SYSTEMATIC ELICITATION AND GATHERING FOR AUTOMATIC REQUIREMENT PROCESSING IN DECISION SUPPORT SYSTEMS

Master of Informatics Engineering (MEI)

Author: *Victoria Gabante Guerra*

Directors: *Oscar Romero*
Petar Jovanovic

Universitat Politècnica de Catalunya
October, 2018

Abstract

Decision Support Systems (DSS) are a crucial component in the Business Intelligence world. They are used to give valuable and actionable insights as well as relevant information in a business or organization. These systems are mainly built based on the stakeholders' requirements and the underlying data sources. As data becomes more and more abundant, the necessity for properly storing and later consuming this information becomes crucial. In addition to this, as enterprises start to require real-time decision making and quicker adaptations on the system's requirements, DSS need to be designed and built in an agile and efficient way.

Therefore, this thesis aims to analyze and compare two possible methodologies for designing a Multidimensional Model (MD) Schema and Extraction, Transformation and Loading (ETL) back – end process, needed to ultimately build the DSS system: the traditional and the semi-automatic approach. The traditional approach refers to typically used tools in the BI commercial market and apply the manual labor required for creating the design. The semi-automatic approach refers to all tools that allow to create the design almost automatically. For this, we will be analyzing a semi-automatic tool developed at Universitat Politècnica de Catalunya called Quarry. Quarry assists different stakeholders in the incremental Data Warehouse (DW) design. Mainly, it receives the system's requirements as an input, and performs the physical DW design automatically.

Additionally, the intention of this work is to base the analysis of the mentioned methodologies in an existing use case: the WHO Chagas disease surveillance system's case study. Therefore, we will first mention the Requirements Engineering (RE) process applied to obtain the final DSS system's requirements: the Requirements Engineering for Decision Support Systems (RE4DSS). This RE approach has proven in previous works to be effective for DSS projects, so we will use it as a preparation phase before starting to analyze the two design methodologies that will be introduced in the present thesis. Finally, at the end we will give some comparative aspects between the traditional and semi-automatic approaches in order to have a clearer view of the benefits and limitations for each of them.

Acknowledgements

My deepest gratitude for my thesis directors. Special thanks to Petar Jovanovic, who provided his continuous guidance and support through all the work.

To my family, who no matter the distance continued encouraging and inspiring me to pursue this degree. Thanks to their love and support I have been able to get the strength to improve myself.

To my friends, who have accompanied me through this process and have ultimately become family.

To God, for giving me this amazing opportunity.

Contents

List of Figures

List of Tables

Chapter I
Introduction

In the last few years of this technology era, the amount of information and data available has become overwhelming. Many new technology trends and modern devices have intervened to help track and obtain more and more information about daily transactions, customer details, marketing trends, and more. For instance, in the health industry, it is possible to have complete health reports about patients around the world, getting and tracking specific diseases as well as specific characteristics of the place where the disease is more frequently found, or details about the people that is getting infected. However, it is important to have the appropriate means to make sense of all of this.

According to Gartner [1], BI is a term that includes all the applications, infrastructure, best practices and tools that enable the analysis of information, in order to improve and optimize decisions and performance in an organization. Thus, there is an important necessity to process and organize data from different sources, in order to extract useful information for businesses. This information will allow managers and leaders inside their organizations to make crucial decisions in order to help the company grow and develop. For that reason, the IT department needs to deploy a common repository, with properly integrated data to be later consumed through different applications efficiently. To populate this repository, they will need to make use of a proper Extract, Transform and Loading process (ETL process) in order to extract and transform data coming from many heterogeneous data sources, and then load it into the mentioned data repository. This task may be difficult not only due to the complexity of the ETL process, but because of the extensiveness of the Requirement Engineering process. Furthermore, it is demonstrated that the majority of projects fail due to the lack of clearness in the requirements elicitation process from end users. In fact, according to a study from the Standish Group, the inappropriate analysis of requirements is one of the top reasons that leads to project failures. Moreover, during this study the figures revealed the following: “43% of the IT projects were challenged (late, over budget, and/or with less than the required features and functions) and 18% failed (cancelled prior to completion or delivered and never used)” [10].

A traditional methodology commonly used for these kind of projects is to elicit requirements first, and then starting to develop the ETL design with commercially known software (such as Pentaho, for example). This process requires human intervention for the creation of the Multidimensional Model (MD) model and ETL design but it may be highly error prone, and it does not have a clear methodology to guarantee the completion of all the requirements demanded by the stakeholders. Under these circumstances, to help developers and business owners through this process, there has been a number of recently developed semi-automatic tools that assist the IT team through the development of the MD model and ETL design. In fact, these tools receive a number of inputs such as the data sources and the requirements in a machine-readable format, and they are able to produce the MD model as well as the ETL design in a semi-automatic fashion. Thus, for the purpose of this thesis, we will explain the role of a new tool called Quarry, developed by professors

from Universitat Politècnica de Catalunya. Quarry supports both sides during the requirements elicitation process and the development phase. The tool comes with a user interface that assists end users to describe their information requirements, which are later translated into a machine-readable format, allowing their automatic processing and translation into the MD model and ETL design. Therefore, this approach guarantees that the system will be created fulfilling all the requirements and it helps the stakeholders to have a clearer picture of the general goals for the project.

However, although this type of tools may be very helpful for creating a BI system more efficiently, there are steps in the process that are difficult to be automated. For this reason, this project's main objective is to perform a comparison between the traditional and semi-automatic methods (in this case Quarry) to generate the MD design and ETL process design. This analysis will be done using a real-life use case and applying the typically used tools for each approach. Then, we will analyze the benefits and limitations of the semi-automatic approach comparing it to the traditional approach.

Particularly, the real-life use case consisted in developing a BI system for the World Health Organization that will help them track and control the Chagas disease. The World Health Organization (WHO) is a United Nations (UN) agency responsible to provide leadership on global health matters by setting standards and providing technical support to monitor the health trends [2]. Among the many studies made by this organization, there was a report launched in 2010 about Neglected Tropical Diseases (NTDs) [3] which included the Chagas disease. This disease is considered life-threatening, caused by a parasite named "protozoan parasite", which is transmitted to humans by the feaces of infected triatome bugs. According to the report, Chagas disease is mostly found in 21 Latin American countries and it is estimated that around seven to eight million people are infected worldwide. As a consequence of this worrying situation, the WHO decided to present a resolution directed to control and eliminate Chagas disease. A few years later, in 2013 a new strategy was formulated, which consisted in creating a Chagas Information Database (CID) for surveillance on the disease's trends around the world. In particular, the idea was to exploit this information through analytical tools such as dashboards visualization of statistics, etc [3].

With all this in mind, in the following sections the goal is to discuss about two different approaches to create the ETL design and MD model given the stated use case: the traditional approach, followed by the semi-automatic approach. First, In Chapter II the reader may find a brief context for the thesis. In Chapter III, we will begin to explain in more detail the use case and some basic terminology. Then, in the Chapter IV we will continue detailing the requirement elicitation process as a preparation phase for applying the traditional and the semi-automatic methodologies. Finally, we give details about how these two were implemented to later show a comparison between both of them, as well as a few conclusions for the analysis.

Chapter II
Background and Context

Before introducing the real use case and the methodologies applied in this project, it is important to review some key concepts, which are important to have a basic knowledge for a better comprehension of the work. With this in mind, in the following sections we will introduce a brief summary about the technical foundations for this thesis. We will start explaining Decision Support Systems and their importance, followed by Data Warehouses. Particularly, we will explain Multidimensional (MD) models and Extraction, Transformation and Loading (ETL) processes and their role inside a Data Warehouse. Then, we will continue by discussing the field of Requirements Engineering, and explain a specific approach very useful for this type of systems: The Requirements Engineering for Decision Support Systems (RE4DSS approach), which will be used for the requirements elicitation process before applying the selected methodology for the design. Finally, we will make an overview of Quarry, which we will use in the comparison as an example semi-automatic tool.

2.1 Decision Support Systems

Until the mid-1980s, the most crucial information that a business requires to operate was stored in regular operational databases. These databases were populated as a result of daily operational activities, such as sales and management tasks. However, in the digital era, the amount of information that needs to be stored for analytical and strategic purposes is getting higher every year. From transactional operations in sales, providers, and clients to customer satisfaction and preferences, all this information can be recorded and documented for further analysis. For this reason, it became more important to properly organize this information in order to exploit it as much as possible.

With this in mind, nowadays it is very common to have systems that allow managers and directors to visualize information and therefore, obtain proper knowledge about their businesses. They are known by the concept of Decision Support Systems (DSS), which may be defined as *a set of interactive tools designed for processing and analyzing data, and for supporting managers in decision making processes [5]*.

Typically, DSS are management systems (interactive dashboards for example), that are connected to a knowledge engine. Data warehouses (DW) are the most common examples of DSS. They store huge amount of information that may come from heterogeneous sources into one single repository, in an integrated clean way.

2.2 Data warehouse

Data Warehouses are created to support DSSs, acting as a knowledge engine, or a huge data repository where all the information must be properly modeled in a clean, accurate way. In summary, according to Golfarelli & Rizzi's book a data warehouse is "*a collection of methods, techniques, and tools used to support knowledge workers to conduct data analyses that help with performing decision-making processes and improving information resources*"[5]. Given so, DWs must be integrated and consistent repositories, because they extract data from multiple and heterogeneous sources, such as internal production databases or external third party's information systems, which may typically have incomplete or inconsistent data. Therefore, to satisfy the BI system's requirements effectively, this data needs to be transformed and processed before being loaded into the DW.

Furthermore, in DWs it is common to use a special type of querying different to the one used in relational databases. This type of querying is called Online Analytical Processing (OLAP), and it allows to perform complex analyses of the store data. Essentially, it processes huge amounts of records in order to obtain numeric measures, which are able to convey valuable insights about the enterprise. To enable such analysis, data inside the DW must be modeled following the MD model.

2.2.1 Multidimensional Model

2.2.1.1 Overview

A Multidimensional (MD) Model is a very common and widely used conceptual data model in the DW design. Among its many advantages, it allows to make queries and analysis more efficiently and simple, through the OLAP querying [6]. The approach mainly focuses on identifying the key processes within a business (which will be called **facts**) and modelling them. In other words, it may be defined as a bottom-up approach, that will be iteratively built [5]. For example, a **fact** may be the *yearly revenue* of a business.

Furthermore, to obtain information about a **fact**, we typically need to analyze a series of events. Meaning, each fact is described by a set of relevant measures that come from each of the events. So, continuing with the previous example, to obtain the *yearly revenue* of a business we must consider the *daily incomes* and *outcomes* as events to later calculate the *yearly revenue*, which will be our desired fact. In a MD model we also have **measures**. These correspond to numeric values that give a measurement for each of the events and allows to place them in the n-dimensional space. When doing an OLAP query, we define the granularity of the analysis by selecting the detail level for the dimensions. Then, after performing calculations with the selected events, we obtain the measure that gives us insights about the desired fact [5].

Certainly, in an enterprise environment events are too many to be analyzed one by one. For this reason, it is convenient to place them in an n-dimensional space and summarize them (e.g., by

year or *geographical location*). In the n-dimensional space, axes are called **dimensions**, and they are defined as different perspectives used to analyze events [5]. For example, time is considered a dimension and it may be analyzed through many detail levels (years, months, days or even hours). This level of detail defines the granularity of the analysis. This means, the query can be performed in any desired level of detail by doing aggregations or selections (explained in further sections), and the obtained data may be more or less specific according to the granularity of its dimensions.

This concept of dimensions gave life to represent data as cubes in a n-dimensional space, where cube cells are single events that can be quantitatively described by numeric measures, and the cube edges are the dimensions. These dimensions can be analyzed at different detail levels specified by hierarchically structured attributes [5]. For example, we may have incomes (or outcomes) for a particular set of days, months or even years. In the following figure, it is explained how data may be visualized and organized as a cube, according to its dimensions and measures.

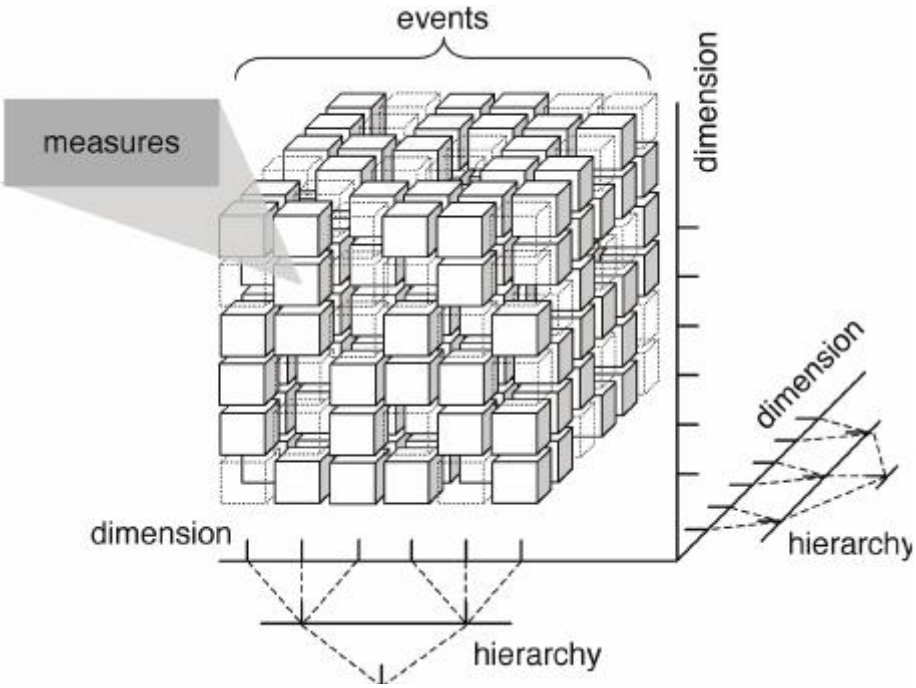


Figure 1. Illustration of Measures and Dimensions for a Cube Multidimensional Model [29]

Normally, the information in multidimensional cubes is difficult for users to manage without using automatic tools, because of the large amount of data. So essentially, there are techniques to reduce the quantity of data and obtain valuable information: restriction and aggregation.

2.2.1.1.1 Restriction (Selections/Projections)

Restricting data means separating part of the data from the cube, to create an analysis field. It is also called making selections and/or projections. The simplest example may be data slicing. This one consists in setting one or more dimensions to a specific value, which will decrease the cube's dimensionality [5].

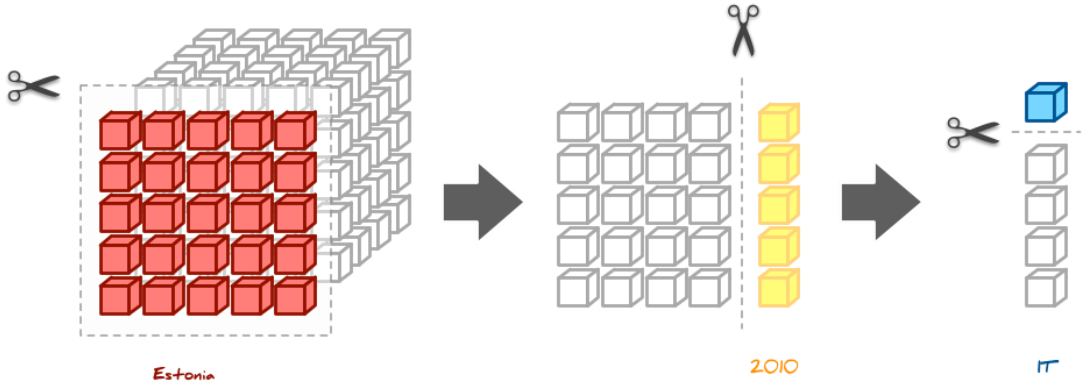


Figure 2. Illustration for Restrictions (Selections and Projections) [30]

2.2.1.1.2 Aggregation

Is a fundamental technique used in multidimensional databases. The main idea is to generate one single event by applying an aggregation function to a group of events, which will summarize the information contained in those events [5]. For example, in the figure below we show the simplest examples for aggregation. The original data has each item by date, product and store. If we aggregate it to have a more general perspective of the information, we could have the amount of products aggregated by month, type and city or simply by month and type.

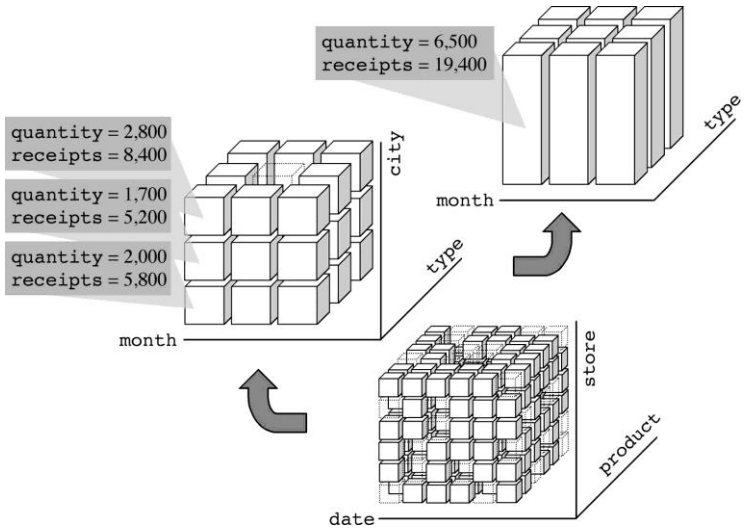


Figure 3. Illustration for aggregations in a Multidimensional Model[5]

2.2.1.2 Conceptual Modeling phase: Dimensional Fact Model

Continuing with the Multidimensional design concept, we will explain a useful approach to help on the design of the MD model during the development phase, the Dimensional Fact Model (DFM)
.

DFM is a graphical representation devised to support the conceptual modeling phase in a Data Warehouse project. It is very intuitive and sufficiently simple to be used by analysts and non-technical users as well [5]. In the figure, we can observe an example for a DFM, where we can visualize some attributes: facts, measures and dimensions.

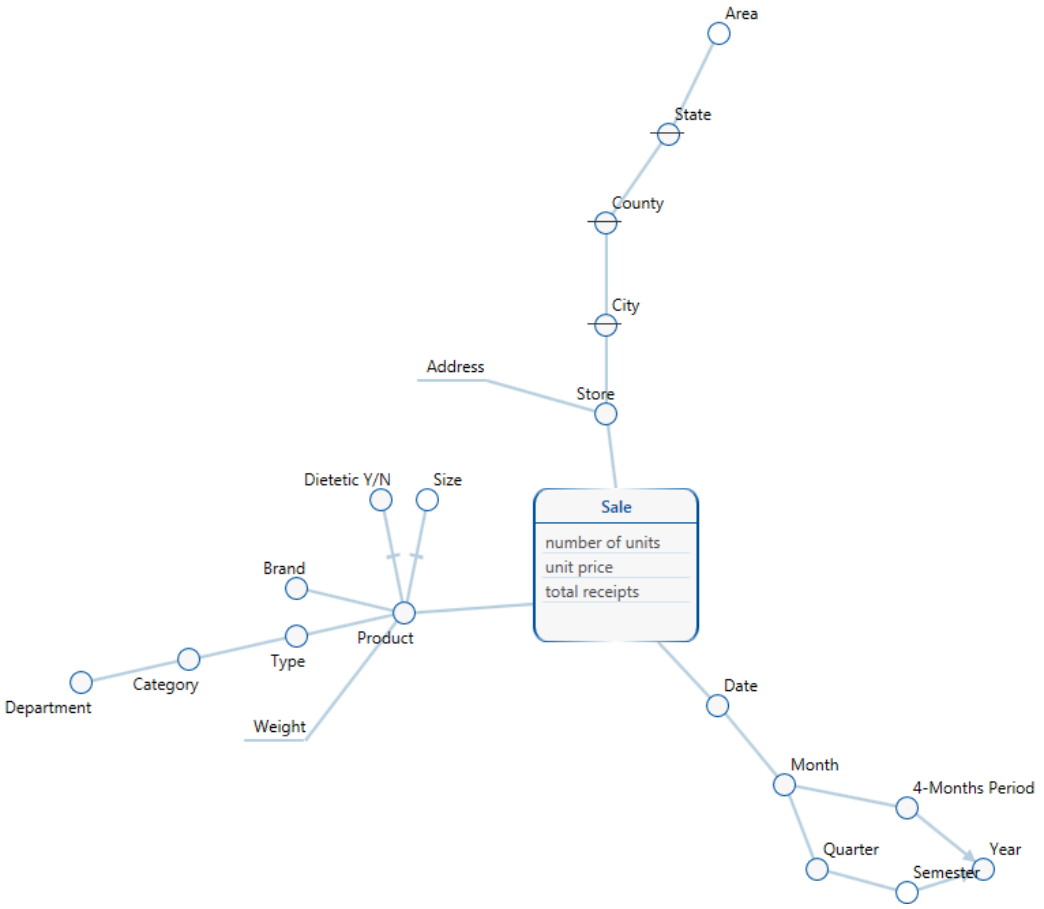


Figure 4. Dimensional Fact Model example [8]

- Facts: The concepts to be analyzed. Typically, events that will take place in a company and the ones we wish to extract information about.

- Measures: Is a numerical property of a fact, which provides a quantitative attribute relevant to the analysis.

- Dimensions: They are finite properties, that help to define the granularity of the representation and analysis. Dimensions may also have attributes, which give additional information for the analyzed property

- Hierarchy: Represented as a tree whose nodes may model many to one associations. They are fundamental to define possible aggregations and selections in the analysis of the MD model.

An advantage of using this approach is to have a clear representation of multidimensional concepts (e.g., attributes, measures and hierarchies). For this reason, it is considered very flexible and it can be used from the initial DW life-cycle steps, to rapidly devise a conceptual model to share with customers[8].

After having a conceptual data model clearly defined that satisfies the requirements for the system, we may begin creating the ETL design. In the next section, we will briefly explain in what ETL exactly consists.

2.2.2 Data Staging and ETL processes

In a data warehouse, information may come from multiple and heterogeneous sources and it is necessary to store them in a well-defined architecture that can be scalable and efficient. This architecture needs to be flexible enough, to take into consideration the business growth and possible expansion. This process is usually complex, as the data usually needs to be cleaned and filtered.

Independently of the chosen architecture, in every DW, data staging and ETL processes are very much needed. ETL refers to Extraction, Transformation and Loading of the data. An example of a generic architecture can be found in the following figure.

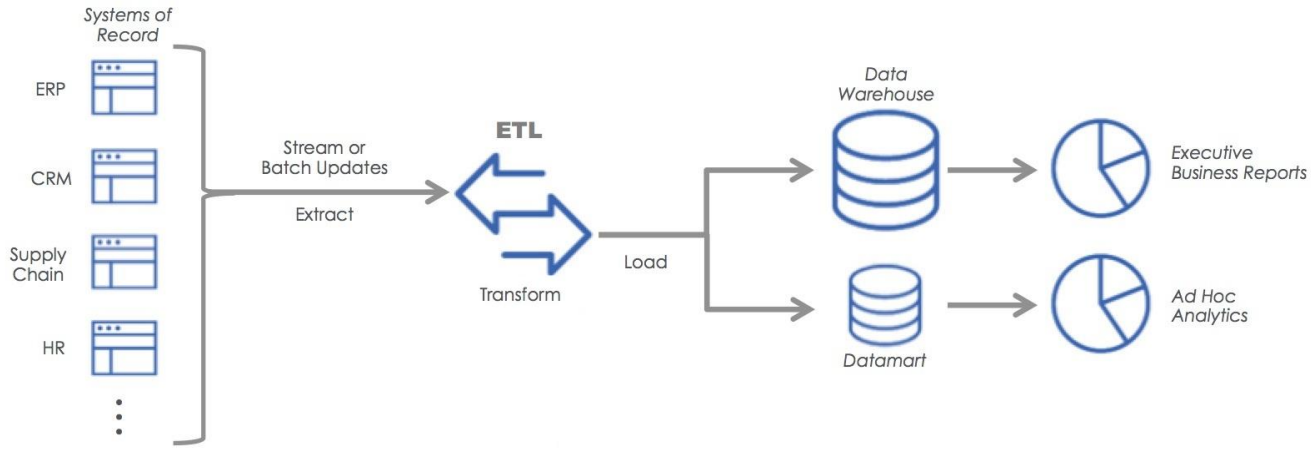


Figure 5. Extract, Transform and Loading process for a Data Warehouse [9]

As can be seen in Figure 5, first data is extracted from the different available data sources. Then, we start the ETL process in order to clean and integrate data into one single data repository. The stages of an ETL process can be defined as follows:

2.2.2.1 Extraction

Is the phase where relevant data is obtained from the sources. The frequency of this step may vary depending on the use case.

2.2.2.2 Cleansing

Process that takes care of frequent data mistakes, such as: duplicate data, inconsistent values, missing data, among others.

2.2.2.3 Transformation

Converts data from the operational database into the data warehouse multidimensional schema. This step is made by applying complex processes that allow the mapping between the source data structures into the data warehouse schema modeled following MD integrity constraints.

2.2.2.4 Loading

Last step of the process, where previously transformed and cleaned data is actually loaded into the data warehouse.

2.3 Requirements Engineering

2.3.1 Requirements Engineering overview

For the context of this thesis, it is very important to introduce the main concepts of Requirements Engineering (RE), given that it sets the basics for defining the requirement elicitation process before developing the DW system. For this reason, in the following sections we will give a brief overview for RE concepts, and summarize the RE framework used for this thesis.

When developing an IT project, the whole design and implementation process depends on the users' requirements and what they expect to achieve with the system. This is why there must be a methodology to gather these requirements in an effective way. Later in this thesis, this methodology will be described specifically for DSS projects, but in this section, the objective is to introduce the concept of Requirements Engineering, for any IT project.

According to a study from the Standish Group, the inappropriate analysis of requirements is one of the top reasons that leads to project failures. Furthermore, during this study the figures revealed the following: "43% of the IT projects were challenged (late, over budget, and/or with less than the required features and functions) and 18% failed (cancelled prior to completion or delivered and never used)" [10]. Actually, according to the same study, successful projects (completed on-time and on-budget) were only 39%, which is less than half. This study gives enough evidence to conclude that RE should receive special attention during the development of any project. In fact, researchers agree that the RE process has moved from being the first phase in

software development cycle to a key activity that spans across the complete software development life-cycle in many organizations.

An accurate definition for RE could be the following [11]:

Requirements Engineering is a cooperative, iterative and incremental process which aims to ensure the following:

1. All relevant requirements are explicitly known and understood at the required level of detail.

2. Sufficient agreement about the system requirements is achieved between the stakeholders and the involved actors.

3. All requirements are documented and specified in compliance with the relevant documentation/specification formats and rules.

2.3.2 Requirements Engineering Framework and Requirements Engineering for Decision Support Systems (RE4DSS)

In order to introduce the RE4DSS framework, we will first introduce the Requirements Engineering framework presented by Klaus Pohl, which consists on a process that helps to materialize the project into a vision, making it possible to define the major structural elements of the Requirements Engineering process.

Basically, this framework provides a well-structured base for the fundamentals of requirements engineering. It is fairly simple to adapt and utilize regardless of the software project and it has been successfully validated and transferred to the industry using it as a reference for training their staff. Its main goal is to convert the stakeholder's needs to *goals*, *scenarios* and *solution-oriented requirements* by iterating the following activities: *elicitation*, *documentation*, and *negotiation [12]*.

The "Requirements Engineering Framework" consists of three main building blocks (System Context, Core Activities, and Requirement Artifacts) and two cross-sectional activities (Validation and Management), as shown in the figure.

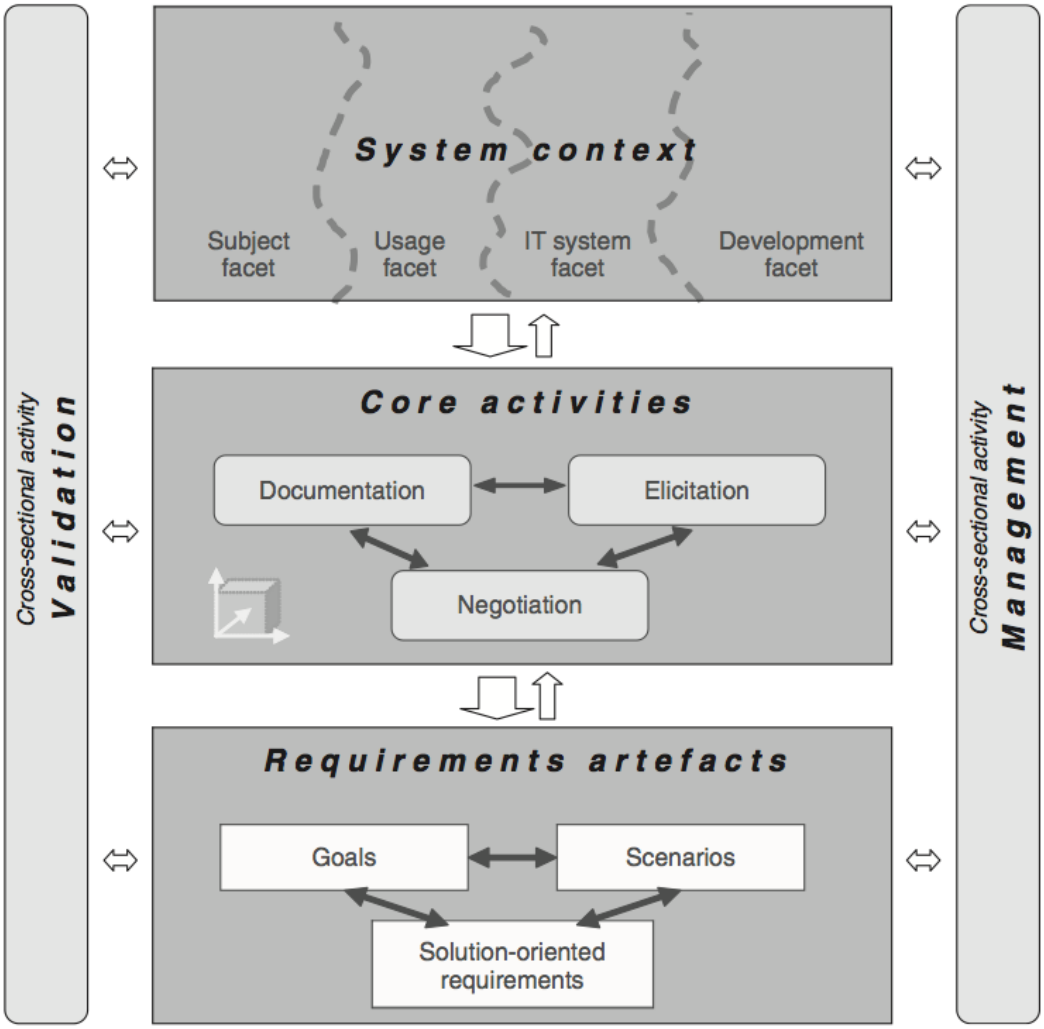


Figure 6. Requirements Engineering Framework [12]

This framework is the starting point for creating the RE4DSS approach. In the following section, we will explain in detail its integrating components.

2.3.2.1 RE4DSS overview

As mentioned before, the framework from Klaus Pohl, constitutes the base for creating a more specific framework created at UPC specifically designed for DSSs, which is called Requirements Engineering for Decision Support Systems (RE4DSS). This block-oriented approach aims to improve these practices previously described by systematizing the requirements process, and accompanied by a system called Quarry (previously called GEM - *Generating ETL and MD designs [13])* it provides a semi-automatic approach to further facilitate the design of DWs. Basically, it means translating each of the requirements obtained from the RE4DSS framework, to finally produce a validated and verified ETL and MD designs.

However, independently of using a semi-automatic approach or not, it is a very useful technique to gather requirements for DSS systems, so in this thesis it will be used as a preparation phase before applying both methodology approaches: the traditional and the semi-automatic one.

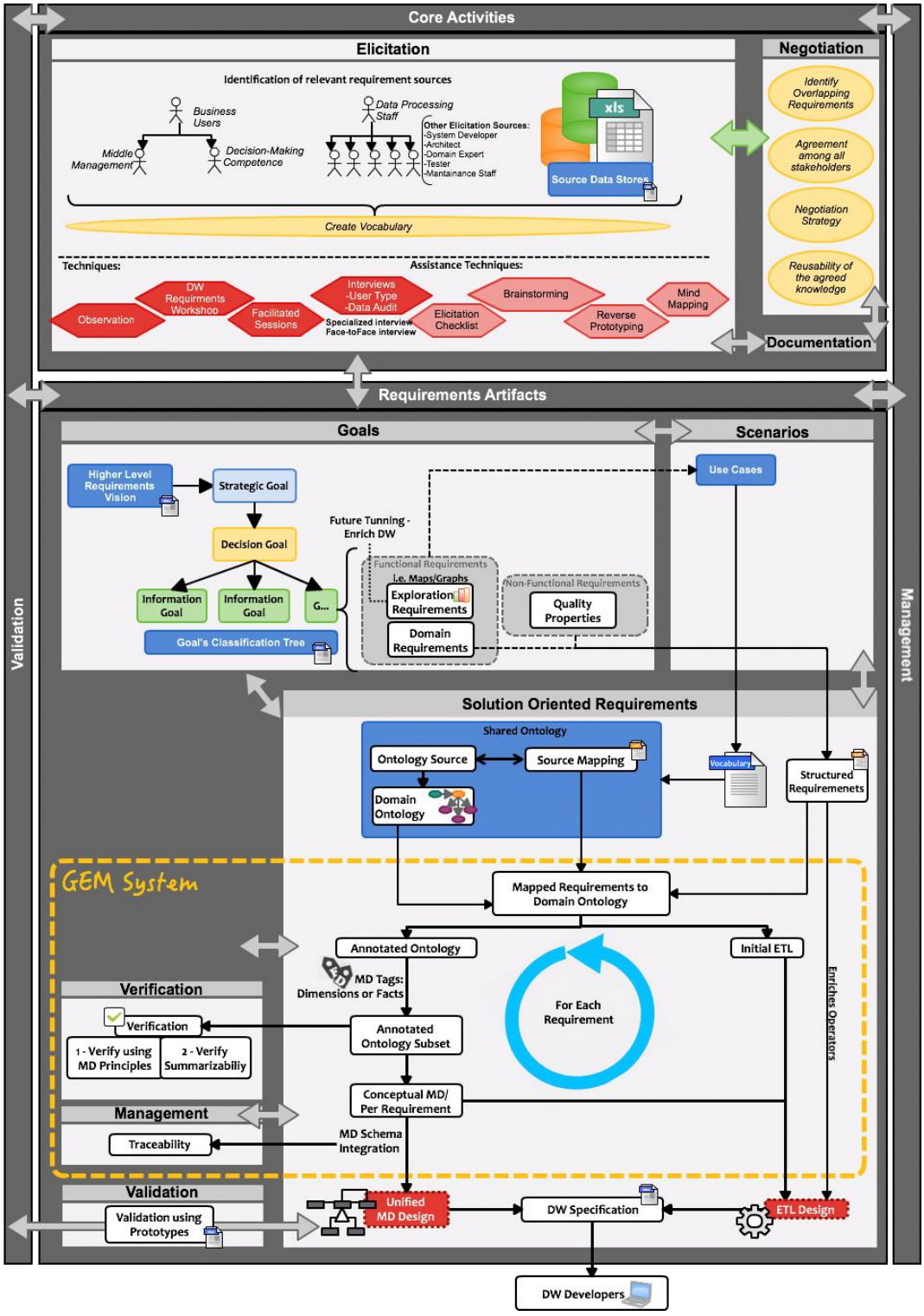


Figure 7. Requirements Engineering for Decision Support Systems (RE4DSS) approach [12]

The RE4DSS approach architecture can be observed in the Figure 7. According to *[12]*, the building blocks can be explain as follows

2.3.2.1.1 Core Activities

The Core activities are mainly three - Elicitation, Documentation, and Negotiation, which are performed iteratively with the aim to establish the vision within the existing context. Documentation allows to create a common reference among stakeholders, which may come from different backgrounds. Elicitation, on the other side, aims to elicit requirements from the stakeholders by using different techniques, for example, interviews, conversations, workshops, etc. It starts with the main sources and business users or stakeholders. Finally, Negotiation comes from the inevitable case in which requirements conflict with each other. Thus, it involves communication to reach an agreement between stakeholders. These activities are used as a base in order to elicit, negotiate and document the Requirements Artifacts.

2.3.2.1.2 Cross-sectional Activities

Validation and Management are the cross-sectional activities that support the core activities. These should be present during the entire project and aim to validate if the core activities satisfy the initial criteria, as well as observing and supervising the activities and execution of the requirement artifacts.

2.3.2.1.3 Requirements Artifacts

Requirement Artifacts consist in three components: Goals, scenarios, and solution-oriented requirements. Goals define the specific objectives that need to be satisfied. Scenarios represent interactions between the system and its actors. They also reveal useful insights about the RE process by illustrating if these scenarios satisfy the requirements or not. As a consequence of the previous two elements, the solution-oriented requirements are defined. They are created to specify a deeper required level of details that must be agreed by all stakeholders as completely and free of conflicts as possible between goals.

Goals and Scenarios

One of the most important roles of elicitation activity is to elicit the goals starting from the high-level requirements. In this approach, goals can be modeled by using a Goal Classification tree. This tree is divided into three levels: strategic, decision and information requirements. At the lowest level, information goals (requirements) must be then classified as one of these three: *Exploration Requirements*, *Domain Requirements* or *Quality Properties*.

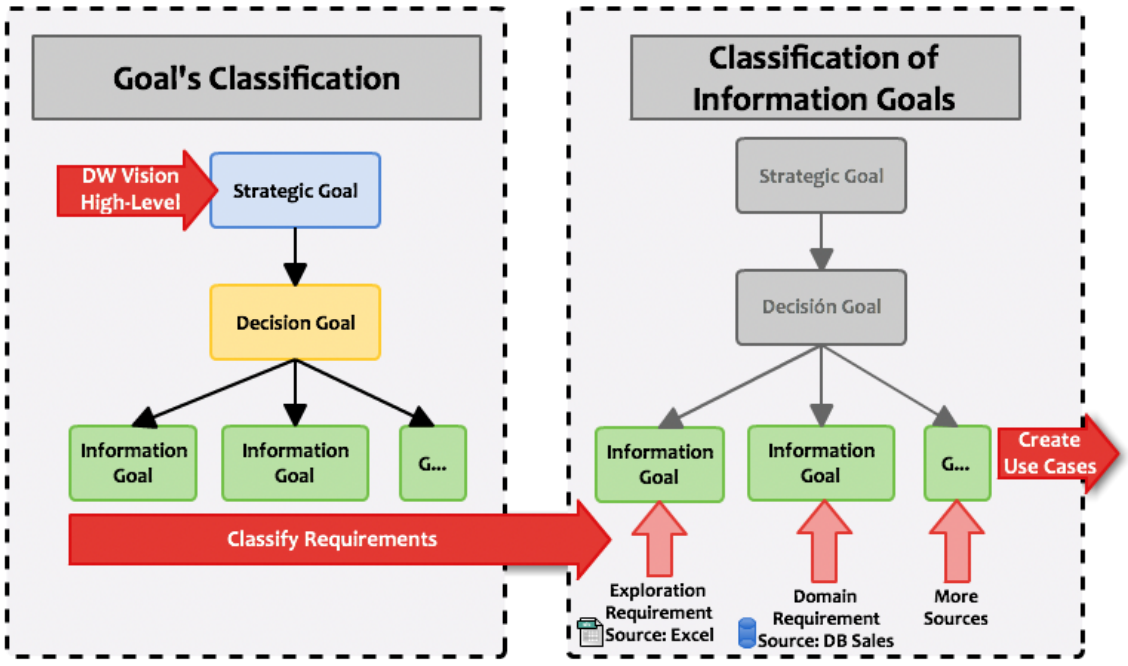


Figure 8. Goal Classification Tree [12]

Information Requirements

As mentioned before, information requirements are the most specific goals in the classification tree. We insisted in making this concept clear at this point, as it will be later referenced along this thesis. Information requirements can be classified into:

- Exploration Requirements: Requirements that involve direct end-user interaction (e.g. visualization dashboards), mainly applied at the end of the process. They generally involve requirements that the user can observe in the UI, and how we will be able to interact with it.
- Domain Requirements: They are mainly functional requirements that describe the behavior or functionality for the system. In a DSS, they describe exactly what is expected from the system to later exploit this information through UI tools or reports. These requirements define how the system will be designed.
- Quality Properties: Also known as non-functional requirements, describe how the system should behave as a constraint upon the system's behavior.

In addition, from the information goals, we also extract the scenarios (Use Cases) that come from the *Exploration Requirements* and the *Domain Requirements* (functional requirements).

Solution Oriented Requirements

After completing the Goals and Scenarios activities, we can proceed to define Solution-Oriented requirements. As it can be seen in the RE4DSS describing figure, this step is crucial for defining the

main inputs to the Quarry system, which will semi automatically generate ETL tools and MD model to create our desired system. However, they are also useful to define requirements in a structured manner before starting the development for the design, independently of the chosen methodology.

Solution Oriented requirements may be organized in the following way:

- Structured Requirements
- Shared Ontology:
    - Vocabulary
    - Ontology source and domain ontology
    - Source Mappings

First, the Structured Requirements are obtained from the *Domain Requirements* and the *Quality Properties*. These, as we mentioned before, are information goals, the ones located at the lowest level in the Goal classification tree. These requirements must be collected and then expressed in an XML file where multidimensional concepts are identified (measures and its dimensions).

If we recall the section about multidimensional models, dimensions are the aspects to analyze data, that allow to specify different levels of detail. Measures are the quantitative values for data in a specific dimension. For example, in the following figure we can see the requirement "examine income per customer" expressed in the required XML format. We can observe that dimensions in this case are the customers, which can certainly be analyzed in different levels of detail (for example, different types of companies). Also, we can observe the measures and how they must be calculated.

```
<dimensions>
    <concept id = "customers"/>
        <dimensions>

<measures>
    <concept id = "income"/>
<measures>


<measures>
    <concept id="revenue">
          <q_requirement> p_priceATRIBUT * p_discountATRIBUT</q_requirement>
          <nfr kind="freshness" format="HH24:MM:SS">&lt;01:00:00</nfr>
    </concept>
</measures>
```

Figure 9. Source Mappings example [12]

The next component in the Solution Oriented Requirements is the Shared Ontology. This one encompasses both the Vocabulary and the Ontology Source, which helps to create a common reference that helps to describe the structure of the different data sources, as well as to homogenize terms between stakeholders.

First, the Vocabulary document provides a glossary to enhance and ease the communication between business users (domain users), stakeholders, IT people, and the sources. It is mainly considered to avoid inconsistencies regarding the technical terms between business users and IT or tech-related people.

On the other hand, the Shared Ontology is also a document but in a machine-readable format which provides an integrated and summarized view of the terminology and data structure, describing all the different data sources involved in the creation of the multidimensional model. The main idea is that the shared ontology should provide a cross-reference structure which includes the data structure coming from the data sources, but also maintaining the traceability of the terms.

An ontology can be defined as the "specification of a conceptualization" *[5]*. The ontology proposed to be used as an input to the Quarry system is the Web Ontology Language (OWL). This is used to explicitly represent the meaning of the terms in vocabularies as well as the meanings of those terms. Besides, this ontology is intended to be used to be processed by applications (automatic processing of information). In the figure below there is an example that illustrates an ontology representation of an existent data source.

```
<!-- http://www.semanticweb.org/vegabante/ontologies/2018/2/chagas-disease#CollectiveDiagnosis -->

<owl:Class rdf:about="http://www.semanticweb.org/vegabante/ontologies/2018/2/chagas-disease#CollectiveDiagnosis">
    <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/vegabante/ontologies/2018/2/chagas-disease#Diagnosis"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://www.semanticweb.org/vegabante/ontologies/2018/2/chagas-disease#CollectiveDiagnosis_CollectiveDiagnosisInfo"/>
            <owl:qualifiedCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:qualifiedCardinality>
            <owl:onClass rdf:resource="http://www.semanticweb.org/vegabante/ontologies/2018/2/chagas-disease#CollectiveDiagnosis"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

 <!-- http://www.semanticweb.org/vegabante/ontologies/2018/2/chagas-disease#populationOfGeoArea -->

 <owl:DatatypeProperty rdf:about="http://www.semanticweb.org/vegabante/ontologies/2018/2/chagas-disease#populationOfGeoArea">
     <rdfs:domain rdf:resource="http://www.semanticweb.org/vegabante/ontologies/2018/2/chagas-disease#CollectiveDiagnosis"/>
     <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
 </owl:DatatypeProperty>


<!-- http://www.semanticweb.org/vegabante/ontologies/2018/2/chagas-disease#CollectiveDiagnosis_CollectiveDiagnosisInfo -->

<owl:ObjectProperty rdf:about="http://www.semanticweb.org/vegabante/ontologies/2018/2/chagas-disease#CollectiveDiagnosis_CollectiveDiagnosisInfo">
    <rdfs:domain rdf:resource="http://www.semanticweb.org/vegabante/ontologies/2018/2/chagas-disease#CollectiveDiagnosis"/>
    <rdfs:range rdf:resource="http://www.semanticweb.org/vegabante/ontologies/2018/2/chagas-disease#CollectiveDiagnosisInformation"/>
</owl:ObjectProperty>
```

Figure 10. Ontology (OWL) example

In the figure, there are different aspects that can be represented about a relational data source:

- Source table: In the example, the table "Region" is represented as an ontology class (owl:Class)

- Attributes of the source tables: The different attributes from the table(s) are represented in as different datatype properties in the ontology, for example r_regionkey INTEGER (primary key) and r_name VAR- CHAR.
- Concept Relationships: Relationships (references) between tables in data sources can be also represented in the ontology. For example, the relation existent in the table “Nation” which includes the reference to the source table “Region”.
- Cardinalities between Relationships: Cardinalities in the relationships between tables are also represented in the ontology. In the example, the tables Region and Nation cardinalities are defined to be 1..1 and 1..* respectively.

Finally, the Source Mappings represent the structured manner to define mappings that relate the Ontology Source with the real data stores. These should be defined in an XML structure, and its goal is that for each of the concepts in the ontology there should be its correspondent mapping inside the XML structure. Considering the types of aspects that may be represented in the Ontology Source, there are different kinds of mappings:

- Mapping of an ontology class
- Mapping of an ontology datatype property
- Mapping of an ontology object property

In the figure below, it is shown the Source Mappings XML structure that corresponds to the previous figure that illustrated Ontology Source.

```
<OntologyMapping sourceKind="relational">
    <Ontology type="property">
        http://www.owl-ontologies.com/unnamed.owl#Region_r_nameATRIBUT
    </Ontology>
    <Mapping>
        <Tablename>region</Tablename>
        <Projections>
            <Attribute>r_regionkey </Attribute>
            <Attribute>r_name </Attribute>
        </Projections>
    </Mapping>
</OntologyMapping>
```

Figure 11. Source mapping (XML) example [12]

In conclusion, the Ontology Source and Domain Ontology, the Source Mappings and the Vocabulary, constitute the Shared Ontology, which is a necessary input for Quarry, in addition to the Structured Requirements explained earlier. Also, these Requirement Artifacts allow to structure Requirements in a structured way, which becomes useful for designing the DSS using

traditional methodologies.

2.4 Quarry system overview

In business intelligence projects, the common practice is to obtain the requirements through traditional methods such as interviews and workshops, and then translating them into an ETL design and conceptual MD schema. However, this process can be highly error prone and demands several rounds of discussions and conciliations between different stakeholders. To this end, in 2011, professors from Universitat Politècnica de Catalunya (UPC) presented the GEM framework [13], which later evolved and was renamed to *Quarry*.

Quarry is a tool that produces in a semi-automatic way both the Multidimensional Design and the ETL Process Designs of the resulting data warehousing system, taking into consideration the set of requirements and the data sources as inputs for the system. One of the improvements that Quarry presented in comparison to GEM framework is that it includes a graphical assistance tool, which helps users independently of their background, to enter the obtained information requirements to the system. It actually starts by analyzing high-level information requirements having a subject of analysis (or a fact that wants to be known) as well as its analysis dimensions. Moreover, Quarry performs an automated process of validation to each requirement regarding the MD integrity constraints and its translation into MD schema and ETL process designs. Finally, it takes into consideration user-defined quality requirements in order to produce an optimal DW solution adapted to the user's needs [14].

Before using this system, it is recommended to apply a RE process to properly define the desired requirements for the system. Then, these requirements will be entered in Quarry through the UI tool. Also, it is suggested the RE4DSS approach to obtain the necessary inputs before proceeding making use of the semi-automatic tool. The inputs for Quarry that are produced by the RE4DSS approach are the ontology source and domain ontology, the vocabulary and the source mappings, written in a machine-readable format.

2.4.1 Features

The Quarry system mainly consists in four components: Requirements Elicitor, Requirements Interpreter, Design Integrator, and Design Deployer, as can be seen in the Figure. The Requirement Elicitor helps the end user to input all the gathered requirements. This component connects to another one which is the Requirement Interpreter, which generates a validated MD schema and ETL process design for each one of the input requirements. The third component (Design Integrator) allows to process each of the partial MD schema and ETL process designs into a unified designed solution. Finally, the Design Deployer performs the final deployment over the DW schema and the whole ETL process that populates it [14].

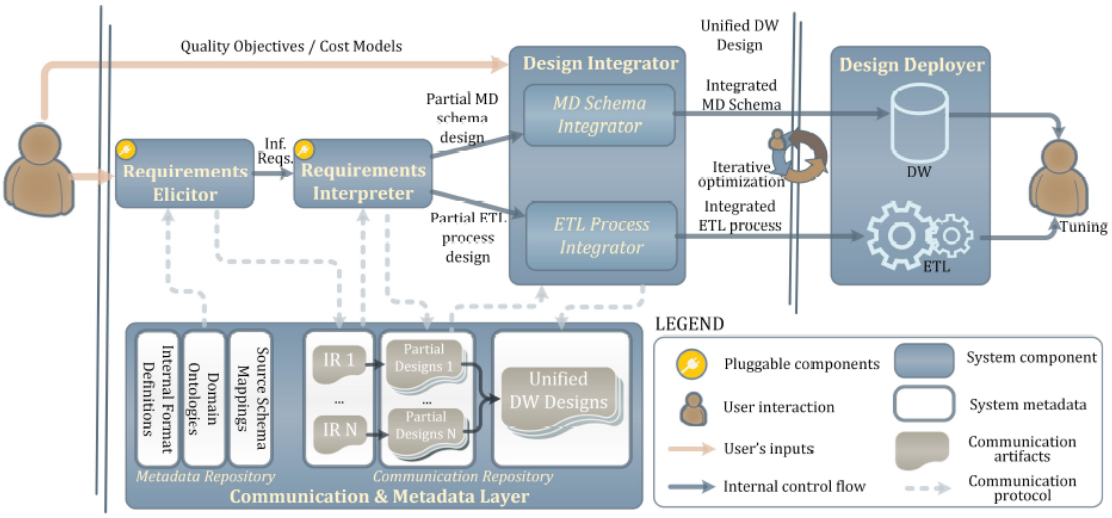


Figure 12. Quarry System Overview [14]

2.4.1.1 Requirements Elicitor

The first step of the process, consists of a User Interface that uses a graphical representation of a domain ontology, which remarks aspects of interest regarding the data sources. As mentioned before, the domain ontology can be enriched with the Vocabulary, to provide a useful guide to all end users before using the system. In fact, the Requirement Elicitor is based on the representation of the ontology in the form of a graph, which allows users to explore the data sources in an integrated manner, using a vocabulary familiar to them. In the following figure, it is shown an example of Requirement Elicitor in Quarry system.

Another useful feature of this component is that analyzes relationships in the domain ontology between different entities, and automatically gives suggestions about possible analytical perspectives.

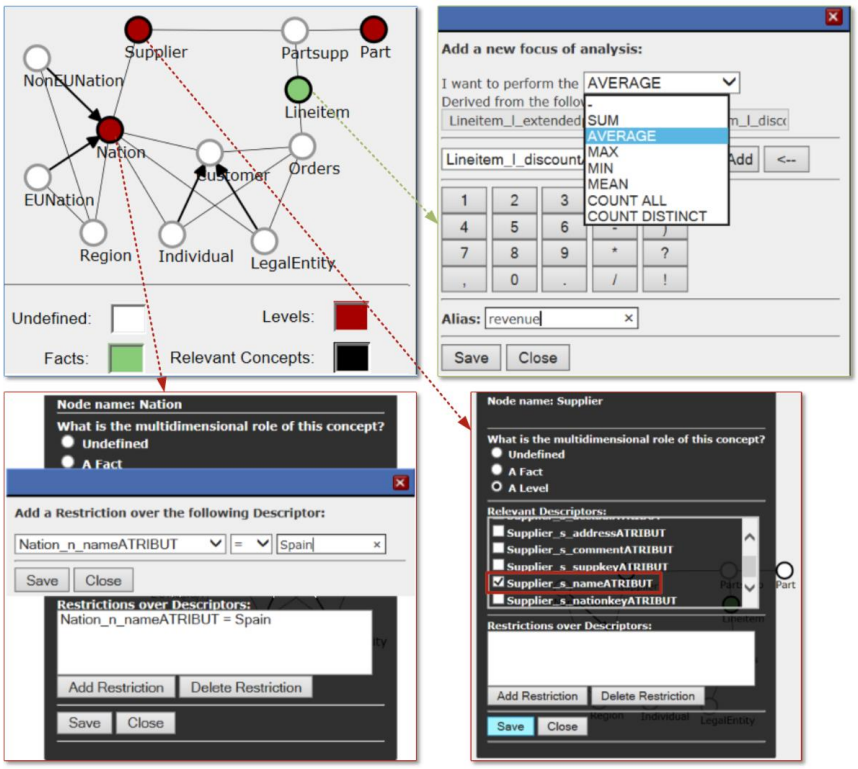


Figure 13. Quarry's Requirement Elicitor component [14]

2.4.1.2 Requirements Interpreter

On this step, Quarry receives the elicited requirements and maps them to the data sources. Specifically, the domain ontology helps capturing the requirements and corresponding them to the source mappings that were also an input to the system. This produces the MD schema and ETL process that satisfy the requirement in a semi-automatic way. An interesting addition to this particular step is that Quarry allows plugging in other external design tools, assuming that the provided designs meet integrity constraints and they satisfy the user requirement. For example, it allows the addition of import and export parsers, to enable support for various external notations such as SQL, Apache PigLatin, ETL Metadata, etc.

2.4.1.3 Design Integrator

This component starts analyzing each of the requirements and takes care of each of the partial MD schema and ETL designs that were previously obtained to incrementally integrate them until generating a unified design solution that satisfies all the requirements. It encompasses mainly two modules: MD Schema Integrator and ETL Process Integrator. The MD Schema Integrator is in charge of integrating partial MD schemas semi automatically. It basically comprises four stages: matching facts, matching dimensions, complementing the MD schema design, and integration. The ETL Process Integrator, on the other hand, is in charge of integrating partial ETL designs, obtained from analyzing each of the requirements, and unifying them into one single ETL design. The key of this integration process is that for each new requirement, maximizes the reuse by looking for the largest overlapping of data and operations in the existing ETL process.

2.4.1.4 Design Deployer

The final stage for the Quarry system is to perform the deployment of the unified design solutions over the storage repositories and execution platforms. An important advantage in the Quarry system is that it's platform-independent, so it can bind to a wide variety of execution platforms.

2.4.1.5 Communication & Metadata Layer

To support internal and cross-platform communication, Quarry uses the Communication & Metadata Layer. This component uses logical, XML-based formats to represent elements that may be exchanged between the different components [14]. Each of these elements in Quarry are represented through a specific format, in order to be properly identified and handled by the components. The elements that will be exchanged are the Information Requirements, the MD schemas and the ETL process designs.

Moreover, the *Communication & Metadata Layer* offers plug-in capabilities for adding import and export parsers, for supporting various external notations (e.g., SQL, Apache PigLatin, ETL Metadata) [14]. Finally, the *Communication & Metadata Layer* serves as a repository for the metadata that is used and created during the DW design lifecycle. Namely, the domain ontologies that capture the semantics of underlying data sources and the source schema mappings used to define the mappings between the terms in the ontology regarding the underlying data sources.

As a conclusion and having exposed the main aspects about the Quarry system, in the following sections we will begin to describe the actual use case where we will analyze the benefits and limitations between the results obtained using traditional methods in comparison to the semi-automatic tool Quarry. We will emphasize on the design process based on some requirements for this particular use case, and point out advantages and improvements that can be done to the semi-automatic tool, in order to have a more accurate final design.

Chapter III
WHO Use Case

In order to make the comparison between the traditional and semi-automated approach for building a DW, we intend to apply them inside a real use case with the intention of obtaining different criteria to perform the comparison. With this in mind, in this chapter we describe the use case, and elaborate on the methodology applied to extract the requirements. Later on, in future chapters, we will start to analyze how the different tools should be applied for the given requirements, and then extract conclusions based on the results.

3.1 The Case Study

The use case is provided by the World Health Organization (WHO), a United Nations (UN) agency. It consists in improving disease control and prevention by creating a Chagas Information Database (CID) system for surveillance, to raise awareness on the Chagas disease by storing and analyzing strategic data related to it [15].

Chagas disease is a mortal illness caused by the protozoan parasite called Trypanosoma cruzi. Its transmission to humans usually occurs through contact with faeces of vector insects (triatomine bugs), which includes the ingestion of contaminated food, transfusion of infected blood, congenital transmission, organ transplantation or laboratory accidents [16]. The study of this insect is particularly important for the comprehension of the disease. Triatomine bugs are members of the Triatominae subfamily, and are also known by their local names used in Latin America, which include *barbeiros, vinchucas, pitos, and chinches*. In Figure 14, we can observe the scientific classification for this insect as well as the classification of biological organisms for reference. There are more than 130 species known of this subfamily and they are mainly found and widespread in the Americas, with a few species present in Asia, Africa, and Australia. A number of species have adapted to living in and around houses and they are important in the transmission

to humans of the Trypanosoma cruzi parasite. However, transmission can be successfully interrupted by controlling the triatomine bugs in and around the houses where they have their resting places.[17]

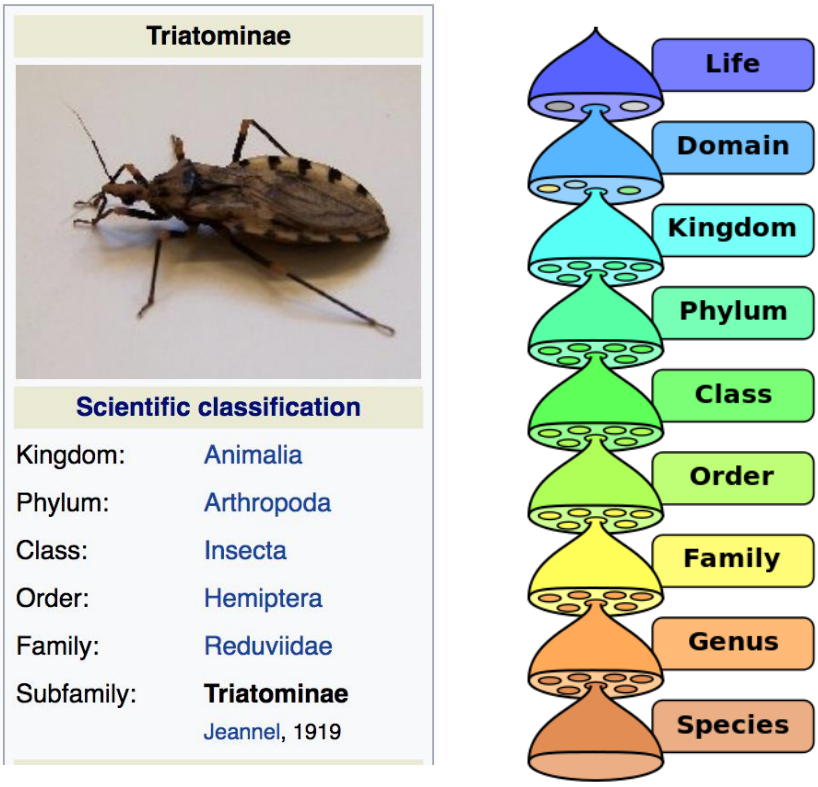


Figure 14. Triatomine bug scientific classification[31][32]

Consequently, this disease has created a big concern due to its great range of extension and gravity. People infected with Trypanosoma cruzi are located mostly in the endemic areas of 21 Latin American countries: Argentina, Belize, the Bolivarian Republic of Venezuela, Brazil, Chile, Colombia, Costa Rica, Ecuador, El Salvador, French Guyana, Guatemala, Guyana, Honduras, Mexico, Nicaragua, Panama, Paraguay, Peru, Bolivia, Suriname and Uruguay [18].

For these reasons, in 2010 WHO released the First report on Neglected Tropical Diseases (NTD) which included Chagas disease. In the same year, the World Health Assembly (WHA) approved the resolution "Chagas disease: control & elimination", with the following objectives [19]:

- Tackling of all transmission routes
- Integration of care of patients with acute and chronic clinical forms of Chagas disease
- Recognition of an increased number of cases of Chagas disease in countries where the disease is not endemic regions

To achieve these goals, in 2013 the WHO released a technological strategy:

"Create an information and surveillance system to facilitate the access to heterogeneous sources,

creating interactive data, providing disease statistics, visualizing maps and diagrams with current patients."

Based on this main strategy, the first step is to perform the requirement elicitation process, and after obtaining the requirements, we can proceed to develop the system's design. In the next section, we will enumerate the requirements used as starting point for this thesis to understand the implementation work explained later.

3.2 Requirements

After defining this general objective, an extensive process for requirements elicitation follows. This means, defining technical and non-technical requirements that will determine how the system is going to be defined. This process applies the Requirements Engineering for Decision Support Systems framework (RE4DSS) explained in chapter II, which is a Requirements Engineering approach designed to structure and organize the requirements elicitation process for this type of systems.

In chapter IV, we will begin to explain the process applied to follow the RE4DSS approach, for a better understanding of the preparation process before applying the traditional and semi-automatic methods. However, in this section we will introduce the requirements that will be implemented in this thesis.

Additionally, in background section 2.3.2.1 we explained the concept for information requirements in the context of a RE process. They are the most detailed requirements in the Goal Classification Tree, and can be classified in Exploration Requirements, Domain Requirements and Quality Properties (also technical and non-technical requirements). These become the starting point for performing the development of the system's design.

With this in mind, in the following subsection we introduce a short goal classification tree extracted from *[12]*, that will represent the horizon for our analysis of the different methodologies (the traditional and the semi-automatic) to generate an ETL design and MD model for this use case.

3.2.1 Elicited Requirements for this use case

As mentioned before, the requirements elicited for the WHO use case where the following:

High-level Requirement (Strategic Goal)

Health Ministry Officer: *"We need a system to control/eliminate the Chagas Disease".*

Strategic Goal Decomposition

(1) Researcher: *"I need to import Excel files containing inspection data in the system automatically so that this data is delivered to the WHO"*
(2) Health Ministry Officer: *"I need to visualize maps showing the presence / absence of triatomines per species, genus and geographical area"*
(3) Health Ministry Officer: *"I need to generate graphics showing the temporal variation of evaluated (inspected) houses by department every two years for La Rioja (Argentina)"*
(4) Health Ministry Officer: *"I need graphics from infested dwelling to be updated every 2 days for La Rioja (Argentina)"*

For the scope of this study, we are interested in comparing the different methods for the generation of ETL design and final MD model that allows complying with the requirements. For this reason, we need to focus the requirements selection even more, to the ones that are relevant to perform this comparison.

With this being said, we will focus our analysis to the decision requirements (2) and (3), which will directly implicate the way the ETL must be performed and how the MD model must be created. On the other hand, requirements (1) and (4) define the system's desired behavior or special features needed, but they do not influence how the ETL design must be made, so they will be disregarded for the purpose of this study.

Therefore, for future references, the requirements used for this study will be enumerated as follows.

IR1. Health Ministry Officer Requirement: "I need to visualize maps showing the presence / absence of triatomines per species, genus and geographical area"

IR2. Health Ministry Officer Requirement: "I need to generate graphics showing the temporal variation of evaluated (inspected) houses by department every two years for La Rioja (Argentina)"

3.2 Sources Available

An important input that must be taken into consideration when performing the requirement elicitation process is the available sources. In essence, the sources were basically two: Source data stores and WHO stakeholders.

In the WHO use case, data sources included three different inputs, from excel sheets to open data files. The used sources were the following *[12]*:

- *Excel Kiss Bug Inspection and Spraying: Data about the raids organized by researchers and / or governments to inspect, analyse or spray dwellings to control the kiss bug presence in lived areas. Source: Excel file.*
- *MetaTri Database: It is a geolocated vectorial-oriented database. Stores information about where kiss bugs can be found in South America. Source: Microsoft SQL Server database. This is a database created by Jorge Rabinovich (Argentinian entomologist). This database contains information regarding all the known species in the world that may transmit the Chagas disease.*
- *FAO Climate and flora: Data about the climate and flora associated to any point in the world. Source: Open data files to be downloaded from FAO.*

With this information in hand, the Requirement Engineering process took place to obtain the requirement artifacts. In the next chapter, we will begin to explain how was the RE4DSS applied in the WHO use case in order to prepare the basis before starting the development phase.

CHAPTER IV

Applying Requirements Engineering for Decision Support Systems (RE4DSS) methodology in the WHO use case

In any IT system, the Requirement Engineering process has become almost a mandatory phase in a project´s lifecycle, in order to guarantee its success. According to previous chapters, the RE4DSS is the suggested approach for the requirements elicitation process in Decision Support Systems, and it has been studied and proved to be effective in the planning of these projects [33]. With this in mind, we have used this approach as a preparatory phase before starting the system's design process. The RE4DSS allows to obtain the Requirement Artifacts that will become the base for developing the MD model and ETL design. In this thesis, we needed to apply this approach as a preparation phase before applying both methodologies that will be compared: the traditional and the semi-automatic one. Therefore, the intention of this chapter is to explain how the RE4DSS approach was performed for the WHO use case, and which are the Requirement Artifacts obtained from this process.

As seen in Chapter II, the RE4DSS framework consists in performing three Core Activities: Elicitation, Negociation and Documentation. These activities are useful to define the "Requirement Artifacts" which basically describe the desired behavior for the system. First, the Documentation activity is performed to preserve a common reference for the elicited requirements, according to certain rules. The Elicitation activity, aims to gather all the requirements for the system. And the Negotiation activity intends to achieve agreements between the different stakeholders due to conflicts that will inevitably arise.

As a result of performing these activities, we will obtain the requirement artifacts, which will help defining and structuring the system's desired behavior. These requirement artifacts are three: Goals, Scenarios and Solution-Oriented Requirements. The Goals are structured requirements that describe exactly what is expected from the system. They are represented in a hierarchical way, from less specific goals to the more specific ones. Scenarios, are another way to illustrate the system's desired behavior. They help delimitating possible situations that the system may face when there is an interaction between the system and the different actors. They also allow to define the expected behavior that the system should have in such cases. Finally, solution oriented requirements are the final output, which is created by using the previous two requirement artifacts. More specifically, the Goal Classification Tree helps defining the Information Requirements that constitute the base for starting the development of the Solution Oriented Requirements. These will later be used directly for the design of the DSS manually (using traditional tools) or semi-automatically (by using tools such as Quarry).

With this introduction being made, in the following sections we proceed to explain how these activities were performed for the WHO use case, and further on, we describe which were the requirement artifacts that we will be basing this study on.

4.1 Core activities

In previous studies for this use case, the IT team in association with the WHO staff performed the Core Activities, in order to obtain the previously mentioned Requirement Artifacts. In following sections, we will begin to describe the work they did, extracted from [12].

4.1.1 Elicitation

Elicitation is the core activity responsible for collecting requirements from the stakeholders based on their different needs. For the WHO required system, there are two important aspects that must be taken into consideration in the elicitation process:

- *Business Requirements*: Which will define the starting point to specify the design and functionality of the system, based on the user's needs. The most critical business users (specialist in Chagas disease and triatomine bugs) were the following: Health Ministry Officer, Researcher and WHO. Their different roles and responsibilities may be summarized as follows [15]):
    - *World Health Organization:* The WHO actor interacts with the system in order to visualize, manage, analyze and import/export information. On the one hand, WHO has access to the entire information system database, therefore, it is able to consult, but not modify, any type of information related to any country.
    - *Health Ministry Officer:* The HMO actor, in a similar way as the WHO actor, interacts with the system in order to visualize, manage and import/export information. Health Ministry Officers are only allowed to provide, modify, and delete information related to their own country.
    - *Researcher:* The Researcher actor, in a similar way as the other actors, interacts with the system in order to visualize, manage, import/export, and analyze information. Researchers can provide information related to healthcare, and transmission interruption.
- *Data Sources*: They define the actual sources where the data is extracted from. There must be considered the different formats available such as Excel, XML or relational databases and the techniques that will be used to extract, transform and load the information.

To succeed during the elicitation process, there are several techniques suggested in the literature, such as Interviews, Workshops, Facilitated Sessions, among others. In this case, the software analysts conducted various interviews to the Chagas experts. These were conversation-like interviews where the interviewee elicited the interviewed opinion regarding some issues. Also, some workshops were necessary during the process, to agree on some high-level objectives. Finally, when there wasn't any available time to meet personally, some questionnaires were performed to the stakeholders as well in order to clarify on some issues.

As a result, this activity allows not only to obtain a first set of the requirements but also to define

a common vocabulary, which will be enriched along the duration of the project. This Vocabulary will help bridge the terminology between the technical or IT staff and the actual business users. In the WHO use case particularly, it became necessary due to the amount of technical terms in both sides, which created a gap between each other to come to an agreement. For example, one of the issues they faced was that the sources do not always match the same terminology. The Chagas bug has different name variations (such as kiss-bug, vinchuca, barbeiro, chinche, chipo, etc), which may vary depending on the country and the specialist. For this reason, there was a need to bridge this gap between the Chagas experts and the software analysts' terminology by creating a Vocabulary, pursuing to identify the real business needs, avoid linguistic inconsistencies between Chagas specialist and software analysts and to homogenize used terms.

4.1.2 Negociation and Documentation

The next core activities according to the RE4DSS framework are Negotiation and Documentation. The first one intends to help the different stakeholders to come into agreements, due to the fact that they may have different wishes in conflict with each other. For example, one common source of discussion in the Negotiation activity involved a conflict of interests between the stakeholders on how the information should be collected.

The second one allows to establish a common reference between the stakeholders and to later document requirements according to the rules. Mainly, with the objective of preserving knowledge for future employees and for reference to new employees.

4.2 Requirement Artifacts

As mentioned before, the core activities are made to reach to the definition of the requirement artifacts. This is the term used in the R4DSS approach to refer to a documented requirement defined in a specific format. For the purpose of this study, in this section we will describe the Goals, Scenarios and Solution-Oriented requirement artifact obtained for the WHO use case which is directly relevant to the work done in this thesis.

4.2.1 Goals

Goals refer to the documentation of stakeholder's intentions about the objectives. They state what is exactly expected or required from the system. In order to identify the main goal of our stakeholders, during the elicitation process it is important to ask: "What do you do (and why) with the system?" instead of simply "What do you want?" (20). By following this recommendation, we extract the main objective which will be the input at the highest level of our Goal Classification Tree. Namely, consists in decomposing goals hierarchically, which helps to refine the system's vision into more specific objectives that can be expected from the Decision Support System.

Having identified the higher-level objectives and its sub-goals, a Goal Classification Tree is built. As explained in the background section, this tree is built according to the level of abstraction, so the different goals can be classified into different categories, namely Strategic Goals, Decision Goals and Information Goals. After some interviews and discussions, the Chagas specialists agreed on what they needed:

"*Decision Support System where they could store their data representing single and integrated view of the Chagas disease characteristics (including dwellings, countries, infected individuals, etc.) intended to be exploited by Chagas specialists." [12]*.

Following this same strategy, we keep refining these decision goals, until reaching the last hierarchy level of the tree (leafs). This last subdivision for the tree may be classified as Information Goals, and it consists of the most specific objectives of the goal classification tree. For instance, "Visualize Maps of Infested Dwelling".

After obtaining the main Goal Tree which defined general objectives for the project, business rules must also be taken into consideration during the goal elicitation, to obtain a more accurate definition of the desired DSS. To do this, Information Goals should be the starting point for refining the obtained information requirements into specific categories, namely *Exploration Requirements*, *Domain Requirements* or *Quality Properties*. For example, "Import Information" is classified as a *Domain Requirement*, "Graph Update every 2 seconds" is classified as a *Quality Properties*, and finally "Visualize Diagrams and Maps" is classified as *Exploration Requirements*.

Following this procedure, the Goal Classification Tree was built and refined for the project. A representative example of this may be seen in the following Figure

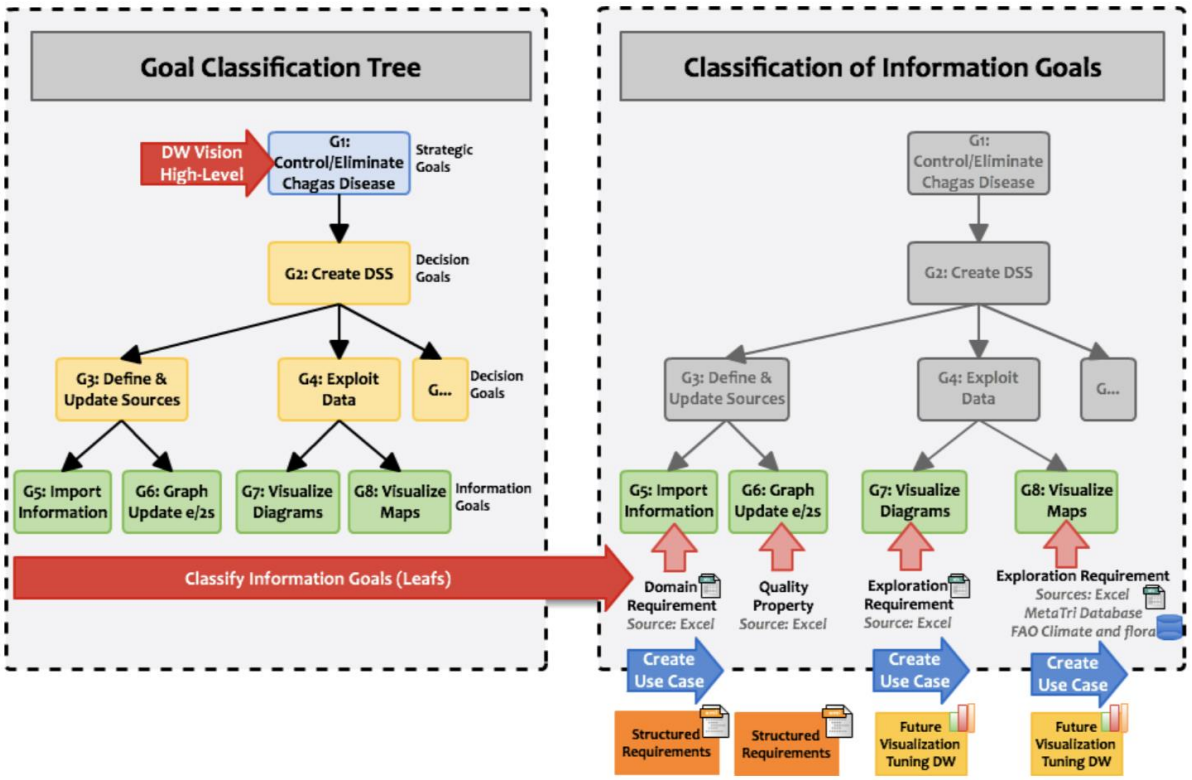


Figure 15. Goal Classification Tree for the WHO use case [12]

The purpose of this thesis is not to show the complete Goal Tree, but to select some meaningful examples from the total elicited requirements and use them as starting point to perform our analysis. The intention was to select representative example requirements to keep the analysis delimited and make the comparison between different methodologies easier. With this being said, we have selected a subset of information requirements which specify the desired functionality in terms of the information that the user wants to obtain from the system. Some representative requirements obtained from this Goal Classification Tree were described in section 3.2.1 from Chapter III, and the ones we will be working with in this thesis are IR1 and IR2.

4.2.2 Scenarios

Scenarios are concrete use cases that describe a possible activity or feature that the end user may perform with the system, and how it should behave accordingly. Namely, it aims to detail the process that must be followed by the system as a consequence of the interaction between the different actors. In the following figure, we may find an example of a possible scenario for the WHO use case, extracted from [15]

Running Example - Use Case Import Information

Use Case Name: Import Information

Active Actor: HMO user

Trigger: HMO user indicates export information

Preconditions: The HMO user must be identified and authenticated. Stakeholders and interests

i. HMO user: obtaining and importing information about diagnosis, treatments, triatomine bug studies, inspections, insecticide applications and systemic and normative information from the system in order to evaluate and analyse it.

Main Success Scenario

i. User indicates import information
ii. System presents the different types of information that can be imported: Inspection information, Triatomine Bug information, Insecticide Application information, Diagnosis information, Treatment information, Systemic information and Normative information.
iii. User selects the information to import
iv. System shows all available geographic objects in the system
v. User selects the geographic objects which he wants to extract information from and then confirms the exportation
vi. System requests the import file name
vii. User enters the import file name

The intention is to use these scenarios to enrich the Vocabulary and improve the Domain Ontology in the Solution Oriented requirements. It is important to notice that the enrichment of the Vocabulary is not limited to this step. Instead, it must be an iterative process during the whole lifecycle of the project

4.2.3 Solution-Oriented requirements

After going through the first stage of the RE4DSS approach iteratively, we obtain the solution oriented requirements which allow to get a conceptual model for the system. This requirement artifact may be used as input for Quarry, but it may be used for traditional methods of ETL design as well. They may be used as a more structured-like source to get a clear conceptual model for the desired final result.

Solution Oriented requirements consist in two parts:

- Structured Requirements: XML file which contains the Domain Requirements and Quality Properties (Information requirements) from the Goal Classification Tree.

- Two components from the Shared Ontology:
  - A Domain Ontology represented with OWL classes, which is later enriched with the Vocabulary.
  - Source Mappings, XML file containing mappings of the ontology concepts.

For the WHO use case, the solution-oriented requirements consisted of the following:

4.2.3.1 Structured Requirements

Presented in a XLM file, they are extracted from the last leaves of the Goal Classification Tree (Domain Requirements and Quality Properties), and represented as measures and dimensions. In the following figure, an example is shown.

Levels:

```
<dimensions>
    <concept id="Species_s_name"/>
</dimensions>
```

Measures:

```
<measures>
    <concept id="EvaluationCoverage">
        <function>
            Inspection_i_inspectedhousesATRIBUT div Inspection_i_evaluatedhousesATRIBUT
        </function>
    </concept>
</measures>
```

Slicers:

```
<slicers> <comparison>
    <concept id="Inspection_i_dateATRIBUT" />
    <function>extracty_year</function>
    <operator>&lt;=</operator>
    <value>2010</value> </comparison>
</slicers>
```

Figure 16. Structured Requirements (XML) example [12]

For Quarry, we don't need to provide the XML file as an input to the system, because Quarry comes with a User Interface to elicit the requirements, as well as measures and dimensions. However, this is a very useful way of organizing requirements in a unified format.

4.2.3.2 Shared Ontology

4.2.3.2.1 Domain Ontology

After defining the different data sources that shall be used, we need to structure them in a machine-readable format, which may allow the system to extract, transform and load the data according to the desired requirements. As explained before in the background chapter, the ontology is represented in OWL format, and allows to detail some characteristics of the data source where the data will be extracted from, such as:

- Source tables
- Attributes
- Relationships
- Cardinalities between relationships

The role of the ontology is twofold:

- It is a common conceptualization of the data sources, that unambiguously and in a domain specific vocabulary define the semantics of the underlying data sources.
- It is structured and stored in a machine-readable format that allows its automated processing and visualization (e.g., Quarry).

In further sections, we will show how the ontology source was made for the purpose of this study, as well as the tools used to do this. The complete ontology created in this thesis for the WHO use case may be found online through this reference [36].

Besides the ontology source, we needed the vocabulary to enrich our shared ontology. As explained before, vocabulary helps homogenize terms between different stakeholders inside the project. However, an additional approach in this use case is to present the vocabulary in a machine-readable format to help the automation process for the ETL design and MD model and to provide a quick and visual guide to develop the system easily.

In the following figures, we show this vocabulary in a UML-like diagrammatic representation. The complete and detailed vocabulary representation may be found in Annex 5.

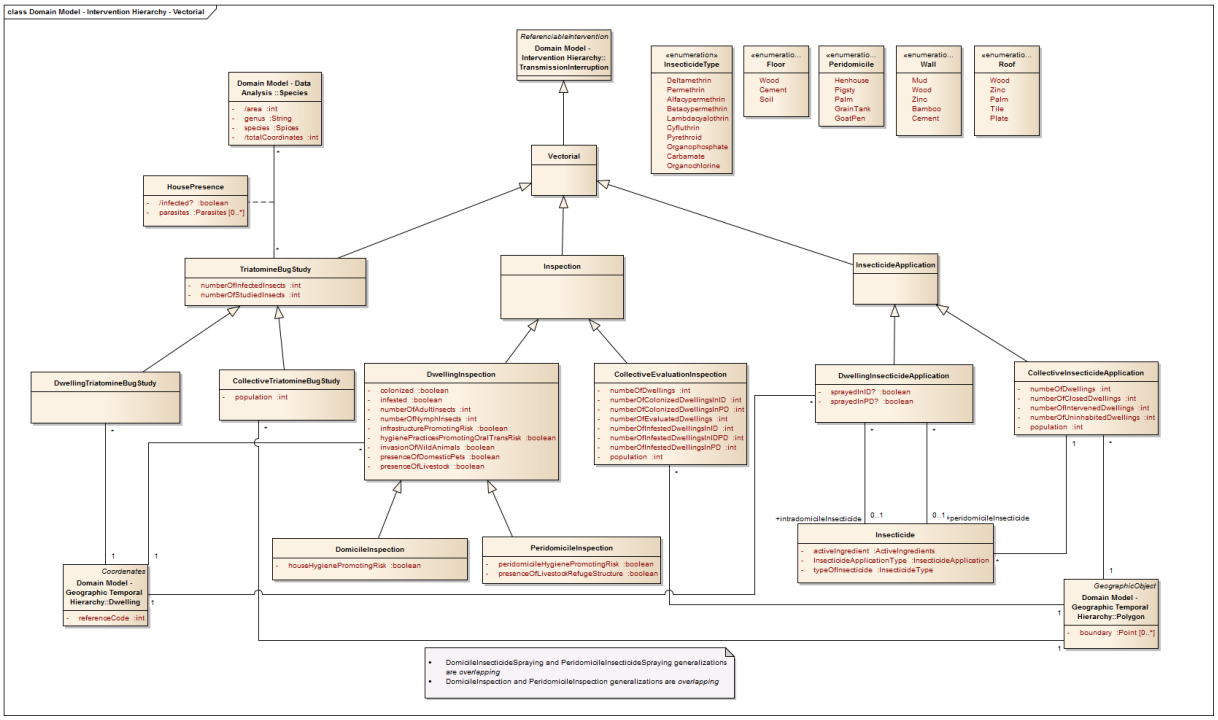


Figure 17. Vocabulary part 1

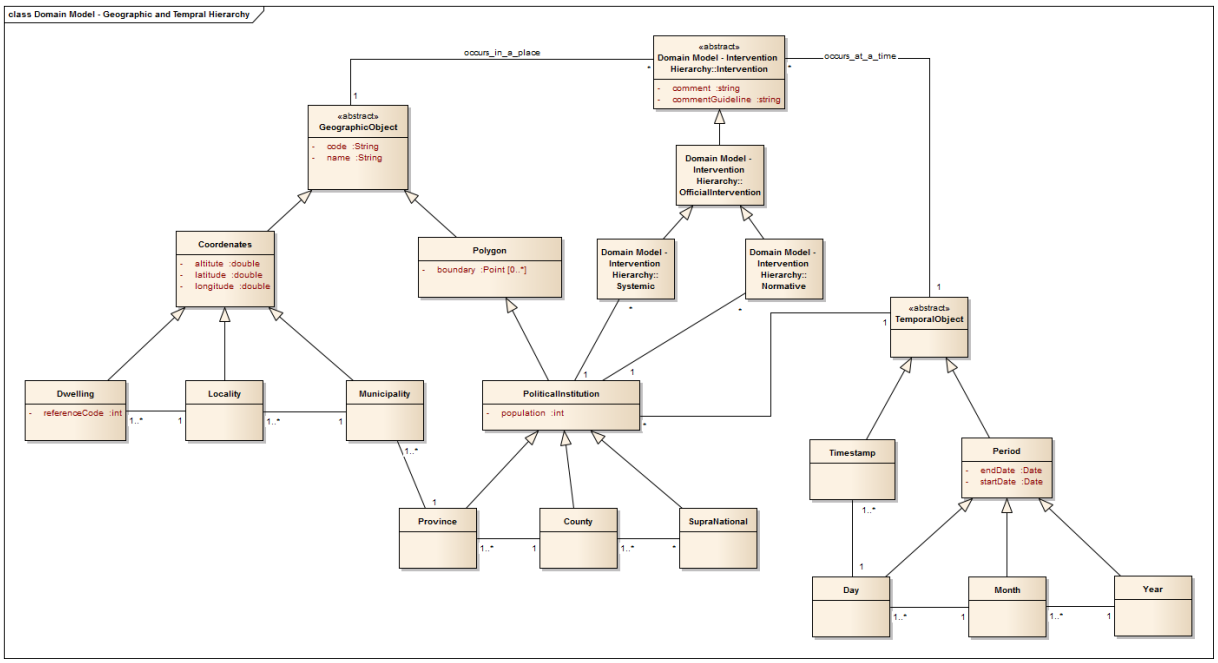


Figure 18. Vocabulary part 2

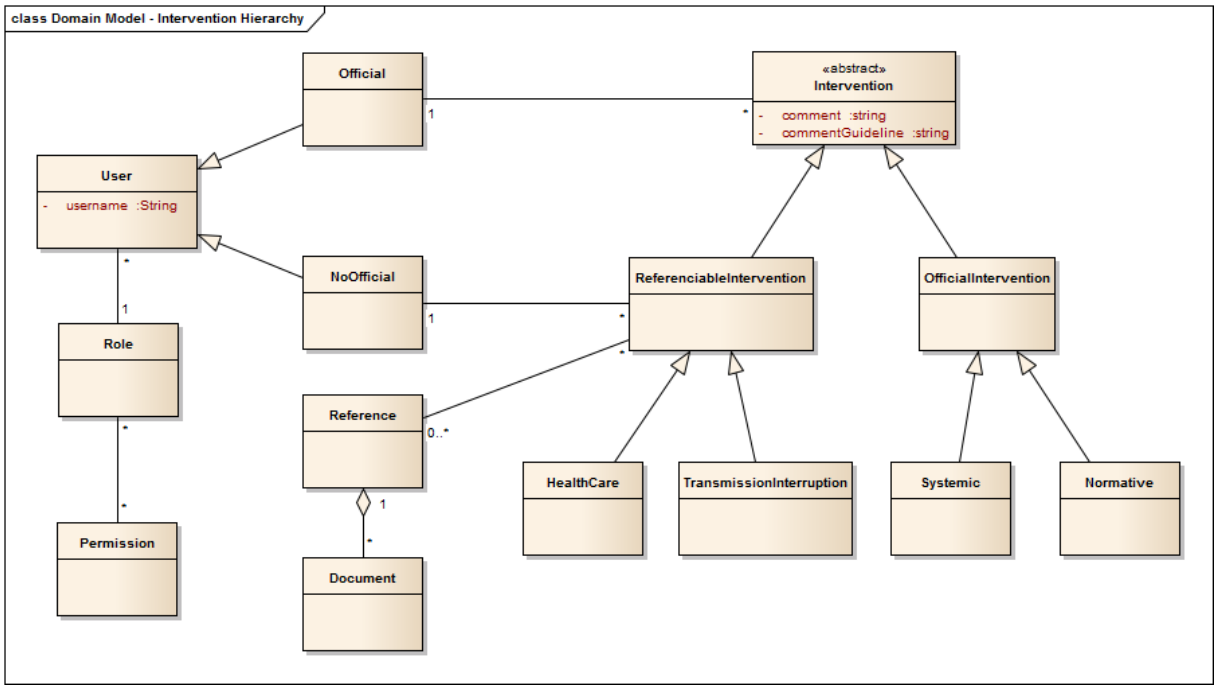


Figure 19. Vocabulary part 3

4.2.3.2.1.1 Development of the Domain Ontology with Protégé

As a preparation phase for our work on developing the two approaches for obtaining the system's design (the traditional and the semi-automatic), at this point we began creating the domain ontology based on the vocabulary and the available data sources for this project. The idea was to obtain the domain ontology artifact in order to use it as an input to the development phase of MD model and ETL process design. Not only as a machine-readable file to be processed by semi-automatic systems, but also for documentation purposes. Mainly, for the conceptualization of the data sources and definition of the semantics for the data sources

For this case study, we used a tool that helped us during the development of the ontology. The tool used is called Protégé, and it assisted us in the manual creation of the ontology. There are other alternatives that also help in the fulfillment of this task, which may be consulted in this source [34]

Protégé is a free, open-source ontology editor and framework for building intelligent systems. It was built by the Stanford Center for Biomedical Informatics Research (BMIR) at the Stanford University School of Medicine [21]. They developed this system to easier develop and maintain ontologies, as well as support the National Center for Biomedical Ontology. This latter one aims to provide a repository and web service for researchers and scientists to use in the biomedical area.

The BMIR also supports the WHO by providing the technological infrastructure for developing and maintaining many important ontologies in the biomedical area such as the International Classification of Diseases and the International Classification of Traditional Medicine, among others. Moreover, the WHO designated BMIR an official Collaborating Centre [21] For these reasons, we decided to use this tool for building the ontology in this work.

Therefore, in this thesis we use Protégé Desktop, which will help us create the ontology for the WHO Chagas use case easier. Protégé consists of a user interface that helps defining key concepts in the ontology, such as classes, data properties and relationships. To do this, there are different views dedicated to create each one of the necessary fields for the ontology. For instance, in the Figure 20-22 there is a screenshot of an example for the Protégé tool, where we specify classes and data properties.

Using the Protégé tool, we proceeded to develop the ontology for the WHO Chagas use case. To do this, we took as a source the UML= diagrammatic representation of the data sources, enriched with the domain vocabulary (see Figures 17-19). The correspondence between the data source concepts and the fields in the ontology can be seen in Table 1.

Table 1. Data source concepts in the Ontology

Data source concepts	Ontology Fields
Classes and Subclasses	owl:Class
Data properties (attributes)	owl:DatatypeProperty
Object Properties	owl:ObjectProperty
Individuals	owl:NamedIndividual

In Figures 20-22 we show some screenshots of the creation of the ontology using protégé. The entire ontology result may be found online, through this reference [36].

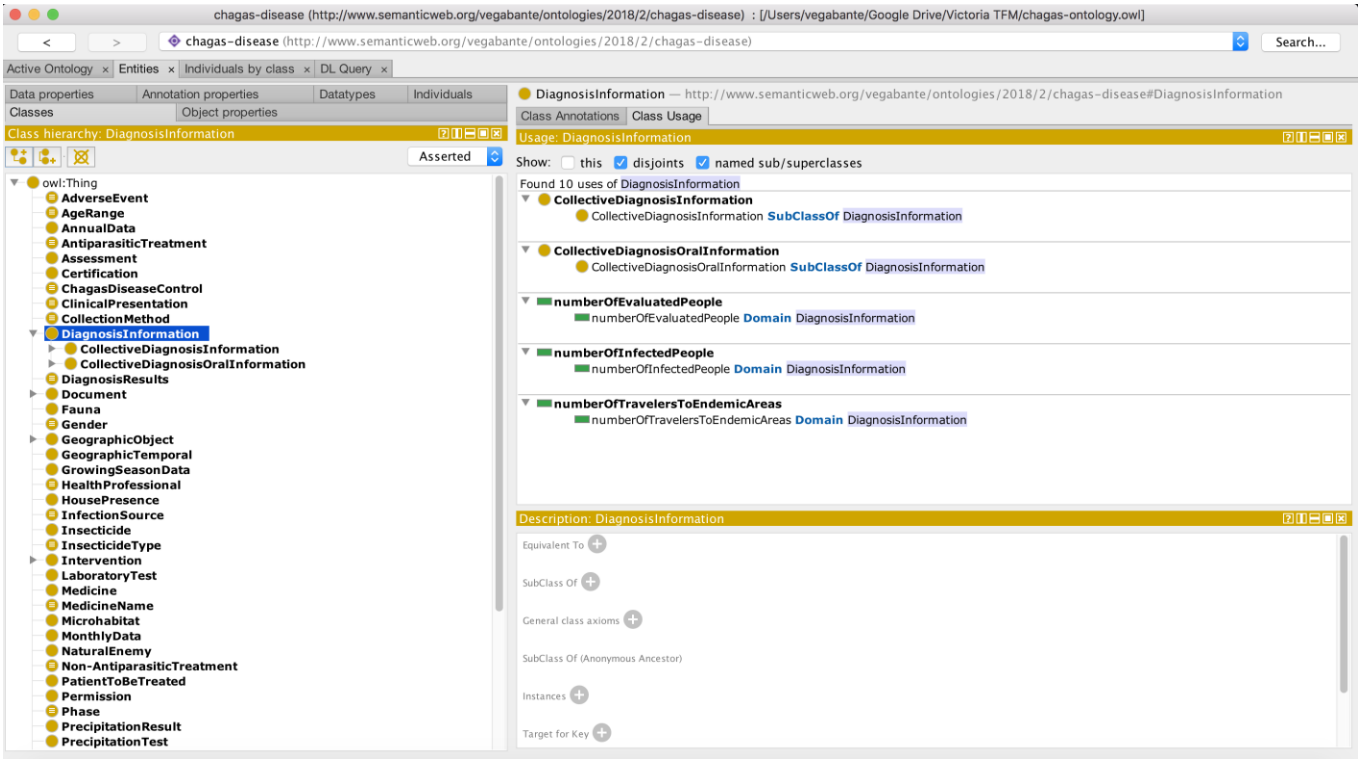


Figure 20. Ontology creation using Protege tool (Classes)

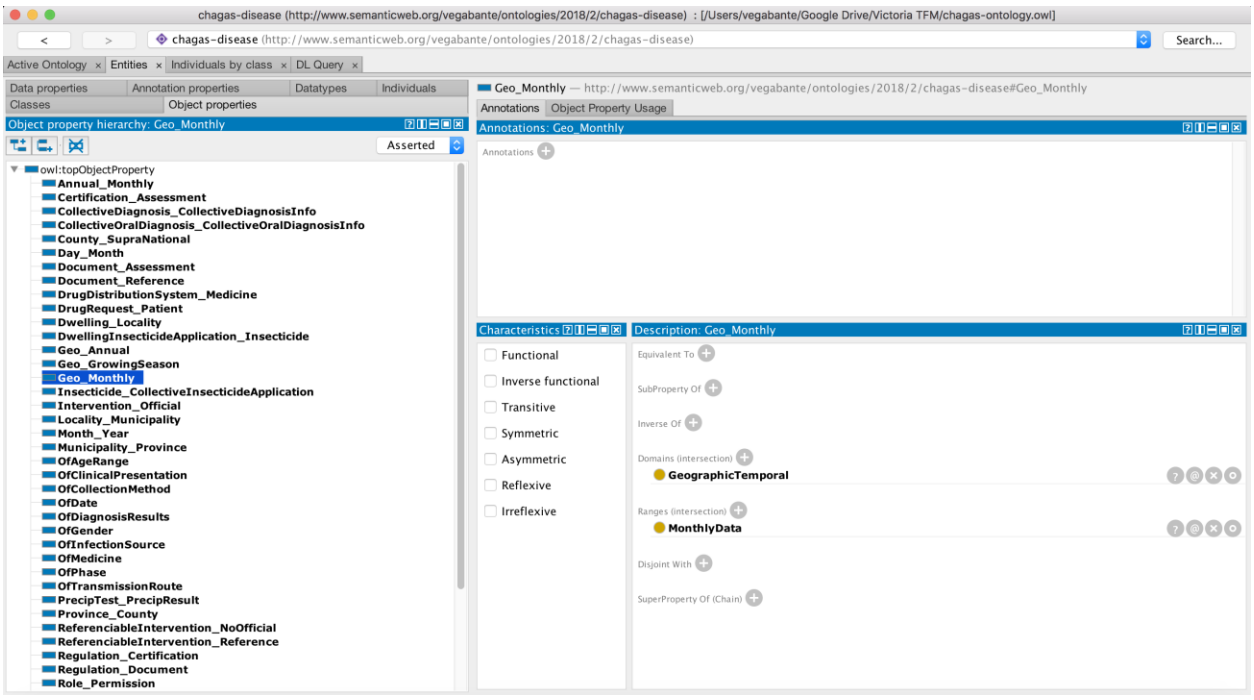


Figure 21. Ontology creation using Protege tool (Object properties)

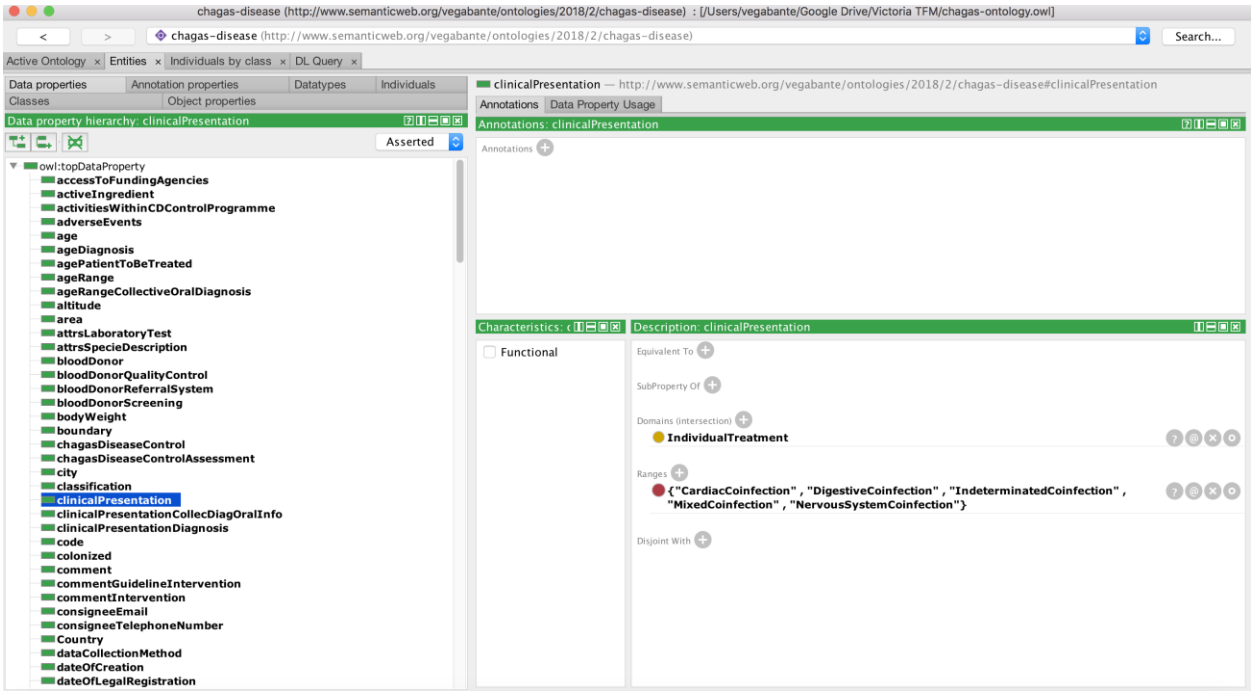


Figure 22. Ontology creation using Protege tool (Data properties)

The Protégé tool helps in the development of the Ontology by detecting inconsistencies and possible syntax errors before introducing a new field, and the user interface helps making the process more visual and intuitive.

However, it was a long and complex process. It required to translate each of the fields in the data sources into their correspondent area in the ontology. In fact, the creation of the ontology took a few weeks of work because it required defining each of the fields described in the Vocabulary, clarifying many ambiguities and consulting domain experts. For instance, according to the Vocabulary extracted from [12], the "Species" class also contains an attribute of type Species. We were not sure how to define this in the Ontology, so we consulted expert sources to understand how this Entity should be represented, according to the Triatomine bug Scientific classification. Part of our research was to understand that, according to [31] Triatomine is a subfamily, that can be later classified into genus and then into Species. Therefore, we represented this accordingly in the Ontology.

Similar to this example, we encountered many cases which we were not sure how to represent into the ontology, so it required weeks of consults, research and iteration.

4.2.3.2.2 Source Mappings

This representation is made in XML format, and it consist of a mapping for each of the concepts from the ontology inside a data source. This may be done starting from the already created Ontology and the available data sources. It consists in matching each concept in the data source with the ones in the ontology, according to what is specified in Table 1. An example is shown in the figure below

```
<OntologyMapping sourceKind="relational">
    <Ontology type="concept">http://www.owl-ontologies.com
        /unnamed.owl#Inspection</Ontology>
    <RefOntology type="property"> http://www.owl-ontologies.com
        /unnamed.owl#Inspection_i_inspectionkeyATRIBUT</RefOntology>
    <Mapping>
        <Tablename>inspection</Tablename>
        <Projections>
            <Attribute>i_inspectionkey</Attribute>
        </Projections>
    </Mapping> </OntologyMapping>
```

Figure 23. Source mapping example [12]

Chapter V

Developing two different methodologies for the creation of the MD model and ETL design: The Traditional and the Semi-Automatical

The main work performed for this project is to compare two methodologies for creating an ETL design and MD model that will comply with the previously stated requirements in the WHO use case (see Section 3.2.1). The traditional methodology involved the typical tools used commercially for creating MD models and ETL designs. The semi-automatic methodology assists the user in the creation of these designs receiving the requirements and source ontology as an input. To perform this comparison, we decided to limit the number of requirements that will be taken into consideration in order to help us make a simpler analysis and to obtain clearer conclusions.

Under those circumstances, in this chapter we will begin to explain how these methodologies were applied to satisfy the given requirements (IR1 and IR2), and in the next chapter we will perform the comparison and explain the conclusions.

5.1 Traditional Approach

In this section, we will explain each step taken to obtain a MD model and ETL design that satisfy the requirements IR1 and IR2, manually, by using selected BI tools. By the end of the chapter, the difference between the two methods will become clearer, as well as the effort and work required by each one of them.

5.1.1 Utilized tools overview

In this thesis, we use Indyco Builder [21] and Pentaho Data Integration [23] tools to develop the design of the MD model and ETL, respectively. Both tools are commonly known in the BI market, and allow the user to connect to a great number of tools and use plugins to design the ETL process, as well as to create a neat conceptual model for the system. In the next sections, we will perform a brief overview about each one before starting to describe the actual work done with them.

5.1.1.1 Indyco Builder

Indyco Builder is a tool designed by an Italian company called Iconsulting. The tool helps creating a conceptual model for visualizing the desired MD model, in order to meet with the previously agreed requirements. In fact, Indyco is a User Interface that helps creating a Dimensional Fact Model (DFM) [8]. This conceptual model technique is a very helpful representation of the desired model for the Data Warehouse because it takes the user requirements into consideration. We explained the concepts of DFM previously, in section 2.2.1.2.

In the following figure, we show a screenshot of the tool. Note that it allows to create facts, measures and dimensions, and organize them until having constructed the entire data marts or DW. It is very helpful for supporting the conceptual modeling of the system.

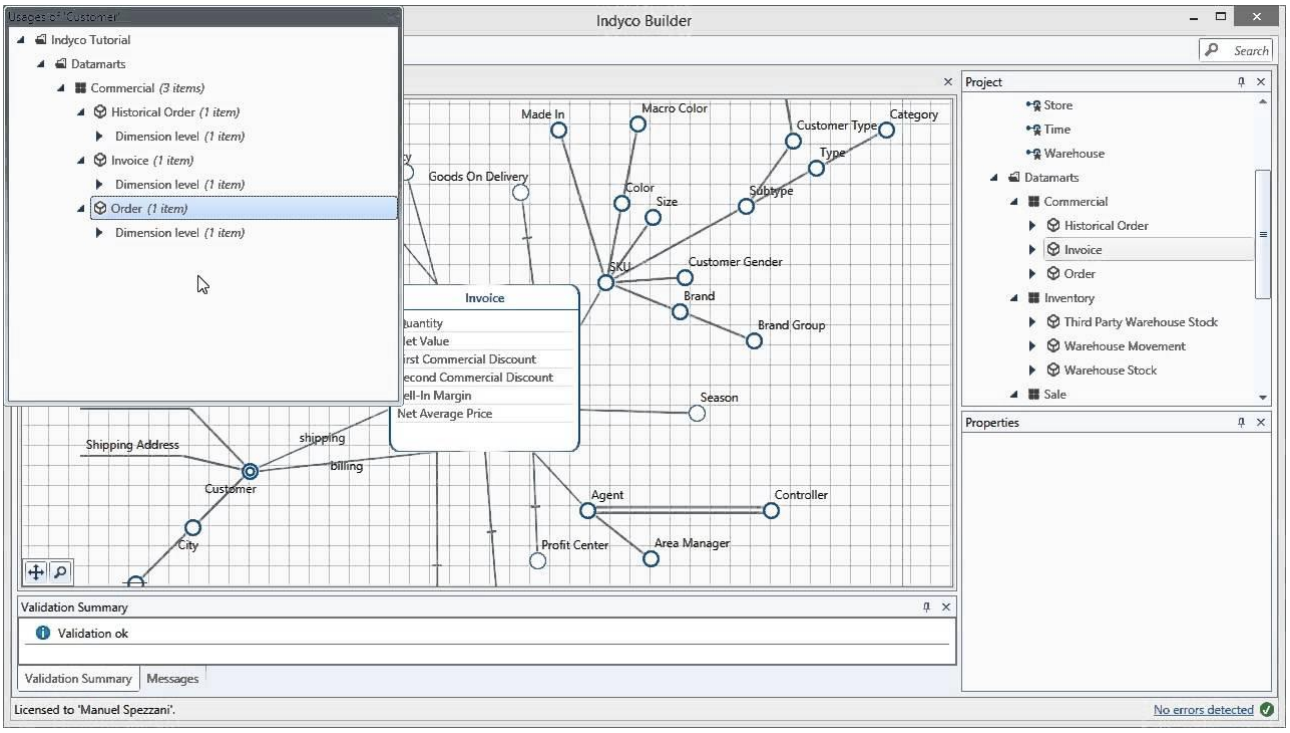


Figure 24. Indyco tool screenshot example

5.1.1.2 Pentaho Data Integration

For the ETL design, we will be using Pentaho. Pentaho provides a suite of business intelligence software that gives support for data integration, OLAP services, reporting, information dashboards, data mining and extract, transform, load (ETL) operations. It includes many number of applications, including server and desktop/client, which provides different solutions in the BI area. These solutions belong to the company's Hitachi Data Systems portfolio since 2015 [23].

In particular, the application that we will use for this thesis is Pentaho Data Integration (PDI), also known as "Kettle", which consists of a data integration ETL engine, and GUIs that allow the user to define data integration jobs and transformations (i.e., Spoon). It supports deployment on single node computers as well as on a cloud, or cluster [23]. In the figure we can observe some screenshots of the application, as well as some examples for its usage.

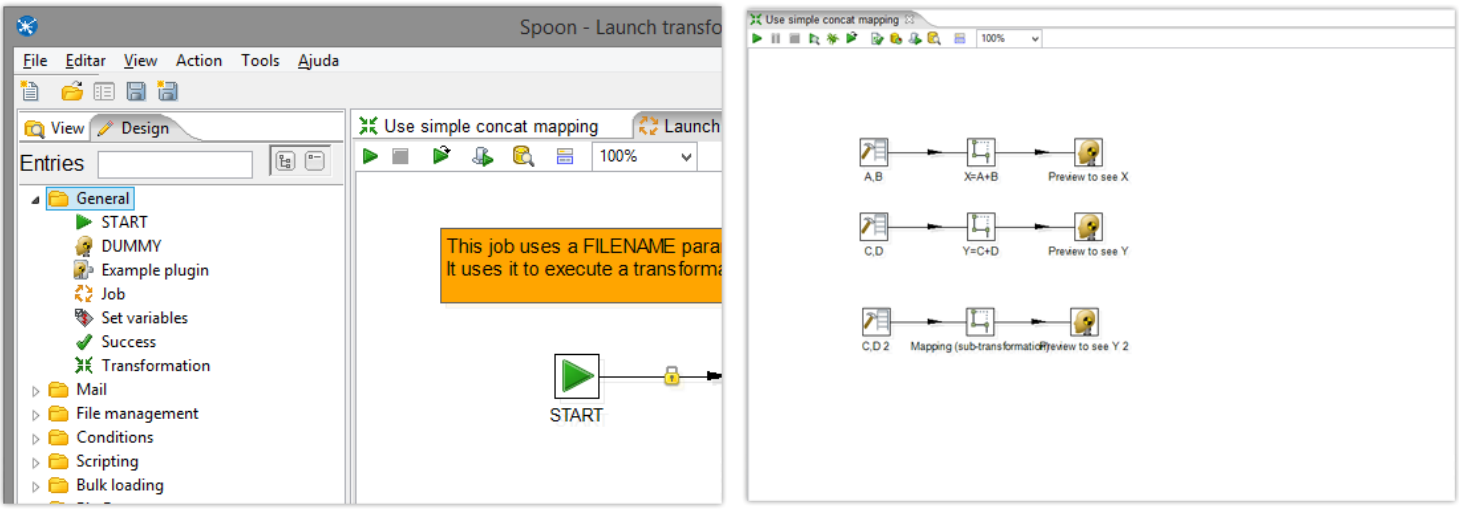


Figure 25. Pentaho tool screenshot example [23]

5.1.2 Design work

Having introduced the tools, we begin to describe the work that has been done for obtaining the MD model and ETL process design following the Traditional approach. In the following sections, we will show each of the steps performed with each tool and the obtained results.

5.1.2.1 Designing MD model with Indyco

As mentioned earlier, for this thesis we will base our analysis in the requirements obtained from the Goal Classification Tree developed during the implementation of the RE4DSS approach. Particularly, on the requirements IR1 and IR2. Now, we will show the work done to satisfy each requirement separately.

Work done to satisfy requirement IR1

The focus of our analysis is to evaluate the available ontology with the objective of obtaining the MD model that satisfies our requirement. Thus, the factual concept that we are interested in obtaining is the Presence or Absence of Triatomines by Area, classified in Species and Genus. This fact will come from a Boolean type field, that indicates if the Triatomine bug is present or not for a determined specie or genus in a determined area. Meaning, we want to analyze Presence/Absence of triatomines in terms of geographical location or area where it is detected and in terms of species and genus of the bug. Thus, the geo-location/area, and triatomine bug specie and genus are the dimensions of the analysis and the number of triatomines obtained for the given dimensions is a measure.

Species and genus constitute the bug's scientific classification (see Figure 14). Therefore, we should be able to classify triatomines first by genus and then by specie, in this order. This order in fact defines the levels of our second dimension and allows to analyze the data at these different granularities. The presence or absence of triatomines for a given location and scientific classification will be determined by the number of insects obtained. If this number equals to zero, then there is no presence of triatomines. On the contrary, there is presence of triatomines.

In this sense, we took the previously developed ontology and vocabulary as a reference to start developing the DFM. First, we need to search for the classes and attributes that gives us the information required to satisfy this fact. Then, taking this into consideration, we built the following DFM developed in Indyco, showed in Figure 26.

As we can observe, the dimension names are actually attributes in the tables from the ontology, and depending on the level of aggregation, the number of Triatomines may be calculated.

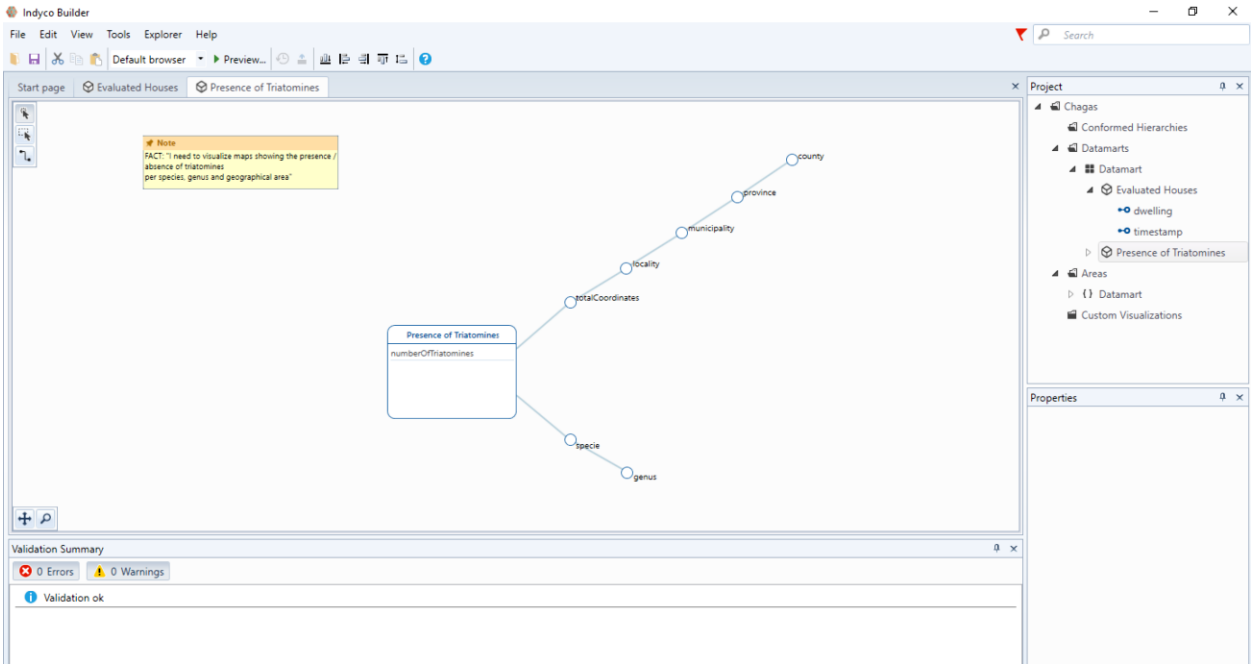


Figure 26. MD model obtained for requirement IR1

Work done to satisfy requirement IR2

Similarly as done before, for the second requirement we need to create the MD model that allows to obtain the information about the correspondent fact, considering the available ontology.

Therefore, we need to analyze "*the temporal variation of evaluated (inspected) houses by department every two years",* specifically for La Rioja (Argentina).

To achieve this, we simply need to get the measure "number of inspected houses" according to the selected timeframe and place. Thus, in this case, we have a time dimension, (with granularity that comes down to year, month, day and timestamp), and a geographical dimension (with granularity of dwelling or house, locality, municipality and province). If we wanted to obtain the number of evaluated houses for la Rioja, we would need to select "La Rioja" as the desired province, and then select two years as timeframe for the analysis. With this, we get the information about this fact, and we define the required fields to build the MD model for this requirement, which is shown is Figure 27.

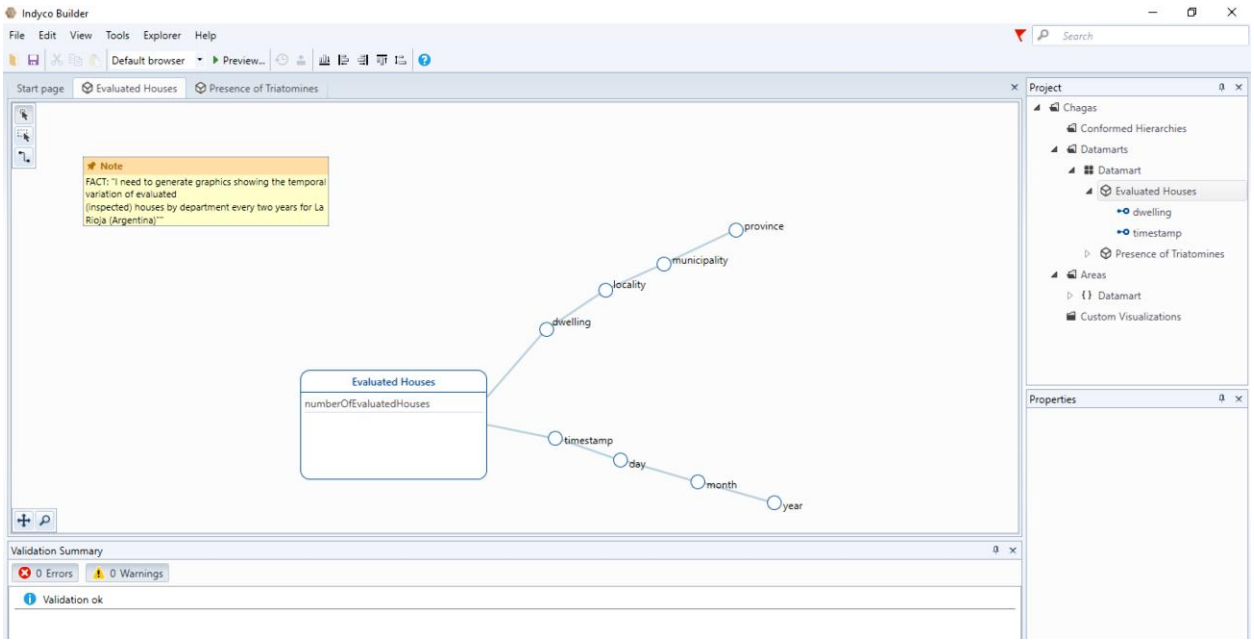


Figure 27. MD model obtained for requirement IR2

5.1.2.2 Designing ETL with Pentaho

After defining the conceptual MD schema model to satisfy the use case requirements, we need to proceed developing the ETL design to extract and transform the data available in the data sources and prepare it for being loaded into the created MD schema.

The first step in an ETL design is to extract the data from the available sources. At the moment, this data is being collected and stored through a system called *WHO Data Integration Platform (WIDP)*. The system is used for collecting data about Neglected Tropical Diseases (including Chagas Disease) and it is currently being used by several departments at WHO. It also includes features

for configuring settings, as well as for visualizing data in real time. This system is based in DHIS2 tool, which may be defined as follows:

*"DHIS2 is a tool for collection, validation, analysis, and presentation of aggregate and patient-based statistical data, tailored (but not limited) to integrated health information management activities. It is a generic tool rather than a pre-configured database application, with an open meta-data model and a flexible user interface that allows the user to design the contents of a specific information system without the need for programming. DHIS2 is a modular web-based software package built with free and open source Java frameworks"[**Error! Reference source not found.**]*

In the DHIS2 system, there is a special application for entering *Events* (also called cases or records) and storing them into their entities (called *Programs* in the DHIS2 system). These Programs belong to a particular *Organization Unit*, which is the geographical area or location where this Program may be configured for. *Organization Units* may be classified in continents, countries, areas, cities and more, and they are previously defined in the system.

The application used to enter Events into the WIDP system is called Event Capture app, and it is used for registering events that occurred at a particular time and place. In DHIS2, events are always linked to a program. The Event Capture app allows to select the organization unit and program and specify a date when an event occurred, before entering information for the event. In the Annex 1-2, it can be seen the different forms in the Event capture app to collect data for the two Programs that we will use for this use case.

To access the data, the WHO technical staff has prepared an API, that distributes the data in JSON format. The WHO Web API adheres to all the principles of a REST architecture API. This means, it uses all HTTP methods such as GET, POST, PUT and DELETE which use JSON data objects. Also, the API has Basic and OAuth2 Authentication as a Security Layer [**Error! Reference source not found.**]. To access the events from a particular Program, we need to indicate the ID of the program as an input parameter for a GET request.

5.1.2.2.1 Extraction from sources

To extract data from the WHO sources for beginning the development process, are interested in obtaining the events from two programs:

- Chagas disease – Dwelling Triatomine Bug Study (For Requirement IR1)
- Chagas disease – Dwelling inspection (For Requirement IR2)

To do this, we first obtained these programs' IDs and then we consumed the WHO API introducing the following REST requests:

For Requirement IR1:

```
GET /api/26/events.json?program=j4Sw9ZEOsYh HTTP/1.1
Host: who-dev.essi.upc.edu:8081
```

For Requirement IR2:

```
GET /api/26/events.json?program=k3pL2hPUqQi HTTP/1.1
Host: who-dev.essi.upc.edu:8081
```

Then, after obtaining the data, we proceed to the design of the ETL in Pentaho. In the next section we will begin to explain the transformation process necessary for each of the requirements and the obtained results.

5.1.2.2.2 Transformation process

Now that we have defined the method for extracting data from the sources, we proceed transforming the data. In the following paragraphs we will explain the steps done to do this for each of the requirements

Requirement IR1

The ETL design applied for the first requirement can be observed in the figure.

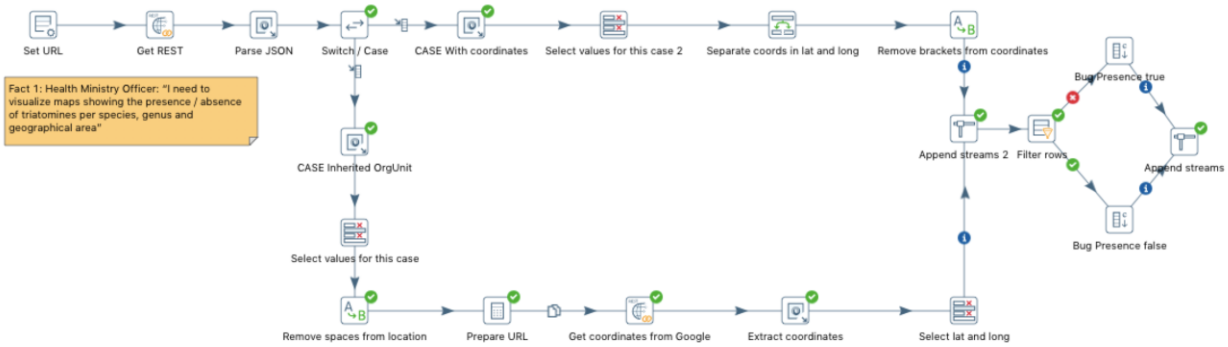


Figure 28. ETL design obtained for requirement IR1

The steps applied were:

1. Generate Rows: Define a constant variable to store the API's URL and Authentication information to extract data from the sources.
2. REST Client: Perform the HTTP GET request operation

3. JSON Input: Parse JSON data into columns and specify its data types. To do this, we used JSONPath expressions, to indicate which keys in the JSON object correspond to each columns (for more information, see [37])
4. Switch: Step to decide which transformation process will be applied next, depending on how the location information for the event was introduced: by Organization Unit (city, country, continent, etc), or by coordinates. Mainly, to create new events through the WHO forms (for example, a new Triatomine bug study performed), we can specify the place of the study in two different ways: by specifying the organizational Unit where it was done, or by specifying its coordinates. This way of introducing geographical location for the events produced several number of problems that we will explain in detail in the next chapter, as part of the conclusions for this thesis.

5. a. Case Inherited Organization Unit: This path is taken by the data objects that were introduced specifying an Organization Unit. Here we extract the JSON fields needed and select the ones we are interested in for our study.

    5.a.1 Remove spaces from location: Here we get the name of the organizational unit, and remove spaces from the string to prepare it to be later introduced in an URL. The intention is to use the Google Maps API to get the coordinates of the place by providing the name of the organizational unit

    5.a.2 Prepare URL: Concatenate different parts of the URL (including the name of the organizational unit, which is a dynamic field), to be ready for use.

    5.a.3 Get coordinates from Google: Perform the HTTP GET request to the Google Maps API to obtain the desired coordinates. An example of the usage of this API is shown in Annex 3, extracted from [25].

    As can be seen, this API extracts coordinates depending on the address introduced. However, this address may be more or less specific, depending on what the user had entered in the data collection forms. For this reason, it is possible that the API gives inaccurate results due to the ambiguity of the address in the input. For example, if we enter “Santa Barbara” in the API, it could mean the Santa Barbara city in California (United States), or it could mean a place in Mexico city (Mexico). Given that, the API has no way of knowing which of the two choices is correct, therefore, it will give the one by default in their system.

    This situation requires additional handling for this type of cases that we are not considering in this work, but they should be taken into account for further improvements of this design.

    5.a.4 Extract coordinates: Extract fields from the JSON response using JSON path expressions
    5.a.5 Select values: Selecting the desired fields

b. Case With Coordinates: This path is taken by the data objects that were introduced specifying Coordinates. Here, same as before, we extract the JSON fields and select the ones of interest.

5.b.1 Split fields and Replace in String: These steps were made for separating the coordinates in two different columns: latitude and longitude. Originally, it came as a single string with brackets on their sides (f.eg "[-30.5, 50.3]").

Finally, the two ways are unified into one single path:

6. Append streams: Which joins the data that came from both paths into one single repository
7. Filter rows, Presence True/False and Append Streams: At this point, we obtain the number of infested bugs in the study as a column. In this step we filter the ones whose number of infested bugs is zero, versus the ones whose number is not zero. Then according to this, we add a new column specifying if there is presence of infested triatomines in the study or not, and then we join data again.

Therefore, we obtain the following results in a single repository

#	ID	program	orgUnit	specie	genus	numBugs	lat	long
1	Wfd1uvR9sbN	j4Sw9ZEOsYh	Global	Panstrongylus rufotubercul...	PANSTRONGYLUS	0	-32.43	50.43
2	iCUsDblmGwj	j4Sw9ZEOsYh	EUR	Panstrongylus lutzi	TRIATOMA	100	20	20
3	ChlCtlqCiXQ	j4Sw9ZEOsYh	Global	Panstrongylus howardi	TRIATOMA	100	-6.25	53.35
4	XBgxhj0DyZf	j4Sw9ZEOsYh	Global	Panstrongylus chinai	RHODNIUS	50	-7.77	53.2734
5	p0A89ciCZKn	j4Sw9ZEOsYh	Bosnia and Herzegovina	Panstrongylus rufotubercul...	TRIATOMA	1500	43.9158...	17.6790...
6	Eh1alONdYfo	j4Sw9ZEOsYh	EUR	Panstrongylus rufotubercul...	MEPRAIA	150	43.9158...	17.6790...

Figure 29. Results obtained for requirement IR1

As an addition to this result, we could have added another geographical API in order to get more details of the geographical location by providing its coordinates (having all data unified into a single way of providing geographical information, through coordinates). From the API, we can obtain more insights such as the city, area, province and country.

Requirement IR2

After developing the ETL design for the first requirement, we proceed to do the same for requirement IR2. Resulting ETL process design is shown in Figure 30.

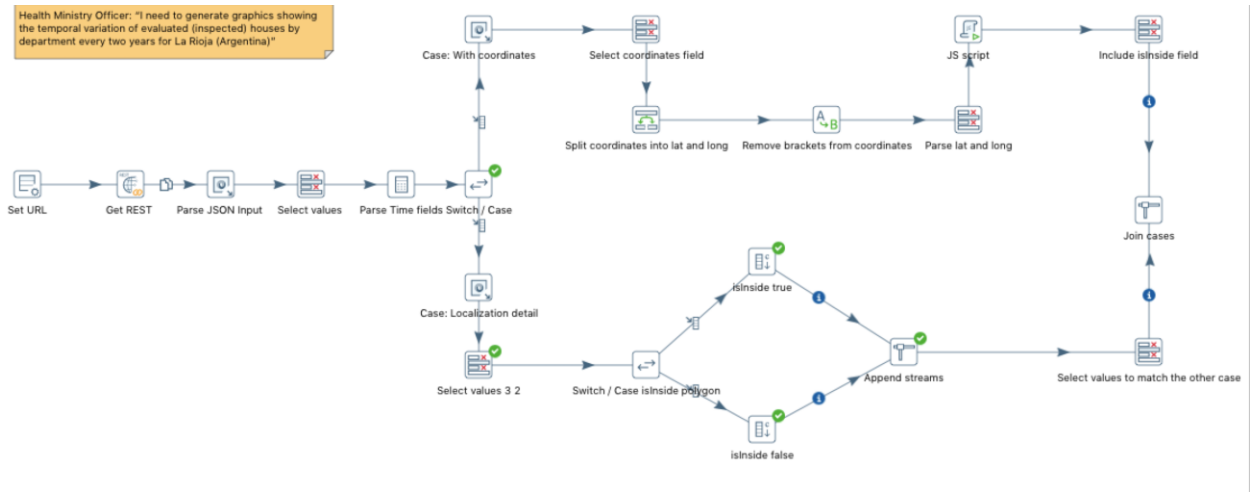


Figure 30. ETL design obtained for requirement IR2

In this case, we applied the following steps:

1. Generate Rows: Define a constant variable to store the API's URL and Authentication information.

2. REST Client: Perform the HTTP GET request operation

3. JSON Input: Parse JSON data into columns using JSONPath expressions and specify its data types.

4. Select: Select the fields that we are interested in

5. Time fields: Parse the date and time of the event into separate columns of year, month and day

6. Switch: Same as before, we may have different ways for specifying geographical location. In this case, we will analyze the case "Localization detail" and "Coordinates". First, we will analyze the case when the event comes with coordinates:

    6.a.1 We perform four steps (Select, Split fields, Replace in String and Select): These are for parsing the "coordinates" field into two separate columns: latitude and longitude and then parsed them as numbers. As explained before, the coordinates field comes like a String value in this format "[latitude, longitude]".

    6.a.2 Modified Javascript Value: In this step, we perform an operation using Javascript code, in order to determine if a specific coordinate is inside or outside of a geographical polygon. For this use case, we are interested in knowing if a certain geographical point belongs to La Rioja province in Argentina or not.

First we had to investigate the surrounding coordinates for La Rioja. To do this, we used an API called MapIt, from an English company called My Society [26]. This API allows to obtain geographical data by providing an area, postal code, coordinates and more. Not only gives general information about the location, but it is able to export files such as GeoJSON, WKT or KML which may be imported in another system and used for obtaining geographical information about the place. In this case, we were interested in obtaining the GeoJSON file for "La Rioja" province, which consists in an array of coordinates that specify the borders of the location (or the polygon in this case). An image of the preview for download in the MapIt webpage can be seen in Figure 31.

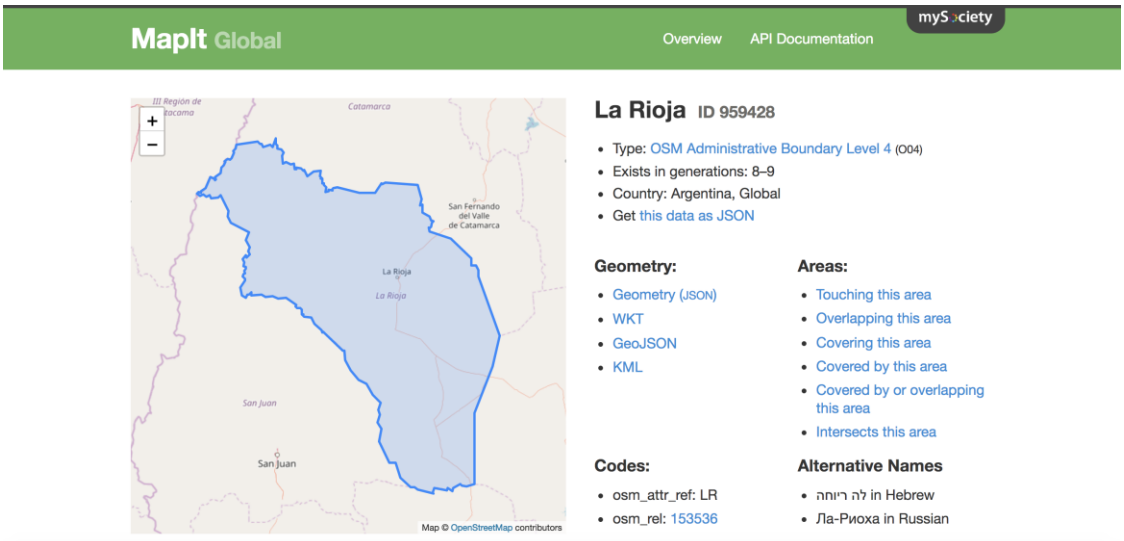


Figure 31. MapIt tool screenshot [26]

After downloading the GeoJSON file and trying to use it in Pentaho as "the polygon" we aim to take as reference, we obtained an error of overloaded memory. Meaning, this GeoJSON file was giving too many coordinates to specify the surrounding polygon. Therefore, we had to apply a tool for simplifying the polygon into fewer border coordinates. To do this, we used "MapShaper", which is an open tool to import a file to define the geometry of the shape, and simplify it specifying a simplification rate and method. [26]. The simplified shape is shown in Figure 32.

Figure 32. MapShaper tool screenshot [27]

After downloading the new GeoJson file, we proceed to use it in our Modified Javascript Value step. The script can be observed in the figure. The function to calculate if a point is inside a polygon was extracted from a public research work that counted with a MIT license [28].

Script 1

```
//Script here
//var laRiojaOriginal = [ [ -66.661282899999989, -31.9210387 ], [ -66.6049825, -31.9230227 ], [ -66.6051144, -31.9336613 ], [ -66.5686971, -31.9
var laRiojaSimplified = [[-31.9982346, -69.6422254] , [-27.7097658, -69.6422254], [-31.9982346, -67.18208248052053], [-27.7097658, -67.18208248

// Example points
var pointInside = [-29.211167, -67.671553];
var pointOutside = [-31.167234, -69.857833];

var coords =  [lat,long];
function inside(point, vs) {
    // ray-casting algorithm based on
    // http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html

    var x = point[0], y = point[1];

    var inside = false;
    for (var i = 0, j = vs.length - 1; i < vs.length; j = i++) {
        var xi = vs[i][0], yi = vs[i][1];
        var xj = vs[j][0], yj = vs[j][1];

        var intersect = ((yi > y) != (yj > y))
            && (x < (xj - xi) * (y - yi) / (yj - yi) + xi);
        if (intersect) inside = !inside;
    }

    return inside;
};

var isInside = inside(coords, laRiojaSimplified);
```

Figure 33. Javascript code used for calculating point in polygon presence

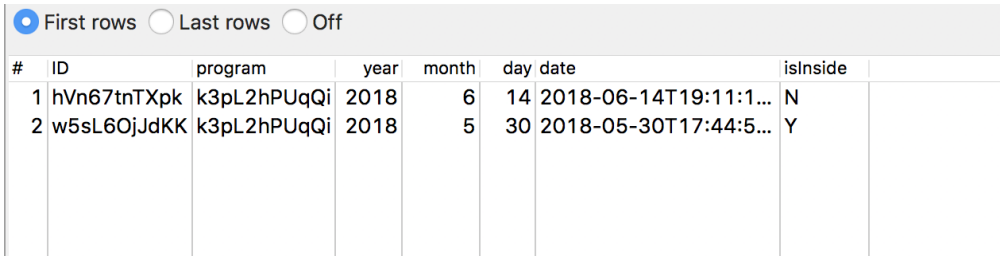
6.a.3 Select: Finally, we select the fields of interest before merging this data with the one coming from the other branch (the events entered with "Localization detail" information).

Now, we can begin to explain the steps followed in the other case: Events with Localization Detail information.

6.b.1 Steps "Localization detail" and Select: These steps extract the required fields from the JSON object and selects and parses the columns of interest for this analysis. Here, we obtain the "Address" field from the input data.

6.b.2 Steps isInside true/false, Append Streams, Select: These steps check if "La Rioja" and "Argentina" strings are included in the address column. If this is true, the "isInside" column will be true. On the contrary, it will be false. Then, the system merges both outputs into one single repository and selects the final fields that will be in the final data model.

7. At the end of the process, we apply the final step "Append Streams", to join the output of the two branches together: the one that came from the "Coordinates" case, and the other which came from the "Localization details" case. Finally, we obtain the following output



#	ID	program	year	month	day	date	isInside
1	hVn67tnTXpk	k3pL2hPUqQi	2018	6	14	2018-06-14T19:11:1...	N
2	w5sL6OjJdKK	k3pL2hPUqQi	2018	5	30	2018-05-30T17:44:5...	Y

Figure 34. Results obtained for requirement IR2

This repository allows to show the temporal variation of inspected houses for La Rioja, if we perform the desired operations of aggregation and selection. We would need to specify the granularity of the temporal variation and calculate the number of rows (events) to obtain each point of the chart. Also, we could simply filter the rows where *isInside* equals true, to find out which are the inspected houses in La Rioja.

Overall, ETL process design is known to be difficult, and as seen here it is rather burdensome even for the couple of small examples of our use case.

5.2 Semi-Automatic method: Quarry

From previous chapters we recall that Quarry is a semi-automatic system capable of creating the ETL design and MD model for DW system, based on some input requirements entered in a machine-understandable format. Some of these inputs required are the ontology, which describes the semantics of data sources under consideration and gives mappings to the data sources. The ontology is enriched with a domain vocabulary to enable better understanding by the end users. In summary, the inputs required by Quarry are:

- Domain ontology and source mappings
- Structured requirements, entered using the UI provided by the tool itself.

From the RE4DSS process, the WHO team was able to define the vocabulary in a diagrammatic representation which helped us create the ontology in the OWL language. Also, the structured requirements are constructed by specifying measures and dimensions which are collected by the UI in Quarry (see Figure 13).

After providing the required inputs, Quarry generates the ETL process given the specified requirements. However, the Quarry system's transformation process is limited to a set of operators, namely [13]:

- Extraction
- Selection
- Union
- Intersection and Minus
- Join
- Aggregation
- Renaming
- Projection
- Function

Therefore, the ETL design structure shown in section 5.1 (specifically, the switches that allow treating data differently according to its structure) may not be possible considering this set of operators. With this being said, we can foresee that results won't be as effective, because source data does not always come with the same structure. As we are obtaining data via REST API, the response comes in JSON format. This implies that data is non-SQL and it does not have a static schema for data to be treated always in the same way. We will explain this in more detail in chapter V, when detailing the Effectiveness of both methodologies.

To illustrate this, we will proceed to conceptually explain how would Quarry design the ETL process given the requirements we are evaluating (IR1 and IR2).

5.2.1 Requirement IR1

To satisfy this requirement, we will need the following entities in the ontology: the Geographical Object, Species and the Triatomine Bug Study. As can be seen in the vocabulary, the Geographical Object may be represented by coordinates and a polygon (area).

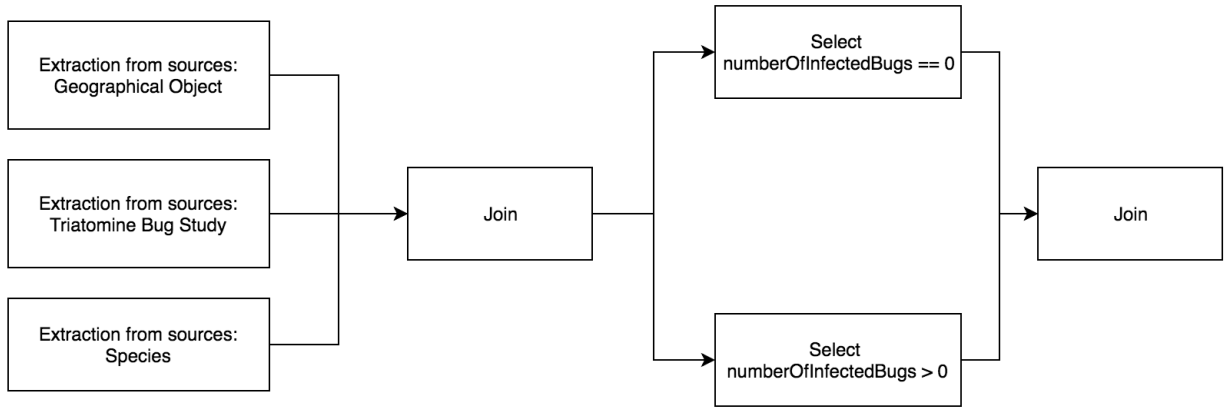


Figure 35. Quarry's conceptual design for requirement IR1

Using the domain ontology, Quarry will proceed to identify the different entities required to satisfy the given information requirement, corresponding them to the source mappings also provided at the beginning of the process. Therefore, after identifying these entities, Quarry proceeds to join them in order to obtain the final MD schema.

5.2.2 Requirement IR2

For this requirement, we need the following entities in the ontology: Dwelling Inspection, Geographical Object and Temporal Object. The ETL process in Quarry can be represented in the following diagram

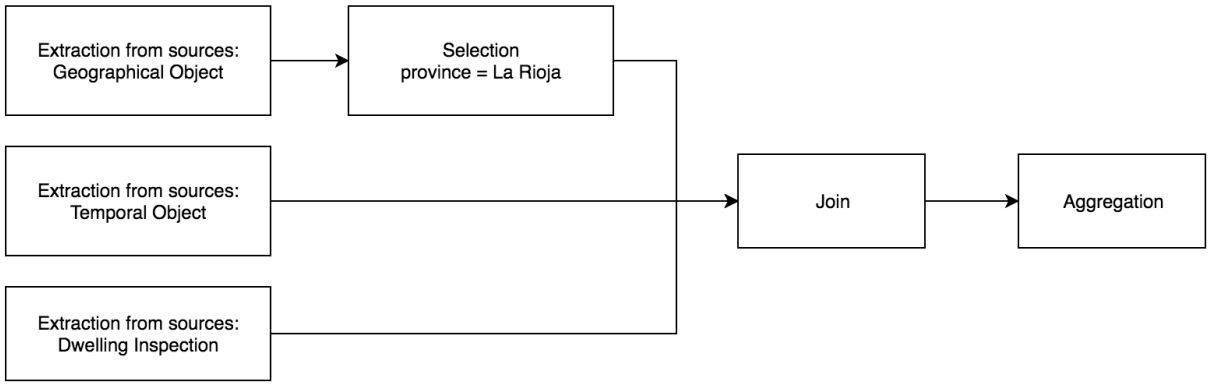


Figure 36. Quarry's conceptual design for requirement IR2

Similarly as before, Quarry identifies the necessary entities from the ontology to satisfy the requirement. Then, for the Geographical Object entity, we select only the objects that correspond to La Rioja, Argentina. We then proceed to join and aggregate them by performing a summation. Finally, we obtain the partial MD schema for this requirement.

Chapter VI

Comparison between the traditional and semi-automatic approaches

Generally speaking, after analyzing the design process with two different approaches, we may reach to several conclusions about the usage of each of the methods for creating a MD model and ETL design. One of the most relevant challenges that we faced during the design process was the fact that the geographical location for the events may be entered in many different ways into the sources. That fact raised several number of inconveniences. For example, the JSON object obtained to represent the events of a program are structured differently depending on the way the location is entered.

Thanks to the flexibility of non-SQL structures, such as JSON, systems are able to allow this behavior in the structure of the response and return all the information in one single object. However, when processing this structure, we need to consider different cases depending on the type of JSON object that we receive.

On the other hand, the manual work and complexity required to create a design for the DW is a lot higher in the traditional methods that are completely manual in comparison with the semi-automatic ones. Also, semi-automatic tools like Quarry add an additional benefit which is the automatic validation of the obtained output which guarantees the soundness of the obtained structures, which cannot be done as this straight forward in the traditional tools.

Altogether, the following table shows a comparison between the traditional methods vs. the semi-automatic methods, according to different criteria.

Table 2. Comparison between traditional methods and semi-automatic methods

Aspect	Traditional methods	Semi-automatic methods
Manual effort	Higher (-)	Lower (+)
Complexity	Higher (-)	Lower (+)
Effectiveness	Higher (+)	Lower (-)
Data sources format	Any (+)	Preferably SQL data sources (-)
External tool integration	Any (+)	Limited (-)
Functional requirement satisfiability	Not guaranteed (-)	Guaranteed (+)

Soundness	Not guaranteed (-)	Guaranteed (+)

As can be seen in the table, we introduced seven relevant aspects or criteria to perform the evaluation: *manual effort*, *complexity*, *effectiveness*, *data sources format*, *external tool integration*, *functional and non-functional requirement satisfiability,* and *soundness*. We further discuss the ability of each approach according to these aspects, based on the results obtained in the analysis from previous chapters. In the next couple of paragraphs, we will explain each of these aspects followed by our general conclusions.

Manual effort

With traditional methods, the manual labor includes lots of human intervention. First, the requirement elicitation involves an important amount of communication between the stakeholders and the IT staff to perform the core activities and obtain the requirement artifacts. After that, the design process implies first the manual design of MD model using tools that require direct human management. Secondly, traditional ETL tools require that the developer manually design each of the steps for properly extracting, transforming and loading data. Not to mention that if a new requirement is added, the whole development process needs to be revised or worse, redone. On the other hand, semi-automatic tools need the required system's inputs, beside the work implied by the requirement elicitation process with the stakeholders using the tool UI. This is, manually creating the input files (ontology and source mappings) once for defining the underlying data sources and then entering the requirements using the UI. In contrast with the traditional method, if we needed to add a new requirement to the system, we would just need to enter it using the UI tool, and it will re-use the previously entered ontology and source mappings. Therefore, it is clear that semi-automatic tools take a lot less manual effort than traditional tools.

Complexity

Similarly, as stated before, the complexity involved in the design using traditional tools is higher than the one implied by using semi-automatic tools. For traditional approaches, we need to analyze each step of the process, which require a lot of research, analysis and continuous validation of the entire process. On the other hand, with semi-automatic tools we need to work on the required inputs, which generally imply using a tool to construct the ontology (like Protégé, for example). The required inputs that are manually built are the source ontology and the source mappings, which need to be done just once. The rest of the definition process may be done interacting directly with the tool and automatically obtaining the design. However, manually building the source ontology and source mappings may be an extensive and complex task. Thus, there are methods to automatically obtain them. For more details, we refer the reader to [34].

In any case, with all these facts considered, traditional methods imply higher complexity than semi-automatic methods. The traditional methods imply using a series of different tools to obtain both

the MD model and ETL process, that satisfy the defined information requirements (as seen in Chapter V). In contrast, Quarry is a single integrated tool that handles both MD model and ETL process designs at the same time.

Effectiveness

There are cases where semi-automatic tools are not that effective in the given results in comparison with traditional methods. When it comes to processing data that is not completely structured in SQL-like schemas such as JSON data, semi-automatic tools need to face a "decision-making" process in order to treat data differently depending on a given structure or the other. Quarry is an example of this. In fact, the system assumes that all data that will be processed is structured according to the given source mappings. It does not expect these variations in the data's structure.

To illustrate this, we show the figure below. There, we see that there are four ways of defining the place of occurrence for an event: GPS coordinates, inherited Organizational Unit, Location detail and GPS coordinates organizational unit. Also, the user can insert data in the four ways, as he or she decides.

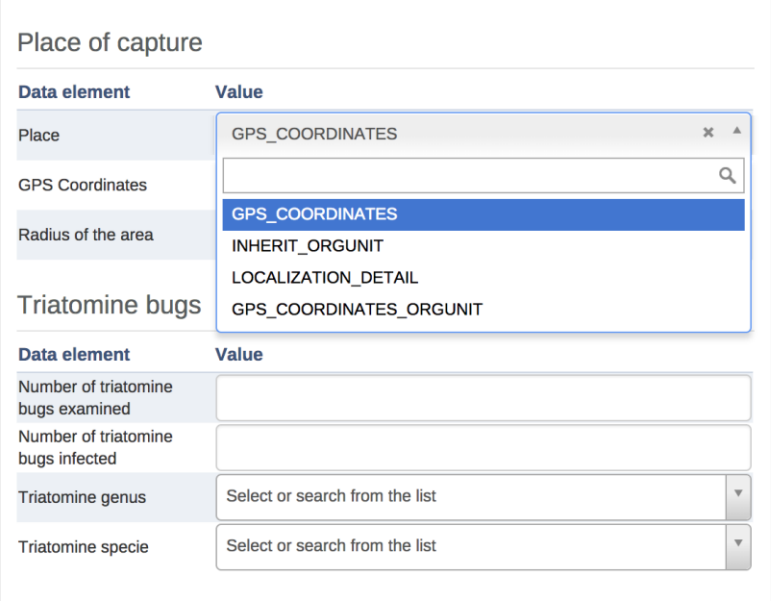


Figure 37. Options for inserting "Place" in the data collection forms for WHO system

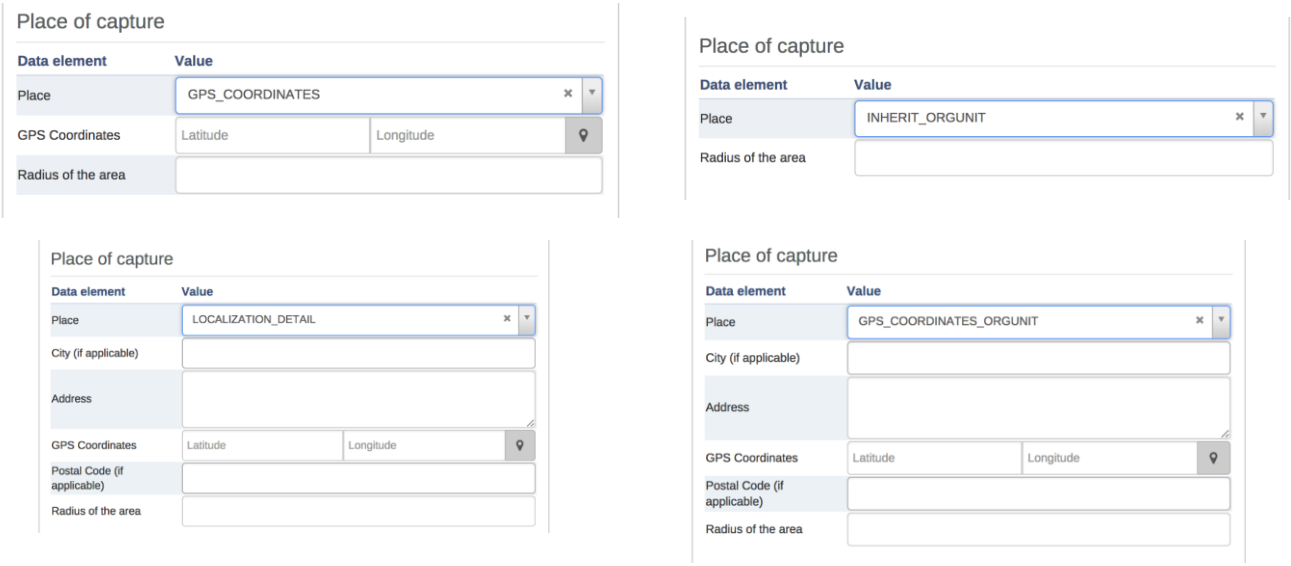


Figure 38. Place description fields according to the selected option in the data collection forms for WHO system

Therefore, when we are extracting data, the structure of the JSON object will change depending on the way this location was entered. In the Annex 4, we show the obtained response after collecting all events inside the Program "Dwelling Triatomine Bug Study". There are two events in the response, and the field "dataValues" contains the values for the columns of the table. If we compare the structure of these fields for both events, we can immediately notice that the structure is different.

This difference in structure may be well represented in JSON format. However, as JSON by definition is a non-structured data representation, sources can come in different formats and therefore, Quarry will not be able to handle this effectively. In contrast, with traditional tools a user is able to define each step of the process required to handle such cases.

Data sources format

As a consequence of the previous aspect, semi-automatic tools will perform better if they work with SQL-like structured data sources. That is, sources with delimited and specified attributes and entities. In the worst-case scenario, where sources may come in flexible structures, data may need to be pre-processed in some way before being entered in Quarry system, to ensure that it always complies with a given structure. Traditional tools, on the other hand, may receive any type of data sources as we will specifically define how this data is going to be processed.

External tool integration

It is known that Quarry offers plug-in capabilities for adding import and export parsers as part of its "Communication & Metadata" Layer, in order to support external notations such as SQL, Apache PigLatin, etc (see Section 2.4.1.5). However, this capability is very limited in comparison to the integration options a traditional tool (like Pentaho) could offer. There are much more protocols

and tools with many different purposes that may be integrated into a traditional tool, as we are able to control exactly how this tool or plug-in is going to be applied to the data. Therefore, the traditional method using existing tools from the market like Pentaho have an advantage in this aspect.

Functional requirement satisfiability

One of the advantages that Quarry offers is that it automatically guarantees each requirement satisfiability as it builds the final MD model and ETL design. First, it creates the designs for each one of the input requirements separately, and then, while integrating them together, it provides an automatic validation of the requirements. Moreover, the obtained results are optimized to comply with the required quality standards such as performance, fault tolerance, etc. In contrast, traditional methods do not provide more validation processes different than the developer can provide itself, which is frequently highly error prone.

Soundness

Similarly, for each new, changed, or removed requirement the Quarry system enables a series of validation processes to guarantee the soundness of the updated design as well as the satisfaction of all the requirements. This means, it verifies that the current design meets all the multidimensional integrity constraints [35]. On the contrary, traditional methods are not able to do this automatically. It would require the developer to ensure this manually, which again, is very error prone.

Overall, as seen in the analysis for each of the different criteria, we conclude that semi-automatic tools seem to be a very good option in terms of simplicity and speed of the design process. However, there are some inevitable limitations that make them weaker in comparison to traditional approaches. For instance, they are weaker in terms of effectiveness in comparison to traditional approaches, due to its limited set of available operators. However, they add important contributions to the design such as the validation process for soundness and requirement satisfiability, but they still need to improve their handling for non-structured data sources.

Chapter VII
Conclusions

Currently, in this technological world and this digital era, Decision Support Systems are crucial to a business or any kind of organization's future. They have become a valuable tool that helps leaders and managers to obtain useful insights about the state of their businesses, which may allow them to take important decisions in order to react on external and internal changes. As information becomes more abundant and the ability to extract and digitalize data becomes easier, the necessity to improve the methods and techniques used to process this information becomes more urgent.

In the health sector, particularly in the WHO organization the initiative to create a centralized database to gather all relevant information may become crucial to accurately make decisions on effective ways for controlling diseases around the world. The WHO organization has an important responsibility in society, so they surely need an effective system to help them achieve these goals.

For these reasons, the role of an efficient and scalable system that is able to achieve desired results is very important, not just for the organization but in a more global scale. An efficient system is evidently done depending on the used methodologies. That is why is crucial to properly analyze and evaluate the used approach for developing the design before starting the actual work. This thesis intended to make this comparison easier, by providing different aspects and criteria that helped the reader to be aware of certain facts regarding each one of the methodologies which may become handy before reaching to a decision.

After having performed the different analysis steps on this project, we came to the following conclusions:

- The Requirement Engineering process has become an integral part for successfully designing any IT project. In fact, it is highly recommended to include it into the project's lifecycle to ensure a solid base before starting the actual development. In Decision Support Systems, we have presented an adapted approach of the RE process called Requirement Engineering for Decision Support Systems (RE4DSS), which we explained in one of the chapters. This approach has been introduced in previous works [12] and has been proven to be effective in this type of systems. In fact, it is very helpful to define and prepare requirements to be easily integrated and considered into the design of the DSS. In this thesis work, the RE4DSS approach has been used as a preparatory phase before developing and analyzing the different design methodologies. Independently of the used RE process, we highly recommend its use in order to ensure solid communication and understanding between the different project's stakeholders and provide useful input artifacts before starting the development phase.
- In technical projects in the health sector such as this one by the WHO, it is important to have a common vocabulary in order to have clarity on the terminology and avoid future

ambiguities and confusions. That is why it is recommended to follow the RE4DSS approach, as it gives tools and methods to handle the creation of such vocabulary among the stakeholder.

- Traditional tools and methodology have a wide variety of options in the market to design this type of systems, according to each business needs and possibilities. However, they involve a huge amount of manual labor and complexity and they could be highly error prone as they require constant human intervention and team management.
- Semi-automatic tools are a newly approach provided mainly in the literature by now. They are a promising option for the future of DSS design, but they need improvements and further developments and research to be of more effective use for the end user. Quarry, particularly, still needs to improve its handling for non-SQL sources, but it has many useful and promising features such as the automatic validation capability.

In essence, it has been clear that the selected methodology before executing the MD model and ETL proess is crucial for defining the system's efficiency and quality. This, in combination with the adequate RE approach, become extremely important when it comes to delivering successful DSS projects. Thus, this thesis aimed to present, analyze and compare the two possible methodologies for creating the DSS system's design. The traditional approach has proven to be more effective due to the wide range of different operators and features that it offers, but it can be highly error prone and far more extensive and complex. The semi-automatic approach on the other hand, demonstrated to be more rigid. This is because it counts with a very limited set of operators and requires the extracted data to be well-structured and defined (non-SQL preferably). However, it is a more simpler and faster approach and it guarantees the validation of some basic aspects for the design such as soundness and requirement satisfaction.

The intention of this work is to provide the reader some differentiator aspects to compare both approaches, with the hope of making the decision easier and clearer. However, there are still many possible extensions of this work, in order to validate and complement the presented information. For instance, the created design following the traditional approach may be improved to make it more accurate and precise. The geolocation information extracted and transformed from the sources should be validated and amplified during the ETL process, in order to prevent possible errors caused by unambiguity. This would definitively give additional insights about the mentioned analysis and would enrich our conclusions.

Ultimately, there is still a long road ahead, but we consider this work a starting point for future discussions and analysis. There is a promising future for DSS systems and we are approaching ourselves to surprising limits in the big data and business intelligence world.

Bibliography

1. Gartner: *Business Intelligence (BI).* Gartner IT Glossary. https://www.gartner.com/it-glossary/business-intelligence-bi/ [Online; accessed 08-October-2018]

2. World Health Organization. About who. http://www.who.int/about/en/, 2014. [Online; accessed 11-July-2014].

3. Lorenzo. Savioli, Denis. Daumerie, D. W. T. Crompton, Patricia. Peters, and World Health Organization.

4. Working to overcome the global impact of neglected tropical diseases : first WHO report on neglected tropical diseases. World Health Organization Geneva, Switzerland, 2010.

5. Matteo Golfarelli and Stefano Rizzi. Data warehouse design: Modern principles and methodologies. McGraw-Hill, Inc., 2009.

6. Oracle. Oracle® OLAP Application Developer's Guide. https://docs.oracle.com/cd/B13789_01/olap.101/b10333/multimodel.htm, 2003. [Online; accessed 08-October-2018]

7. Connolly, Thomas; Begg, Carolyn. Database Systems - A Practical Approach to Design, Implementation and Management (6th ed.). Pearson. Part 9 Business Intelligence, 2014.

8. Indyco. The Dimensional Fact Model. Indyco. https://www.indyco.com/dimensional-fact-model/ [Online; accessed 11-July-2014].

9. Splice Machine. ETL Process Acceleration. Splice Machine, 2018. https://www.splicemachine.com/applications/etl-process-acceleration/ [Online; accessed 11-July-2014].

10. The Standish Group International. Chaos manifesto 2013. http://www.projectsmart.co.uk/docs/chaos-report.pdf, 2013. [Online; accessed 01-February-2014].

11. Klaus Pohl. Requirements Engineering - Fundamentals, Principles, and Techniques. Springer, 2010

12. García, Stephany. A Systematic Requirements Engineering Approach for Decision Support Systems. UPC, 2014.

13. Oscar Romero, Alkis Simitsis, Alberto Abelló: GEM: Requirement-Driven Generation of ETL and Multidimensional Conceptual Designs. DaWaK 2011: 80-95

14. Jovanovic, Petar. Romero, Oscar. Simitsis, Alkis. Abelló, Alberto. Candón, Héctor. Nadal, Sergi. Quarry: Digging Up the Gems of Your Data Treasury. EDBT, 2015: 1-4.

15. Jaume Vinas Navas. Building a data warehouse for the global who information and surveillance system to control/eliminate chagas disease captura de requisits per a un sistema d'informacio i vigilancia a la oms per a la malaltia de chagas. https://upcommons.upc.edu/handle/2099.1/20800?locale=es, 2014. [Master in Computing; accessed 21-February-2014].

16. WHO Staff. Chagas disease (american trypanosomiasis). http://www.who.int/neglecteddiseases/diseases/chagas/en/,2014. [Online; accessed11 _ July _2014].

17. Rozendaal, Jan A. Vector control: Methods for use by individuals and communities. WHO, 1997. http://www.who.int/water_sanitation_health/resources/vector210to222.pdf. [Online; Accessed 08-October-2018]

18. Sustaining the drive to overcome the global impact of neglected tropical diseases: second WHO report on neglected diseases. WHO/HTM/NTD/2013.1.World Health Organization, Geneva, advanced proof edition, 2013. Copies of materials can be requested through: NTD Information Resource Centre

19. Who. Chagas disease: control and elimination. Sixty Third World Health Assembly, 2010.

20. Ralph Kimball. The data warehouse lifecycle toolkit: expert methods for designing, developing, and deploying data warehouses. John Wiley & Sons, 1998.

21. Stanford Center for Biomedical Informatics Research. Protege. Stanford University, 2016. https://protege.stanford.edu/about.php. [Online; Accessed 08-October-2018]

22. Indyco. Indyco features. Iconsulting. https://www.indyco.com/features/ [Online; Accessed 08-October-2018]

23. Hitachi Vantara. Pentaho. https://www.hitachivantara.com/go/pentaho.html [Online; Accessed 08-October-2018]

24. DHIS: DHIS2 Developer guide. Health Information Systems Programme (HISP). https://docs.dhis2.org/ [Online; Accessed on 10-October-2018]

25. Google Maps Platform. Geocoding API - Developer Guide. Google, September 25th 2018. https://developers.google.com/maps/documentation/geocoding/intro. [Online; Accessed 08-October-2018]

26. MapIt UK. MapIt. My Society. https://mapit.mysociety.org/ [Online; Accessed 08-October-2018]

27. Bloch, Matthews. MapShaper. https://mapshaper.org/ [Online; Accessed 08-October-2018]

28. Halliday, James. Github Repository: point-in-polygon. Substack. https://github.com/substack/point-in-polygon. [Online; Accessed 08-October-2018]

29. Stephano Rizzi. Conceptual Modeling Solutions for the Data Warehouse. University Of Bologna, 2008. https://www.researchgate.net/publication/237622501_Chapter_I_Conceptual_Modeling_Solutions_for_the_Data_Warehouse [Online, Accessed 10 October 2018]

30. Stefan Urbanek. OLAP Cubes and Logical Models. Open Knowledge Labs, 2014. http://okfnlabs.org/blog/2014/01/20/olap-cubes-and-logical-model.html. [Online, Accessed 10 October 2018]

31. Wikipedia. Triatominae. Wikipedia, 2018. https://en.wikipedia.org/wiki/Triatominae [Online, accessed 10 October 2018]

32. Wikipedia. Taxonomy (biology). Wikipedia, 2018. https://en.wikipedia.org/wiki/Taxonomy_(biology) [Online, accessed 10 October 2018]

33. Stephany García, Oscar Romero, Ruth Raventós: DSS from an RE Perspective: A systematic mapping. Journal of Systems and Software 117: 488-507 (2016)

34. Rizkallah Touma, Oscar Romero, Petar Jovanovic: Supporting Data Integration Tasks with Semi-Automatic Ontology Construction. DOLAP 2015: 89-98

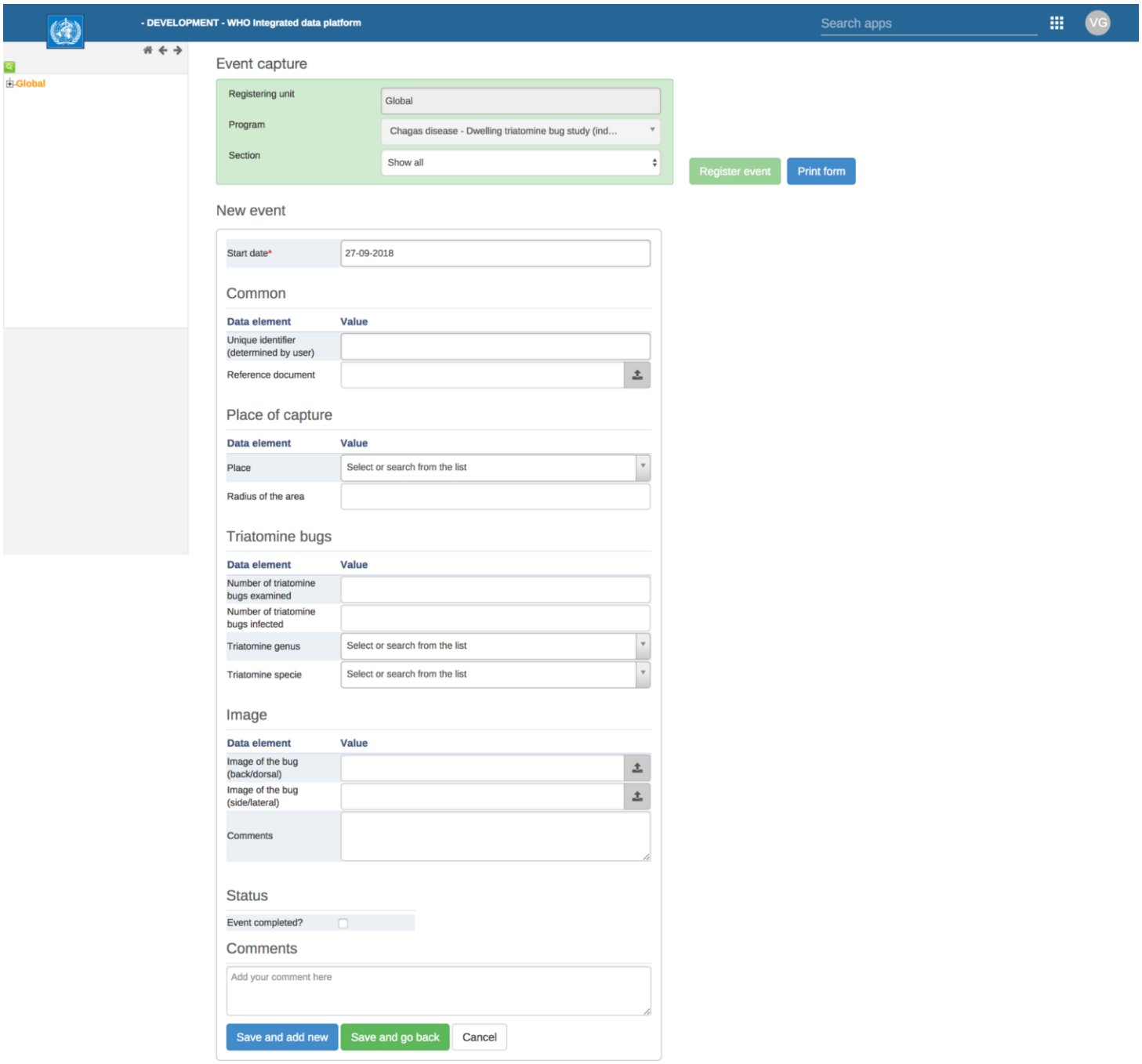
35. Jose-Norberto Mazón, Jens Lechtenbörger, Juan Trujillo: A survey on summarizability issues in multidimensional modeling. Data Knowl. Eng. 68(12): 1452-1469 (2009)

36. Chagas Ontology. http://www.essi.upc.edu/dtim/files/chagas-ontology.owl

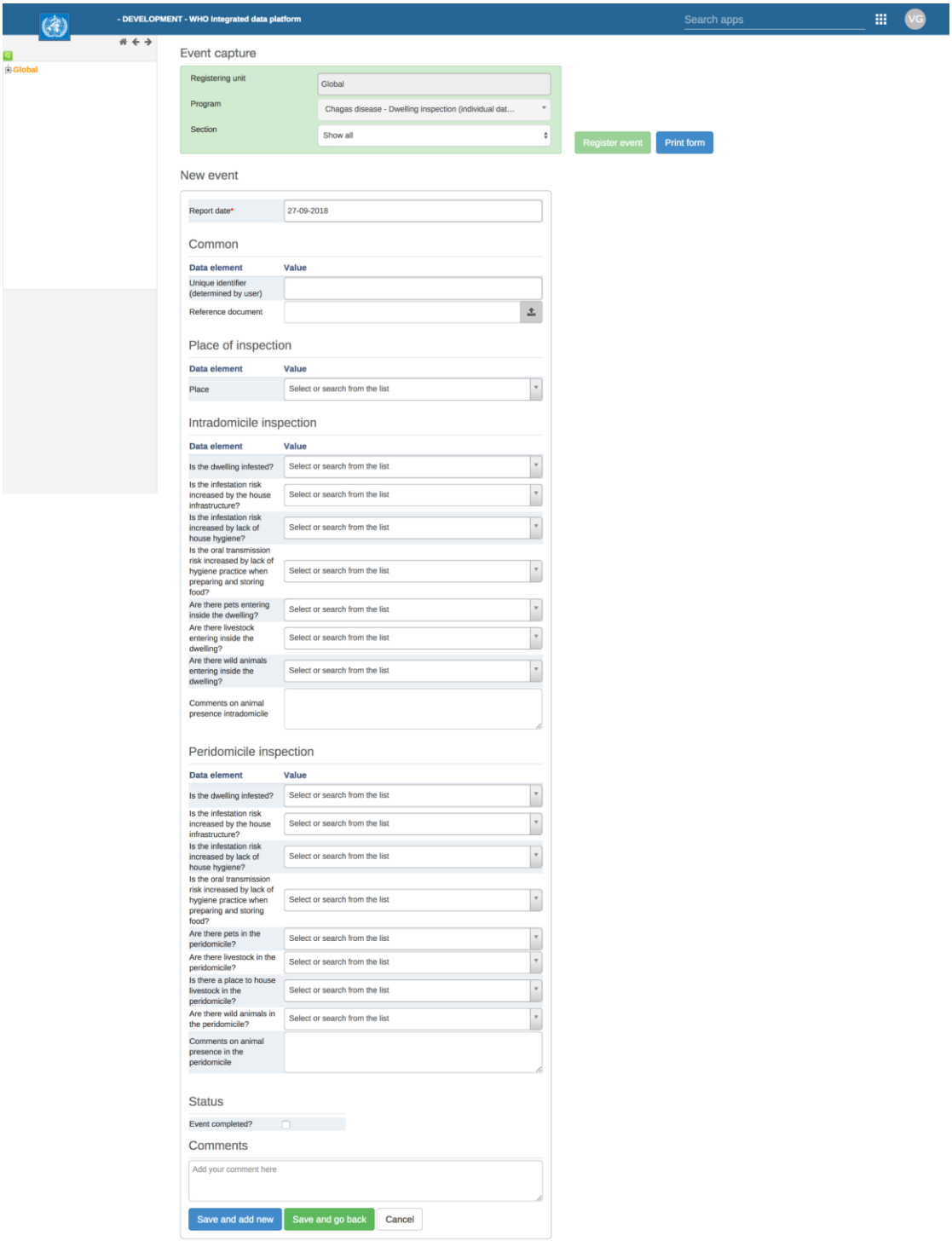
37. Stefan Gössner. Json Path – Xpath for JSON. Fachhochschule Dortmund, 2007. https://goessner.net/articles/JsonPath/ [Online, accessed on 15 October 2018]

Annexes

Annex 1. Data Collection Form in the WHO Integrated Data Platform for the Program "Dwelling Triatomine Bug Study"



Annex 2. Data Collection Form in the WHO Integrated Data Platform for the Program "Dwelling Inspection"



Annex 3. Google Maps API Example REST Request

Request:

```
https://maps.googleapis.com/maps/api/geocode/json?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&key=YOUR_API_KEY
```

Response:


```
{
  "results" : [
    {
      "address_components" : [
        {
          "long_name" : "1600",
          "short_name" : "1600",
          "types" : [ "street_number" ]
        },
        {
          "long_name" : "Amphitheatre Pkwy",
          "short_name" : "Amphitheatre Pkwy",
          "types" : [ "route" ]
        },
        {
          "long_name" : "Mountain View",
          "short_name" : "Mountain View",
          "types" : [ "locality", "political" ]
        },
        {
          "long_name" : "Santa Clara County",
          "short_name" : "Santa Clara County",
          "types" : [ "administrative_area_level_2", "political" ]
        },
        {
          "long_name" : "California",
          "short_name" : "CA",
          "types" : [ "administrative_area_level_1", "political" ]
        },
        {
```

```
          "long_name" : "United States",
          "short_name" : "US",
          "types" : [ "country", "political" ]
        },
        {
          "long_name" : "94043",
          "short_name" : "94043",
          "types" : [ "postal_code" ]
        }
      ],
      "formatted_address" : "1600 Amphitheatre Parkway, Mountain View, CA 94043, USA",
      "geometry" : {
        "location" : {
          "lat" : 37.4224764,
          "lng" : -122.0842499
        },
        "location_type" : "ROOFTOP",
        "viewport" : {
          "northeast" : {
            "lat" : 37.4238253802915,
            "lng" : -122.0829009197085
          },
          "southwest" : {
            "lat" : 37.4211274197085,
            "lng" : -122.0855988802915
          }
        }
      },
      "place_id" : "ChIJ2eUgeAK6j4ARbn5u_wAGqWA",
      "types" : [ "street_address" ]
    }
  ],
  "status" : "OK"
}
```

Annex 4. WHO Web API Example JSON Response

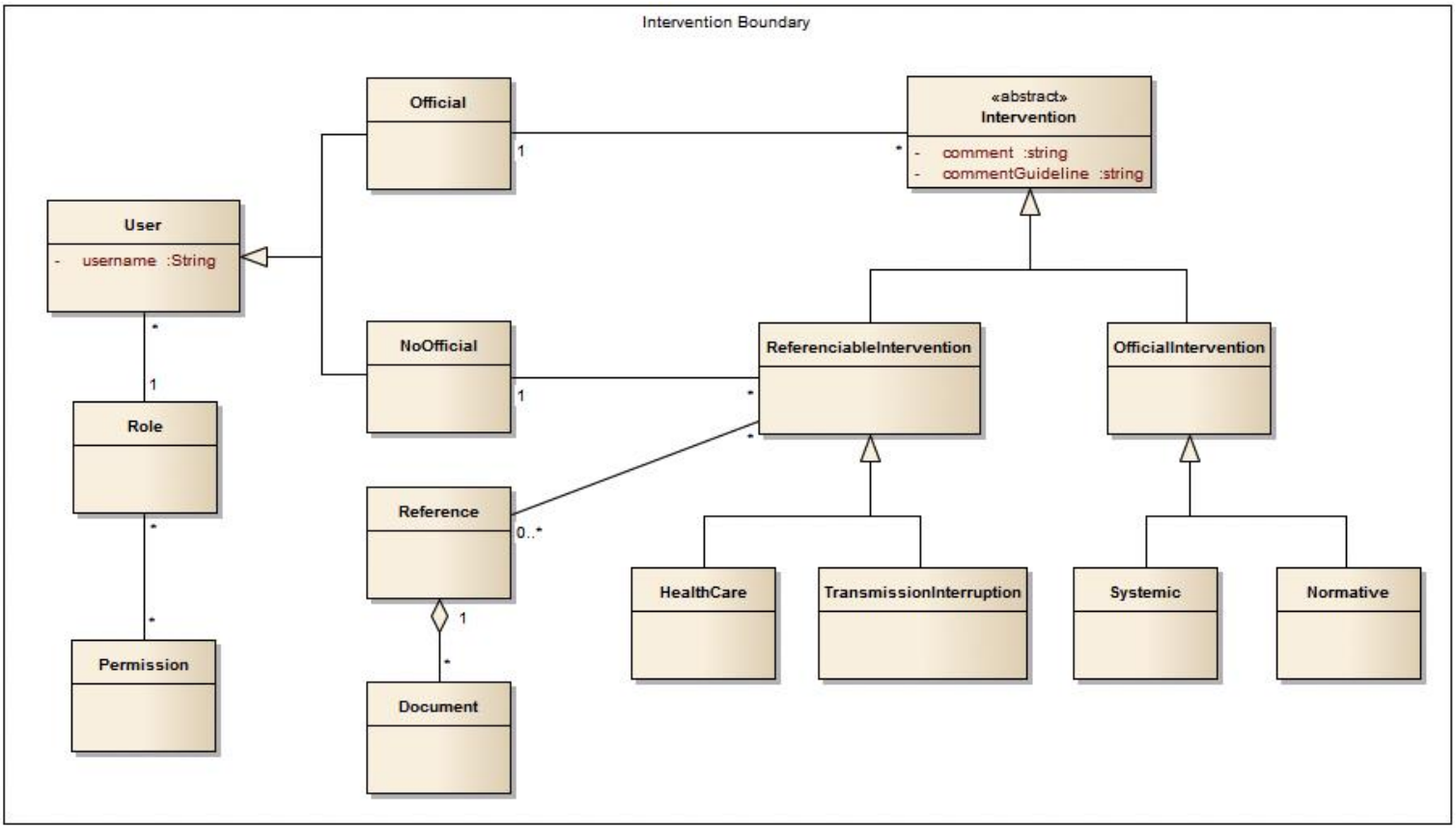

```
{
  "pager": {
    "page": 1,
    "pageCount": 1,
    "total": 0,
    "pageSize": 50
  },
  "events": [
    {
      "storedBy": "vgabante",
      "dueDate": "2018-06-14T19:11:16.745",
      "program": "k3pL2hPUqQi",
      "href": "http://who-dev.essi.upc.edu:8081/api/events/hVn67tnTXpk",
      "event": "hVn67tnTXpk",
      "programStage": "JIgwt1yACL6",
      "orgUnit": "H8RixfF8ugH",
      "status": "ACTIVE",
      "orgUnitName": "Global",
      "eventDate": "2018-06-13T00:00:00.000",
      "attributeCategoryOptions": "Y7fcspgsU43",
      "lastUpdated": "2018-06-14T19:11:16.745",
      "created": "2018-06-14T19:11:16.745",
      "deleted": false,
      "attributeOptionCombo": "Xr12mI7VPn3",
      "coordinate": {
        "latitude": 0,
        "longitude": 0
      },
      "dataValues": [
        {
          "lastUpdated": "2018-06-14T19:11:16.747",
          "storedBy": "vgabante",
          "created": "2018-06-14T19:11:16.747",
          "dataElement": "NVv20fxQPXn",
          "value": "Op_YesNo_Yes",
          "providedElsewhere": false
        },
        {
          "lastUpdated": "2018-06-14T19:11:16.747",
          "storedBy": "vgabante",
          "created": "2018-06-14T19:11:16.747",
```

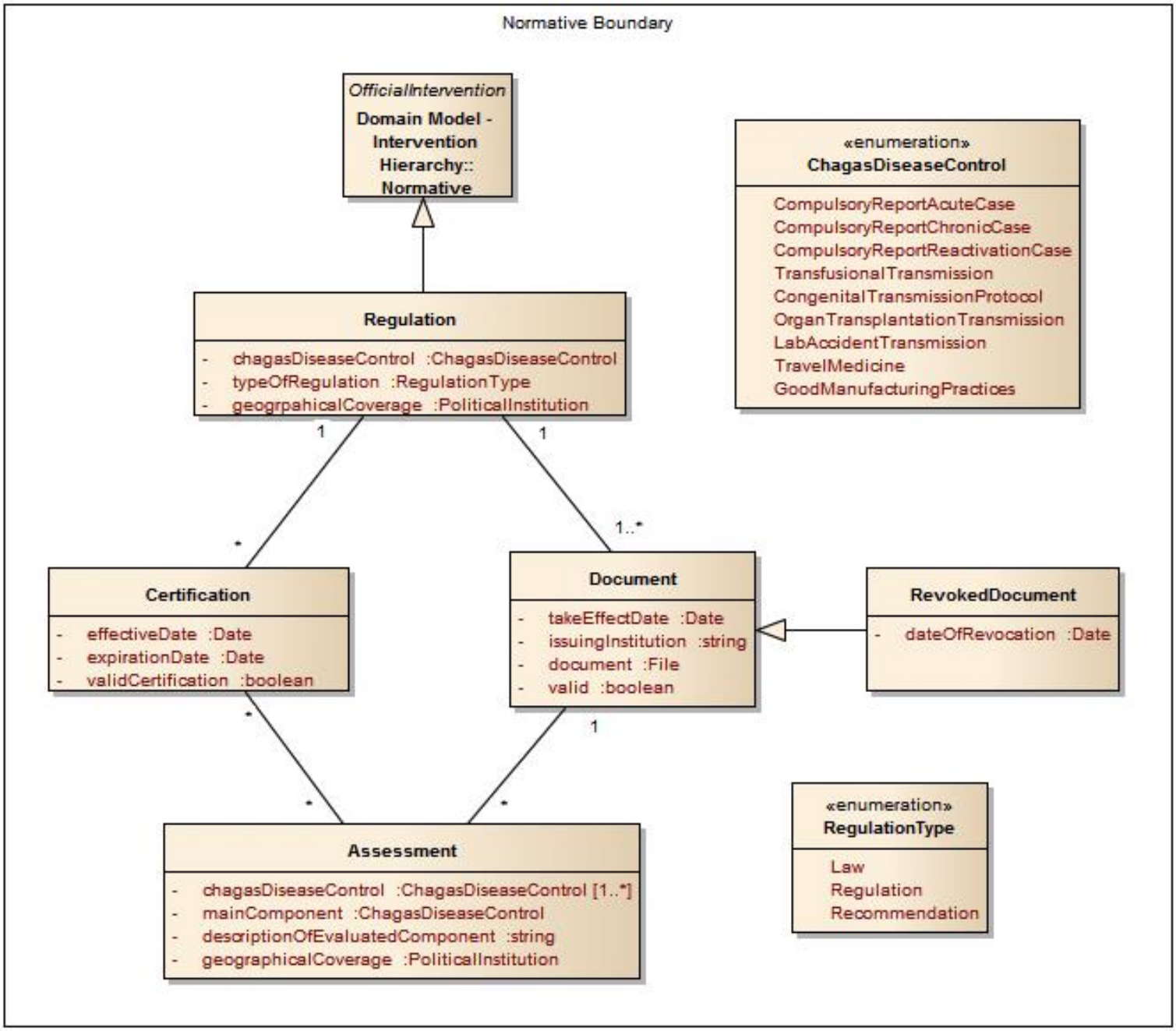
```
            "dataElement": "u1FwJXFMXXi",
            "value": "GPS_COORDINATES",
            "providedElsewhere": false
          },
          {
            "lastUpdated": "2018-06-14T19:11:16.747",
            "storedBy": "vgabante",
            "created": "2018-06-14T19:11:16.747",
            "dataElement": "ZkC7aKeDvgb",
            "value": "[-66.856458,-29.413454]",
            "providedElsewhere": false
          }
        ],
        "notes": []
      },
      {
        "storedBy": "vgabante",
        "dueDate": "2018-05-30T17:44:51.901",
        "program": "k3pL2hPUqQi",
        "href": "http://who-dev.essi.upc.edu:8081/api/events/w5sL6OjJdKK",
        "event": "w5sL6OjJdKK",
        "programStage": "JIgwt1yACL6",
        "orgUnit": "H8RixfF8ugH",
        "status": "ACTIVE",
        "orgUnitName": "Global",
        "eventDate": "2018-05-17T00:00:00.000",
        "attributeCategoryOptions": "Y7fcspgsU43",
        "lastUpdated": "2018-05-30T17:44:51.901",
        "created": "2018-05-30T17:44:51.901",
        "deleted": false,
        "attributeOptionCombo": "Xr12mI7VPn3",
        "coordinate": {
          "latitude": 0,
          "longitude": 0
        },
        "dataValues": [
          {
            "lastUpdated": "2018-05-30T17:44:51.903",
            "storedBy": "vgabante",
            "created": "2018-05-30T17:44:51.903",
            "dataElement": "irVFfZTKi9b",
            "value": "Op_YesNo_Yes",
            "providedElsewhere": false
          },
```

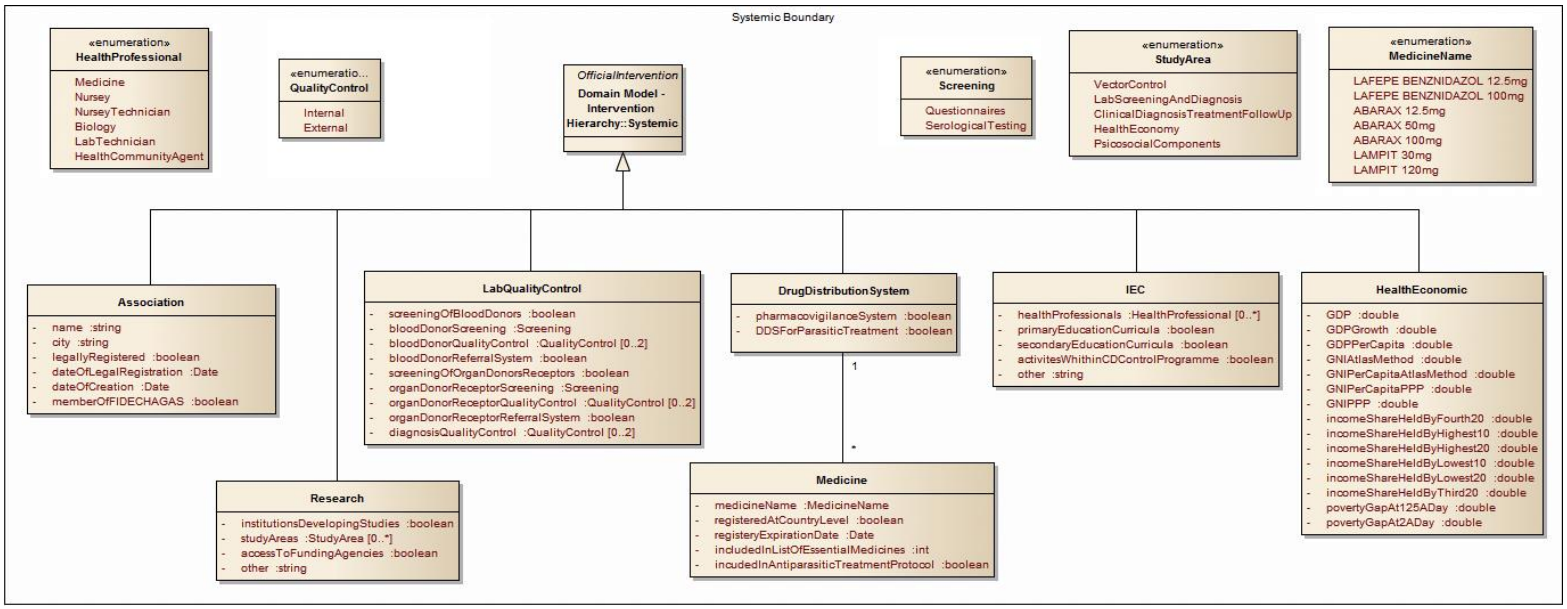
```
        {
          "lastUpdated": "2018-05-30T17:44:51.903",
          "storedBy": "vgabante",
          "created": "2018-05-30T17:44:51.903",
          "dataElement": "tElR7z25dJL",
          "value": "Op_YesNo_Yes",
          "providedElsewhere": false
        },
        {
          "lastUpdated": "2018-05-30T17:44:51.903",
          "storedBy": "vgabante",
          "created": "2018-05-30T17:44:51.903",
          "dataElement": "RdEJi3kXCRb",
          "value": "200",
          "providedElsewhere": false
        },
        {
          "lastUpdated": "2018-05-30T17:44:51.903",
          "storedBy": "vgabante",
          "created": "2018-05-30T17:44:51.903",
          "dataElement": "pTS0K7yFUcW",
          "value": "500",
          "providedElsewhere": false
        },
        {
          "lastUpdated": "2018-05-30T17:44:51.903",
          "storedBy": "vgabante",
          "created": "2018-05-30T17:44:51.903",
          "dataElement": "NVv20fxQPXn",
          "value": "Op_YesNo_Yes",
          "providedElsewhere": false
        },
        {
          "lastUpdated": "2018-05-30T17:44:51.903",
          "storedBy": "vgabante",
          "created": "2018-05-30T17:44:51.903",
          "dataElement": "jefpnPXSAtE",
          "value": "200",
          "providedElsewhere": false
        },
        {
          "lastUpdated": "2018-05-30T17:44:51.903",
          "storedBy": "vgabante",
          "created": "2018-05-30T17:44:51.903",
```

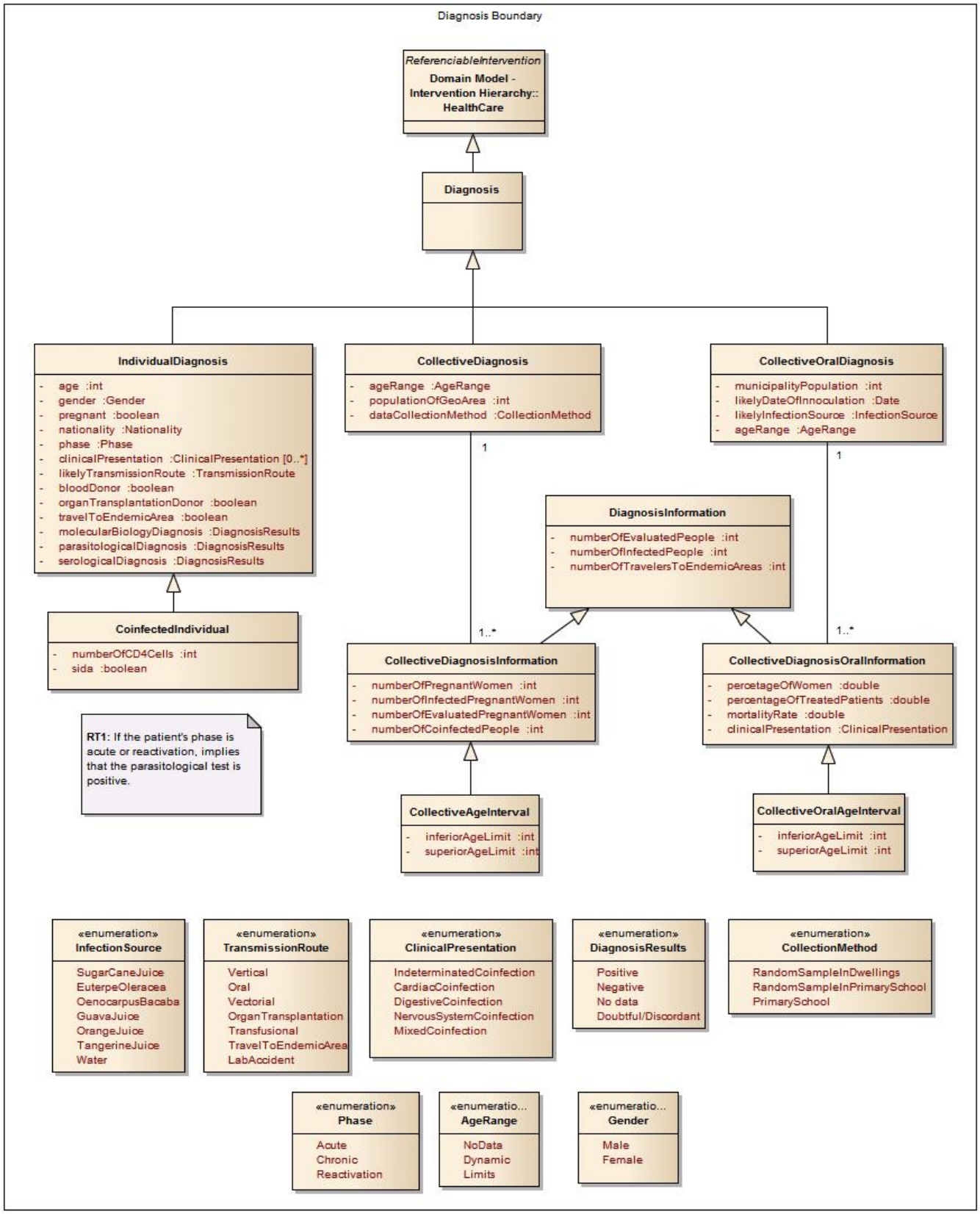
```
            "dataElement": "u1FwJXFMXXi",
            "value": "LOCALIZATION_DETAIL",
            "providedElsewhere": false
          },
          {
            "lastUpdated": "2018-05-30T17:44:51.903",
            "storedBy": "vgabante",
            "created": "2018-05-30T17:44:51.903",
            "dataElement": "JPhV30kNEFB",
            "value": "La Rioja",
            "providedElsewhere": false
          },
          {
            "lastUpdated": "2018-05-30T17:44:51.903",
            "storedBy": "vgabante",
            "created": "2018-05-30T17:44:51.903",
            "dataElement": "zD9zpgUcOt3",
            "value": "Argentina",
            "providedElsewhere": false
          }
        ],
        "notes": []
      }
    ]
}
```

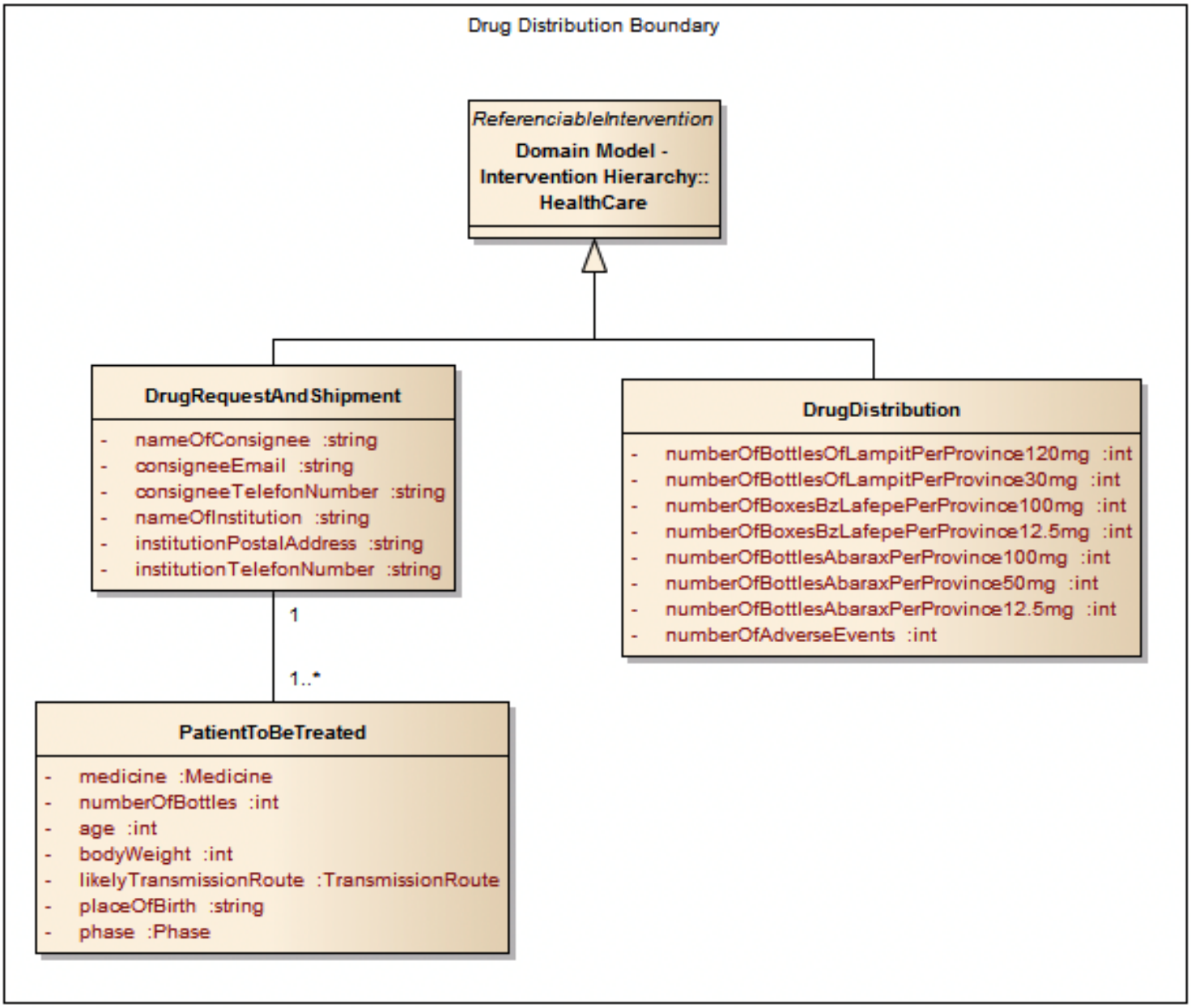
Annex 5. WHO Use case Vocabulary representation

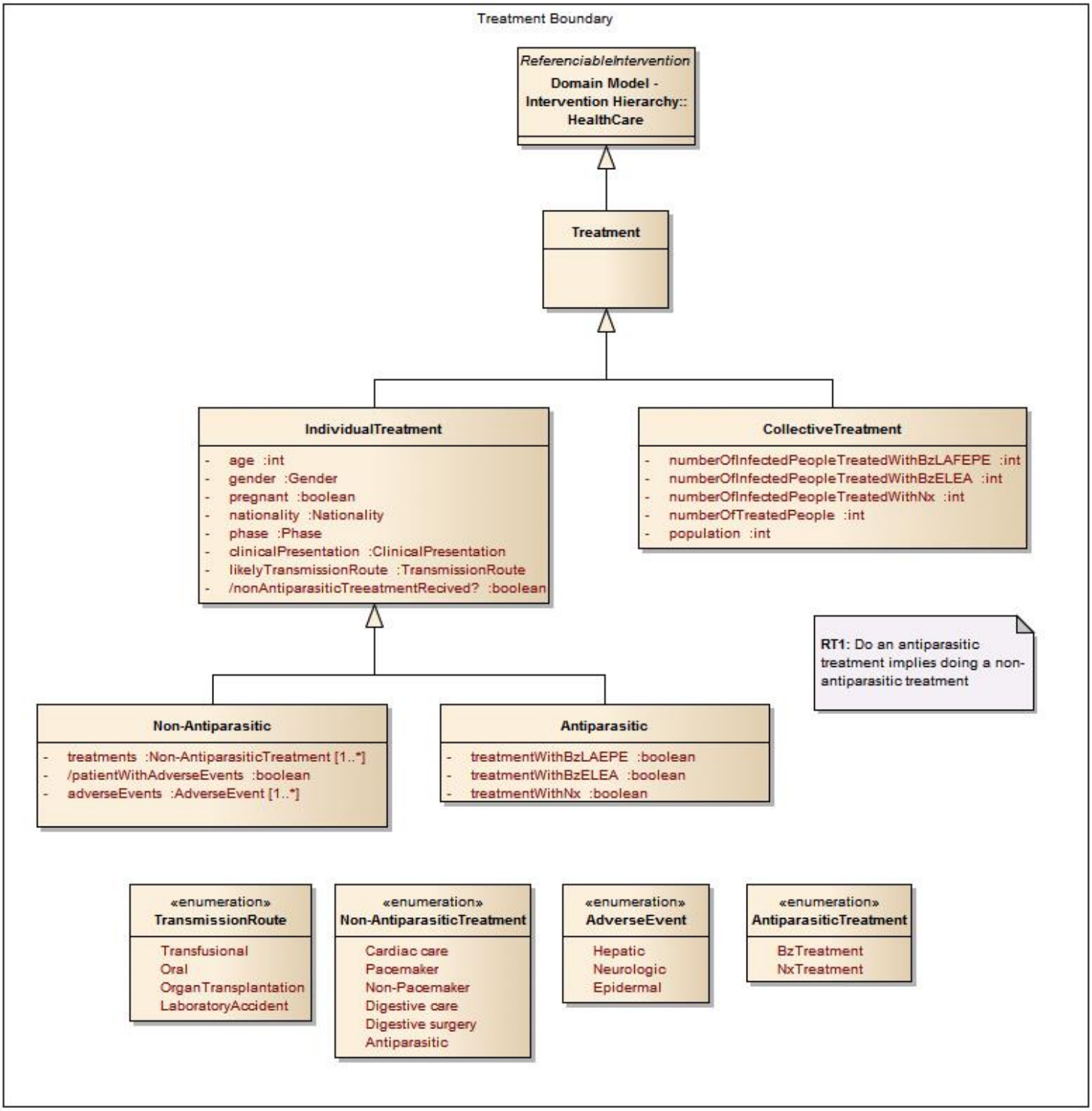


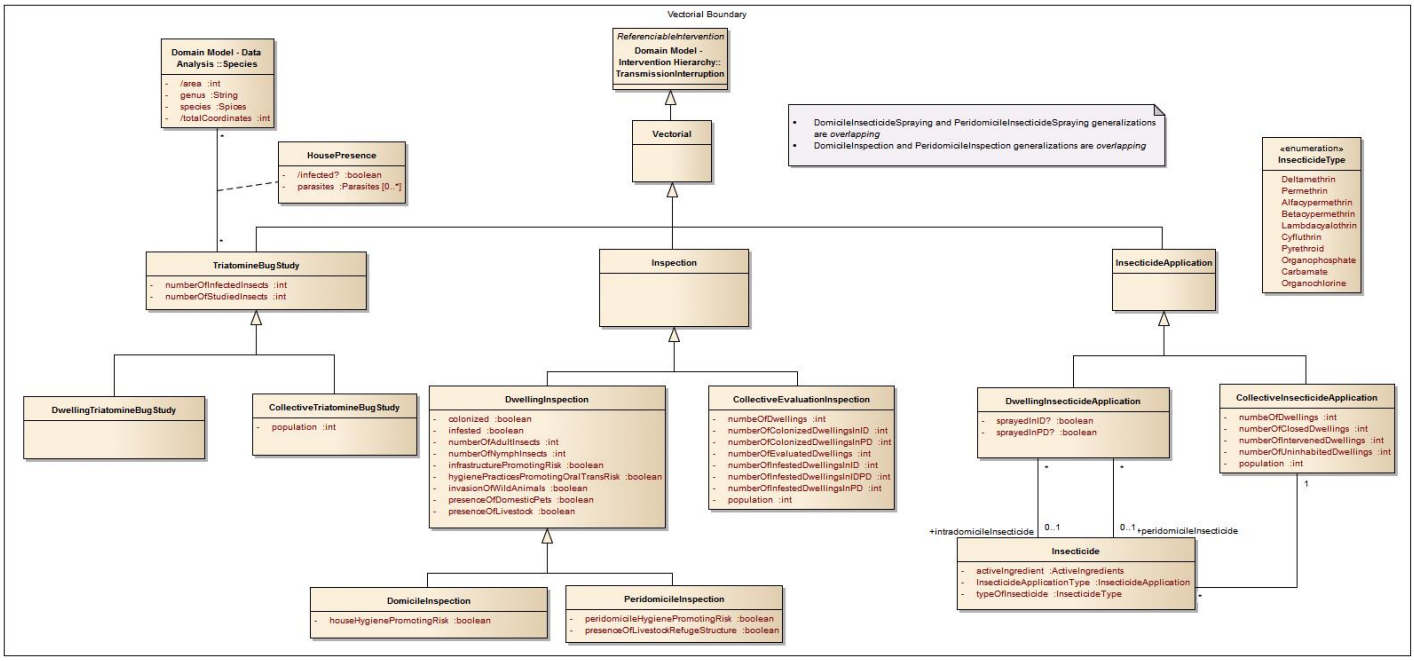

Normative Boundary
OfficialIntervention
Domain Model - Intervention Hierarchy:: Normative
«enumeration»
ChagasDiseaseControl
CompulsoryReportAcuteCase
CompulsoryReportChronicCase
CompulsoryReportReactivationCase
TransfusionalTransmission
CongenitalTransmissionProtocol
OrganTransplantationTransmission
LabAccidentTransmission
TravelMedicine
GoodManufacturingPractices
Regulation
- chagasDiseaseControl :ChagasDiseaseControl
- typeOfRegulation :RegulationType
- geogrpahicalCoverage :PoliticalInstitution
1
1
*
1..*
Certification
- effectiveDate :Date
- expirationDate :Date
- validCertification :boolean
Document
- takeEffectDate :Date
- issuingInstitution :string
- document :File
- valid :boolean
RevokedDocument
- dateOfRevocation :Date
*
1
*
*
Assessment
- chagasDiseaseControl :ChagasDiseaseControl [1..*]
- mainComponent :ChagasDiseaseControl
- descriptionOfEvaluatedComponent :string
- geographicalCoverage :PoliticalInstitution
«enumeration»
RegulationType
Law
Regulation
Recommendation

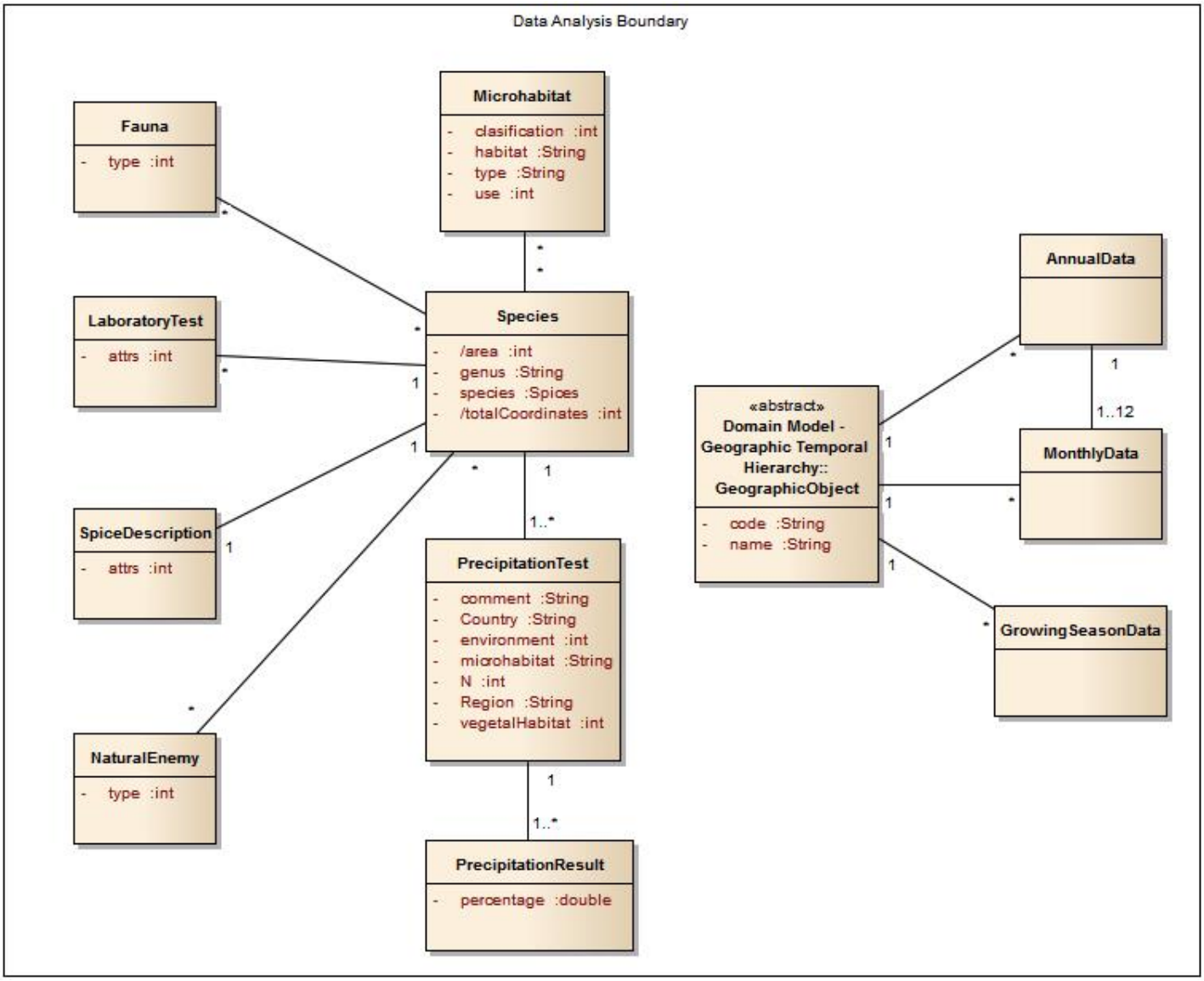

Systemic Boundary
«enumeration»
HealthProfessional
Medicine
Nursey
NurseyTechnician
Biology
LabTechnician
HealthCommunityAgent
«enumeratio...
QualityControl
Internal
External
OfficialIntervention
Domain Model - Intervention Hierarchy::Systemic
«enumeration»
Screening
Questionnaires
SerologicalTesting
«enumeration»
StudyArea
VectorControl
LabScreeningAndDiagnosis
ClinicalDiagnosisTreatmentFollowUp
HealthEconomy
PsicosocialComponents
«enumeration»
MedicineName
LAFEPE BENZNIDAZOL 12.5mg
LAFEPE BENZNIDAZOL 100mg
ABARAX 12.5mg
ABARAX 50mg
ABARAX 100mg
LAMPIT 30mg
LAMPIT 120mg
Association
- name :string
- city :string
- legallyRegistered :boolean
- dateOfLegalRegistration :Date
- dateOfCreation :Date
- memberOfFIDECHAGAS :boolean
LabQualityControl
- screeningOfBloodDonors :boolean
- bloodDonorScreening :Screening
- bloodDonorQualityControl :QualityControl [0..2]
- bloodDonorReferralSystem :boolean
- screeningOfOrganDonorsReceptors :boolean
- organDonorReceptorScreening :Screening
- organDonorReceptorQualityControl :QualityControl [0..2]
- organDonorReceptorReferralSystem :boolean
- diagnosisQualityControl :QualityControl [0..2]
DrugDistributionSystem
- pharmacovigilanceSystem :boolean
- DDSForParasiticTreatment :boolean
1
*
IEC
- healthProfessionals :HealthProfessional [0..*]
- primaryEducationCurricula :boolean
- secondaryEducationCurricula :boolean
- activitesWhithinCDControlProgramme :boolean
- other :string
HealthEconomic
- GDP :double
- GDPGrowth :double
- GDPPerCapita :double
- GNIAtlasMethod :double
- GNIPerCapitaAtlasMethod :double
- GNIPerCapitaPPP :double
- GNIPPP :double
- incomeShareHeldByFourth20 :double
- incomeShareHeldByHighest10 :double
- incomeShareHeldByHighest20 :double
- incomeShareHeldByLowest10 :double
- incomeShareHeldByLowest20 :double
- incomeShareHeldByThird20 :double
- povertyGapAt125ADay :double
- povertyGapAt2ADay :double
Research
- institutionsDevelopingStudies :boolean
- studyAreas :StudyArea [0..*]
- accessToFundingAgencies :boolean
- other :string
Medicine
- medicineName :MedicineName
- registeredAtCountryLevel :boolean
- registeryExpirationDate :Date
- includedInListOfEssentialMedicines :int
- incudedInAntiparasiticTreatmentProtocol :boolean


Diagnosis Boundary
ReferenciableIntervention
Domain Model - Intervention Hierarchy:: HealthCare
Diagnosis
IndividualDiagnosis
- age :int
- gender :Gender
- pregnant :boolean
- nationality :Nationality
- phase :Phase
- clinicalPresentation :ClinicalPresentation [0..*]
- likelyTransmissionRoute :TransmissionRoute
- bloodDonor :boolean
- organTransplantationDonor :boolean
- travelToEndemicArea :boolean
- molecularBiologyDiagnosis :DiagnosisResults
- parasitologicalDiagnosis :DiagnosisResults
- serologicalDiagnosis :DiagnosisResults
CollectiveDiagnosis
- ageRange :AgeRange
- populationOfGeoArea :int
- dataCollectionMethod :CollectionMethod
CollectiveOralDiagnosis
- municipalityPopulation :int
- likelyDateOfInnoculation :Date
- likelyInfectionSource :InfectionSource
- ageRange :AgeRange
DiagnosisInformation
- numberOfEvaluatedPeople :int
- numberOfInfectedPeople :int
- numberOfTravelersToEndemicAreas :int
CoinfectedIndividual
- numberOfCD4Cells :int
- sida :boolean
CollectiveDiagnosisInformation
- numberOfPregnantWomen :int
- numberOfInfectedPregnantWomen :int
- numberOfEvaluatedPregnantWomen :int
- numberOfCoinfectedPeople :int
CollectiveDiagnosisOralInformation
- percetageOfWomen :double
- percentageOfTreatedPatients :double
- mortalityRate :double
- clinicalPresentation :ClinicalPresentation
CollectiveAgeInterval
- inferiorAgeLimit :int
- superiorAgeLimit :int
CollectiveOralAgeInterval
- inferiorAgeLimit :int
- superiorAgeLimit :int
1
1..*
1
1..*
RT1: If the patient's phase is acute or reactivation, implies that the parasitological test is positive.
«enumeration» InfectionSource
SugarCaneJuice
EuterpeOleracea
OenocarpusBacaba
GuavaJuice
OrangeJuice
TangerineJuice
Water
«enumeration» TransmissionRoute
Vertical
Oral
Vectorial
OrganTransplantation
Transfusional
TravelToEndemicArea
LabAccident
«enumeration» ClinicalPresentation
IndeterminatedCoinfection
CardiacCoinfection
DigestiveCoinfection
NervousSystemCoinfection
MixedCoinfection
«enumeration» DiagnosisResults
Positive
Negative
No data
Doubtful/Discordant
«enumeration» CollectionMethod
RandomSampleInDwellings
RandomSampleInPrimarySchool
PrimarySchool
«enumeration» Phase
Acute
Chronic
Reactivation
«enumeratio... AgeRange
NoData
Dynamic
Limits
«enumeratio... Gender
Male
Female


Drug Distribution Boundary
ReferenciableIntervention
Domain Model - Intervention Hierarchy:: HealthCare
DrugRequestAndShipment
- nameOfConsignee :string
- consigneeEmail :string
- consigneeTelefonNumber :string
- nameOfInstitution :string
- institutionPostalAddress :string
- institutionTelefonNumber :string
DrugDistribution
- numberOfBottlesOfLampitPerProvince120mg :int
- numberOfBottlesOfLampitPerProvince30mg :int
- numberOfBoxesBzLafepePerProvince100mg :int
- numberOfBoxesBzLafepePerProvince12.5mg :int
- numberOfBottlesAbaraxPerProvince100mg :int
- numberOfBottlesAbaraxPerProvince50mg :int
- numberOfBottlesAbaraxPerProvince12.5mg :int
- numberOfAdverseEvents :int
1
1..*
PatientToBeTreated
- medicine :Medicine
- numberOfBottles :int
- age :int
- bodyWeight :int
- likelyTransmissionRoute :TransmissionRoute
- placeOfBirth :string
- phase :Phase


Treatment Boundary
ReferenciableIntervention
Domain Model - Intervention Hierarchy:: HealthCare
Treatment
IndividualTreatment
- age :int
- gender :Gender
- pregnant :boolean
- nationality :Nationality
- phase :Phase
- clinicalPresentation :ClinicalPresentation
- likelyTransmissionRoute :TransmissionRoute
- /nonAntiparasiticTreeatmentRecived? :boolean
CollectiveTreatment
- numberOfInfectedPeopleTreatedWithBzLAFEPE :int
- numberOfInfectedPeopleTreatedWithBzELEA :int
- numberOfInfectedPeopleTreatedWithNx :int
- numberOfTreatedPeople :int
- population :int
RT1: Do an antiparasitic treatment implies doing a non-antiparasitic treatment
Non-Antiparasitic
- treatments :Non-AntiparasiticTreatment [1..*]
- /patientWithAdverseEvents :boolean
- adverseEvents :AdverseEvent [1..*]
Antiparasitic
- treatmentWithBzLAEPE :boolean
- treatmentWithBzELEA :boolean
- treatmentWithNx :boolean
«enumeration» TransmissionRoute
Transfusional
Oral
OrganTransplantation
LaboratoryAccident
«enumeration» Non-AntiparasiticTreatment
Cardiac care
Pacemaker
Non-Pacemaker
Digestive care
Digestive surgery
Antiparasitic
«enumeration» AdverseEvent
Hepatic
Neurologic
Epidermal
«enumeration» AntiparasiticTreatment
BzTreatment
NxTreatment


Vectorial Boundary
Domain Model - Data Analysis ::Species
- /area :int
- genus :String
- species :Spices
- /totalCoordinates :int
ReferenciableIntervention
Domain Model - Intervention Hierarchy:: TransmissionInterruption
Vectorial
DomicileInsecticideSpraying and PeridomicileInsecticideSpraying generalizations are overlapping
DomicileInspection and PeridomicileInspection generalizations are overlapping
«enumeration» InsecticideType
Deltamethrin
Permethrin
Alfacypermethrin
Betacypermethrin
Lambdacyalothrin
Cyfluthrin
Pyrethroid
Organophosphate
Carbamate
Organochlorine
HousePresence
- /infected? :boolean
- parasites :Parasites [0..*]
TriatomineBugStudy
- numberOfInfectedInsects :int
- numberOfStudiedInsects :int
Inspection
InsecticideApplication
DwellingTriatomineBugStudy
CollectiveTriatomineBugStudy
- population :int
DwellingInspection
- colonized :boolean
- infested :boolean
- numberOfAdultInsects :int
- numberOfNymphInsects :int
- infrastructurePromotingRisk :boolean
- hygienePracticesPromotingOralTransRisk :boolean
- invasionOfWildAnimals :boolean
- presenceOfDomesticPets :boolean
- presenceOfLivestock :boolean
CollectiveEvaluationInspection
- numbeOfDwellings :int
- numberOfColonizedDwellingsInID :int
- numberOfColonizedDwellingsInPD :int
- numberOfEvaluatedDwellings :int
- numberOfInfestedDwellingsInID :int
- numberOfInfestedDwellingsInIDPD :int
- numberOfInfestedDwellingsInPD :int
- population :int
DwellingInsecticideApplication
- sprayedInID? :boolean
- sprayedInPD? :boolean
CollectiveInsecticideApplication
- numbeOfDwellings :int
- numberOfClosedDwellings :int
- numberOfInterveneDwellings :int
- numberOfUninhabitedDwellings :int
- population :int
DomicileInspection
- houseHygienePromotingRisk :boolean
PeridomicileInspection
- peridomicileHygienePromotingRisk :boolean
- presenceOfLivestockRefugeStructure :boolean
+intradomicileInsecticide
0..1
0..1
+peridomicileInsecticide
Insecticide
- activeIngredient :ActiveIngredients
- InsecticideApplicationType :InsecticideApplication
- typeOfInsecticide :InsecticideType
1


Data Analysis Boundary
Fauna
- type :int
Microhabitat
- clasification :int
- habitat :String
- type :String
- use :int
AnnualData
LaboratoryTest
- attrs :int
Species
- /area :int
- genus :String
- species :Spices
- /totalCoordinates :int
«abstract»
Domain Model -
Geographic Temporal
Hierarchy::
GeographicObject
- code :String
- name :String
MonthlyData
SpiceDescription
- attrs :int
PrecipitationTest
- comment :String
- Country :String
- environment :int
- microhabitat :String
- N :int
- Region :String
- vegetalHabitat :int
GrowingSeasonData
NaturalEnemy
- type :int
PrecipitationResult
- percentage :double