
Export to R format
Export to SPSS format
Export to SQL format
Export to CSV format

**Figure 10 – Use case diagram for the export tool**

## 3.2 Functional and non-functional requirements

A requirement table is an important tool to define the objectives to meet in a software project development. By functional requirements we understand the behavior of any given system, while non-functional requirements describe aspects of the system that do not relate to its execution.

The obtained table of functional and non-functional requirements for the export tool is shown below (Table 4).

| *Requirement* | *Functional* | *Non-functional* |
|---|---|---|
| *OS Environment Independent* | X | |
| *Ability to connect to different web services* | X | |
| *Ability to adapt to different web service protocols* | X | |
| *Easy to be used by non-clinical staff* | | X |
| *Choose the templates to export* | X | |
| *Choose variables to export* | X | |
| *Export to standard formats* | X | |

**Table 4 – Functional and non-functional requirements for the export tool**

This table was built having in mind the need to develop a versatile software which could be easily adopted and built on a free and known platform (enabling further upgrades). Since various operating systems platforms are present in several healthcare units (sometimes even within the same healthcare unit network), it was found important that the tool could operate regardless of the running operating system. This is particularly important in countries where the percentage of open source operating systems in public and private healthcare sector is

significantly higher than the one verified in Portugal (e.g. Brazil).
There are numerous configurations and design patterns found in different web services around the world, each of them with its own peculiarities. The increasing implementation of the REST web services, due to lightweight communications over a well-known protocol par opposition to a more elaborated and slower protocol such as SOAP motivated the definition of the flexibility regarding web service types and protocols.
Even though the objective of the tool is not being used by administrative staff, situations may arise where some information must be retrieved by personnel with no clinical knowledge (e.g. unavailability of the regular operator of the tool). In this case, the tool should provide the most user-friendly interface possible to facilitate its use.
Lastly, it is necessary to assure that the tool is able to allow the user to choose the required data to be selected and exported for reporting or analysis purposes. This was the grounding basis for the definition of the last three functional requisites of the software.

*This page was intentionally left blank*

# 4 Architecture

The system's architecture follows the standard client-server architecture, with requests made by the client to a specific address being answered by the server at the desired information. As this architecture is well defined though approved and implemented standards, there was no need to use or implement different communication parameters that would only difficult the adoption of the tool as a result of the required investment by healthcare organizations.

## 4.1 Used technologies

One of the first steps towards the completion of this task was to define the programming language to develop the tool. Being one of the requirements of this tool the possibility of being used independently from the operating system in which it will be run, the choice fell on Java programming language. This would guarantee the same application behavior regardless of the system environment in which it is executed.

## 4.2 openEHR export tool communication

The communication with the EHR server is made by means of web service consumption. A REST or SOAP web service can then be found at the other end of the communication channel by the application, thus making the handling of different protocols a necessity of this application. One other aspect to keep in mind is that the method used to retrieve the available templates and their variables can have different names/URIs from web service to web service and it is essential to choose the right method, which must be validated by the application.
The server-client architecture is rather simple and straightforward as it can be perceived from Figure 11.

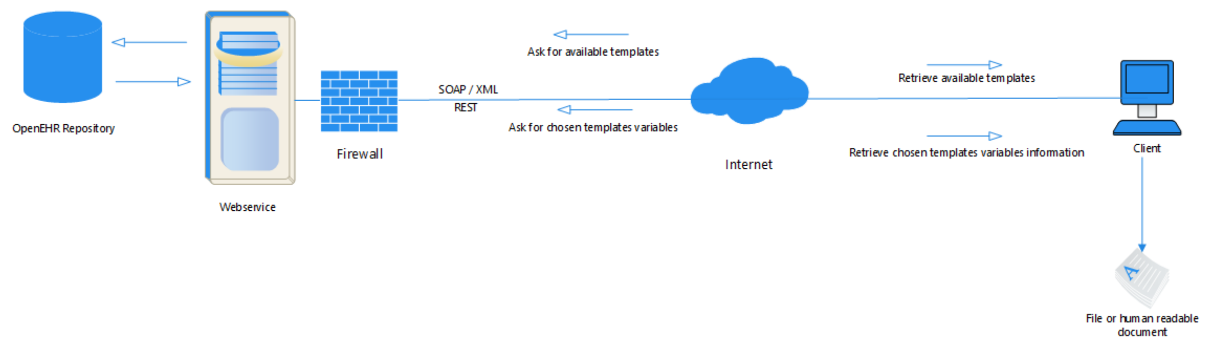

OpenEHR Repository
Webservice
Firewall
SOAP / XML
REST
Ask for available templates
Ask for chosen templates variables
Internet
Retrieve available templates
Retrieve chosen templates variables information
Client
File or human readable document

**Figure 11 – openEHR export tool server-client architecture diagram**

When the client connects to the web service, it must able to show the user every available methods so it can be possible to proceed to template retrieval and choosing. This is accomplished by using a parser for the expected XML or JSON response from the server. Again, and for every subsequent request-response action between the client and the server, the same parsing is applied in order to enable the application to handle data internally and applying the Service Oriented Architecture, which allows for vendor, product or technology independence. In this case, regardless of the database type to which the web service connects to for data retrieval, the client application may alter its features without having to change the request-response parsing already defined, thus providing user interface modification without compromising the client functionality. Without this approach, the functional request for web service type and protocol independency could not be fulfilled.

# 5 Implementation

While the interface design phase was occurring, a test web service was being built at University of Porto so it could be possible to connect and test the export tool. So, one of the first tasks was to create a mockup of the application. Creating a mockup of an application behavior aids the developer to maintain a course until the end of the project. There are two versions of the mockups and, from the first (Figure 12) to the second version (Figure 13), the only difference was at the web service address insertion screen, so it could be provided to the user the ability to re-use a web service previously inserted. Below there is the first and second versions of this screen. The complete set of mockups of each version can be found at the end of this thesis as an annex.

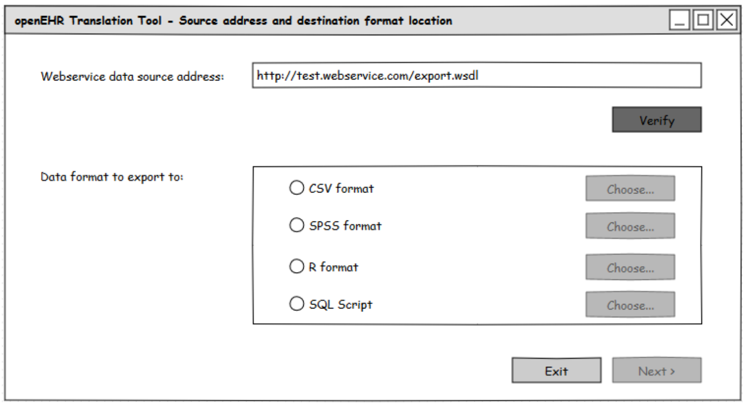

openEHR Translation Tool - Source address and destination format location
Webservice data source address:
http://test.webservice.com/export.wsdl
Verify
Data format to export to:
CSV format
SPSS format
R format
SQL Script
Choose...
Exit
Next >

Figure 12 – First version source address and destination format location screen mockup

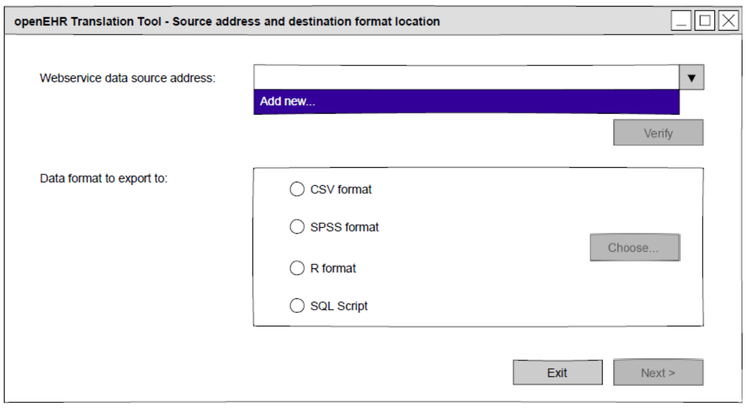
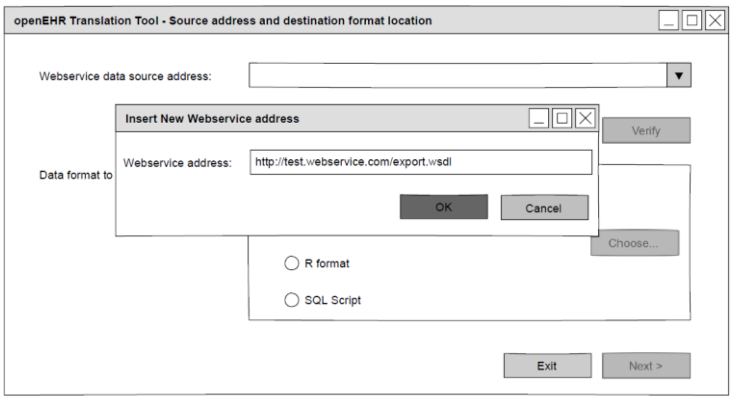

openEHR Translation Tool - Source address and destination format location
Webservice data source address:
Add new...
Verify
Data format to export to:
CSV format
SPSS format
R format
SQL Script
Choose...
Exit
Next >

Figure 13 - Second version source address and destination format location screen mockup

As it can be observed, it was decided to include the possibility to save a previously used web service address for posterior use. This could make it easier for the end user to choose from a list of previously used addresses to connect without having to ask for this information.

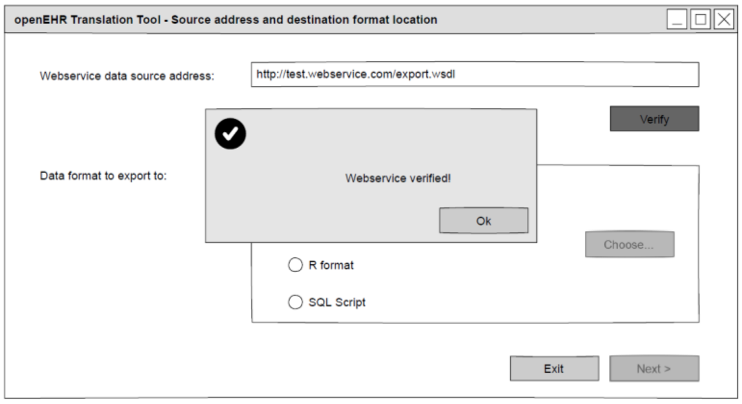
The insertion of the web service address (Figure 14) would then be validated by the software to check for its availability. If available, the address would be added to the list of previously connected web services (Figure 15). If not, the address would be discarded of that list (Figure 16).


openEHR Translation Tool - Source address and destination format location

Insert New Webservice address

Webservice address: http://test.webservice.com/export.wsdl

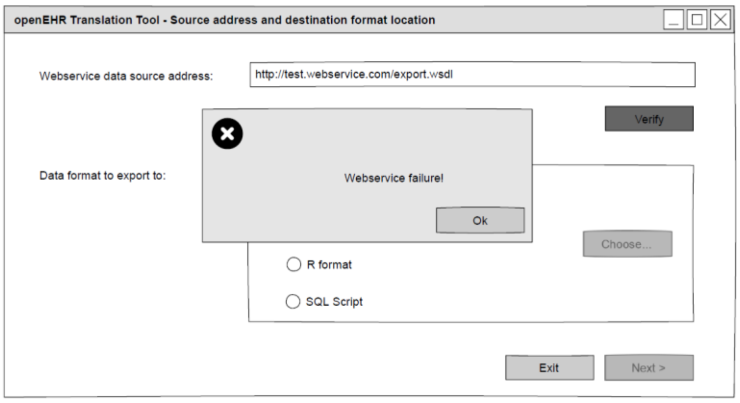
Figure 14 – Web service address insertion screen mockup


openEHR Translation Tool - Source address and destination format location

Webservice data source address: http://test.webservice.com/export.wsdl

Webservice verified!

Figure 15 – Successful web service address verification screen

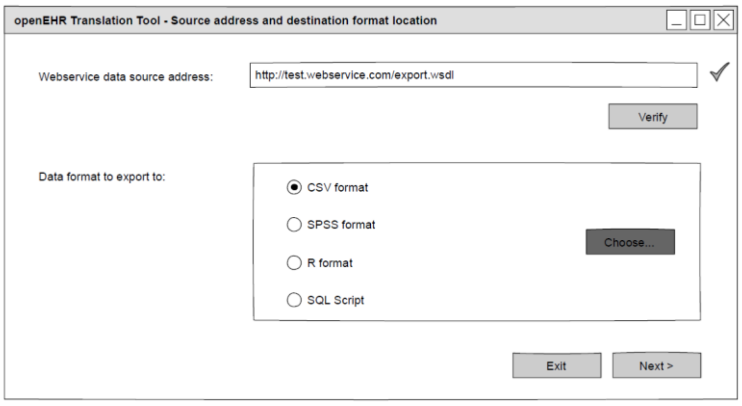

openEHR Translation Tool - Source address and destination format location

Webservice data source address: http://test.webservice.com/export.wsdl

Webservice failure!

Figure 16 – Unsuccessful web service address verification screen

On a successful verification, the next screen (Figure 17) presents the user with the option to choose the desired output format for the data. Only one format can be chosen and a screen allowing to define the location for the exported data file is presented afterwards (Figure 18).

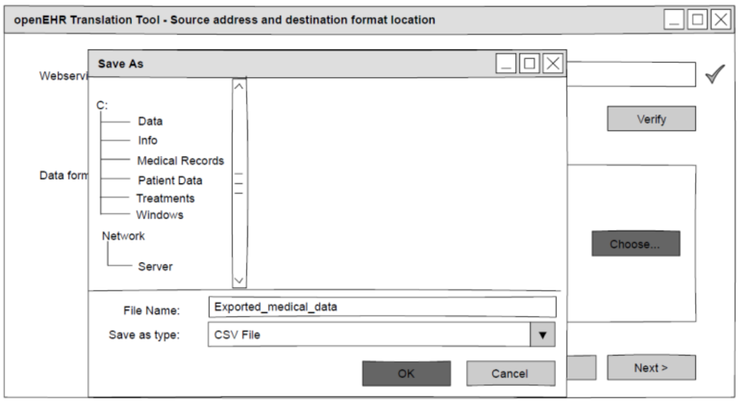

openEHR Translation Tool - Source address and destination format location

Webservice data source address: http://test.webservice.com/export.wsdl ✓

Data format to export to:

- [✓ CSV format]
- SPSS format
- R format
- SQL Script

**Figure 17 – Output file format choice screen**


openEHR Translation Tool - Source address and destination format location

Save As

C:
- Data
- Info
- Medical Records
- Patient Data
- Treatments
- Windows

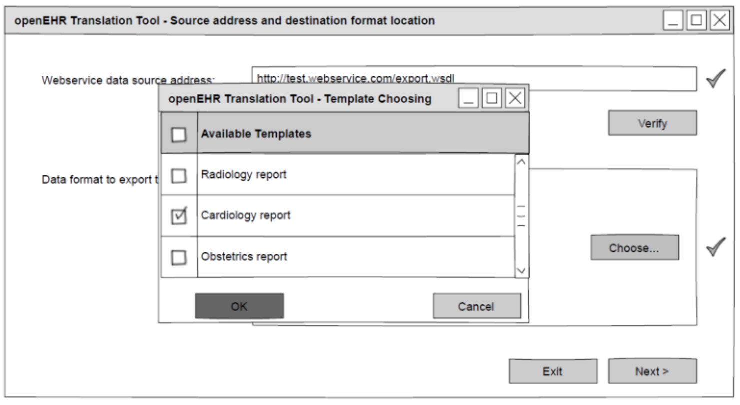
Network
- Server

File Name: Exported_medical_data

Save as type: CSV File

**Figure 18 – Output file location choice screen**

After defining where to connect and where to save exported information, the user is presented two screens: one to choose from a list of available templates the desired ones to proceed (Figure 19) and a second screen which, based on the selected templates in the previous one, presents a list of available fields (Figure 20).

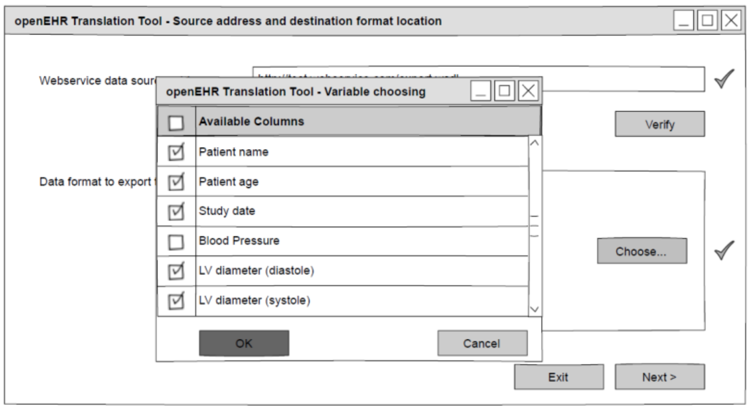

openEHR Translation Tool - Source address and destination format location

Webservice data source address: http://test.webservice.com/export.wsdl ✓

openEHR Translation Tool - Template Choosing

| ☐ | Available Templates |
|---|---|
| ☐ | Radiology report |
| ✓ | Cardiology report |
| ☐ | Obstetrics report |

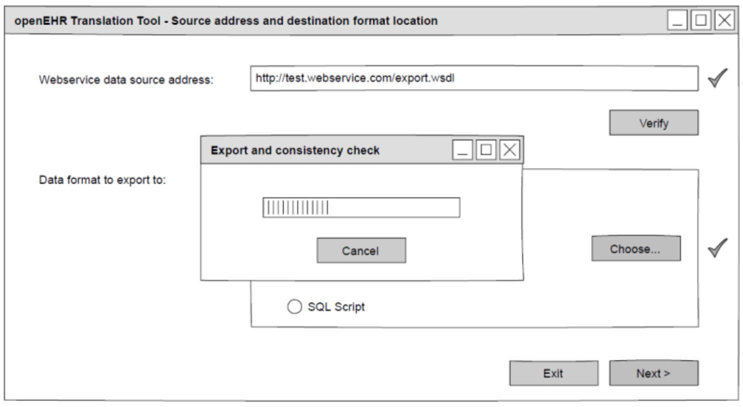
**Figure 19 – Web service available templates screen**


openEHR Translation Tool - Source address and destination format location
Webservice data sour
Data format to export t
openEHR Translation Tool - Variable choosing
Verify
Choose...
OK
Cancel
Exit
Next >

- ☐ Available Columns
- ☑ Patient name
- ☑ Patient age
- ☑ Study date
- ☐ Blood Pressure
- ☑ LV diameter (diastole)
- ☑ LV diameter (systole)

Figure 20 – Chosen template variable selection screen

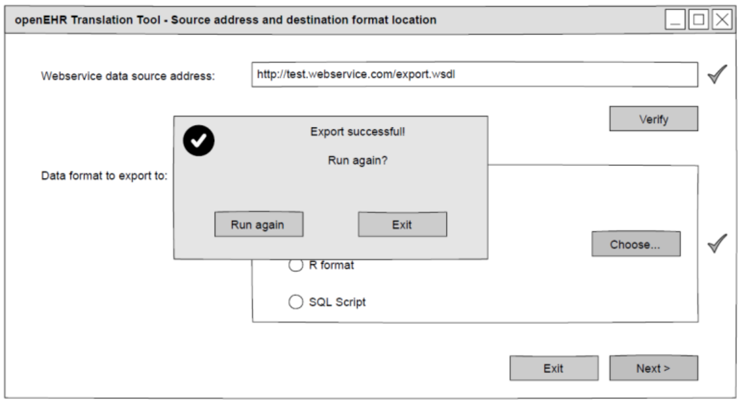
The export and consistency check task would then take place to ensure that the exported data is the same that is received from the web service (Figure 21) and an option to re-run the tool or to exit is offered to the user (Figure 22).


openEHR Translation Tool - Source address and destination format location
Verify
Export and consistency check
Cancel
Choose...
SQL Script
Exit
Next >

Webservice data source address: http://test.webservice.com/export.wsdl

Data format to export to:

Figure 21 – Export and consistency check screen


openEHR Translation Tool - Source address and destination format location
Verify
Run again
Exit
Choose...
R format
SQL Script
Exit
Next >

Webservice data source address: http://test.webservice.com/export.wsdl

Data format to export to:

Export successful!

Run again?

Figure 22 – Re-run or exit option screen

It is a known fact that the mockup screens initially drawn for an application rarely last through the whole project and this has been the case with the development of this tool. During the coding phase, several conditions forced the modifications either to the user interface or to the available functionalities of the application. These modifications are documented to justify the differences between what was idealized and what was effectively implemented.

## 5.1 Graphic user interface

The graphic user interface for the tool was implemented through the use of the CardLayout Java Swing component to build a wizard-like interface to guide the end user on the process of exporting data. Swing is the main GUI widget toolkit and is part of the Java Foundation Classes (JFC) API for Java GUI interfaces.

## 5.2 Adopted methodology

After defining use cases, system architecture, functional and non-functional requirements, the tool's mockups and graphic user interface, the next step consisted in implementing the tool's connection to the web service and determining its type. As the test server was running a SOAP web service, methods were coded to determine the online status and then to present a list of available methods so the user could start by choosing the correct one and then continue to select the desired information to export to the selected format. REST adaptability would be implemented then as a fork of the SOAP implementation branch, having its breaking point in the analysis of the web service address, but maintaining the same orientation and rejoining the main course at the moment of data export.

Retrieving information from the web service and then exporting it to CSV format after validating all information would constitute the next step. Other web services would be tested for the tool's adaptability capabilities at a later point to confirm the success of the tool's operation.

After the minimal features of the tool were achieved, the number of export formats should be increased with the addition of the other predefined three options: SQL format, SPSS format and R format.

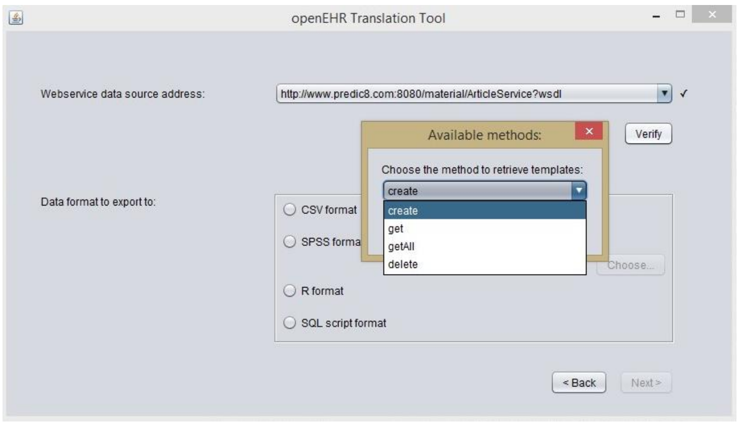
*This page was intentionally left blank*

# 6 Results

Due to technical difficulties related with the web service integration motivated by data privacy questions and previous project dependent modules scheduling, some functionalities could not be implemented in the test web service. As a result, a public web service (http://www.service-repository.com/service/overview/1875112799) with a similar mechanism was used in the development of the software, since the conclusions would be comparable. The results obtained with the tool have contributed to draw important conclusions regarding the gathering and export of data and the orientations to do so.

## 6.1 Tool operation

When implementing the SOAP connection and retrieving the available methods, the need to have enumerated a description for each one of them (with a label or a tooltip, for instance) is of major importance, since the user may be misled by the method name presented (Figure 23). This issue can be circumvented by establishing the method names for each action and certify that these names are used regardless of the web service address or developer, being unnecessary to choose the first method to invoke.

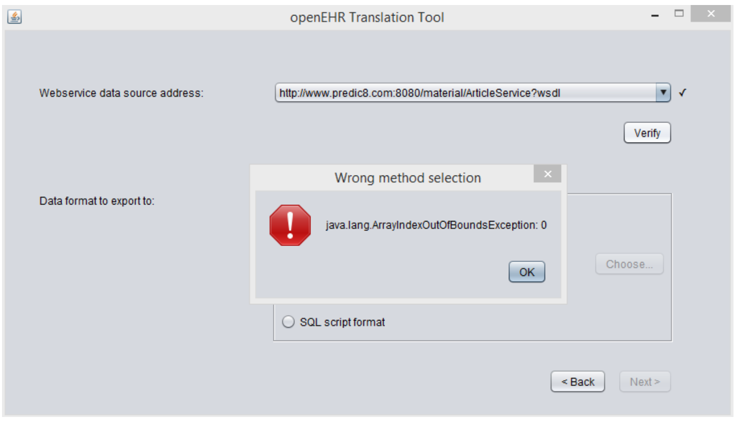


Webservice data source address: http://www.predic8.com:8080/material/ArticleService?wsdl

Choose the method to retrieve templates: create

**Figure 23 – openEHR Translation Tool retrieve templates method selection screen**

In this particular case, the available method names are suggestive and it is easy for a person with no medical or technological background to deduct the correct method to choose. Otherwise, an error message is displayed with the intention of informing the user that another method must be selected to proceed (Figure 24).



Webservice data source address: http://www.predic8.com:8080/material/ArticleService?wsdl

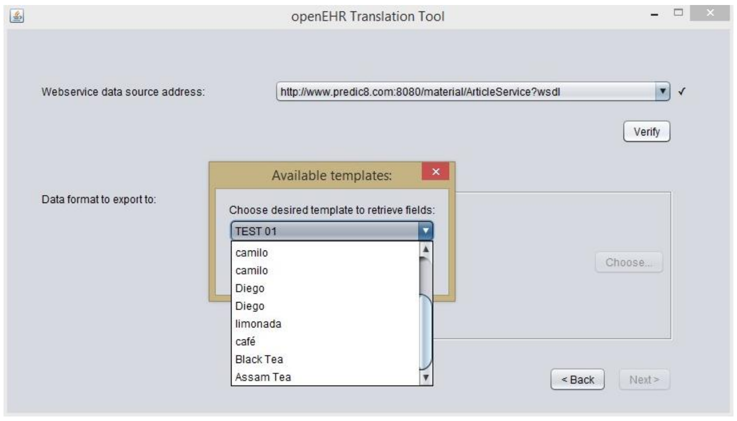
Data format to export to:

java.lang.ArrayIndexOutOfBoundsException: 0

SQL script format

**Figure 24 – Wrong method selection error message screen**

Contrary to what was initially projected, the most logical screen to be presented subsequently to the user was not the file location and format dialog. Instead, the order was altered to show the available templates option dialog (Figure 25) and then the available fields selection dialog (Figure 26).



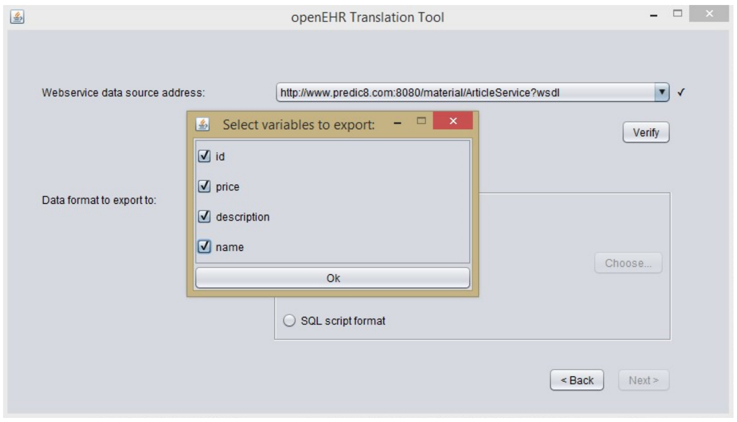
Webservice data source address: http://www.predic8.com:8080/material/ArticleService?wsdl

Data format to export to:

Choose desired template to retrieve fields: TEST 01

camilo
camilo
Diego
Diego
limonada
café
Black Tea
Assam Tea

**Figure 25 – Available templates option screen**



Webservice data source address: http://www.predic8.com:8080/material/ArticleService?wsdl

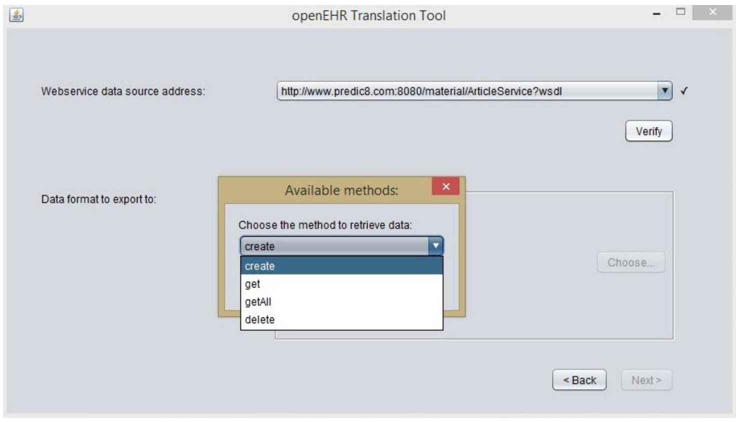
Data format to export to:

✓ id
✓ price
✓ description
✓ name

SQL script format

**Figure 26 – Available fields to export selection screen**

Between the previous two screen's related tasks, it is necessary to invoke a method to retrieve the available fields from the selected template. Again, the user must choose the right method in order to enable the application to display those fields and, as mentioned before, the user may be confused by the displayed method names (Figure 27).


openEHR Translation Tool

Webservice data source address: http://www.predic8.com:8080/material/ArticleService?wsdl ✓

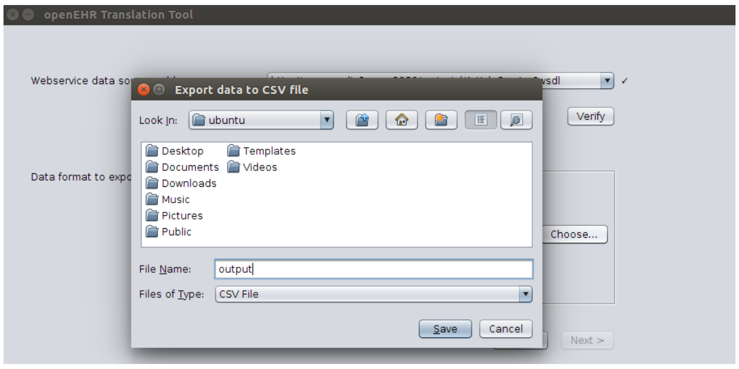
Data format to export to:

Available methods:

Choose the method to retrieve data: create

Figure 27 - openEHR Translation Tool retrieve variables method selection screen

After receiving the variable data associated with the chosen template, the application unlocks the file format options, enabling the user to select the location for the file in the desired format. This screen can be observed in Figure 28, now in a Linux environment.


openEHR Translation Tool

Export data to CSV file

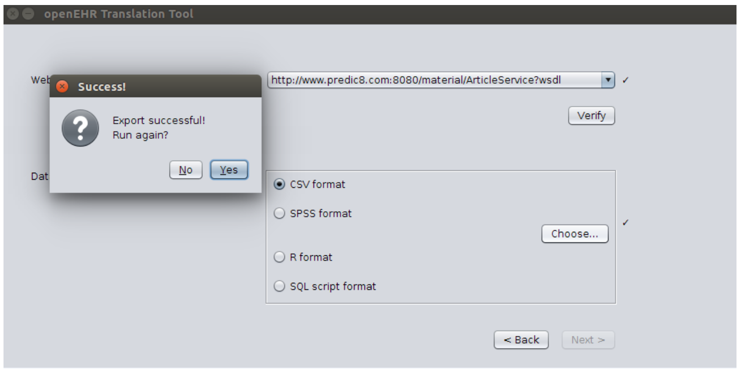
Look In: ubuntu

File Name: output

Files of Type: CSV File

Figure 28 – File save dialog screen (Linux environment)

The last screen of the application questions the user about running the tool again or not and informs regarding the status of the operation (Figure 29).


openEHR Translation Tool

http://www.predic8.com:8080/material/ArticleService?wsdl ✓

Success!

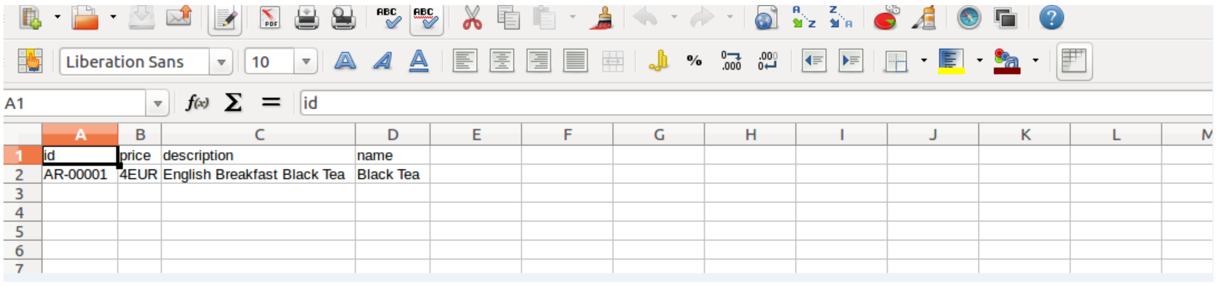
Export successful!
Run again?

● CSV format
○ SPSS format
○ R format
○ SQL script format

Figure 29 – Successful export operation screen (Linux environment)

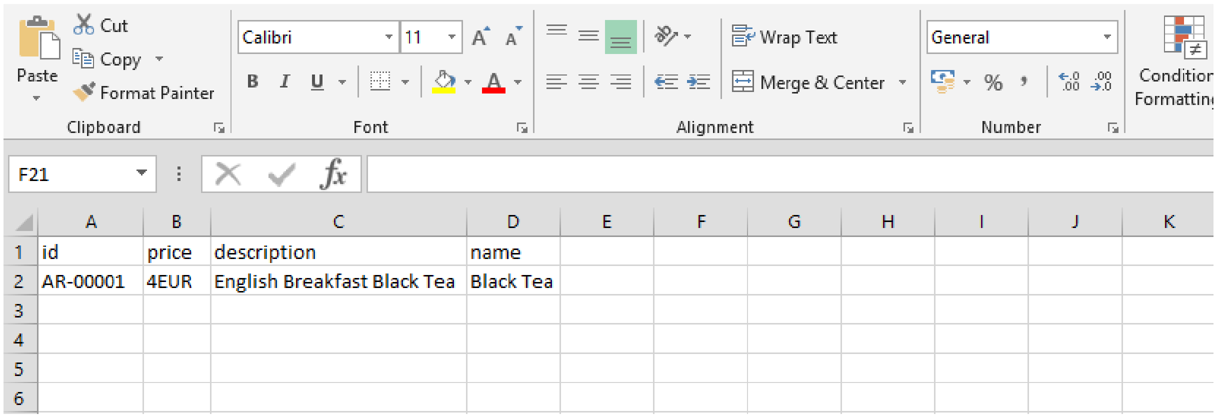
## 6.2 Tool exported data use

The csv file generated by the tool can then be opened in any software that can read this extension file type (Figure 30, Figure 31 and Figure 32). For the common user, the most used applications are Microsoft Excel or OpenOffice Calc, while for most specific objectives, namely data analysis or data mining projects, application like Rapid Miner, Weka or Talend Open Studio are more popular.



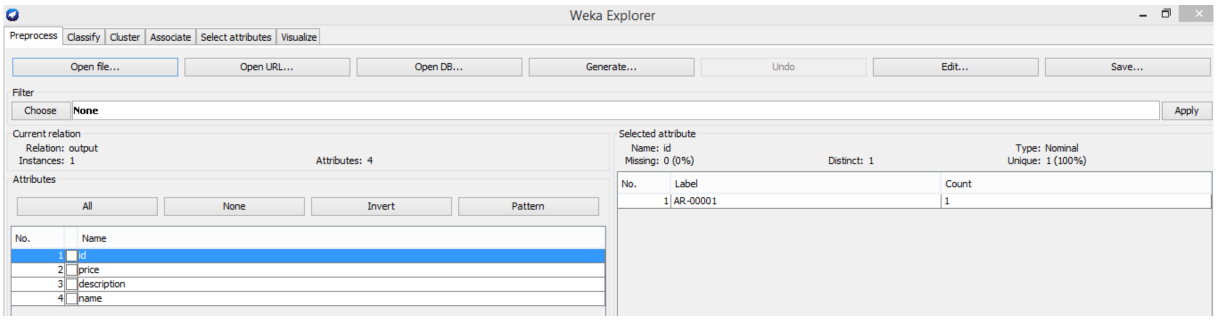
| | A | B | C | D |
|---|---|---|---|---|
| 1 | id | price | description | name |
| 2 | AR-00001 | 4EUR | English Breakfast Black Tea | Black Tea |

Figure 30 – File import in OpenOffice Calc software



| | A | B | C | D |
|---|---|---|---|---|
| 1 | id | price | description | name |
| 2 | AR-00001 | 4EUR | English Breakfast Black Tea | Black Tea |

Figure 31 - File import in Microsoft Excel software



Filter: None

Current relation
Relation: output
Instances: 1
Attributes: 4

Selected attribute
Name: id
Missing: 0 (0%)
Distinct: 1
Type: Nominal
Unique: 1 (100%)

| No. | Label | Count |
|---|---|---|
| 1 | AR-00001 | 1 |

Attributes

| No. | Name |
|---|---|
| 1 | id |
| 2 | price |
| 3 | description |
| 4 | name |

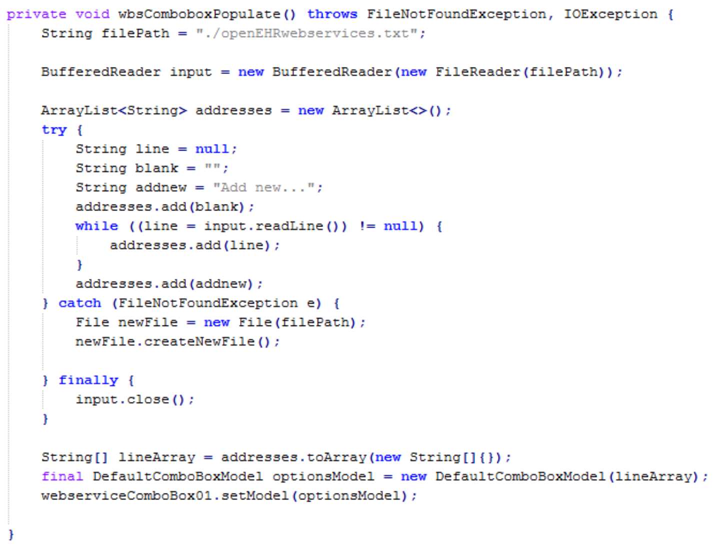
Figure 32 - File import in Weka software

The results demonstrate that data exported from an openEHR repository to csv format can be imported in other systems or applications, allowing it further analysis. Even though a public web service was used to develop the tool, its mechanism is similar to the defined openEHR web service system architecture and is still valid for conclusion inference.

## 6.3 Tool application code

Some tool operations code implemented are presented below. The complete project will be delivered with this document and can be tested against the specified web service.

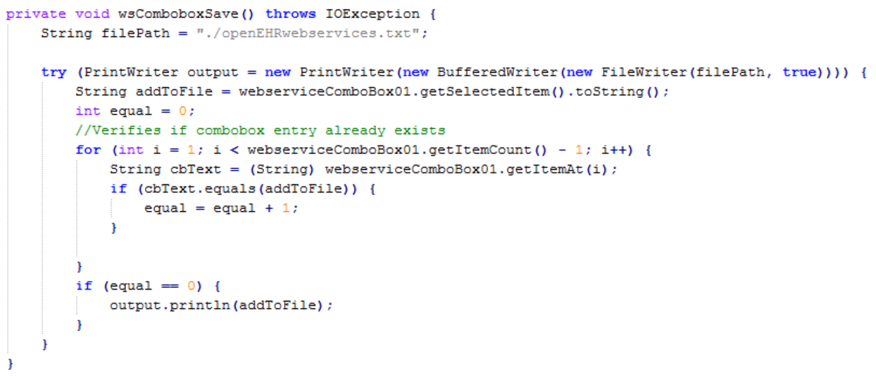
Regarding the first user input screen, the application loads a previous created file with accessed web service addresses for selection or, in case that the file is not found, it creates a new one (Figure 33).



```java
private void wbsComboboxPopulate() throws FileNotFoundException, IOException {
    String filePath = "./openEHRwebservices.txt";

    BufferedReader input = new BufferedReader(new FileReader(filePath));

    ArrayList<String> addresses = new ArrayList<>();
    try {
        String line = null;
        String blank = "";
        String addnew = "Add new...";
        addresses.add(blank);
        while ((line = input.readLine()) != null) {
            addresses.add(line);
        }
        addresses.add(addnew);
    } catch (FileNotFoundException e) {
        File newFile = new File(filePath);
        newFile.createNewFile();

    } finally {
        input.close();
    }

    String[] lineArray = addresses.toArray(new String[]{});
    final DefaultComboBoxModel optionsModel = new DefaultComboBoxModel(lineArray);
    webserviceComboBox01.setModel(optionsModel);

}
```

Figure 33 – Destination web service combo box population function code

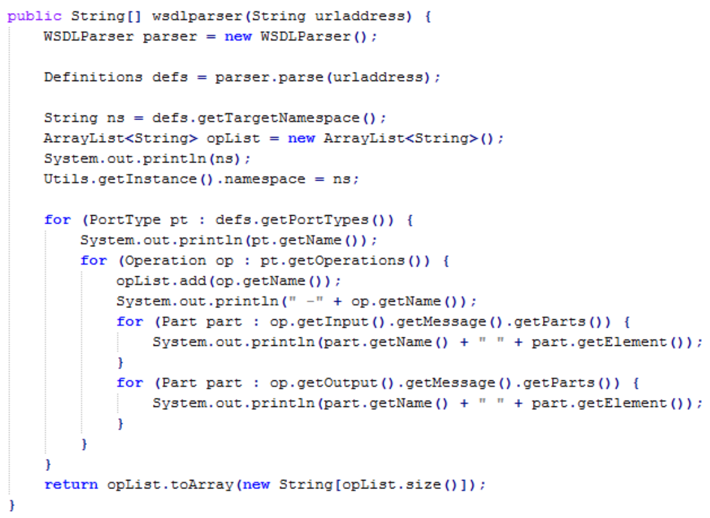
After verifying for successful web service connection the tool then adds the inserted address to the created or existing file. In case that address already exists, it is not added (Figure 34).



```java
private void wsComboboxSave() throws IOException {
    String filePath = "./openEHRwebservices.txt";

    try (PrintWriter output = new PrintWriter(new BufferedWriter(new FileWriter(filePath, true)))) {
        String addToFile = webserviceComboBox01.getSelectedItem().toString();
        int equal = 0;
        //Verifies if combobox entry already exists
        for (int i = 1; i < webserviceComboBox01.getItemCount() - 1; i++) {
            String cbText = (String) webserviceComboBox01.getItemAt(i);
            if (cbText.equals(addToFile)) {
                equal = equal + 1;
            }

        }
        if (equal == 0) {
            output.println(addToFile);
        }
    }
}
```

Figure 34 - Destination web service address save function code

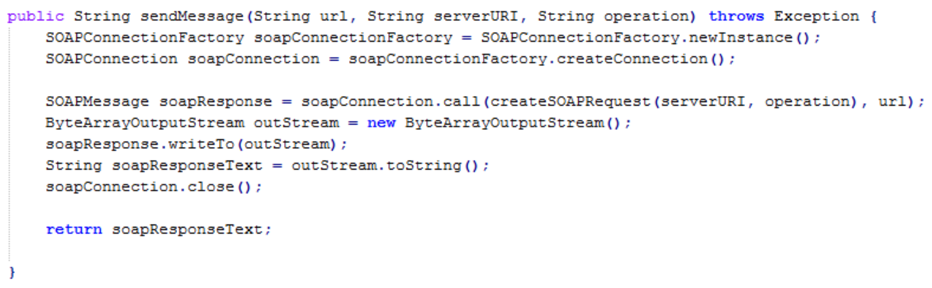
Available methods at the given web service address are parsed with the next function and returned to the combo box for user selection. Iteration throughout all XML nodes is performed by a WSDL parser (Figure 35).



```java
public String[] wsdlparser(String urladdress) {
    WSDLParser parser = new WSDLParser();

    Definitions defs = parser.parse(urladdress);

    String ns = defs.getTargetNamespace();
    ArrayList<String> opList = new ArrayList<String>();
    System.out.println(ns);
    Utils.getInstance().namespace = ns;

    for (PortType pt : defs.getPortTypes()) {
        System.out.println(pt.getName());
        for (Operation op : pt.getOperations()) {
            opList.add(op.getName());
            System.out.println(" -" + op.getName());
            for (Part part : op.getInput().getMessage().getParts()) {
                System.out.println(part.getName() + " " + part.getElement());
            }
            for (Part part : op.getOutput().getMessage().getParts()) {
                System.out.println(part.getName() + " " + part.getElement());
            }
        }
    }
    return opList.toArray(new String[opList.size()]);
}
```

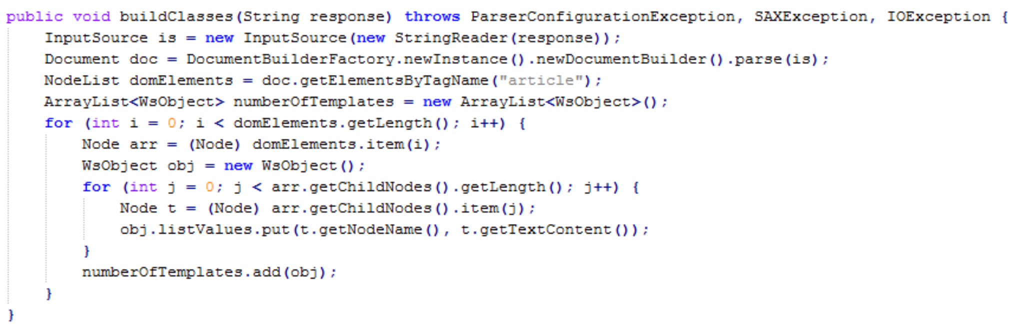
**Figure 35 – Retrieve available web service methods function**

After invoking the method to retrieve all templates, the tool receives the response (Figure 36) and forwards it to a XML parser which enumerates all available templates (Figure 37). In this particular case, it was needed to adapt the tool to the web service used.



```java
public String sendMessage(String url, String serverURI, String operation) throws Exception {
    SOAPConnectionFactory soapConnectionFactory = SOAPConnectionFactory.newInstance();
    SOAPConnection soapConnection = soapConnectionFactory.createConnection();

    SOAPMessage soapResponse = soapConnection.call(createSOAPRequest(serverURI, operation), url);
    ByteArrayOutputStream outStream = new ByteArrayOutputStream();
    soapResponse.writeTo(outStream);
    String soapResponseText = outStream.toString();
    soapConnection.close();

    return soapResponseText;

}
```

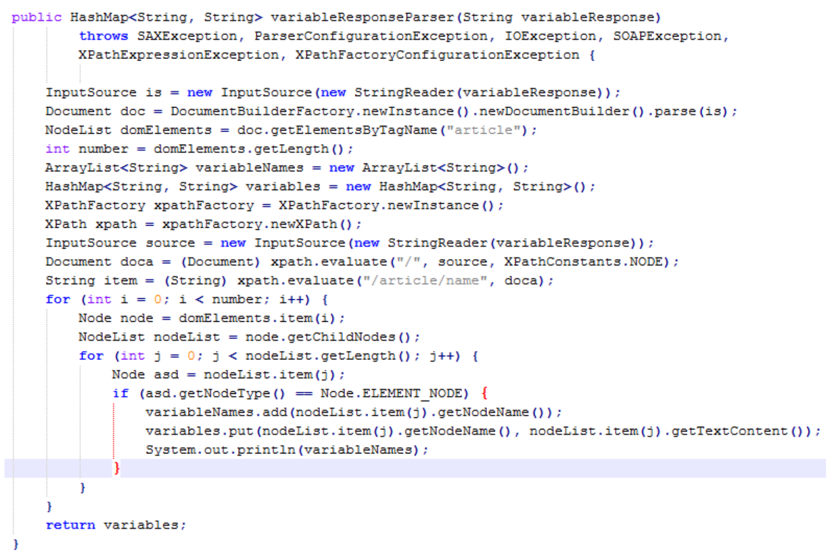
**Figure 36 – Web service SOAP response handling function**



```java
public void buildClasses(String response) throws ParserConfigurationException, SAXException, IOException {
    InputSource is = new InputSource(new StringReader(response));
    Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(is);
    NodeList domElements = doc.getElementsByTagName("article");
    ArrayList<WsObject> numberOfTemplates = new ArrayList<WsObject>();
    for (int i = 0; i < domElements.getLength(); i++) {
        Node arr = (Node) domElements.item(i);
        WsObject obj = new WsObject();
        for (int j = 0; j < arr.getChildNodes().getLength(); j++) {
            Node t = (Node) arr.getChildNodes().item(j);
            obj.listValues.put(t.getNodeName(), t.getTextContent());
        }
        numberOfTemplates.add(obj);
    }
}
```

**Figure 37 – Retrieving template names function**

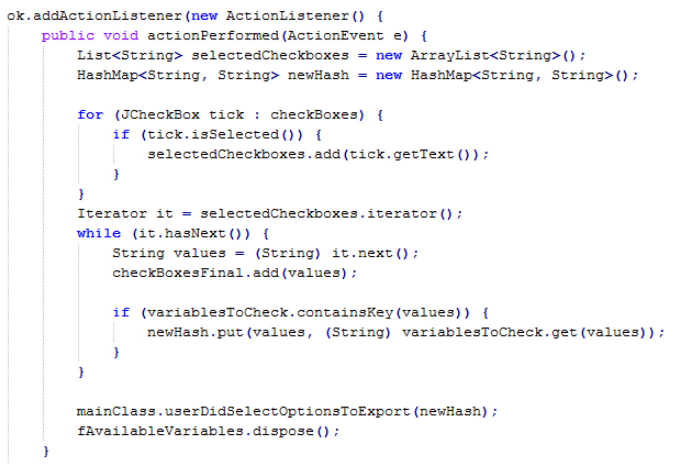
The user then is presented with the template names list to choose and following to that, another screen to choose the method to invoke. After that, and based on the template name and id saved in a background process, the application invokes the method selected by the user to present the available variables to pick in a check box screen. The code for retrieving variable names is presented in Figure 38.



```java
public HashMap<String, String> variableResponseParser(String variableResponse)
        throws SAXException, ParserConfigurationException, IOException, SOAPException,
        XPathExpressionException, XPathFactoryConfigurationException {

    InputSource is = new InputSource(new StringReader(variableResponse));
    Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(is);
    NodeList domElements = doc.getElementsByTagName("article");
    int number = domElements.getLength();
    ArrayList<String> variableNames = new ArrayList<String>();
    HashMap<String, String> variables = new HashMap<String, String>();
    XPathFactory xpathFactory = XPathFactory.newInstance();
    XPath xpath = xpathFactory.newXPath();
    InputSource source = new InputSource(new StringReader(variableResponse));
    Document doca = (Document) xpath.evaluate("/", source, XPathConstants.NODE);
    String item = (String) xpath.evaluate("/article/name", doca);
    for (int i = 0; i < number; i++) {
        Node node = domElements.item(i);
        NodeList nodeList = node.getChildNodes();
        for (int j = 0; j < nodeList.getLength(); j++) {
            Node asd = nodeList.item(j);
            if (asd.getNodeType() == Node.ELEMENT_NODE) {
                variableNames.add(nodeList.item(j).getNodeName());
                variables.put(nodeList.item(j).getNodeName(), nodeList.item(j).getTextContent());
                System.out.println(variableNames);
            }
        }
    }
    return variables;
}
```

**Figure 38 – Retrieve variable names function**

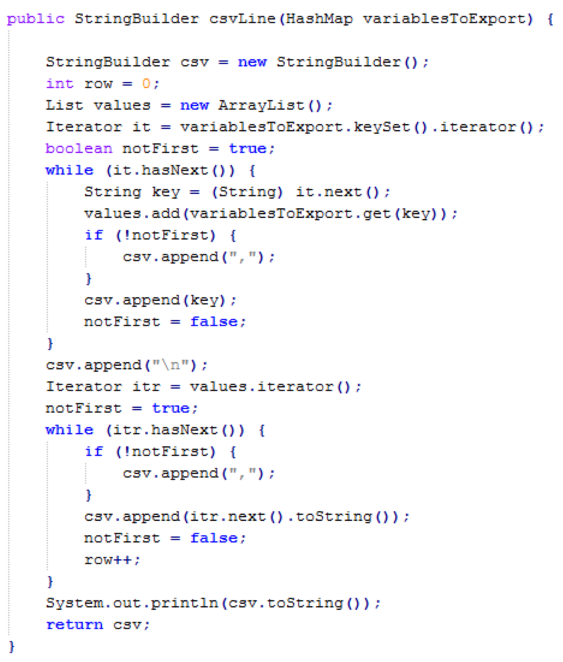
Since the full variable data has been stored from the moment of its retrieval, the next function iterates through it to match the selected checkboxes with the values stored (Figure 39).



```java
ok.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        List<String> selectedCheckboxes = new ArrayList<String>();
        HashMap<String, String> newHash = new HashMap<String, String>();

        for (JCheckBox tick : checkBoxes) {
            if (tick.isSelected()) {
                selectedCheckboxes.add(tick.getText());
            }
        }
        Iterator it = selectedCheckboxes.iterator();
        while (it.hasNext()) {
            String values = (String) it.next();
            checkBoxesFinal.add(values);

            if (variablesToCheck.containsKey(values)) {
                newHash.put(values, (String) variablesToCheck.get(values));
            }
        }

        mainClass.userDidSelectOptionsToExport(newHash);
        fAvailableVariables.dispose();
    }
```

**Figure 39 – Retrieve variable values function**

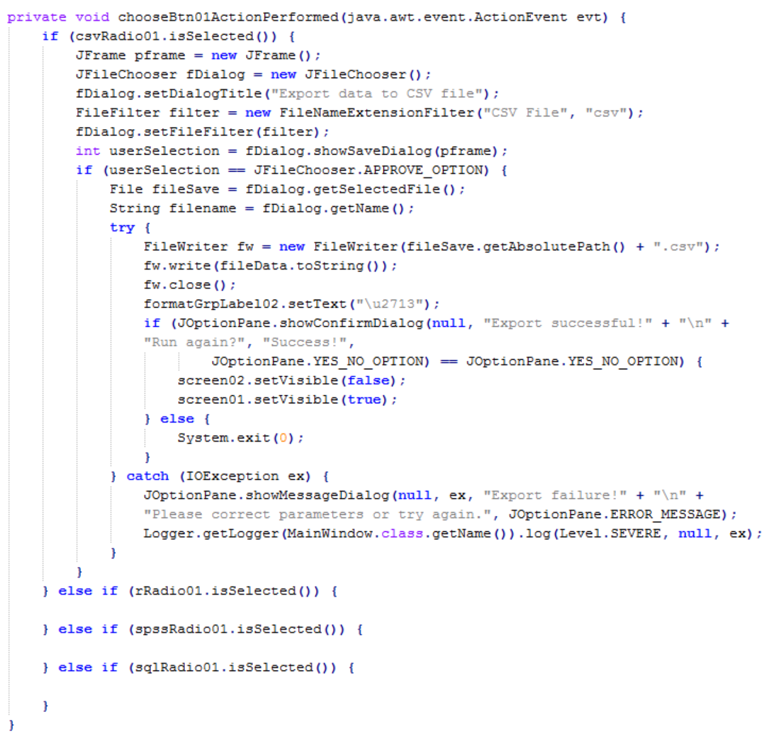
Variable names and values are then sent to be treated in a function that builds the csv structure to be written in a file. The code for that function is presented in the next image (Figure 40).



```java
public StringBuilder csvLine(HashMap variablesToExport) {

    StringBuilder csv = new StringBuilder();
    int row = 0;
    List values = new ArrayList();
    Iterator it = variablesToExport.keySet().iterator();
    boolean notFirst = true;
    while (it.hasNext()) {
        String key = (String) it.next();
        values.add(variablesToExport.get(key));
        if (!notFirst) {
            csv.append(",");
        }
        csv.append(key);
        notFirst = false;
    }
    csv.append("\n");
    Iterator itr = values.iterator();
    notFirst = true;
    while (itr.hasNext()) {
        if (!notFirst) {
            csv.append(",");
        }
        csv.append(itr.next().toString());
        notFirst = false;
        row++;
    }
    System.out.println(csv.toString());
    return csv;
}
```

**Figure 40 – CSV structure creation function**

Finally, and after the user has selected the output file format, the “Choose” button is pressed and a file dialog for file saving is displayed. The file creation code for exporting data afterwards is displayed below (Figure 41).



```java
private void chooseBtn01ActionPerformed(java.awt.event.ActionEvent evt) {
    if (csvRadio01.isSelected()) {
        JFrame pframe = new JFrame();
        JFileChooser fDialog = new JFileChooser();
        fDialog.setDialogTitle("Export data to CSV file");
        FileFilter filter = new FileNameExtensionFilter("CSV File", "csv");
        fDialog.setFileFilter(filter);
        int userSelection = fDialog.showSaveDialog(pframe);
        if (userSelection == JFileChooser.APPROVE_OPTION) {
            File fileSave = fDialog.getSelectedFile();
            String filename = fDialog.getName();
            try {
                FileWriter fw = new FileWriter(fileSave.getAbsolutePath() + ".csv");
                fw.write(fileData.toString());
                fw.close();
                formatGrpLabel02.setText("\u2713");
                if (JOptionPane.showConfirmDialog(null, "Export successful!" + "\n" +
                "Run again?", "Success!",
                        JOptionPane.YES_NO_OPTION) == JOptionPane.YES_NO_OPTION) {
                    screen02.setVisible(false);
                    screen01.setVisible(true);
                } else {
                    System.exit(0);
                }
            } catch (IOException ex) {
                JOptionPane.showMessageDialog(null, ex, "Export failure!" + "\n" +
                "Please correct parameters or try again.", JOptionPane.ERROR_MESSAGE);
                Logger.getLogger(MainWindow.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    } else if (rRadio01.isSelected()) {

    } else if (spssRadio01.isSelected()) {

    } else if (sqlRadio01.isSelected()) {

    }
}
```

**Figure 41 – CSV data output file save function**

As it can be observed in Figure 41, the code is prepared to be completed for other file formats export. However, and according the defined methodology, this part is referred to future work.

## 6.4 Comparative tests

No comparative tests were performed, since no other comparable web service could be found for testing, neither the initial web service could be timely implemented. Additionally, no comparable REST web service could be found to measure the application performance between SOAP and REST connections.

## 6.5 Results summary

Several results were obtained with this work. Table 5 shows the name and description for each obtained item.

| Name | Description |
|---|---|
| *Reading revision* | Previous investigation and work performed under the scope of the project |
| *Journal article* | Article submitted to HCist conference in the scope of the project |
| *Use case diagram* | User interaction with the system representation with relationship description |
| *Functional and non-functional requirements* | List of functionalities defining what a system is supposed to accomplish and how it is supposed to be |
| *System architecture* | System functionality representation |
| *Mockup Screens* | Prototype of the user interface during the use case course with all predictable outcomes. |
| *openEHR Translation Tool software* | openEHR Translation tool executable file |
| *openEHR Translation Tool source code* | openEHR Translation tool editable code for compilation |
| *Output file tests* | Generated file import in specific software |

**Table 5 – Project obtained results**

In the course of this work, the reading revision contributed to the delimitation of the investigation problem, analysis of previous investigations, deviation from erroneous investigation lines, gain of methodological perspectives and to identify further investigation recommendations.

Elaboration of the use case diagram and requisites analysis permitted to build prototypes of the user interface which established guidelines for the tool implementation.

openEHR Translation Tool software and source code allowed the gathering of further enhancements of the software itself as well as the enumeration of necessary conditions to achieve the initially proposed objectives that could not be satisfied.

Finally, the application generated output file tests substantiated the importance of this project to the growing openEHR community and implementation of the standard worldwide.

*This page was intentionally left blank*

# 7 Discussion

As it may seemed a relatively simple web service client implementation project in the beginning, it has revealed itself a reasonably complex task. The main reason for this was the fact that the tool should be flexible enough to connect to any web service despite of its location and type, but no other implementations of this kind are documented. Usually, web service clients are developed to connect to a specific web service and having full knowledge of its implementation. In this project, the web service type adaptability could not be fully implemented due to the fact that no defined base URI for REST web services exists in any specification. Even though some companies like IBM support WSDL in REST web services, others, like Oracle, promote the use of WADL. Either of these specifications can be used to describe a REST web service to the client, but only WSDL can be used in SOAP web services. Since there's no consensus or orientation in this matter, the result is a miscellaneous of available REST web services implemented worldwide with no rule and with little or no documentation other than internal to the developer of the client application.

The need to endow the tool with a high level of dynamic adaptability has greatly increased the software's development process difficulty, due to the necessary research for mechanisms that could allocate the different web service descriptions found at the endpoint of the web service. However, research has come to a point where the level of abstraction needed for the tool development needed to achieve its proposed objectives was too high for the length of this work.

## 7.1 General objective

The general objective of this work was to develop a methodology to export data from an openEHR repository to standard formats, thus enabling data analysis and data mining on specific systems which are based on tabular format through the development of a software tool. The tool should contribute to the increase of systems integration level and address the lack of tools within the standard.

This objective was achieved, since the tool is able to export valid data which can successfully be imported by both specific and general software towards the most various objectives, as demonstrated in the performed tests.

## 7.2 Specific objectives

Specific objectives defined for the project were related to the application's capabilities, namely the operating system independency, non-clinical staff usability, the ability to connect to different types of web services using different protocols and permitting the export of data from an openEHR repository. The obtained software satisfies the operating system independency requisite and is able to export retrieved data to one of the four proposed standard data formats. However, even though the tool is simple enough to be used by non-medical staff, the web service type flexibility feature could not be implemented, although orientations and suggestions are made towards it.

## 7.3 Limitations

Some limitations are identified related with the application development:

- The need for the user to choose the correct methods to invoke during the process for the data export can occur;
- Only one template can be exported at a time at this point;
- REST web service adaptability was not implemented due to insufficient information or test server to implement;
- Only the CSV file export format function is functional, due to lack of free Java libraries which can be used for the other three formats (R, SPSS and SQL).

## 7.4 Future work

From the previously mentioned limitations and other identified problems during the project course, several improvements and suggestions are following outlined.

Hence, the identified software point of failure related with the need for the user to choose the right method to retrieve the available templates and can be solved with the definition of the method names to be invoked by the application. The same happens with the REST implementation: if technologies and endpoints are defined, it is possible to integrate the REST communication functionality and greatly improving the software utility.

Regarding the limitation of the number of templates that can be exported at a time, this can be solved by using an ArrayList of HashMaps or using other storing structure that can deal with the levels of possible stored information retrieved (e.g. one single episode for two separated events, like the birth of twins).

Other identified problem was the lack of free libraries to enable the export of data to the other three proposed formats. There was found a free library (xporter1.6.84.jar), but it's dated from

2009 and it is strongly possible that some incompatibilities with newer SPSS versions are detected, since it was acquired by IBM. Commercial libraries were also found and disregarded by falling out of the requisites for the application. No libraries (commercial or not) for SQL and R file types were found, which uncovers the need for its development for easier integration with the tool.

The last issue relates with the communication security. Either in REST or SOAP protocols, security can be implemented to ensure data integrity and confidentiality. The use of the HTTPS protocol for REST and the WS-Security extension for SOAP web services can increase data privacy over communication in public networks.

## 7.5 Conclusion

Medical documentation comprises several kinds of information reflecting different objectives and uses. According to the objective of data, diverse methods are employed.

The general idea is that medical information relates only to patient medical history, diagnosis, therapies or symptoms. However, caregiving/billing documentation or epidemiological databases/reports also contributes to medical documentation, even though it will be use in a more financial aspect of healthcare.

Since storing every piece of information regarding patient diagnosis, discharge summaries, laboratory results, billing information and others would result in an enormous need of physical space for its storage in its paper form and its quick and accurate access would be near impossible. Privacy of patient data is important and must be one of the first (if not the first!) concerns of legislators when regulating technological approaches on this matter, but it is also important to assure quick and easy access to information and balance is not always maintained.

Modern medicine relies not only on the skills and knowledge of health professionals but also on new technologies to enable communication between them and professionals from other related areas such as hospital and governmental administrations, quality management, education and clinical research. Obtained data from medical encounters, procedures and their outcomes can then be analyzed from various perspectives and, recurring to several methodologies or techniques, improvements can be made so that the decision making process can be more efficient and effective. However, given the myriad of existing and emerging systems, it is a fact that data storage technologies do not always allow an easy and straightforward integration with standard data analysis systems. Therefore, the need for tools which can promote systems integration arises.

This work is part of the OpenObsCare project, which aims to investigate the way the openEHR standard can be implemented in obstetrics medical records and also the result of the investigation regarding applying general data analysis methods and technologies to the healthcare domain along with the development of a methodology to enable exporting data from an openEHR repository to standard formats. The results demonstrate that data export from an openEHR repository to standard formats is feasible, but require the establishment of a protocol between developers, organizations and stakeholders regarding communication parameters standardization to assure the correct interaction between the web services and the software and to enable the performance and functionality of the tool, namely the application of already existing standards and specifications.

# 8 References

[1] Shorliffe, E.H., & Cimino, J.J. (2005). Biomedical Informatics: Computer Applications in Health Care and Biomedicine (3rd ed.).

[2] Hersh, W.R. (2002). Medical informatics: Improving health care through information. JAMA, 288:1955-1958.

[3] Embi, P.J., & Payne, P.R. (2009). Clinical research informatics: challenges, opportunities and definition for an emerging domain. J Am Med Inform Assoc., 16:316-327

[4] AHRQ outcomes research fact sheet. Available at: http://www.ahrq.gov/clinic/outfact.htm. Accessed March 30, 2014.

[5] Taylor, P. (2006). From Patient Data to Medical Knowledge - The Principles and Practice of Health Informatics.

[6] Bates, D.W., Ebell, M., Gotlieb, E., Zapp, J., & Mullins, H.C. (2003). A proposal for electronic medical records in U.S. primary care. J Am Med Inform Assoc., 10:1-10.

[7] Milovic, B., & Milovic, M. (2012). Prediction and Decision Making in Health Care using Data Mining. International Journal of Public Health Science.

[8] Kawamoto, K., Houlihan, A.C., Balas, A.E., & Lobach, F.D. (2005). Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success. BMJ.

[9] Kaushal, R., Shojania, G.K., & Bates, W.D. (2003). Effects of computerized physician order entry and clinical decision support systems on medication safety - A systematic review. Arch Intern Med., 163:1409-1416.

[10] Sahama, R.T., & Croll, R.P. (2007). A Data Warehouse Architecture for Clinical Data Warehousing. Conferences in Research and Practice in Information Technology, 68:227-232.

[11] Ledbetter, S.C., & Morgan, W.M. (2001). Toward Best Practice: Leveraging the Electronic Patient Record as a Clinical Data Warehouse. Journal Of Healthcare Information Management, 15:119-131.

[12] Vassiliadis, P., Vagena, Z., Skiadopoulos, S., Karayannidis, N., & Sellis, T. (2001). Arktos: towards the modeling, design, control and execution of ETL processes. Information Systems, 26:537-561.

[13] Buechi, M., Borthwick, A., Winkel, A., & Goldberg, A. (2003). ClueMaker: A Language

for Approximate Record Matching. Proceedings of the Eighth International Conference on Information Quality (ICIQ-03), 207-223.

[14] Batini, C., & Scannapieco, M. (2006). Data Quality: Concepts, Methodologies and Techniques.

[15] Institute of Electrical and Electronics Engineers, http://www.ieee.org/index.html

[16] European Patients Smart Open Services, http://www.epsos.eu/home.html

[17] Healthcare Information and Management Systems Society. (2014). Strategic Interoperability in Germany, Spain & the UK - The Clinical and Business Imperative for Healthcare Organisations.

[18] Garde, S., Knaup, P., Hovenga, E.J.S., & Heard, S. (2007). Towards Semantic Interoperability for Electronic Health Records - Domain Knowledge Governance for openEHR Archetypes. Methods Inf Med., 46(3):332-343.

[19] Sauter, V.L. (2011). Decision Support Systems for Business Intelligence (2nd Ed.).

[20] Kaushal, R., Shojania, K.G., & Bates, D.W. (2003). Effects of Computerized Physician Order Entry and Clinical Decision Support Systems on Medication Safety - A Systematic Review. Arch Intern Med., 163:1409-1416.

[21] Koppel, R., Metlay, J.P., Cohen, A., Abaluck, B., Localio, A.R., Kimmel, S.E., & Strom, B.L. (2005). Role of Computerized Physician Order Entry Systems in Facilitating Medication Errors. JAMA, 293:1197-1203.

[22] Inmon, W.H. (2005). Building the Data Warehouse (4th Ed.).

[23] Kimbal, R., & Ross, M. (2002). The Data Warehouse Toolkit: The complete guide to dimensional modeling (2nd Ed.).

[24] Chaudhuri, S., & Dayal, U. (1997). An Overview of Data Warehousing and OLAP Technology. ACM SIGMOD Record, 26:65-74.

[25] Santos, S.R., & Gutierrez, A.M. (2004). Implementações de Data Warehouse na Área da Saúde.

[26] Santos, S.R., Almeida, L.A., Tachinardi, U., & Gutierrez, A.M. (2006). Data Warehouse para a Saúde Pública: Estudo de Caso SES-SP. X  Congresso Brasileiro de Informática em Saúde.

[27] Prather, J.C., Lobach, D.F., Goodwin, L.K., Hales, J.W., Hage, M.L., & Hammond, W.E. (1997). Medical Data Mining: Knowledge Discovery in a Clinical Data Warehouse. Proceedings: a conference of the American Medical Informatics Association / AMIA Annual Fall Symposium, 101-105.

[28] Kantardzic, M. (2011). Data Mining: Concepts, Models, Methods and Algorithms.
[29] Fayyad, U., Piatetsky-Shapiro, G., & Smyth P. (1996). From Data Mining to Knowledge Discovery in Databases. AI Magazine, 17(3):37-54.
[30] Nguyen, M.T., Tjoa, M.A., & Trujillo, J. (2005). Data Warehousing and Knowledge Discovery: A Chronological View of Research Challenges. DaWaK, 530-535.
[31] Kurgan, L.A., & Musilek, P. (2006). A survey of Knowledge Discovery and Data Mining process models. The Knowledge Engineering Review, 21(1):1-24.
[32] Weiss, S.M., & Indurkhya, N. (1995). Rule-based Machine Learning Methods for Functional Prediction. Journal of Artificial Intelligence Research, 3:282-403.
[33] Azevedo, A., & Santos, M.F. (2008). KDD, SEMMA and CRISP-DM: A Parallel Overview. IADIS European Conference Data Mining, 183-185.
[34] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0: Step-by-Step Data Mining Guide.
[35] Tufte, E.R. (2007). The visual display of quantitative information (2nd Ed.).
[36] Hawkins, D.M. (2004). The Problem of Overfitting. Journal of chemical information and computer sciences, 44:1-12.
[37] Shannon C.E. (1948). A Mathematical Theory of Information. The Bell System Technical Journal.
[38] Hssina, B., Merbouha, A., Ezzikouri, H., & Erritali, M. (2014). A comparative study of decision tree ID3 and C4.5. International Journal of Advanced Computer Science and Applications, Special Issue on Advances in Vehicular Ad Hoc Networking and Applications, 13-19.
[39] Rokach, L., & Maimon, O. (2010). Data Mining and Knowledge Discovery Handbook (2nd Ed.).
[40] Larose, D.T. (2004). Discovering Knowledge in Data: An Introduction to Data Mining.
[41] International Organization for Standardization, http://www.iso.org/iso/home.htm
[42] Velte, L., Pedrosa, T., Cost, C., & Oliveira, L.J. (2012). An OpenEHR repository based on a native XML database. Proceedings of the International Conference on Health Informatics, 386-389.
[43] openEHR, www.openehr.org
[44] IHE, http://www.ihe.net/About_IHE/
[45] Heard, S., & Beale, T. (2007). What's next for openEHR. [Online].

http://www.mz.gov.si/fileadmin/mz.gov.si/pageuploads/eZdravje/Novice/gradiva_predstavitve_dogodkov/Open_EHR/4_whats_next_for_openEHR.pdf, Accessed June 28, 2014.

[46] Beale, T. (2002). Archetypes: Constraint-based domain models for future-proof information systems. Workshop on Behavioural Semantics.

[47] Kalra, D. (2006). Electronic Health Record Standards. Yearb Med Inform, 136-44.

[48] Eichelberg, M., Aden, T., Riesmeier, J., Dogac, A., & Laleci, B.G. (2005). A Survey and Analysis of Electronic Healthcare Record Standards. ACM Computing Surveys, 4:277–315.

[49] Beale, T., & Heard, S. (2007). Archetype Definitions and Principles. [Online]. http://iten.behdasht.gov.ir/uploads/256_1197_archetype_principles.pdf, Accessed June 28, 2014.

[50] Yu, P. (2003). Archetypes, GEHR, openEHR and Electronic Health Records. The Journal on Information Technology in Healthcare, 369–380.

[51] Beale, T., & Heard, S. (2008). openEHR Architecture Overview. [Online]. http://www.openehr.org/releases/1.0.2/architecture/overview.pdf, Accessed June 28, 2014.

[52] Beale, T., & Heard, S. (2008). Archetype Definition Language, [Online]. http://www.openehr.org/releases/1.0.2/architecture/am/adl.pdf, Accessed June 28, 2014.

[53] Sinha, K.P., Sunder, G., Bendale, P., Mantri, M., & Sande, A. (2013). Electronic Health Record: Standards, Coding Systems, Frameworks, and Infrastructures.

[54] Garde, S., Hovenga, E., Gränz, J., Foozonkhah, S., & Heard, S. (2006). Towards a Repository for Managing Archetypes for Electronic Health Records. IC 2006 and HINZ 2006: Proceedings. Brunswick East, Vic.: Health Informatics Society of Australia, 61-65.

[55] Clinical Knowledge Manager, http://openehr.org/ckm/

[56] Chen, R., Georgii-Hemming, P., & Ahlfeldt, H. (2009). Representing a Chemotherapy Guideline Using openEHR and Rules. Stud Health Technol Inform., 150:653-657.

[57] Frade, S., Freire, S.M., Sundvall, E., Patriarca-Almeida, J.H., & Cruz-Correia, R. (2013). Survey of openEHR storage implementations. IEEE 26th International Symposium on Computer-Based Medical Systems (CBMS), 303-307.

[58] Freire, S.M., Sundvall, E., Karlsson, D., & Lambrix, P. (2012). Performance of XML Databases for Epidemiological Queries in Archetype-Based EHRs. Scandinavian Conference on Health Informatics, 51-57.

[59] Williams, A. (2005). Performance of relational databases versus native XML databases (Dissertation). [Online]. http://hdl.handle.net/10523/1200, Accessed October 13, 2014.

[60] Ding, Q., & Sundarraj, G. (2008). Mining Association Rules from XML Data. Data Mining

and Knowledge Discovery Technologies, 59-71.

[61] Wan, J.W.W., & Dobbie, G. (2003). Extracting Association Rules from XML Documents using XQuery, WIDM '03 Proceedings of the 5th ACM international workshop on Web information and data management, 94-97.

*This page was intentionally left blank*

# 9 Annexes

*This page was intentionally left blank*

Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 16 (2014) 1391 – 1396

Procedia Technology

CENTERIS 2014 - Conference on ENTERprise Information Systems / ProjMAN 2014 - International Conference on Project MANagement / HCIST 2014 - International Conference on Health and Social Care Information Systems and Technologies

# Exporting data from an openEHR repository to standard formats

Jorge Almeida[a,b], Samuel Frade[b], Ricardo Cruz-Correia[b]

[a]*Superior School of Engineering and Management – Polytechnic Institute of Leiria, Portugal*
[b]*Center for Research in Health Technologies and Information Systems - CINTESIS, Faculty of Medicine of Univerity of Porto, Portugal*

**Abstract**

A meaningful electronic health record (EHR) improves the ability for healthcare professionals to enact evidence-based knowledge management and aids decision making in health care. EHRs can have a positive impact in quality of care, patient safety, and efficiencies.
OpenObsCare project aims to investigate how the openEHR standard can be implemented in obstetrics medical records. This project studies the dynamic generation of clinical record modules, the development of forms based in clinical concepts (openEHR templates and archetypes) defined by the international openEHR community, and the way that patient data can be stored, optimizing its future reuse in, for example, clinical management and research.
This paper presents the methodology for exporting data from an openEHR repository to standard tabular formats, enabling further form reports and data analysis using other software. The result will be a tool that can connect to any openEHR repository-based web service and export data to standard formats, therefore enabling data analysis to extract knowledge.
Connection with the EHR server is made by means of web service consumption. Some difficulties arise such as the handling of the different web service protocols, different method names invoked to perform the same task varying from web service to web service and their validation by the application.

*Keywords:* Electronic Health Records; Obstetrics; openEHR repository; data analysis; knowledge extraction;

## 1. Introduction

An electronic health record (EHR) is a digital version of a patient's paper record, allowing access to information about its demographics, allergies, medication, laboratory test results, previous treatments and other medical data of a patient in an integrated manner instantly and to authorized users. This information must be understandable, regardless of the medical institution that accesses it. A meaningful electronic health record (EHR) improves the

doi:10.1016/j.protcy.2014.10.157

ability for healthcare professionals to enact evidence-based knowledge management and aids decision making in health care. EHRs can have a positive impact on quality of care, patient safety, and efficiencies. However, without accurate and appropriate content in an usable and accessible format, these benefits will not be achieved, which means data must be captured, addressing quality concerns, in order not to be prone to propagate errors downstream to data warehouses and other data analysis systems throughout the increase of information shared between organizations [1].

An EHR is stored and managed by an EHR system, which can then be shared with other institutions. To accomplish this, standards are developed and implemented. A standard can be defined in many forms, but essentially it comprises a set of rules and definitions that specifies how to carry out a process or produce a product, and is useful because it allows two or more dissociated people to work in some cooperative way [2].

OpenEHR is an open source standard for data management, access, exchange and storage, allowing any organization or provider shared access to standardized data through a compatible application. Its' specifications are defined and maintained by the openEHR Foundation, which has led the objective of achieving semantic interoperability and its architecture can be separated in Reference Model and Archetype Model. The first allows interoperability by exchanging data between systems only in terms of standard open reference model instances recurring to domain-specific ontologies. As for the Archetype Model, it allows semantic interoperability through the use of structured statements based on the information (reference) model and defines valid data types, structures and values to keep the information both flexible and structured. There is also a Service Model that describes EHR centered basic services in a health information environment [3].

OpenObsCare project aims to investigate the way the openEHR standard can be implemented in obstetrics medical records. This project studies the dynamic generation of clinical record modules, the development of forms based in clinical concepts (openEHR templates and archetypes) defined by the international openEHR community, and the way that patient data can be stored, optimizing its future reuse in, for example, clinical management and research. The project intends to have, as results, several scientific publications and at least one installation of an openEHR based obstetrics clinical record (VCOpenObsCare) in a health institution and therefore being a proof-of-concept.

The openEHR standard promotes the use of data structures that are not easily fit into simple relational databases using normalized tables and relationships. Only 6 out of 16 known openEHR based EHR use relational databases and even these do not use it in a standardized way [4], which means that in an openEHR repository, data can be found in native XML databases, relational databases or relational databases with XML support, each of them with different structures and performance levels [5][6]. This presents a big challenge in the process of exporting data to a standard relational database through use of standard formats, since most data or statistical analysis packages, for management or research, are based on data in spreadsheets. We envision that without proper tools that facilitate the extraction of openEHR data to tabular formats, the proliferation of openEHR systems will be much more difficult.

## 2. Aim

This paper presents the methodology for exporting data from the openEHR repository to standard tabular formats, enabling further form reports and data analysis using other softwares. The result will be a tool that has the ability to connect to any openEHR repository-based web service and export selected data into standard formats, such as CSV, R, SQL or SPSS, thus enabling data analysis by third parties to extract knowledge.

## 3. Full project description

### *3.1. The VCOpenObsCare System*

The VCObsGynCare is a clinical record software designed to be used by obstetrics and gynecology doctors, anesthesiologists, nurses and administrative staff to register inpatient admission and discharge, childbirth and newborn data, as well as surgical procedures, nursing records and gynecological interventions. It is currently in use at Hospital de S. João in Oporto and it is currently being installed in other hospitals in the north of Portugal. The version of this software that will be openEHR-based (VCOpenObsCare) is currently in development, and its repository service layer will be the initial data source for this work's exportation tool.

### *3.2. Project tasks*

The project was divided in five tasks, each one sequentially dependent from the previous.

1. Project management - this task aims to broadcast the results, other projects, institutions and investigators, establish connections and do project promotion through a web site, social networks and other media channels.
2. OpenEHR archetype and template creation - the objective of this task is the definition of the necessary information to gather within the obstetrics health care scope, so that it is possible to automatically produce forms and data repositories in the following tasks. With this in mind, this task was divided in five sub-tasks:
    a. Current Obstetrics health record concept analysis
    b. Exploration of existing Clinical Knowledge Managers (CKM) for useful archetypes
    c. Creation of new openEHR archetypes and adaptation of the existing ones
    d. Definition of openEHR templates, given the existing and new archetypes
    e. Submission of new archetypes to the international community
3. Template to web form conversion tool - The conversion of the obtained templates in the previous task to HTML5 format for web use methodology definition is the goal of this task. The intention is to extend this methodology to any template in a wider form and, therefore, the task was divided in three sub-tasks:
    a. Read and export template information
    b. Define the corresponding matches between form fields and template/archetype elements
    c. Form creation
4. Design of a Data repository - the objective is to develop a data repository to store information obtained from the previous task developed forms. During this task, NoSQL and Object Oriented databases investigation will take place, thus giving place to the following sub-tasks:
    a. Data repository operation design preparation
    b. Application persistence layer creation under the form of services according with the openEHR standard service model specification
    c. Data repository creation
    d. Integration of the data repository with the VCOpenObsCare software for later usage in production
5. Exporting data to standard formats - The aim of this task is to elaborate a methodology for exporting data from the previous task obtained repository, to standard formats, enabling further form reports and data analysis through Data Mining or Business Intelligence systems. Three sub-tasks were defined to accomplish this objective:
    a. Stored openEHR objects content and clinical concepts mapping tool development
    b. Export database definition
    c. Comparative test elaboration

## 4. Exporting data to standard formats task use cases and requirements

As the main purpose of the tool is to allow clinical and non-clinical staff to select and export data available at an openEHR repository web service, enabling its later use in data statistics and analysis related systems, it was necessary to build a use case diagram and to define the functional and non-functional requisites that would define the course of the this last project's task development.

### *4.1. Use case diagram*

The use case diagram for the tool was modeled having in mind the necessary steps to export data from the repository in a simple manner. The next image shows the interaction of the user with the system towards the data export. As it can be observed, to be able to export data from the web service, the user needs to run through six interactions to be able to finally export data:

1. Choose web service address to connect to – in this step, the user inserts the web service address to connect. The program will then try to confirm if the web service is online. If true, the program will move to the next step.
2. Choose method to retrieve available templates – the user will choose the method to retrieve the available templates from the web service methods list. The method will then be invoked.
3. Choose template to display its' variables – Every template has at least one variable. The user will choose the variables to export from the list and the program will retrieve their information.
4. Choose file location – The program will then ask the user to choose file location store the exported data.
5. Choose export file format – The user must choose a file format to save the exported data from four available formats.
6. Export data – After completing the previous steps, the user will then start the process of exporting the data to the desired format.

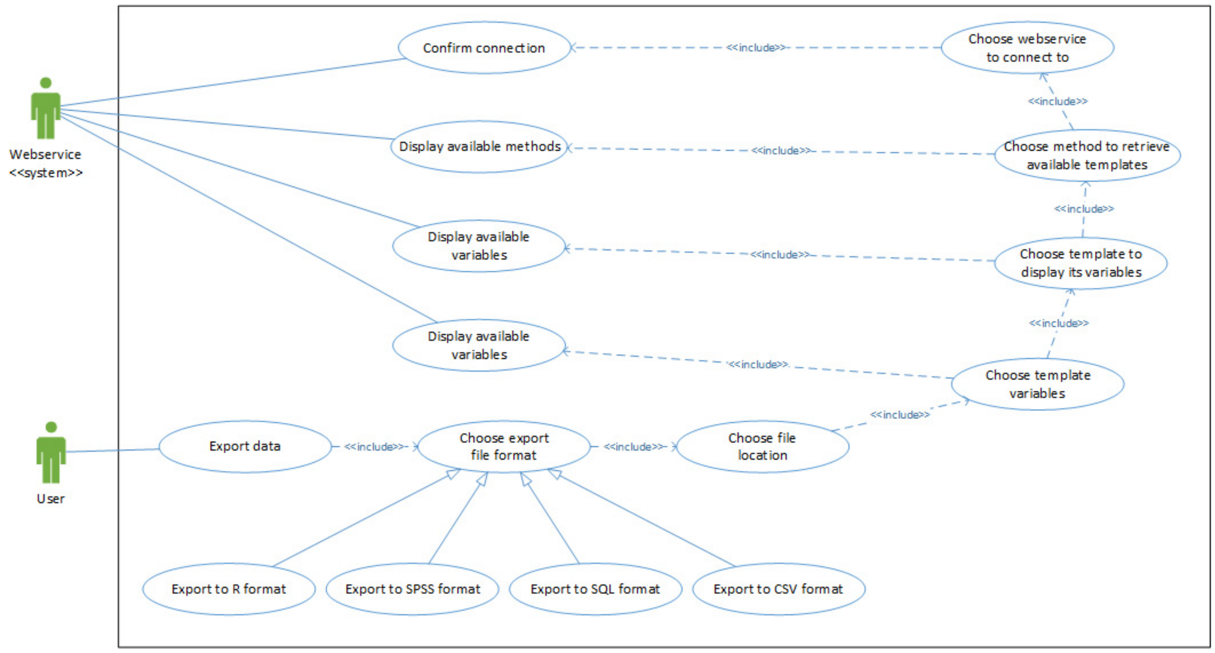


Fig. 1. Use case diagram for the export tool

## *4.2. Functional and non-functional requirement definition*

The obtained table of functional and non-functional requirements for the export tool is shown below. By functional requirements we understand the behavior of any given system, while non-functional requirements describe aspects of the system that do not relate to its execution.

This table was obtained having in mind the need to develop a versatile software which could be easily adopted and built on a free and known platform (enabling further upgrades).

Table 1. Functional and non-functional requirement table

| Requirement | Functional | Non-functional |
|---|---|---|
| OS environment free | X | |
| Ability to connect to different web services | X | |
| Ability to adapt to different web service protocols | X | |
| Easy to be used by non-clinical staff | | X |

| | |
|---|---|
| Choose the templates to export | X |
| Choose variables to export | X |
| Export to standard formats | X |

## 5. Architecture

### *5.1. Used technologies*

One of the first steps towards the completion of this task was to define the programming language to develop the tool. Being one of the requirements of this tool the possibility of being used independently from the operating system in which it will be run, the choice fell on Java programming language. This would guarantee the same behavior regardless of the system environment.

### *5.2. How it communicates with the EHR server*

The communication with the EHR server is made by means of web service consumption. A REST or SOAP web service can then be found at the other end of the communication channel by the application, thus making the handling of different protocols a necessity of this application. One other aspect to keep in mind is that the method used to retrieve the available templates and their variables can have different names/URIs from web service to web service and it is essential to choose the right method, which must be validated by the application.

The server-client architecture is rather simple and straightforward as it can be perceived from the next figure.

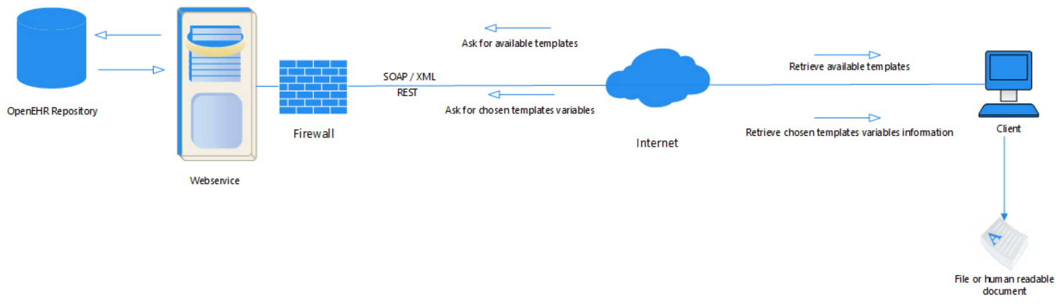

OpenEHR Repository
Webservice
Firewall
SOAP / XML
REST
Ask for available templates
Ask for chosen templates variables
Internet
Retrieve available templates
Retrieve chosen templates variables information
Client
File or human readable document

Fig. 2. Server-client architecture diagram

### *5.3. Application prototype*

A prototype application is being developed and is already able to connect to any given web service and list its available methods while keeping a record of valid web services addresses for later connections. The next step is to implement a mechanism to internally switch between communication protocols according to the web service type found while connecting.

## 6. Discussion

Information nowadays is produced very rapidly. However, not all that information necessarily constitutes knowledge. There's a great volume that is redundant, non-relevant or even misleading. Since quantity is not quality, that information must be refined to extract knowledge. By doing so, we can be contributing to support researchers, managers and decision makers with relevant data, no matter how heterogenic the facilities and institutions systems may be.

One of the main problems found regarding health databases is the lack of quality assurance mechanisms to ensure a proper evaluation and understanding of the electronic health records. The data structure used in each health care institution or organization's information system differs from one another according to the final objective for its data. For instance, if the end purpose of the data is to support research, its granularity and description will not be the same as if it were to be used in a financial context.

Another identified problem is the tabular format limitation of associating two or more events to one episode regarding this same event. Consider a newborn episode: we can register the episode start date, the mother weight, the newborn weight, the newborn Apgar score at the first minute. However, how should this episode be shown in a spreadsheet when there are two or more events in the same episode (e.g.: twins)? For the same episode we have two weights and two Apgar scores, thus having two values in a structure design that shows only one.

The openEHR standard offers a solution to avoid this situation by supplying semantic interoperability. EHR requirements gathered throughout the years contributed to the openEHR specification and architecture design. Due to the fact that this architecture is highly generic, particularly because it is archetype-driven, it satisfies many requirements outside the general concept of "clinical EHR". Nevertheless, in many cases, there is still the need of using the current data analysis systems, whose data structures are different from an openEHR repository and do not allow simple import/export operations or bidirectional communication, and are well quoted and established in the statistical and analytical communities.

Regarding the export tool, when it communicates with the web service, the obtained data will surely be different as the stored templates can have themselves different variables from the same archetype. For instance, a birth template which is composed by more than one archetype can have distinguished variables from one hospital to another. Other big challenge to this task is the possible web service type (REST or SOAP) implemented. The tool will have to possess a mechanism to adapt its used protocol to communicate and retrieve the data. As for the resulting files with exported data, this will require a considerable amount of effort manipulating information in order to create and fill the SPSS, R and SQL file structure so it can be read by its corresponding applications.

We argue that the described export tool is able to address two important issues, which are the lack of tools to implement the standard and to provide clinical and non-clinical staff the ability to export desired data to be used in knowledge building systems and software.

## Acknowledgements

This work is supported by INESCC and CIDES (Faculty of Medicine of University of Porto).
This work was also supported by Gulbenkian Foundation, through the research project "OpenObs.care - Criação de Sistemas de Informação em Saúde baseados em OpenEHR" (MP/P-125963).

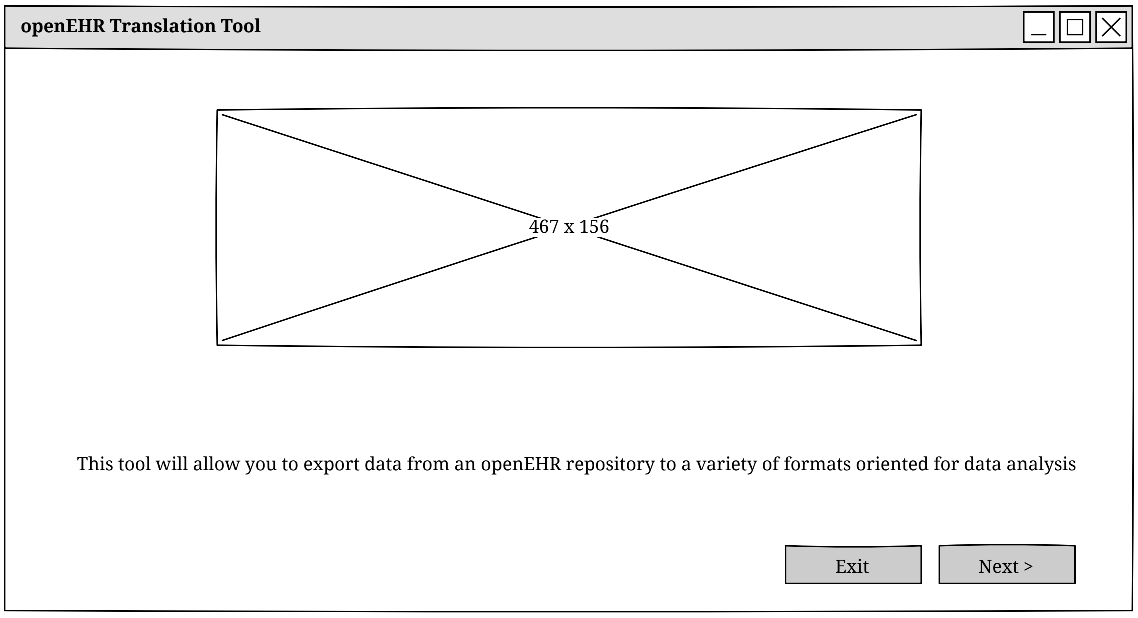
## References

[1] AHIMA, "Assessing and Improving EHR Data Quality (Updated)", Journal of AHIMA 84, number 2, March 2013.
[2] Shortliffe E.H, Cimino J.J., *Biomedical Informatics – Computer Applications in Health Care and Biomedicine*, 3rd ed., Springer, 2006.
[3] Beale T., Heard S., *openEHR Architecture – Architecture Overview*, Ocean Informatics, The openEHR Foundation, Revision 1.1, 2007.
[4] Frade S., Freire S., et al. Survey of openEHR storage implementations. CBMS 2013. Porto. 303-7. DOI: 10.1109/CBMS.2013.6627806.
[5] Freire S., Sundvall E., et al., Performance of XML Databases for Epidemiological Queries in Archetype-Based EHRs, Scandinavian Conference on Health Informatics, 2012
[6] Velte L., Pedrosa T., et al., An Openehr Repository Based on a Native Xml Database, Proceedings of the International Conference on Health Informatics, 2012

# open EHR_ Translation_ Tool_ CSV

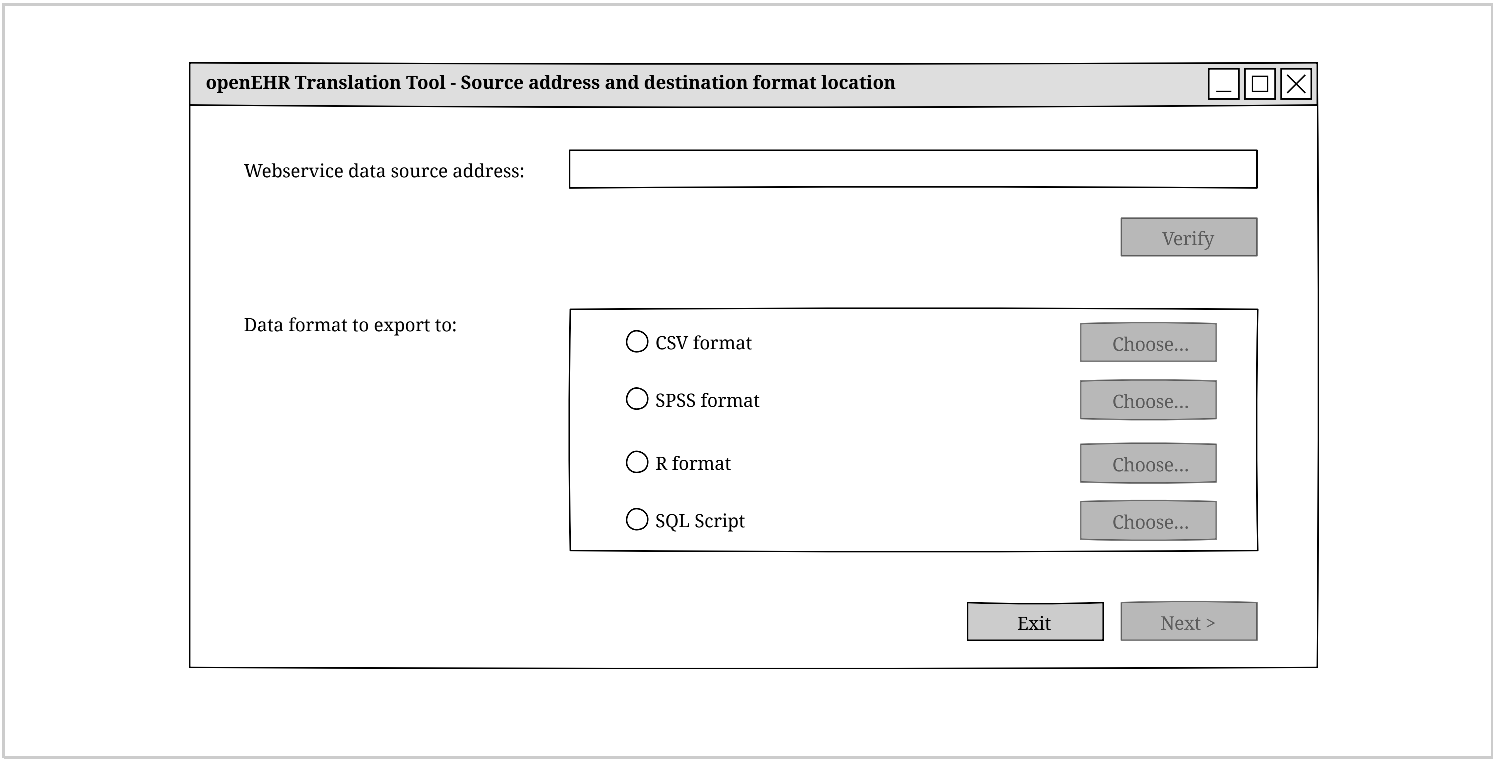
Exported at: Thu Nov 28 2013 21:26:49 GMT+0000 (WET)

## Screen01

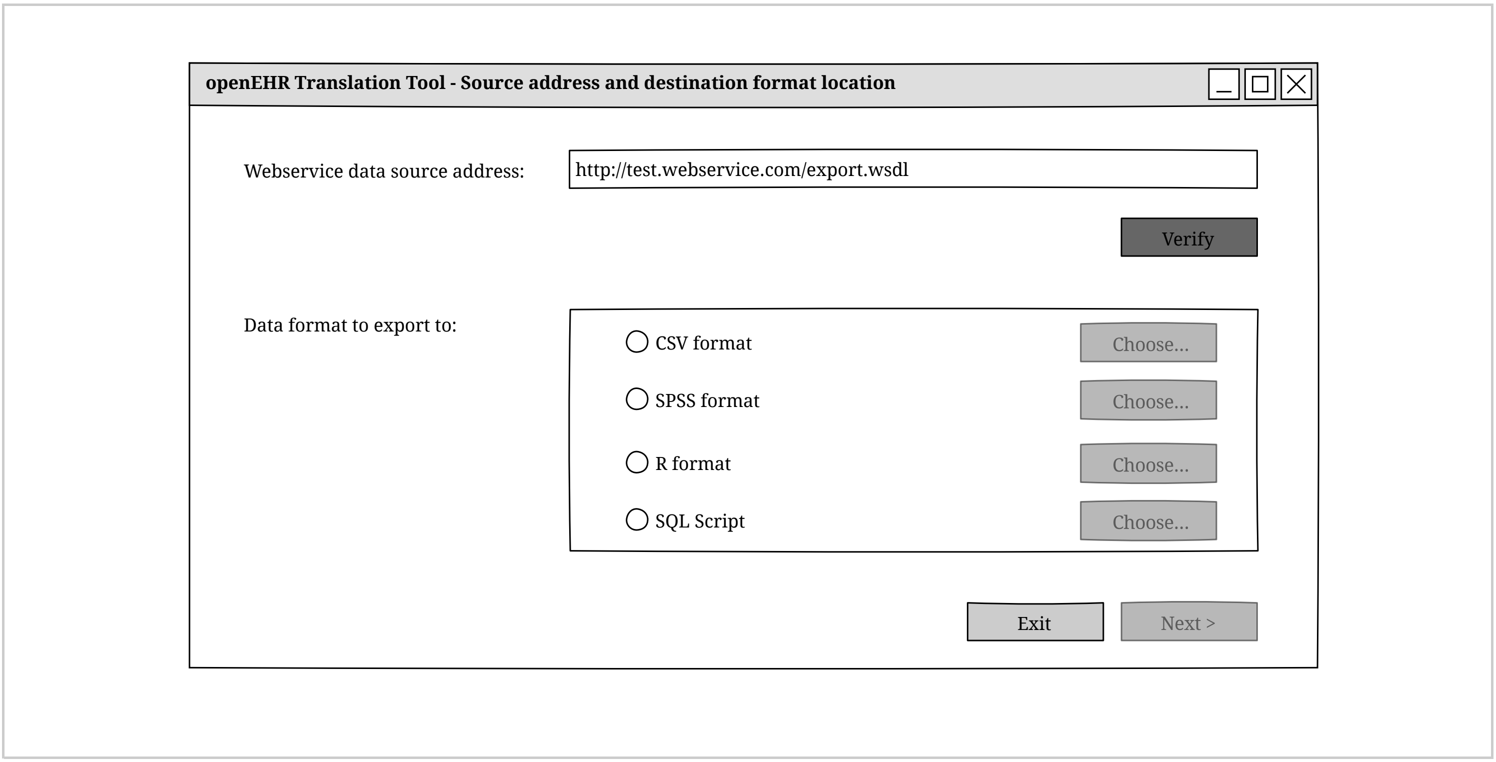


This tool will allow you to export data from an openEHR repository to a variety of formats oriented for data analysis

## Screen02


openEHR Translation Tool - Source address and destination format location
Webservice data source address:
Verify
Data format to export to:
CSV format
Choose...
SPSS format
Choose...
R format
Choose...
SQL Script
Choose...
Exit
Next >

## Screen03



openEHR Translation Tool - Source address and destination format location

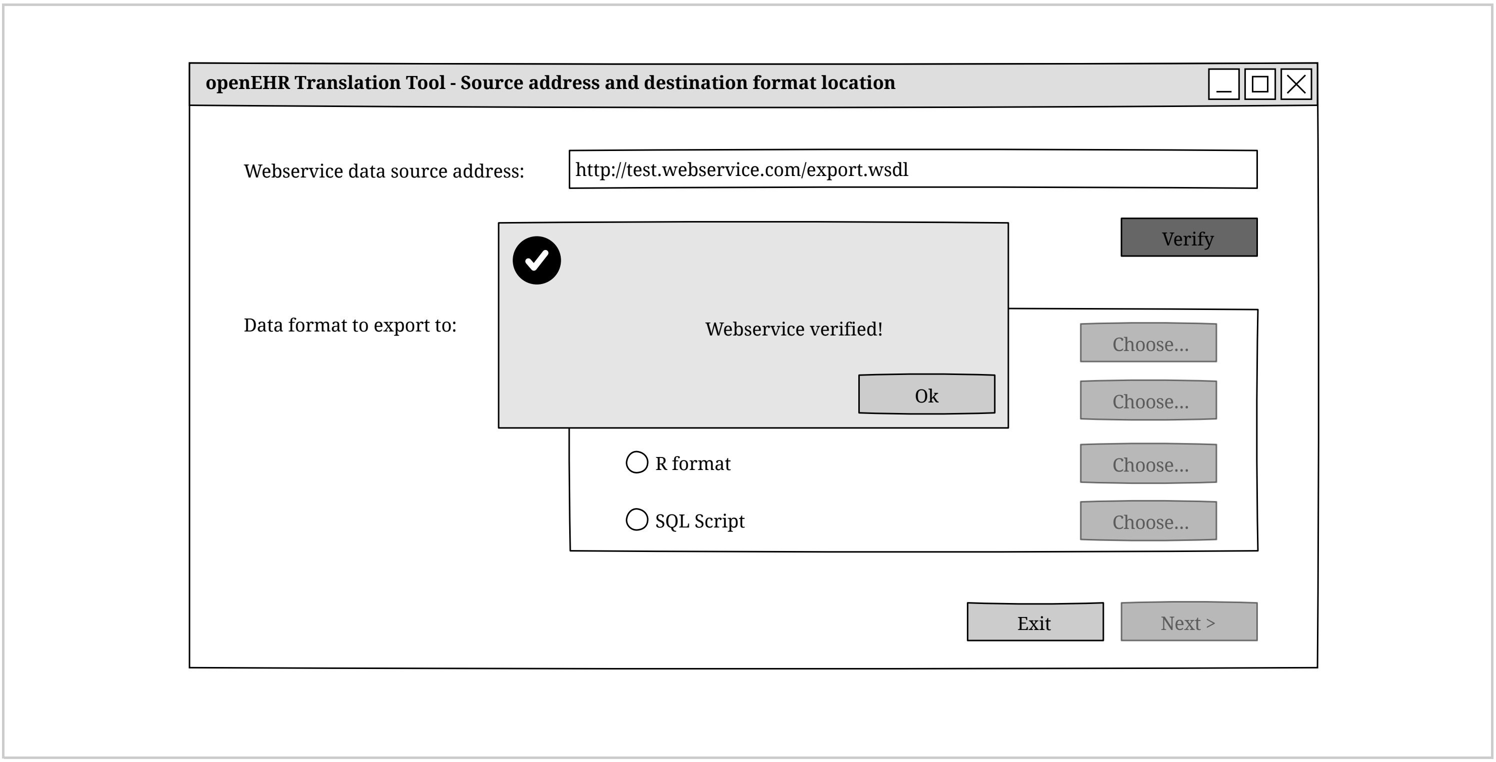
Webservice data source address: http://test.webservice.com/export.wsdl

Data format to export to:

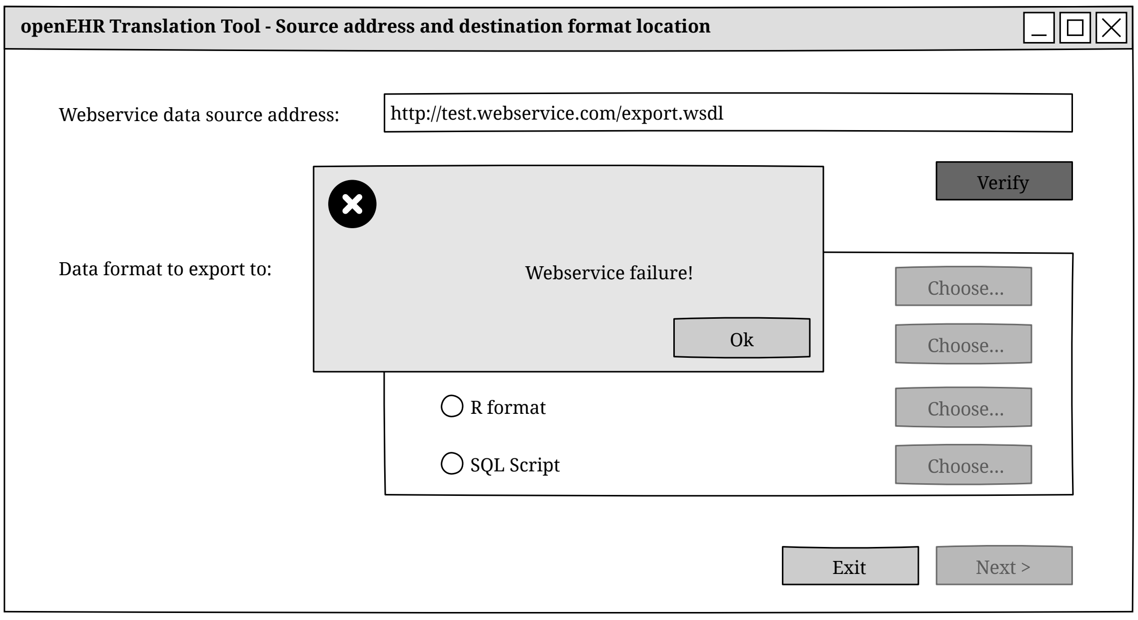
- ◯ CSV format
- ◯ SPSS format
- ◯ R format
- ◯ SQL Script

Choose...
Choose...
Choose...
Choose...
Exit
Next >

## Screen04a


openEHR Translation Tool - Source address and destination format location
Webservice data source address:
http://test.webservice.com/export.wsdl
Verify
Data format to export to:
Webservice verified!
Ok
Choose...
Choose...
R format
Choose...
SQL Script
Choose...
Exit
Next >

## Screen04b



**openEHR Translation Tool - Source address and destination format location**

Webservice data source address: http://test.webservice.com/export.wsdl

Verify

Data format to export to:

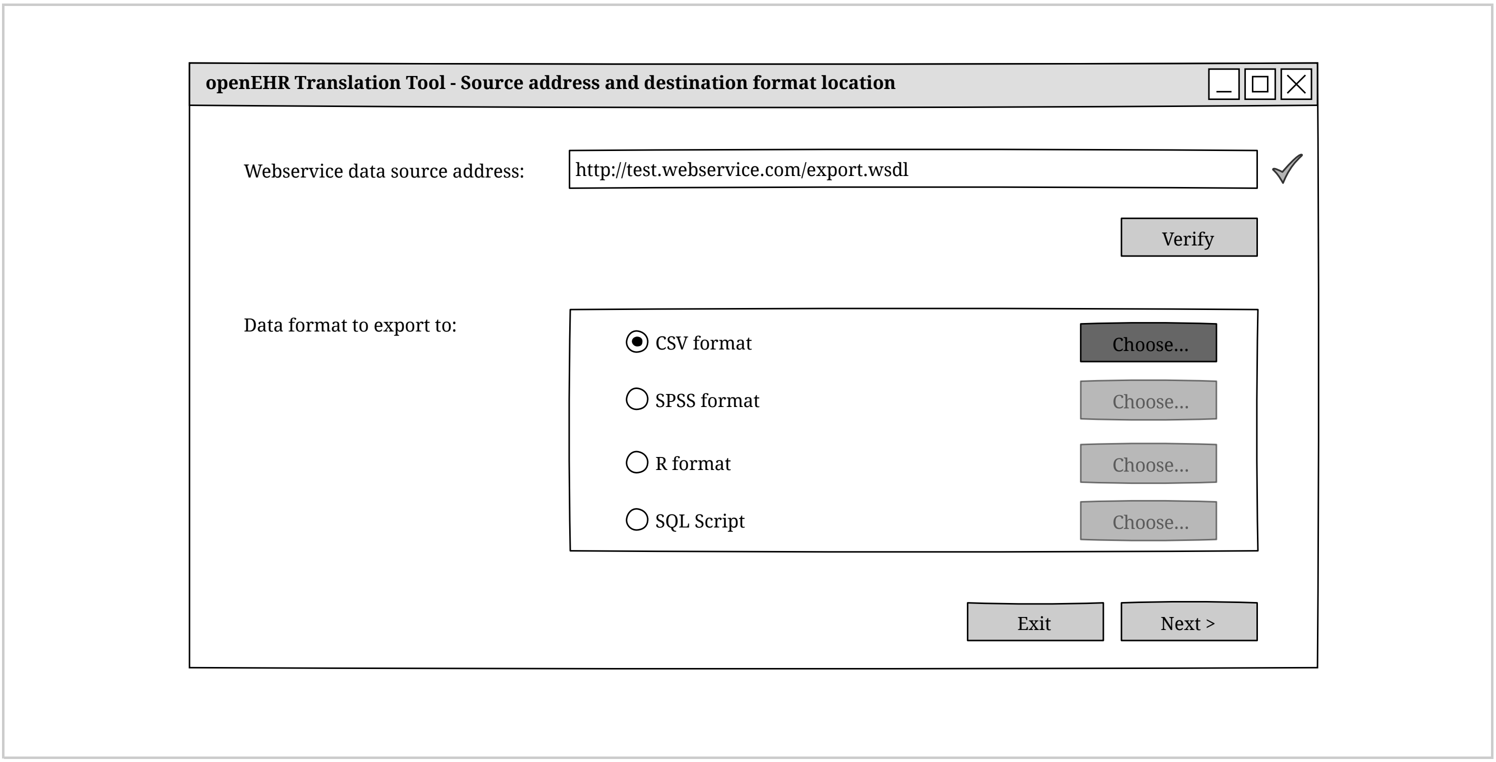
Webservice failure!

Ok

R format

SQL Script

## Screen05



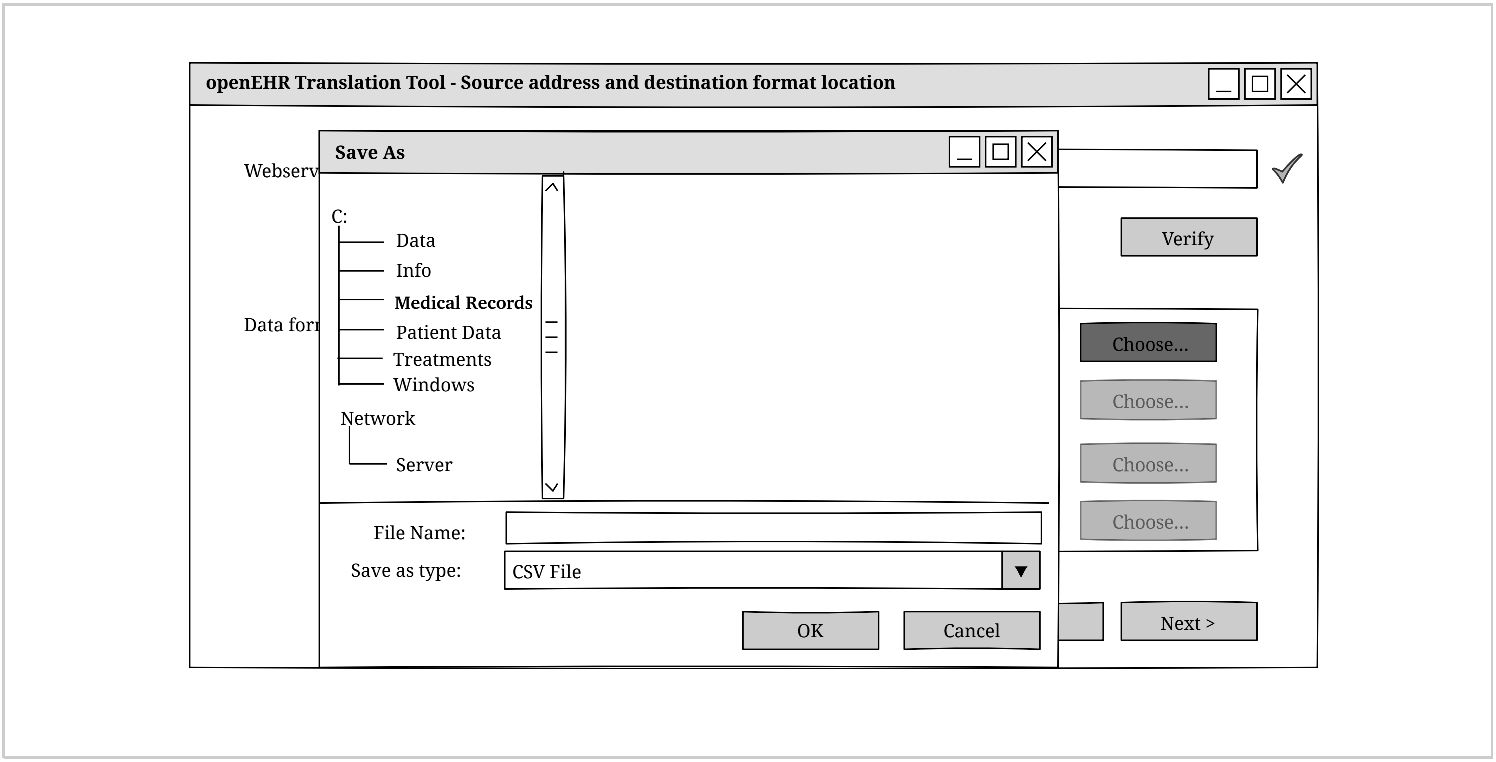
openEHR Translation Tool - Source address and destination format location

Webservice data source address: http://test.webservice.com/export.wsdl ✓

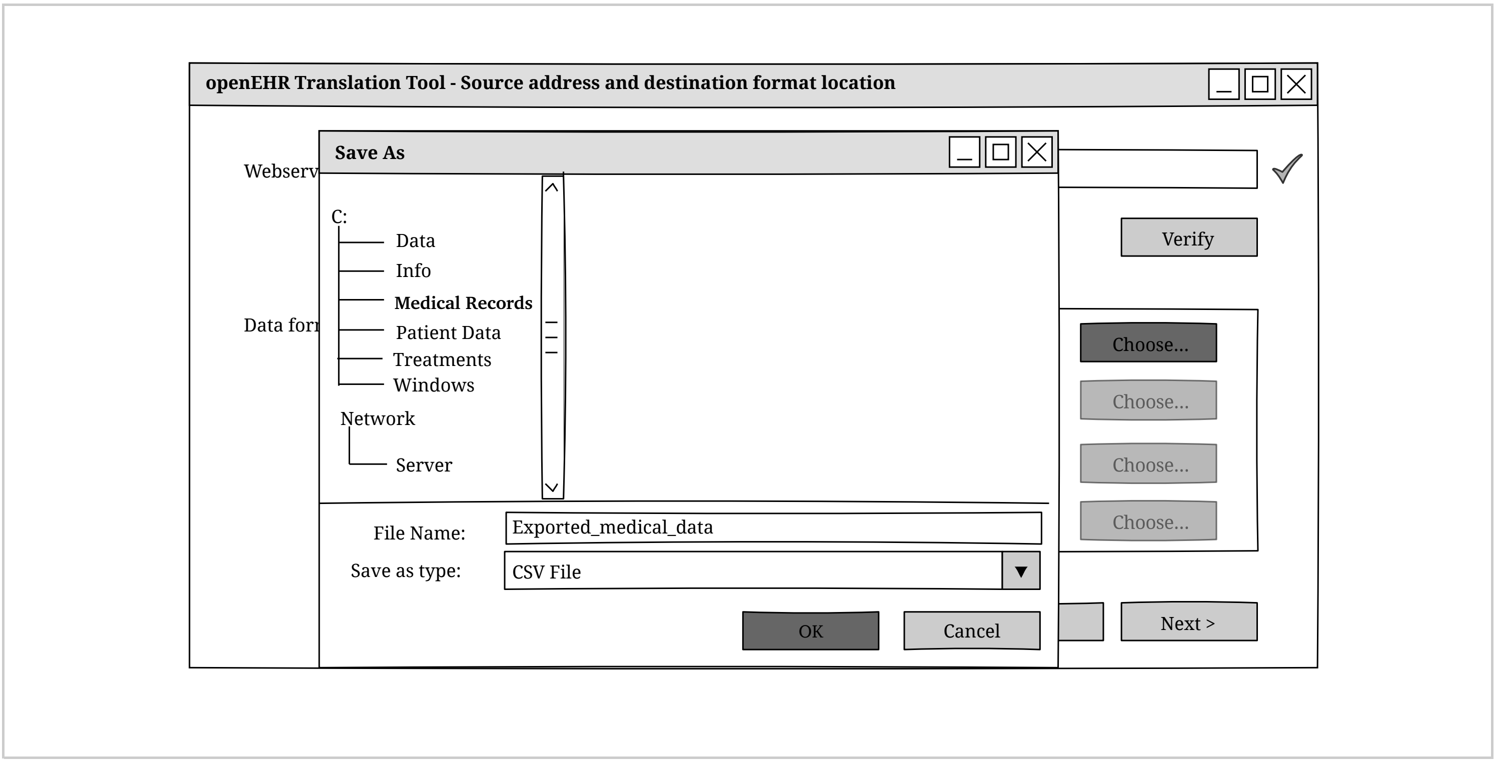
Data format to export to:

● CSV format
○ SPSS format
○ R format
○ SQL Script

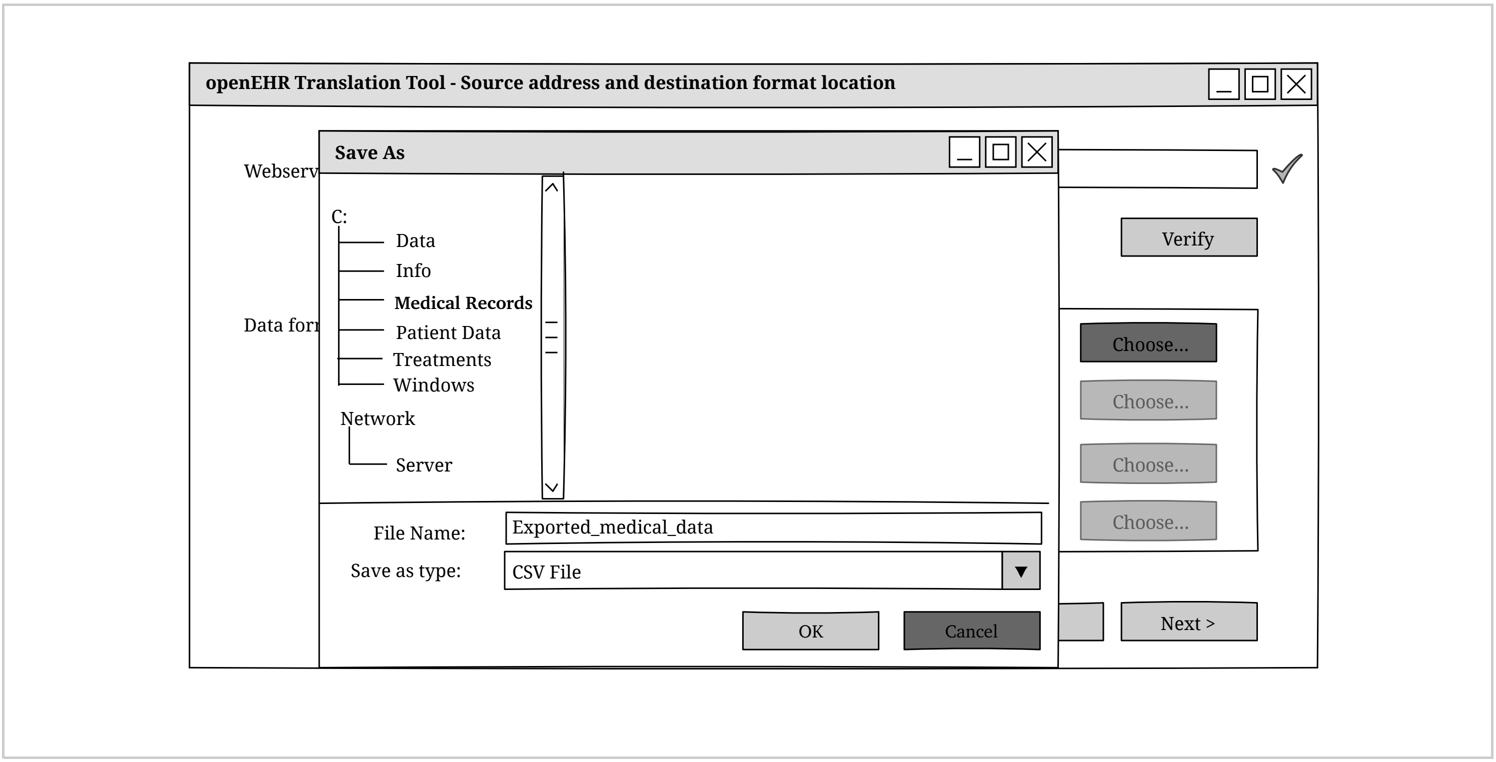
## Screen6


openEHR Translation Tool - Source address and destination format location
Webserv
Data forr
Verify
Choose...
Choose...
Choose...
Choose...
Next >
Save As
C:
Data
Info
Medical Records
Patient Data
Treatments
Windows
Network
Server
File Name:
Save as type:
CSV File
OK
Cancel

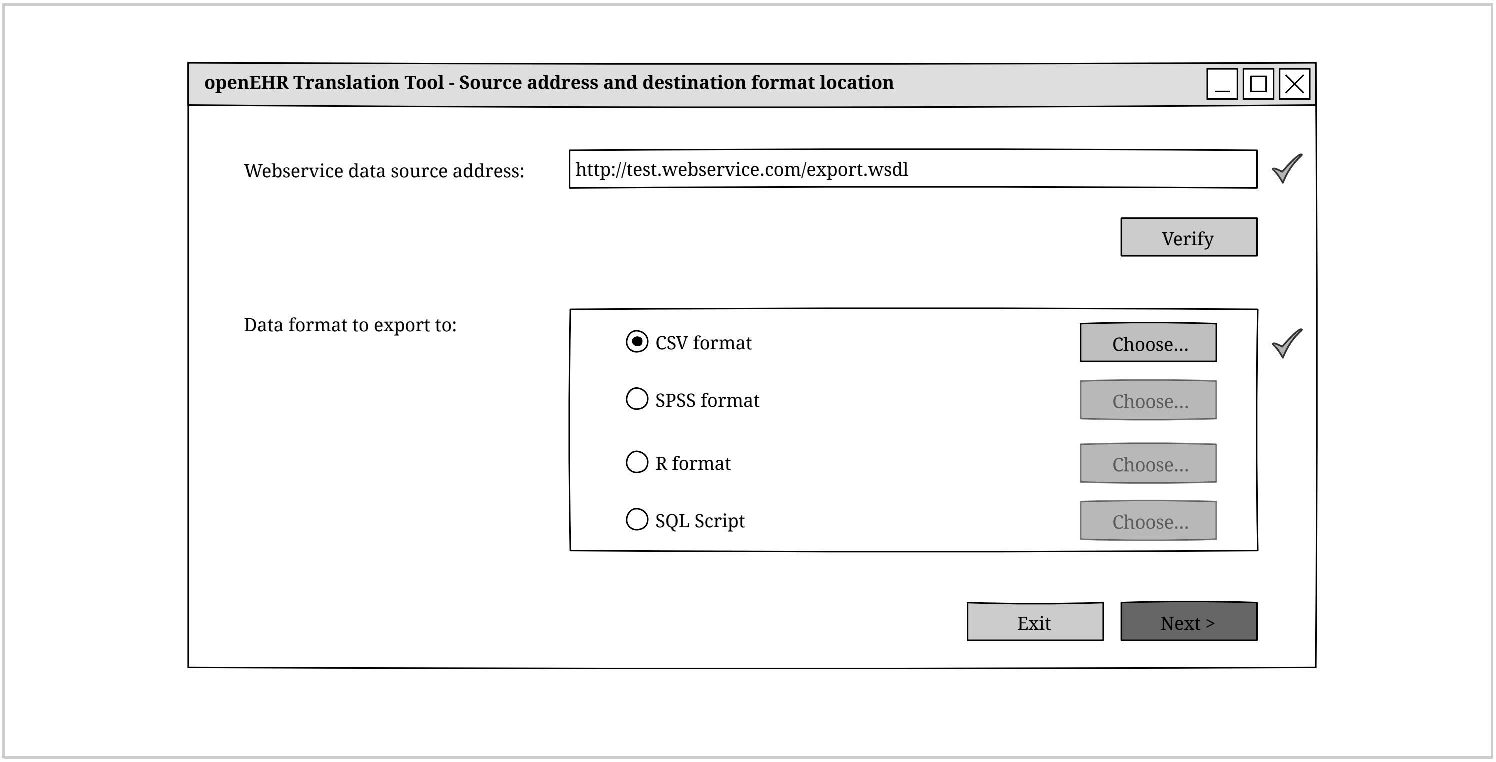
## Screen7a


openEHR Translation Tool - Source address and destination format location
Webserv
Data forr
Verify
Choose...
Choose...
Choose...
Choose...
Next >
Save As
C:
Data
Info
Medical Records
Patient Data
Treatments
Windows
Network
Server
File Name:
Exported_medical_data
Save as type:
CSV File
OK
Cancel

## Screen7b


openEHR Translation Tool - Source address and destination format location
Webserv
Data forr
Verify
Choose...
Choose...
Choose...
Choose...
Next >
Save As
C:
Data
Info
Medical Records
Patient Data
Treatments
Windows
Network
Server
File Name:
Exported_medical_data
Save as type:
CSV File
OK
Cancel

## Screen08

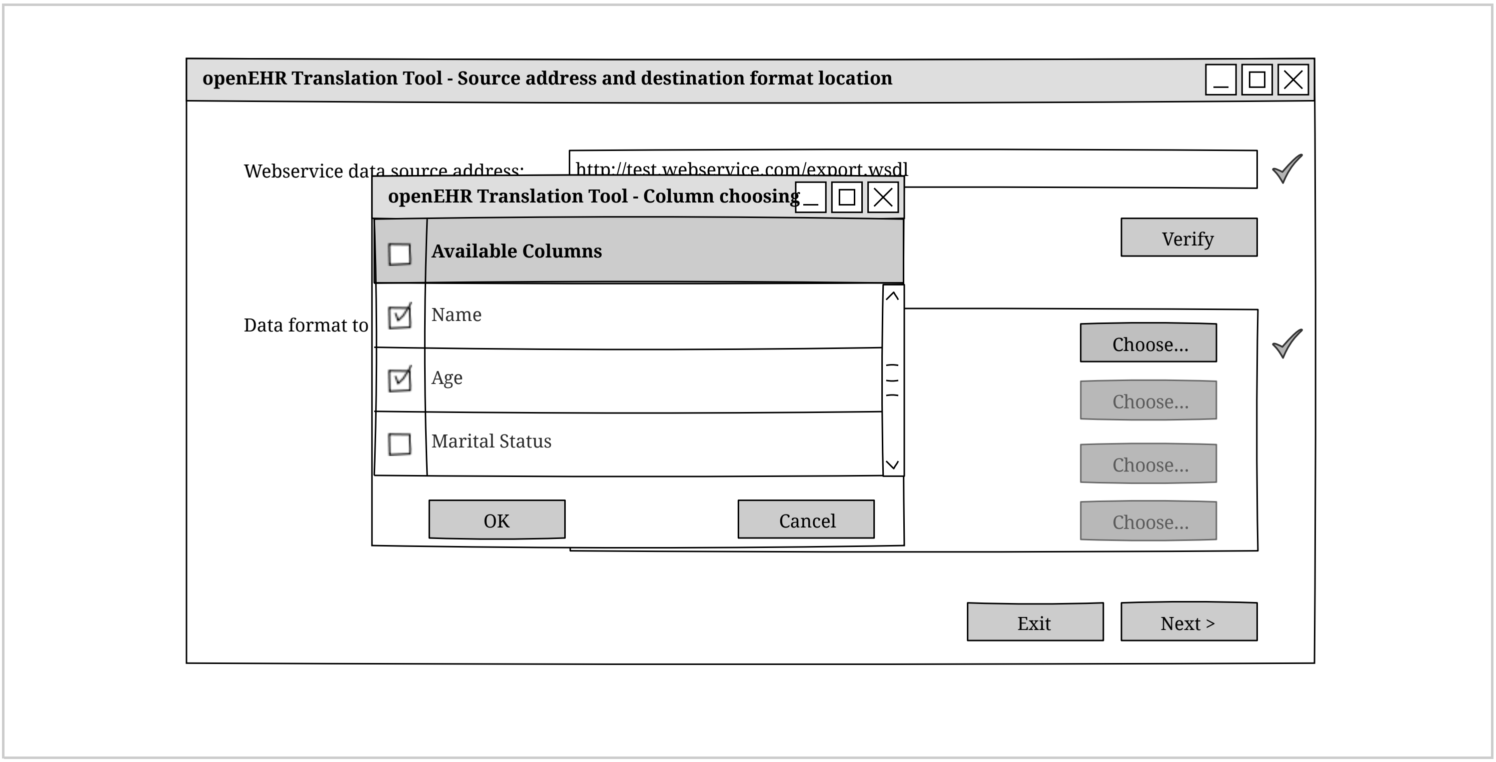


openEHR Translation Tool - Source address and destination format location

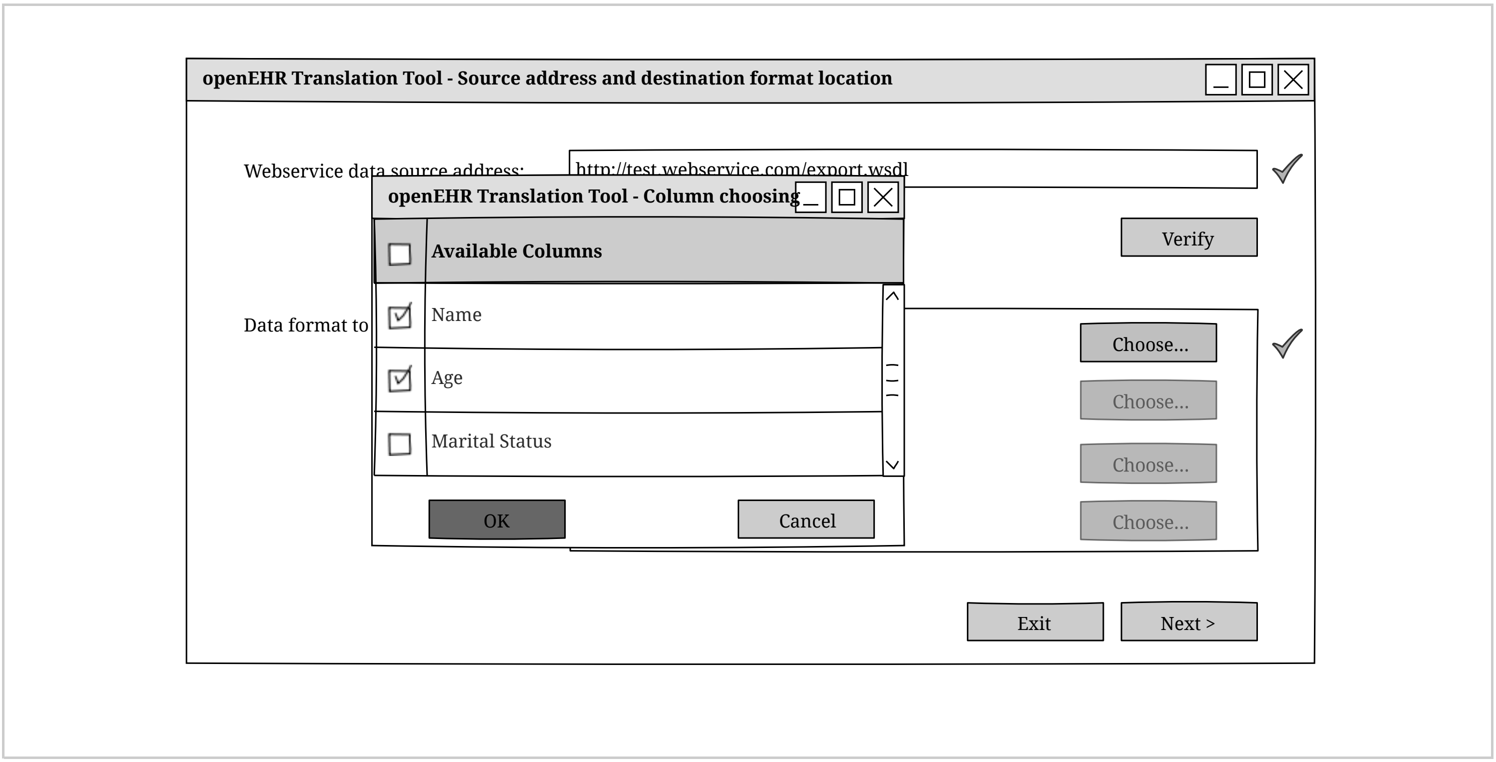
Webservice data source address: http://test.webservice.com/export.wsdl ✓

Data format to export to: [✓ CSV format] SPSS format R format SQL Script ✓

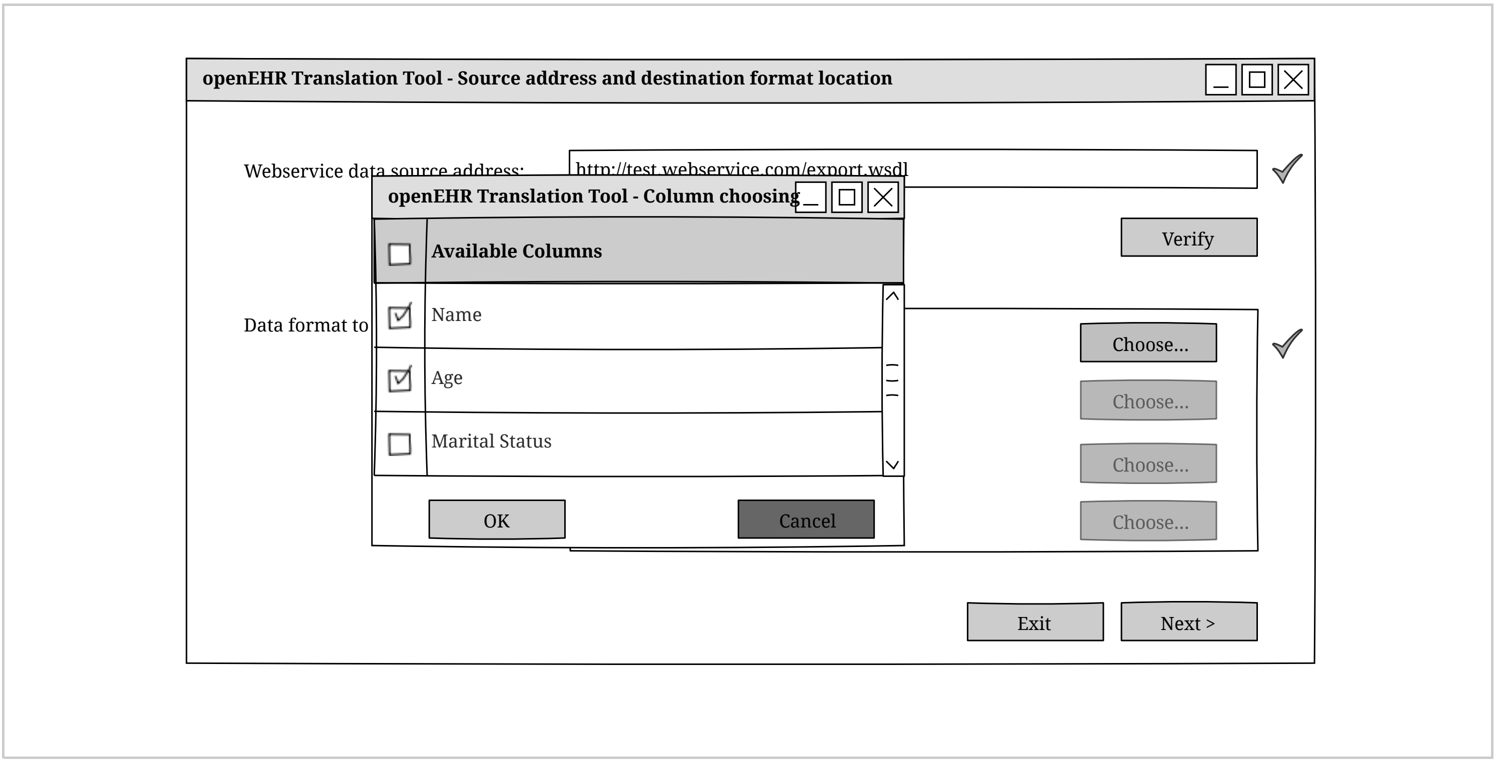
## Screen09


openEHR Translation Tool - Source address and destination format location
Webservice data source address:
http://test.webservice.com/export.wsdl
Verify
Data format to
Choose...
Choose...
Choose...
Choose...
Exit
Next >
openEHR Translation Tool - Column choosing
Available Columns
Name
Age
Marital Status
OK
Cancel

## Screen10a


openEHR Translation Tool - Source address and destination format location
Webservice data source address:
http://test.webservice.com/export.wsdl
Verify
Data format to
Choose...
Choose...
Choose...
Choose...
Exit
Next >
openEHR Translation Tool - Column choosing
Available Columns
Name
Age
Marital Status
OK
Cancel

## Screen10b


openEHR Translation Tool - Source address and destination format location

Webservice data source address: http://test.webservice.com/export.wsdl ✓

Verify

Data format to

Choose... ✓

Choose...

Choose...

Choose...

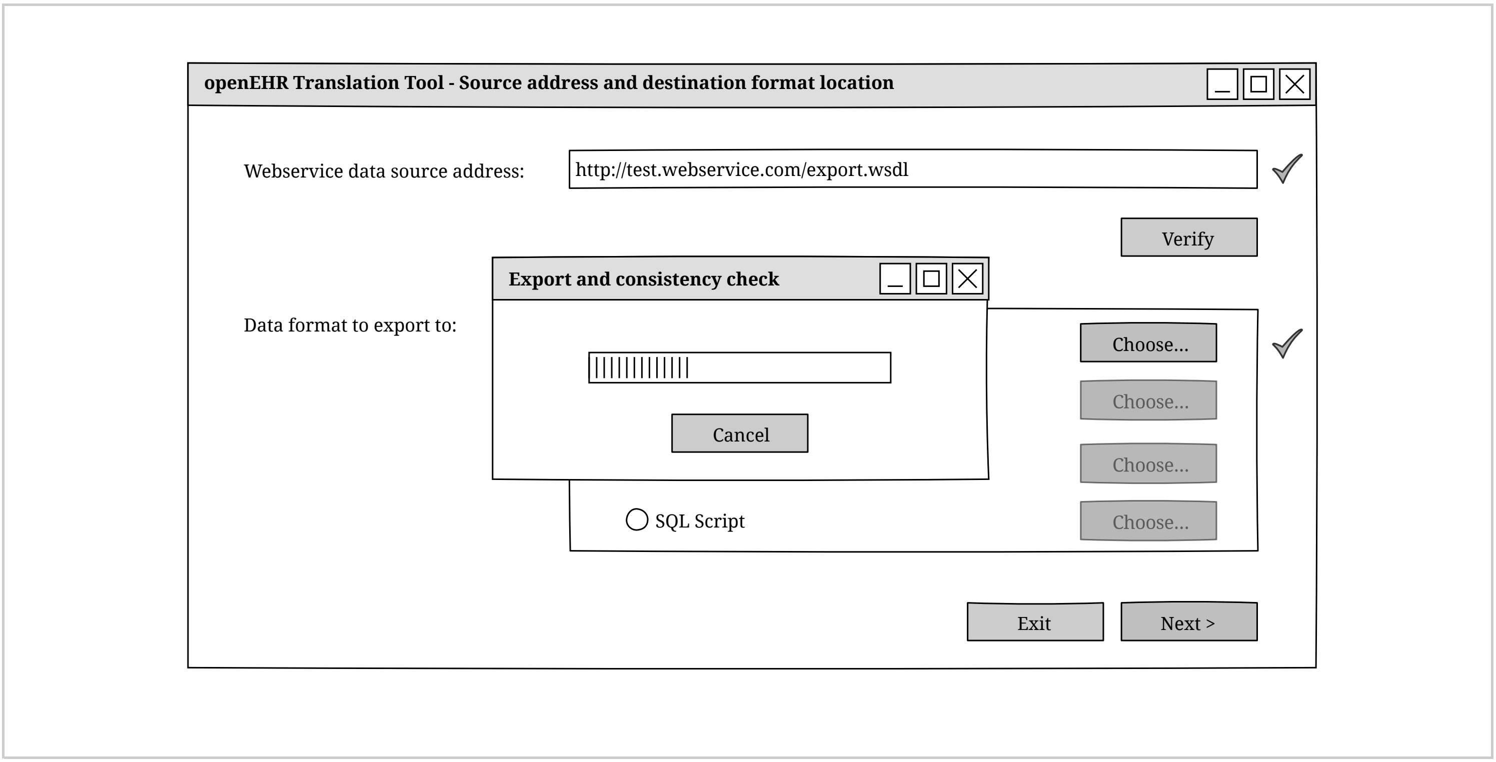
Exit | Next >

openEHR Translation Tool - Column choosing

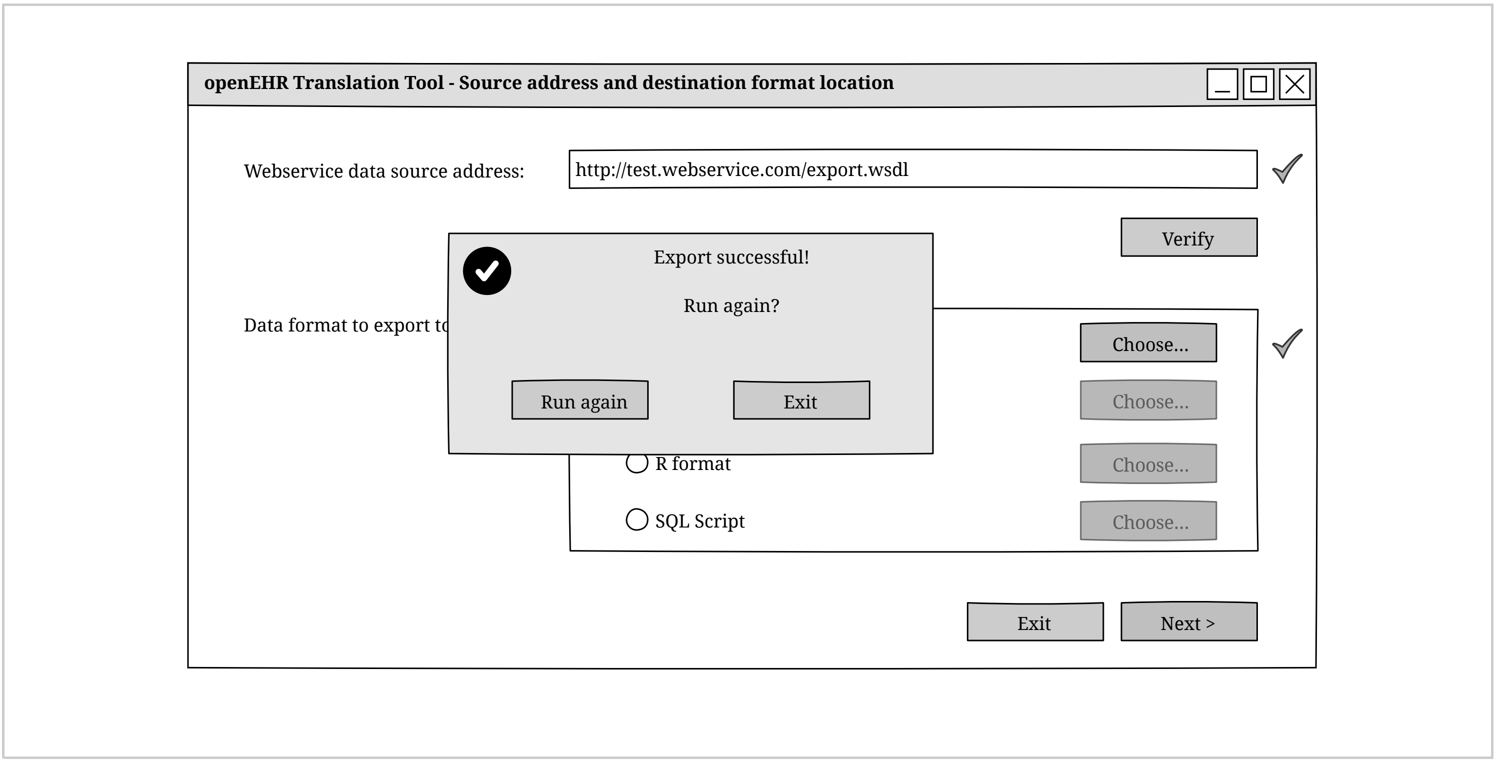
| ☐ | **Available Columns** |
|---|---|
| ☑ | Name |
| ☑ | Age |
| ☐ | Marital Status |

OK | Cancel

## Screen11


openEHR Translation Tool - Source address and destination format location
Webservice data source address:
http://test.webservice.com/export.wsdl
Verify
Data format to export to:
Export and consistency check
Cancel
Choose...
Choose...
Choose...
Choose...
SQL Script
Exit
Next >

## Screen12a



openEHR Translation Tool - Source address and destination format location

Webservice data source address: http://test.webservice.com/export.wsdl ✓

Data format to export to ✓

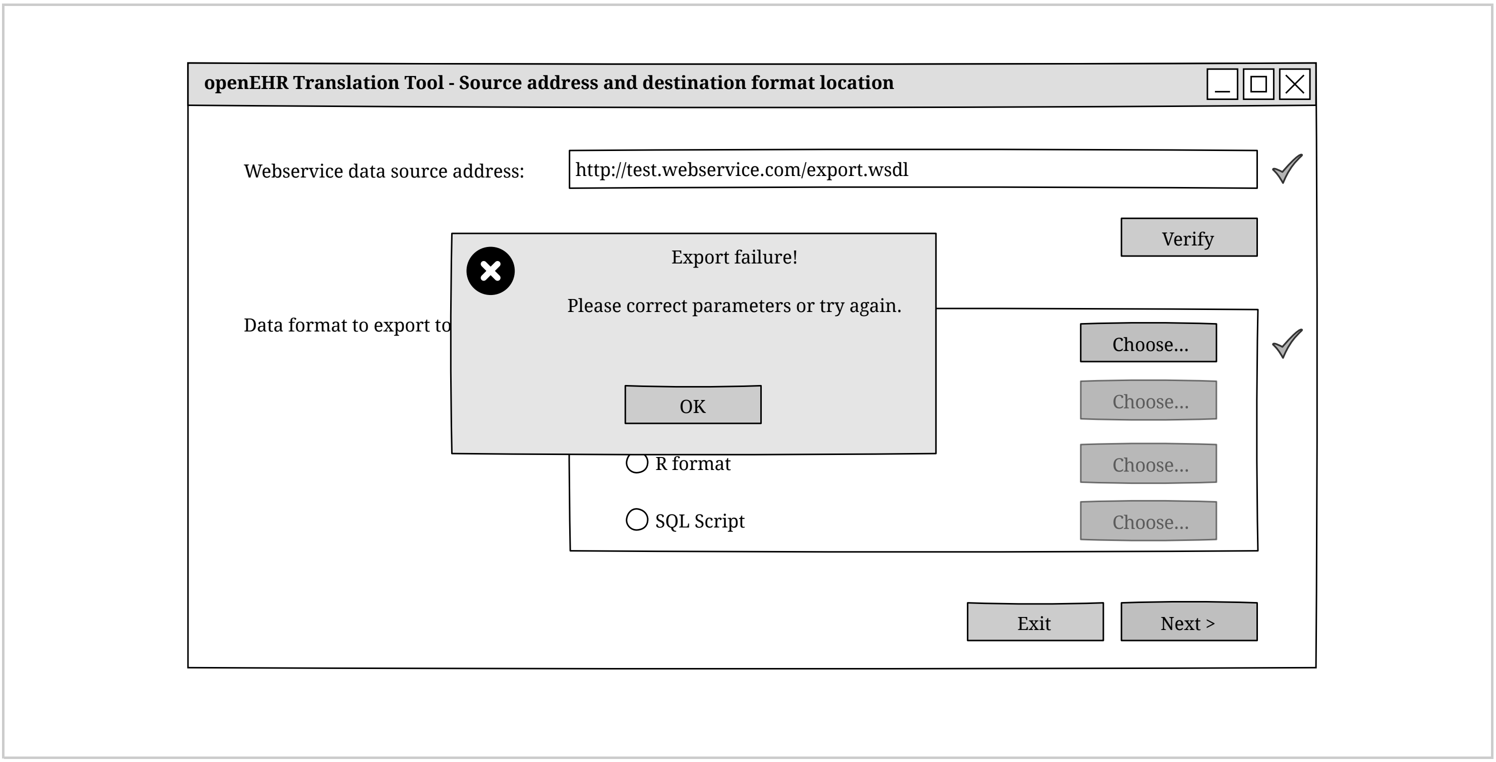
○ R format

○ SQL Script

Export successful!

Run again?

## Screen12b



**openEHR Translation Tool - Source address and destination format location**

Webservice data source address: http://test.webservice.com/export.wsdl ✓

Verify

Data format to export to

R format

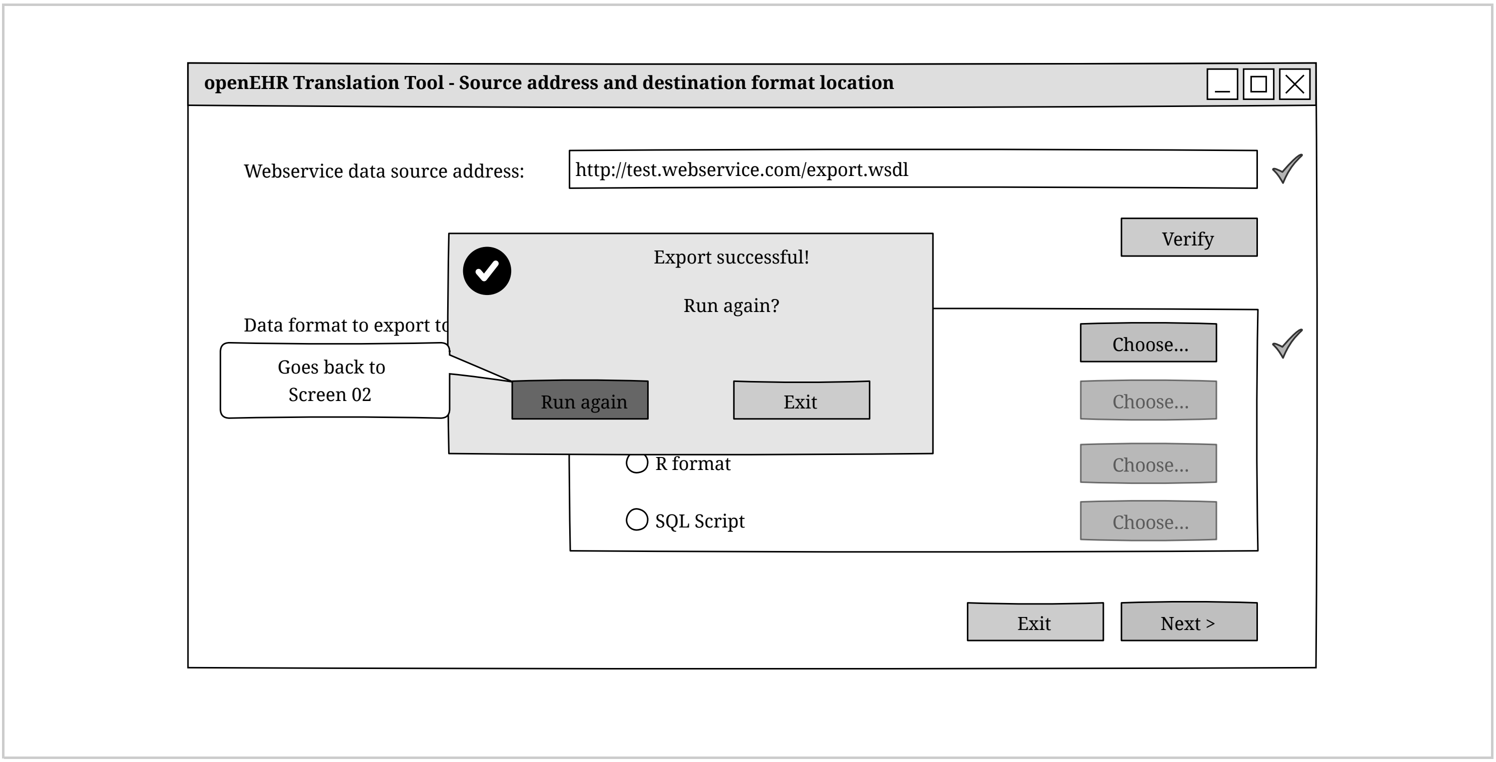
SQL Script

Export failure!

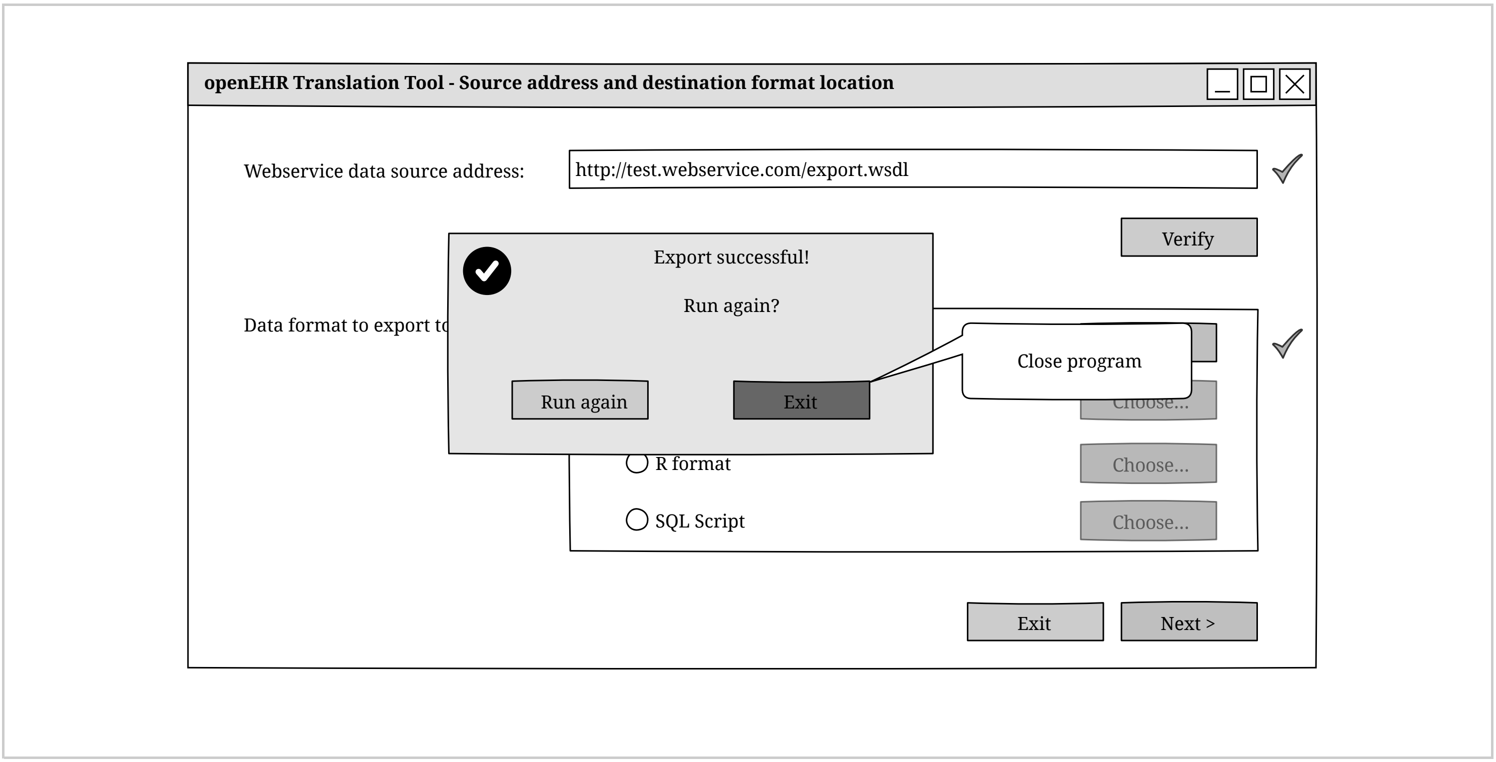
Please correct parameters or try again.

OK

## Screen13a


openEHR Translation Tool - Source address and destination format location
Webservice data source address:
http://test.webservice.com/export.wsdl
Verify
Data format to export to
Choose...
Choose...
R format
Choose...
SQL Script
Choose...
Exit
Next >
Export successful!
Run again?
Run again
Exit
Goes back to Screen 02

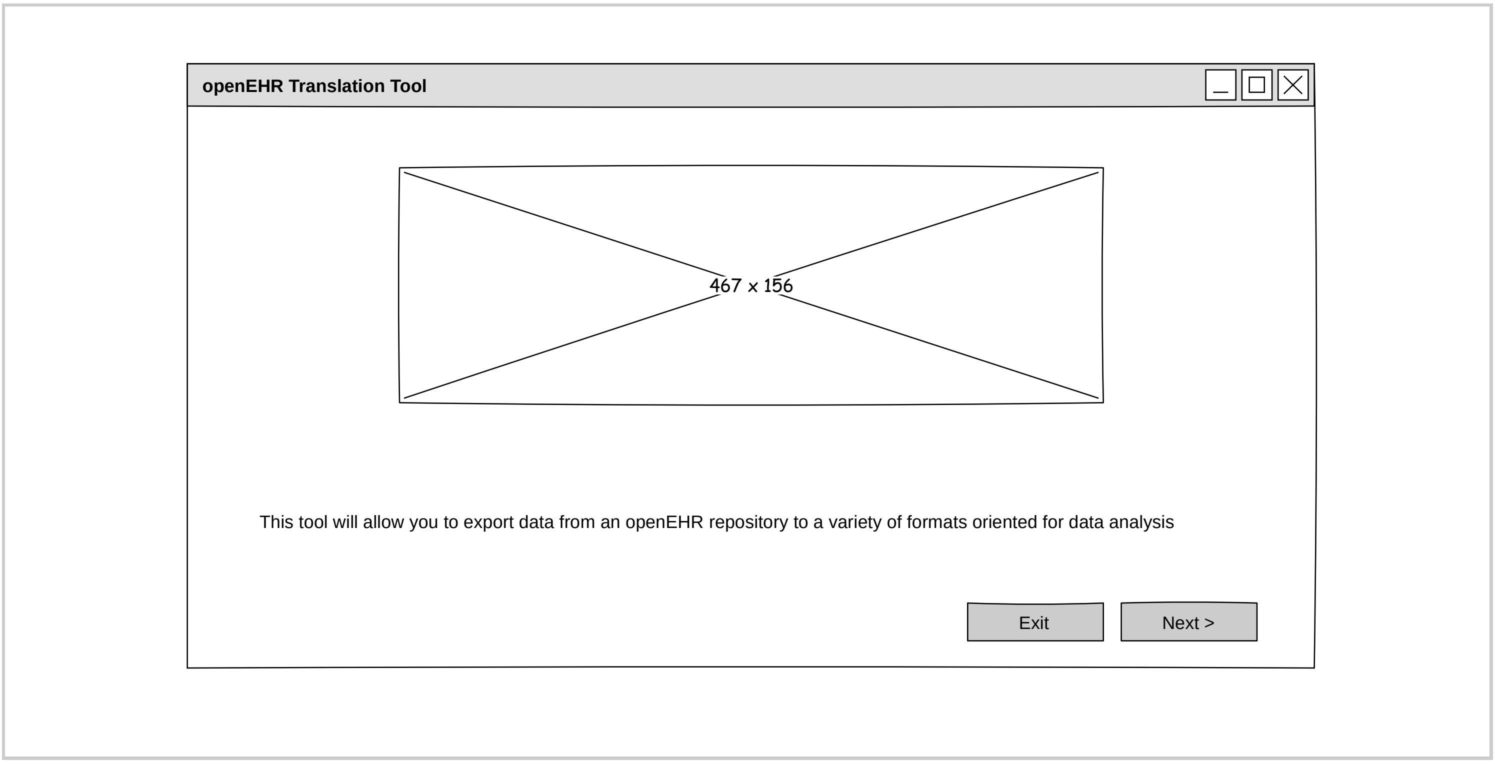
## Screen13b


openEHR Translation Tool - Source address and destination format location
Webservice data source address:
http://test.webservice.com/export.wsdl
Verify
Data format to export to
Export successful!
Run again?
Run again
Exit
Close program
Choose...
R format
Choose...
SQL Script
Choose...
Exit
Next >

# open EHR_ Translation_ Tool_ CSV_ V2

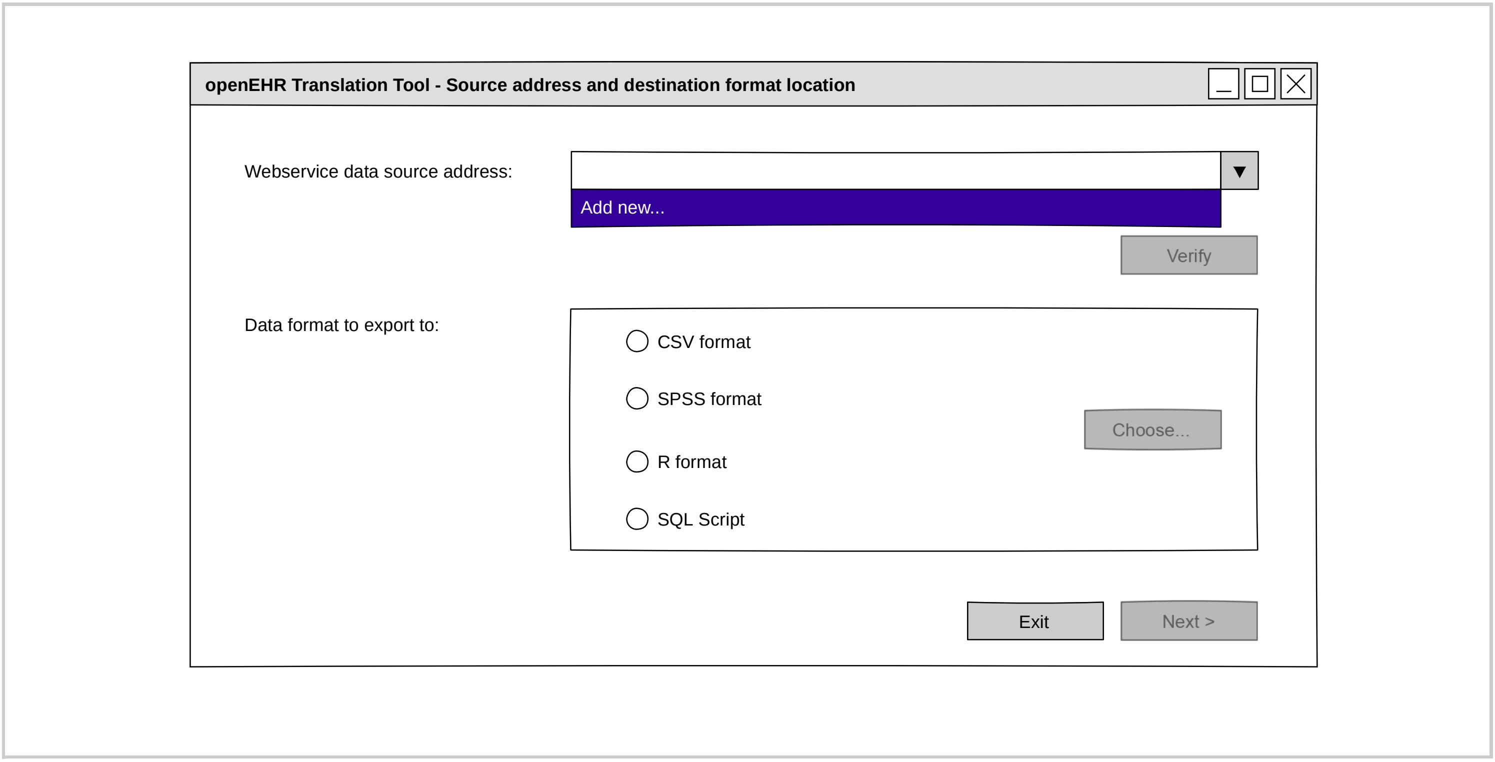
Exported at: Sun Feb 09 2014 19:01:54 GMT+0000 (GMT Standard Time)

## Screen01

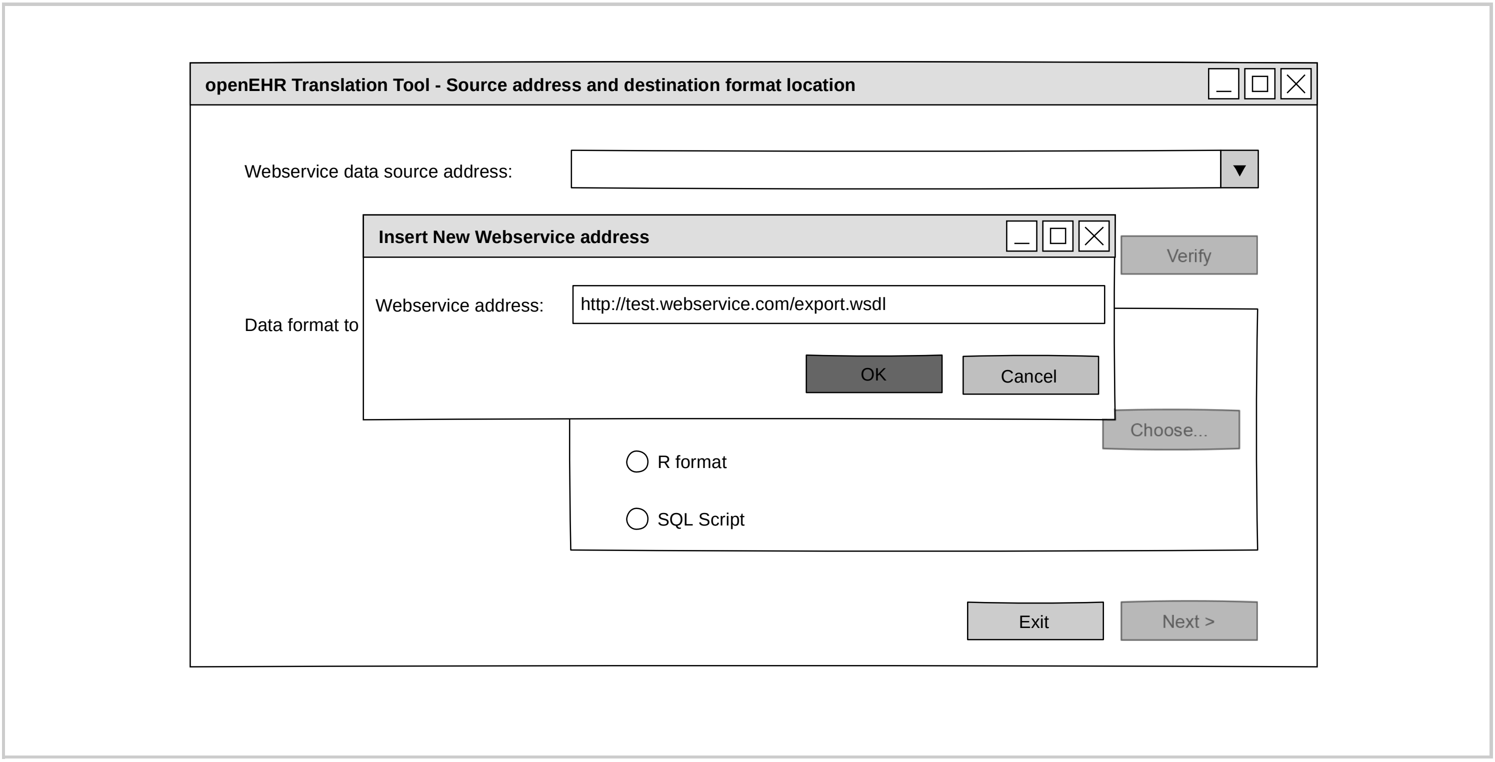


This tool will allow you to export data from an openEHR repository to a variety of formats oriented for data analysis

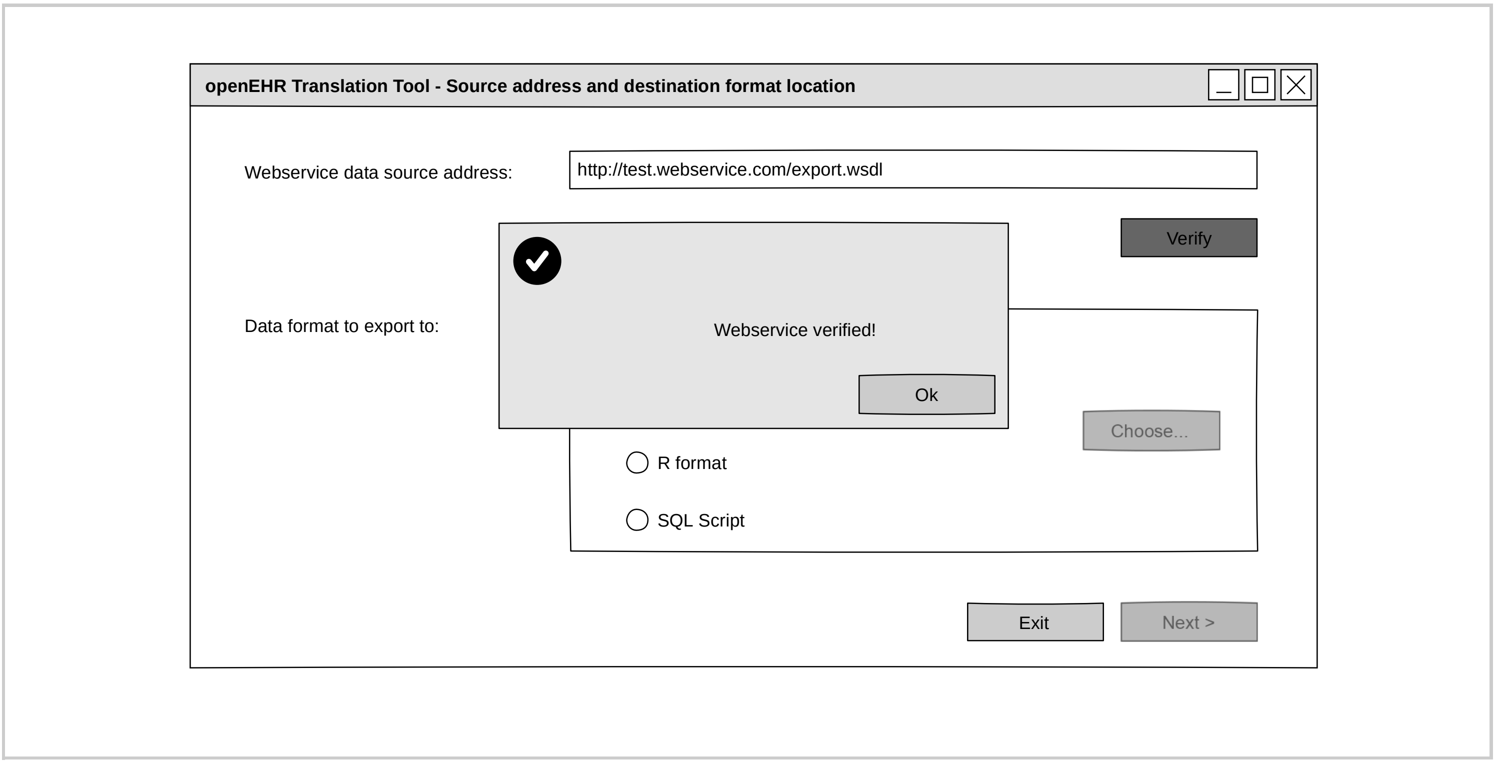
## Screen02


openEHR Translation Tool - Source address and destination format location
Webservice data source address:
Add new...
Verify
Data format to export to:
CSV format
SPSS format
R format
SQL Script
Choose...
Exit
Next >

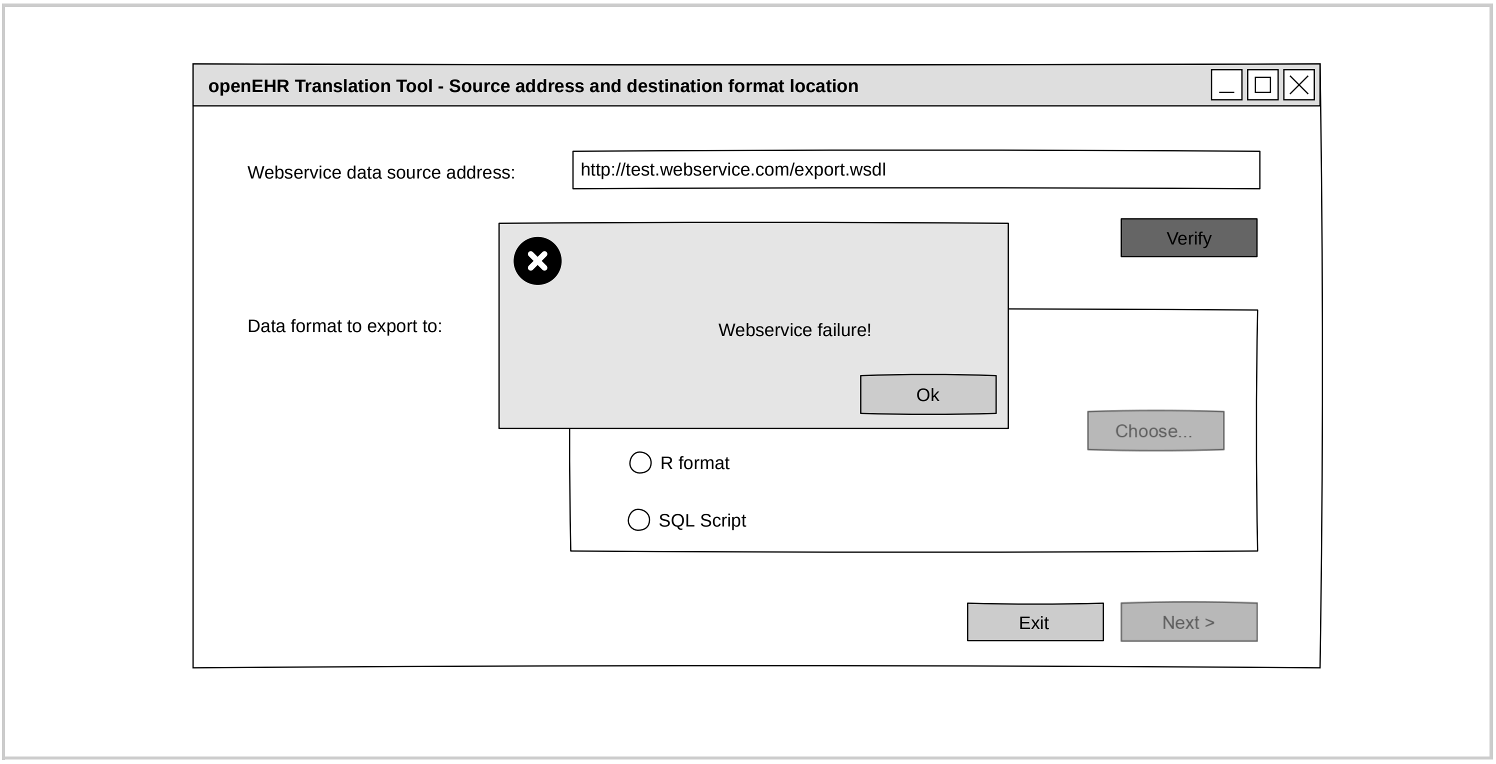
## Screen03


openEHR Translation Tool - Source address and destination format location
Webservice data source address:
Insert New Webservice address
Webservice address:
http://test.webservice.com/export.wsdl
OK
Cancel
Verify
Data format to
Choose...
R format
SQL Script
Exit
Next >

## Screen04a


openEHR Translation Tool - Source address and destination format location
Webservice data source address:
http://test.webservice.com/export.wsdl
Verify
Data format to export to:
Webservice verified!
Ok
Choose...
R format
SQL Script
Exit
Next >

## Screen04b



openEHR Translation Tool - Source address and destination format location

Webservice data source address: http://test.webservice.com/export.wsdl

Verify

Data format to export to:

Webservice failure!

Ok

Choose...

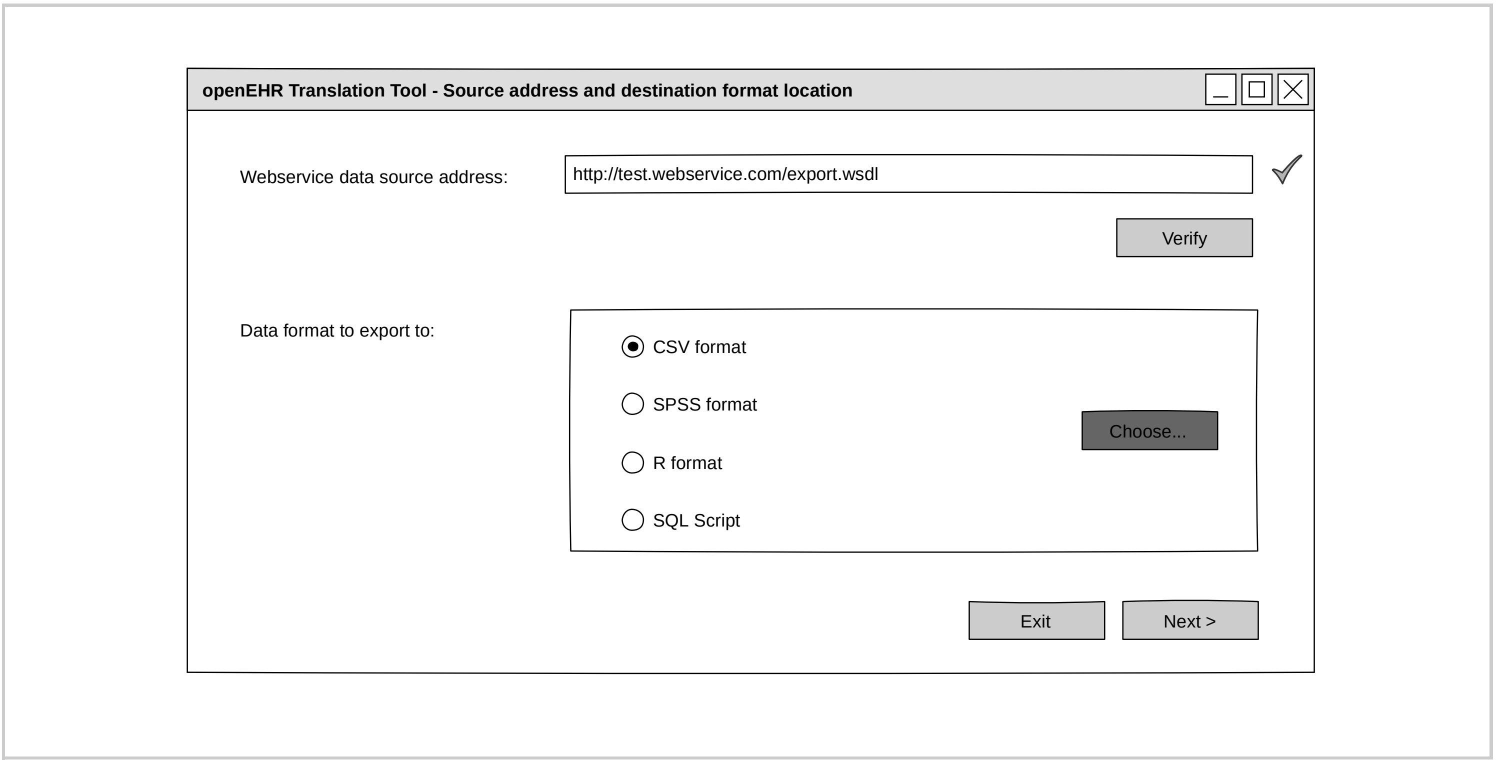
◯ R format

◯ SQL Script

Exit

Next >

## Screen05

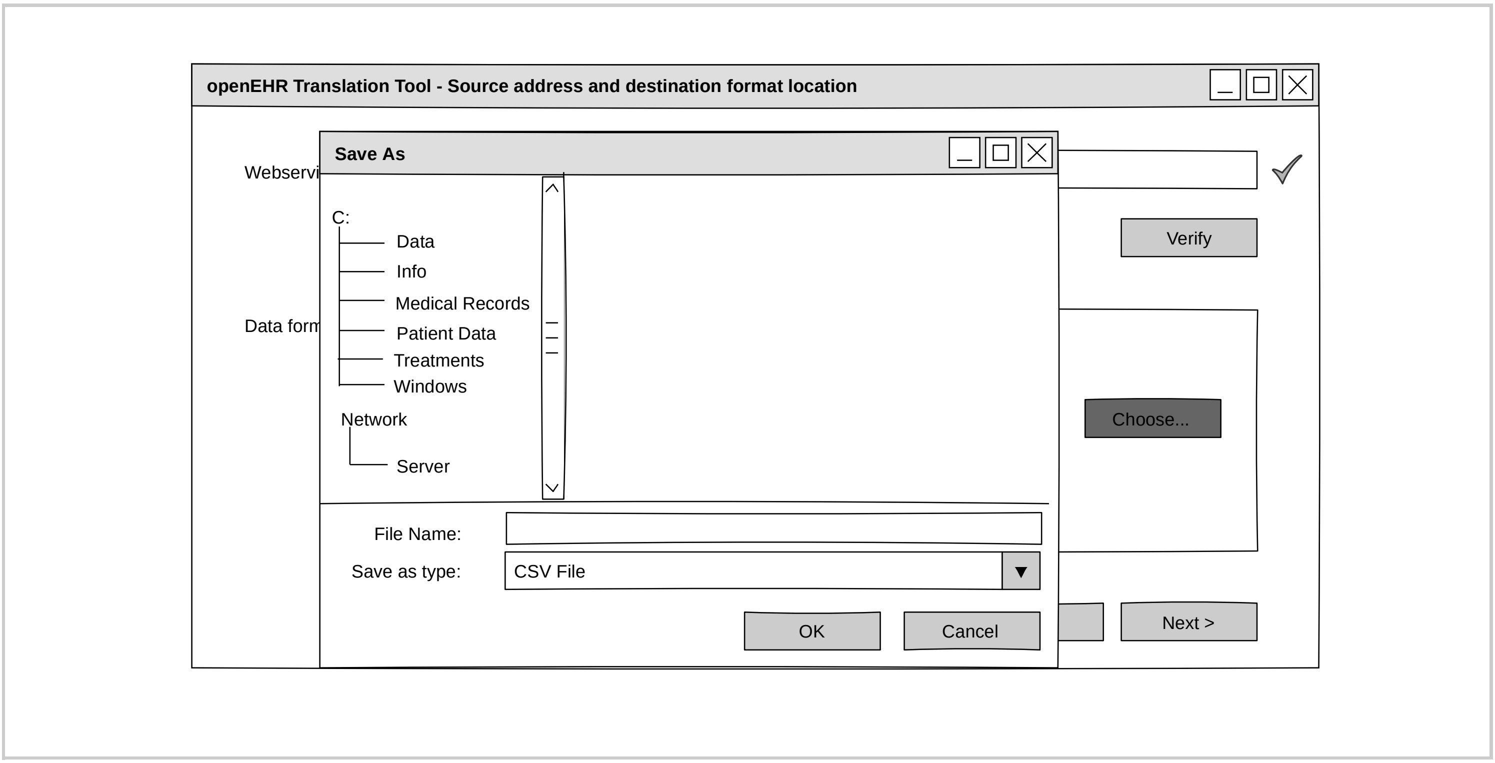

**openEHR Translation Tool - Source address and destination format location**

Webservice data source address: http://test.webservice.com/export.wsdl ✓

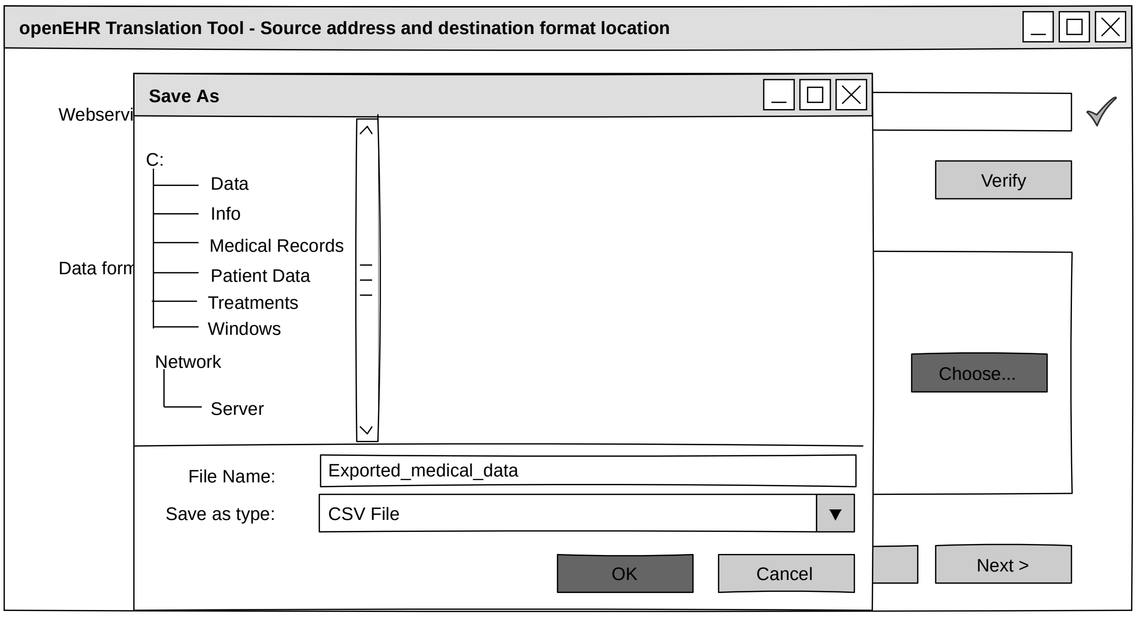
Data format to export to:

[✓ CSV format] SPSS format R format SQL Script

## Screen6


openEHR Translation Tool - Source address and destination format location
Webservi
Data form
Verify
Choose...
Next >
Save As
C:
Data
Info
Medical Records
Patient Data
Treatments
Windows
Network
Server
File Name:
Save as type:
CSV File
OK
Cancel

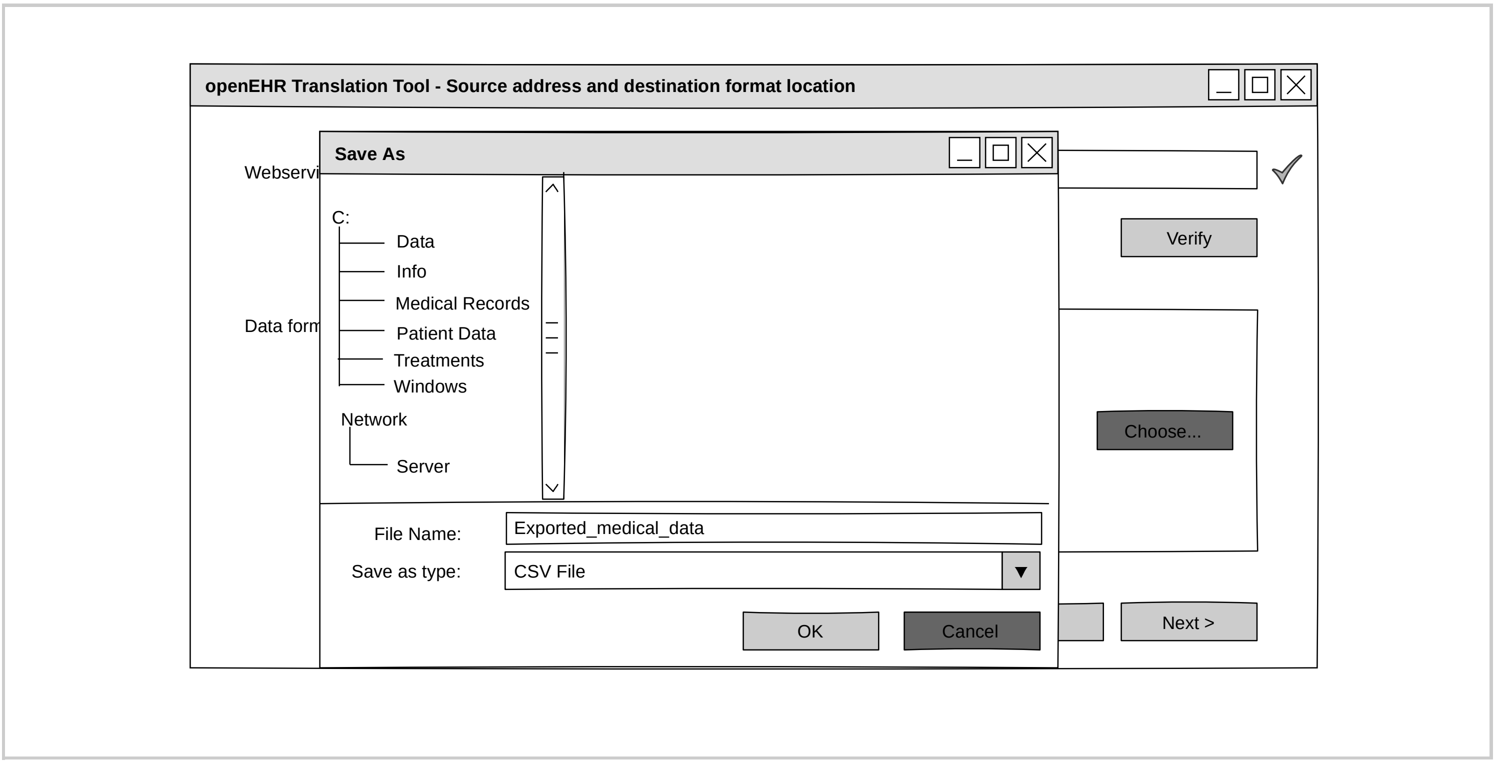
## Screen7a



File Name: Exported_medical_data

Save as type: CSV File

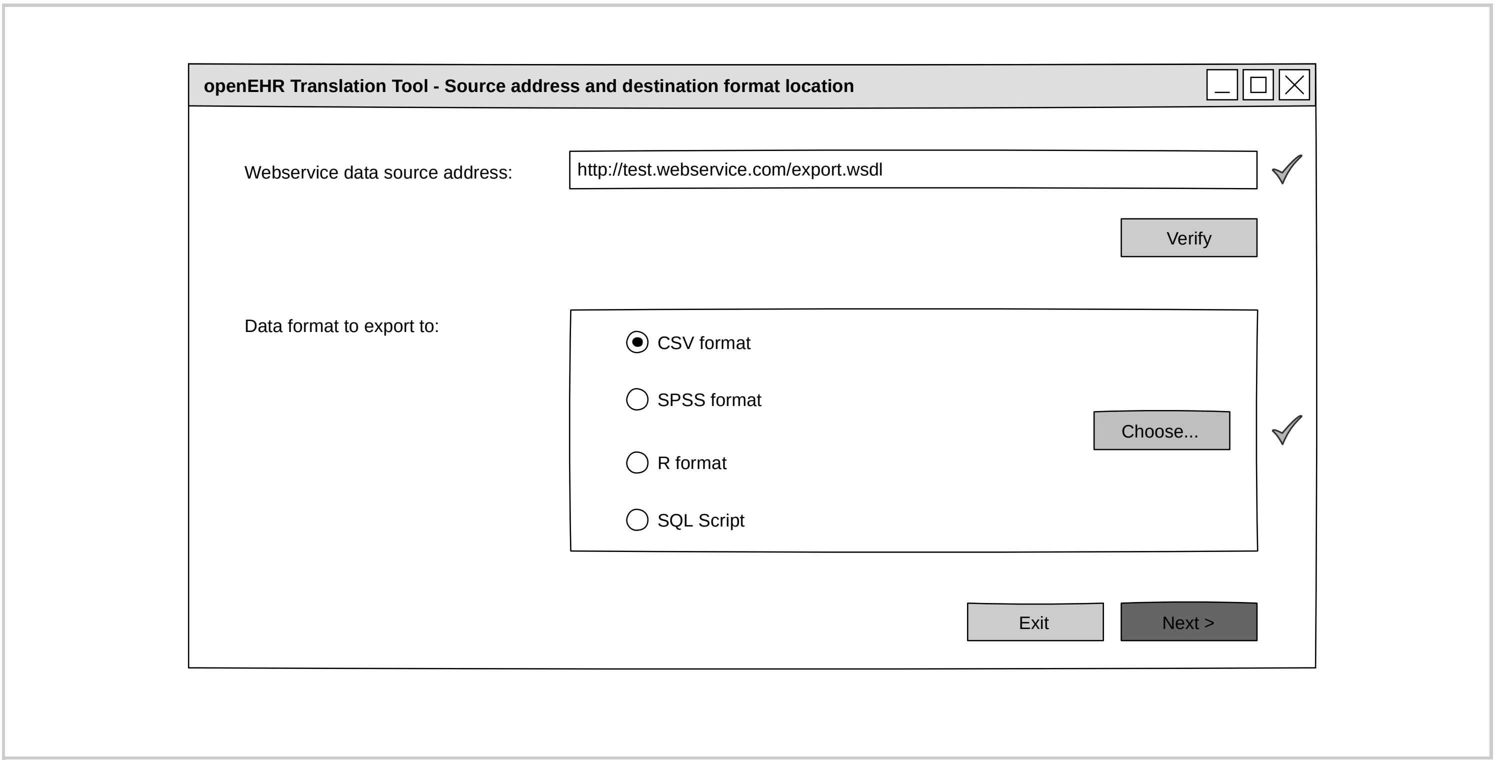
## Screen7b



File Name: Exported_medical_data

Save as type: CSV File

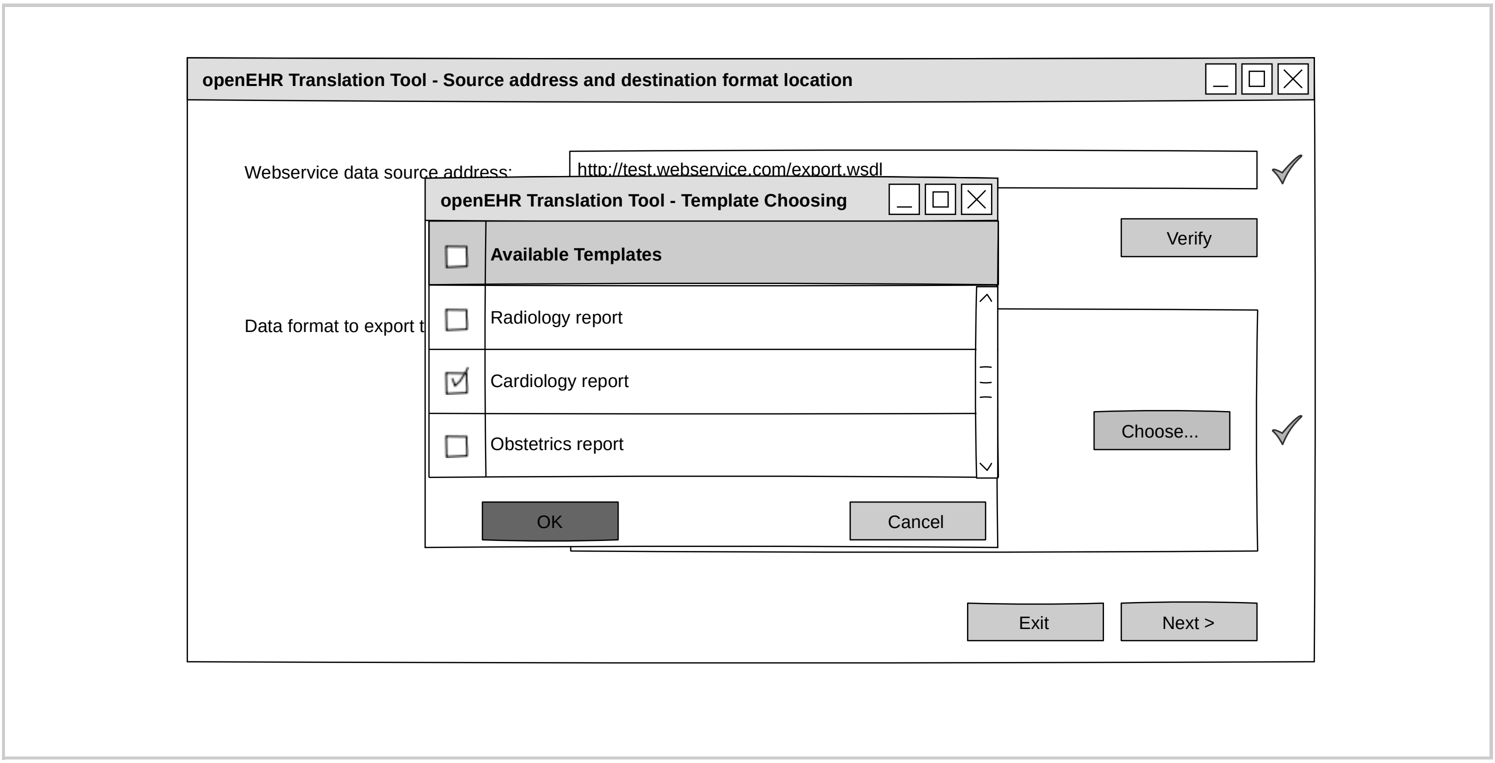
## Screen08


openEHR Translation Tool - Source address and destination format location

Webservice data source address: http://test.webservice.com/export.wsdl ✓

Data format to export to: [✓ CSV format] SPSS format, R format, SQL Script ✓

## Screen09


openEHR Translation Tool - Source address and destination format location

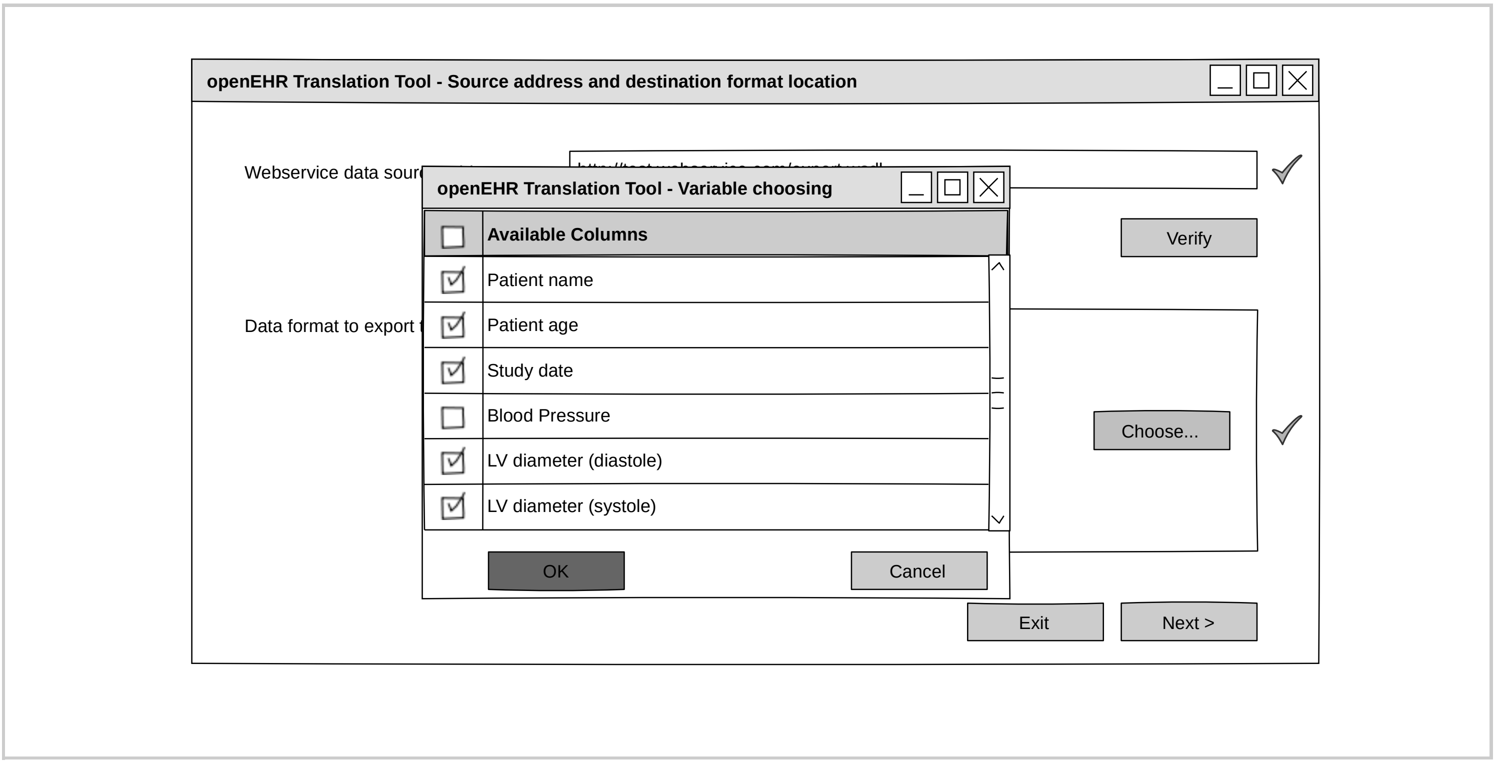
Webservice data source address: http://test.webservice.com/export.wsdl ✓

Data format to export t

openEHR Translation Tool - Template Choosing

| | |
|---|---|
| ☐ | **Available Templates** |
| ☐ | Radiology report |
| ☑ | Cardiology report |
| ☐ | Obstetrics report |

## Screen10


openEHR Translation Tool - Source address and destination format location

Webservice data sourc

http://test.webservice.com/export.wsdl

Verify

Data format to export t

Choose...

Exit

Next >

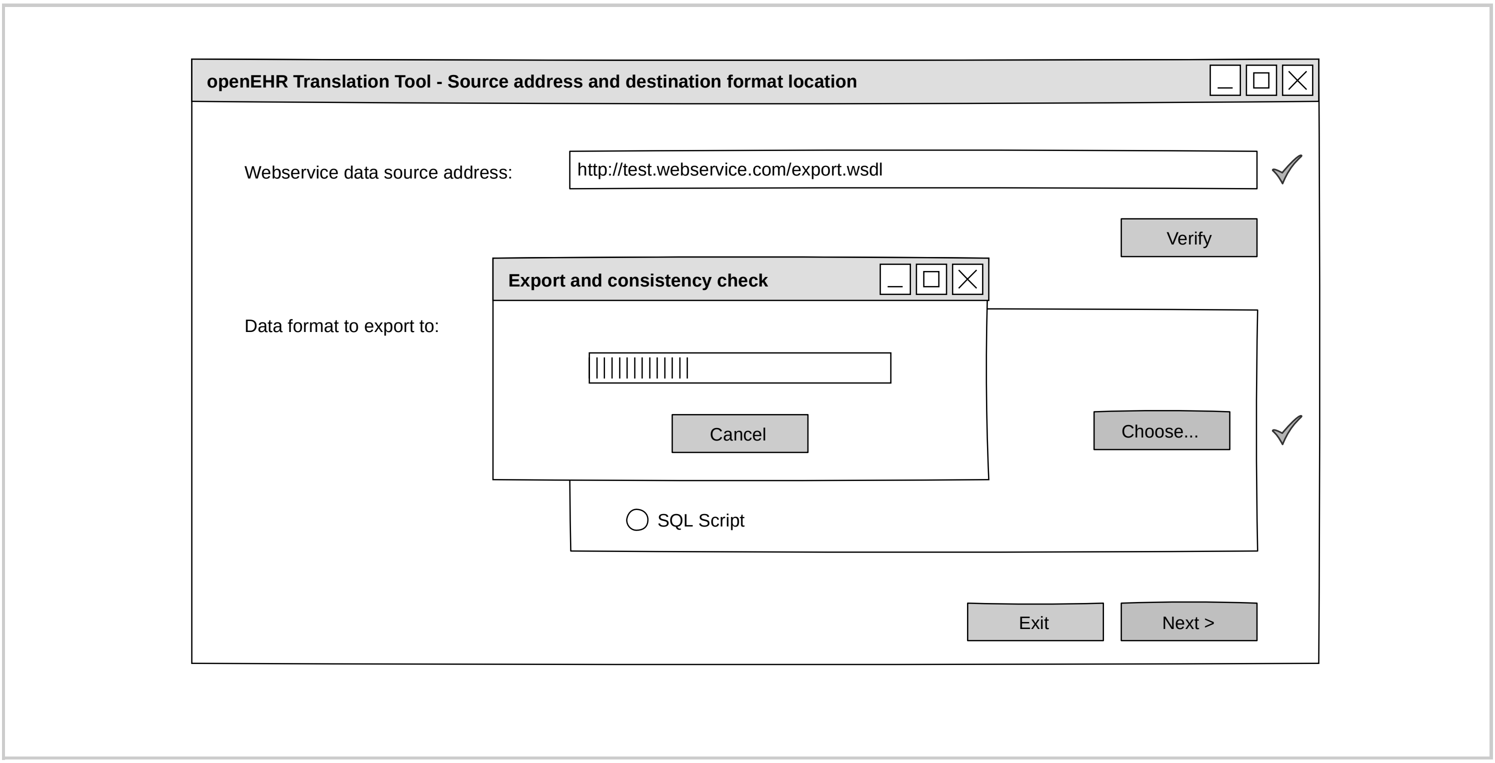
openEHR Translation Tool - Variable choosing

| ☐ | **Available Columns** |
|---|---|
| ✓ | Patient name |
| ✓ | Patient age |
| ✓ | Study date |
| ☐ | Blood Pressure |
| ✓ | LV diameter (diastole) |
| ✓ | LV diameter (systole) |

OK

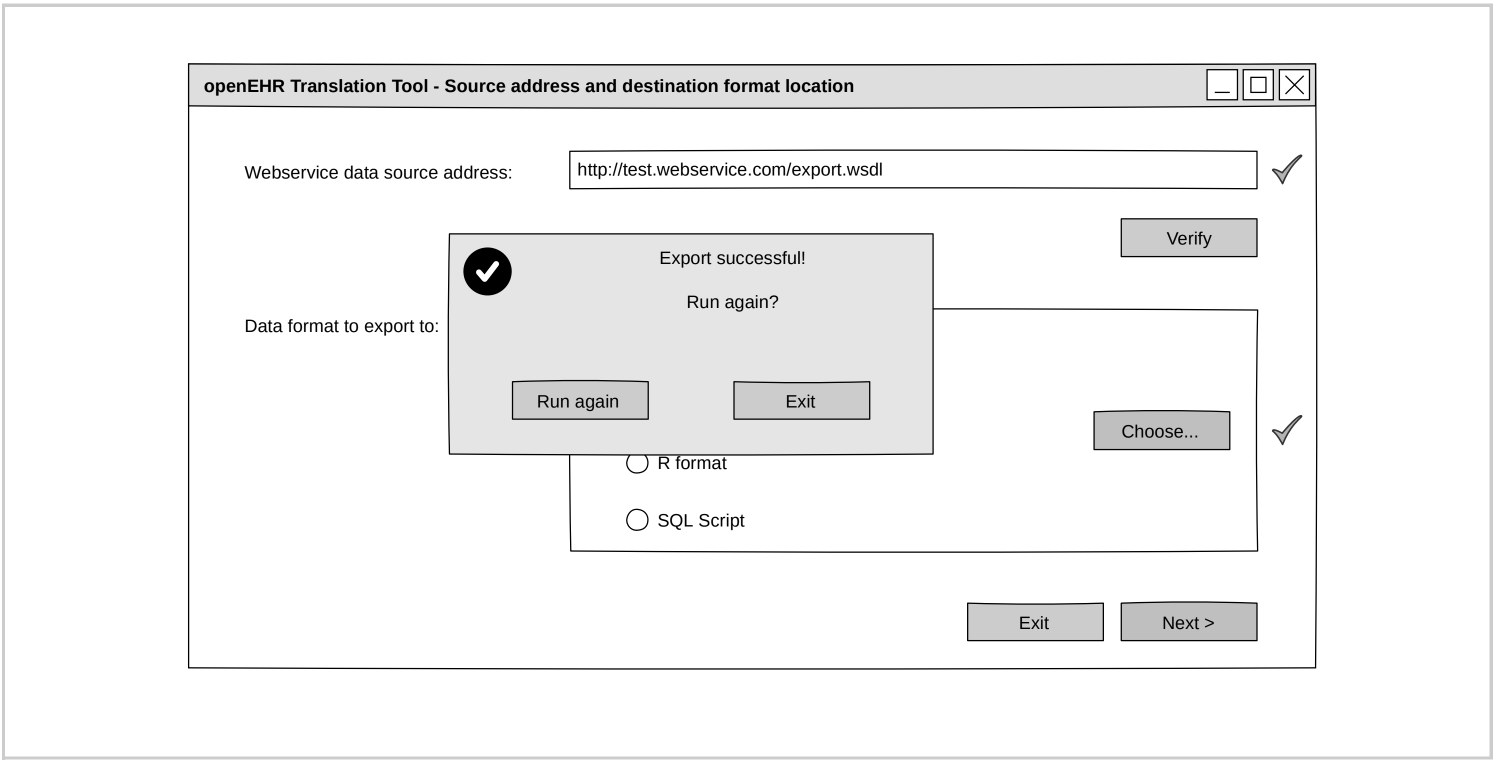
Cancel

## Screen11

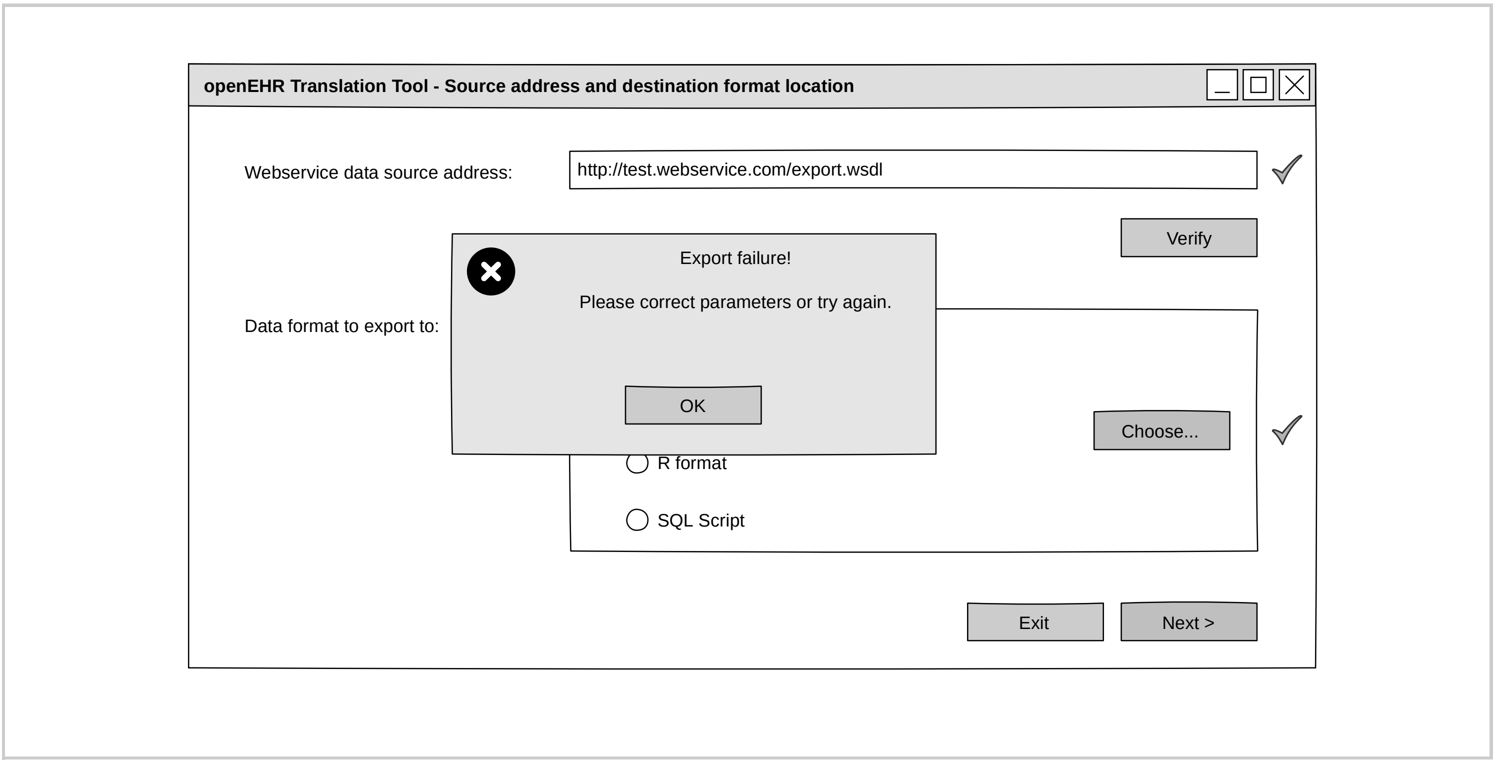


openEHR Translation Tool - Source address and destination format location
Webservice data source address:
http://test.webservice.com/export.wsdl
Verify
Export and consistency check
Data format to export to:
Cancel
Choose...
SQL Script
Exit
Next >

## Screen12a


openEHR Translation Tool - Source address and destination format location
Webservice data source address:
http://test.webservice.com/export.wsdl
Verify
Data format to export to:
Export successful!
Run again?
Run again
Exit
Choose...
R format
SQL Script
Exit
Next >

## Screen12b


openEHR Translation Tool - Source address and destination format location

Webservice data source address: http://test.webservice.com/export.wsdl ✓

Verify

Data format to export to:

Export failure!

Please correct parameters or try again.

OK

Choose... ✓

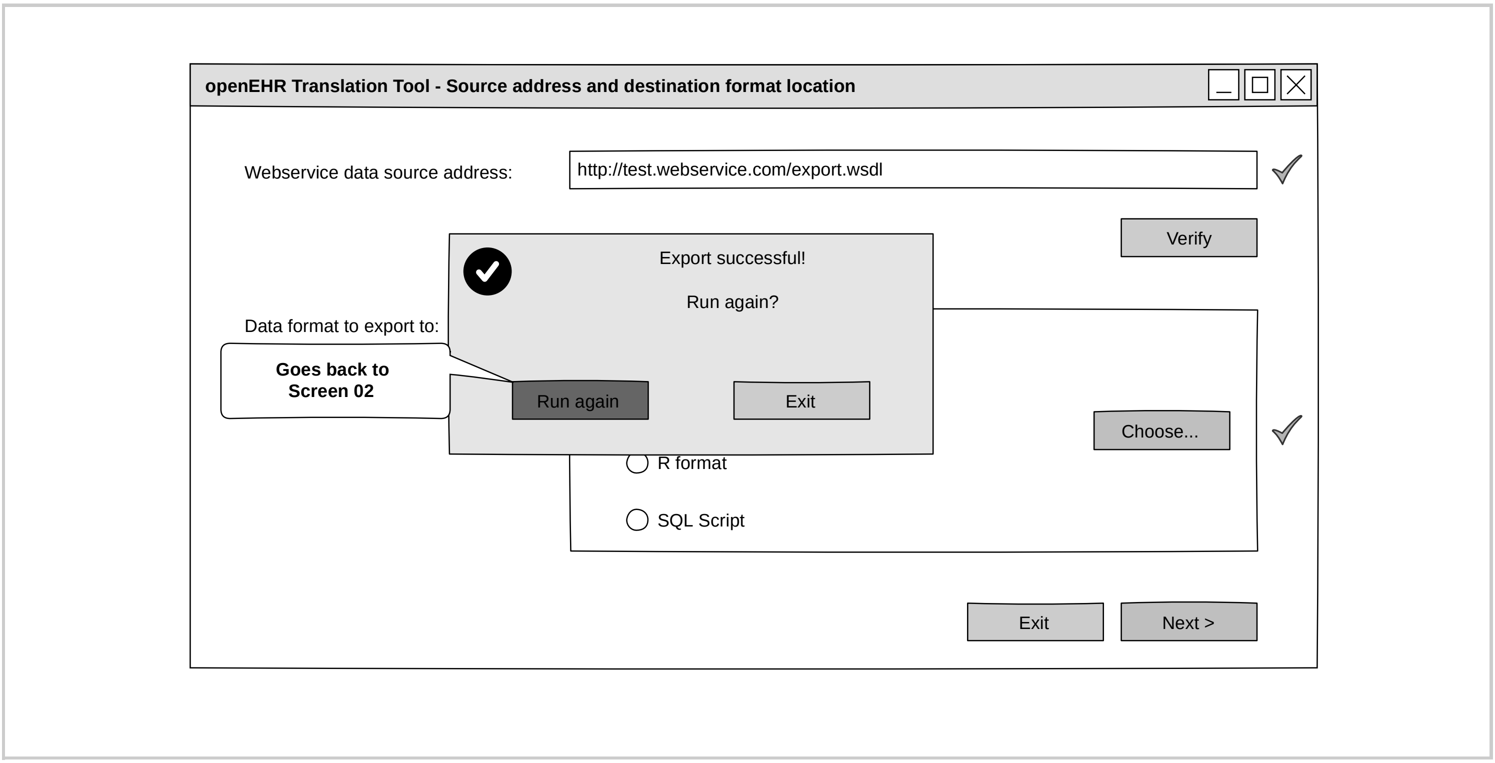
◯ R format

◯ SQL Script

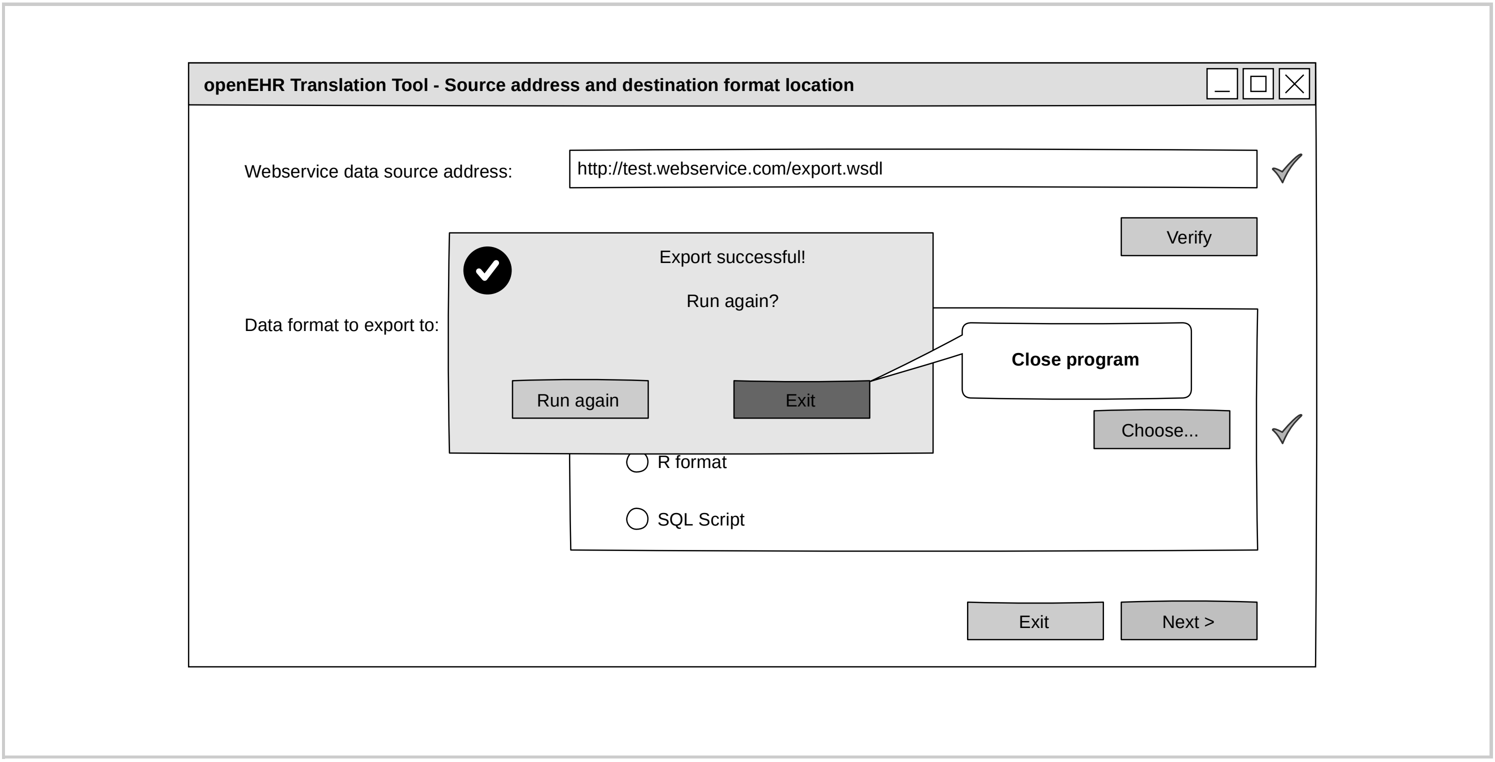
Exit

Next >

## Screen13a


openEHR Translation Tool - Source address and destination format location
Webservice data source address:
http://test.webservice.com/export.wsdl
Verify
Data format to export to:
Export successful!
Run again?
Run again
Exit
Goes back to Screen 02
Choose...
R format
SQL Script
Exit
Next >

## Screen13b



openEHR Translation Tool - Source address and destination format location

Webservice data source address: http://test.webservice.com/export.wsdl ✓

Export successful!

Run again?

Data format to export to:

R format

SQL Script