SETA, a suite-independent agile analytical framework

Master Thesis
Master In Innovation and Research In Informatics (MIRI)
Data Mining and Business Intelligence specialization

Master student: Pedro Javier González Alonso
Supervisor: Alberto Abelló Gamazo
Co-supervisor: Oscar Romero Moral

Facultat d’Informàtica de Barcelona
Universitat Politècnica de Catalunya, BarcelonaTech
Barcelona
October 20, 2016
A mis padres, 

doctores en perseverancia 

carino y bondad
Abstract

Nowadays, business analytical users need agile processes spanning from the selection of relevant data from raw data sources to the generation of data structures prepared to serve as input for OLAP, Data Mining and/or other analytical tools. However, the wide range of analytical needs and the increasingly need of adaptive Business strategies discourages the use of the 'All-In-One' existing suites (i.e., end-to-end Solutions from a single vendor). Oppositely, an agile approach suite-independent is advisable to boost user’s independence from a specific vendor and the analytical capabilities enabled by combining several suites / tools according to the user’s needs. In this thesis we present and develop 'SETA', a suite-independent agile analytical framework by proposing a novel approach combining rich metadata definition and automation components. As proof of validity, we instantiate the developed framework in a real-world project for the WHO Chagas Programme.

This thesis introduces two main contributions. First, an approach to store and integrate a set of heterogeneous data sources into a flexible data store in some intermediate point between the classical Data Warehouse (DW) approaches and the recent Data Lake strategies. We argue that classical DW systems are too rigid to accommodate agile analytical pipelines, whereas Data Lakes and Big Data technologies are not suitable to much of today’s organizations. Thus, a novel approach combining both approaches is presented. Second, a rich definitional system to represent 1) the data components at Source, Global Schema and Domain levels, 2) the data mappings between this levels and 3) the final user analytical requirements. This definitional system provides a flexible view of the data schema at different levels and habilitates the automation of the target data schemas and the ETL to feed them.
Acknowledgments

En primer lugar agradecerles a mi mujer Nuria y a mis hijas Sara y Carmen su comprensión, por un tiempo que siendo suyo, no les he podido dedicar durante el desarrollo de esta tesis y por su paciencia en tantas ocasiones en las que mi mente exploraba formalizaciones de ontologías mientras me hablaban.

Quiero manifestar mi doble agradecimiento a los profesores Alberto Abelló y Oscar Romero. Me dieron la oportunidad de participar en el proyecto WISCC de la OMS y me dedicaron su tiempo y su paciencia para poder hacer de esa experiencia un tema para el desarrollo de mi tesis, a veces en sus despachos, a veces en alguna cervecería en Oslo o Ginebra. Muchas gracias por vuestra cercanía.

A Lise Grout y Pere Albajar del Programme on Control of Chagas disease de la WHO excelentes profesionales y mejores personas. Gracias por vuestra simpatía y hospitalidad.

A todos los compañeros del IT4BI. Siempre me han ofrecido su ayuda cuando la he necesitado. Y siempre tenían localizado el paper correcto!

A mis amigos Jose, Juanma, Chus, Jordi y Sergio. Santas partidas de poker, papel y tertulias posteriores! ¿Cuanto me habré ahorrado en psicólogos en los momentos de stress gracias a vosotros?. También a Perdi, Miquel, Jose Luis y Xavi, siempre organizando nuevas actividades, nuevos retos y nuevas formas de rompernos la crisma por tierra mar o aire. A mi amiga Lucía, una mujer tranquila que siempre tenía las palabras adecuadas.

Finalmente, volver a la universidad después de tantos años para mi no era una decisión fácil y entrañaba sus riesgos. Mi reconocimiento por su labor a los profesores que he tenido durante el MIRI, han hecho que sienta que realmente ha valido la pena.
List of Figures

1.1 The scope of SETA .................................................. 2
1.2 The 'Data LakeHouse' intuition ................................. 4
1.3 Competitive pressure. The Five Forces Model ................. 6
1.4 SETA pillars methodological derivation ........................ 11
1.5 Pillars and Objectives alignment ............................... 12
1.6 SETA user interactions ........................................... 13

2.1 Convergence of automation related works ....................... 29
2.2 MDA general framework .......................................... 30
2.3 MDA framework for DW design ................................. 31

3.1 SETA Overview. The four structural layers ..................... 36

4.1 SETA Definition Layer details ................................... 42
4.2 A dumb data source for examples ............................... 49
4.3 Dumb data source definition example I .......................... 50
4.4 Dumb data source definition example II ........................ 51
4.5 Source-Global Schema mappings example I ..................... 52
4.6 Source-Global Schema mappings example II ..................... 53
4.7 Global Schema-Domain Schema mappings example I ............ 54
4.8 SETA Requirement Elicitor Component .......................... 55
4.9 Quarry: System overview ......................................... 57
4.10 Source monitor and manager. An example of dashboard ........ 59

5.1 WISCC context diagram .......................................... 62
5.2 WISCC data capture subsystem information structure .......... 64
5.3 DHIS2 metadata extractor ......................................... 65
5.4 WISCC extractor and wrapper ..................................... 67
List of Tables

3.1 Comparison of the three different approaches. .......................... 34

4.1 Main SETA $TBOX^*$ classes. ........................................ 44
4.2 Main SETA $TBOX^*$ properties. ................................. 45
4.3 Source Definition $TBOX^*$ annotation properties. .............. 46
Contents

Abstract

Acknowledgments

List of figures

List of tables

1 Introduction

1.1 Contributions

1.2 The current analytical environment in organizations

1.3 Objectives of the proposed framework

1.4 Pillars of the SETA framework

1.4.1 User-centric approach

1.4.2 Automation

1.4.3 Model Driven Architecture (MDA)

1.4.4 Metadata and semantic aware definitions

2 Related work

2.1 User-centric approach

2.1.1 Conclusions

2.2 Automation

2.2.1 Multidimensional design automation

2.2.2 ETL Automation

2.2.3 Conclusions

2.3 Model-Driven Architectures (MDA)

2.3.1 Conclusions

3 SETA overview

3.1 Data warehouses, data lakes and data LakeHouses

3.2 SETA architecture

3.2.1 Definition layer
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.2</td>
<td>Automation layer</td>
<td>39</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Processing Layer</td>
<td>40</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Data Layer</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>SETA framework details</td>
<td>41</td>
</tr>
<tr>
<td>4.1</td>
<td>Definition Layer formalization</td>
<td>41</td>
</tr>
<tr>
<td>4.1.1</td>
<td>SETA Ontology classes and properties</td>
<td>41</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Mappings</td>
<td>47</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Definition layer examples</td>
<td>49</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Analytical requirements</td>
<td>54</td>
</tr>
<tr>
<td>4.2</td>
<td>Automation layer</td>
<td>56</td>
</tr>
<tr>
<td>4.3</td>
<td>Processing Layer</td>
<td>59</td>
</tr>
<tr>
<td>4.4</td>
<td>Data Layer</td>
<td>60</td>
</tr>
<tr>
<td>5</td>
<td>SETA Instantiation</td>
<td>61</td>
</tr>
<tr>
<td>5.1</td>
<td>The WISCC project</td>
<td>61</td>
</tr>
<tr>
<td>5.2</td>
<td>The Definition Layer</td>
<td>65</td>
</tr>
<tr>
<td>5.3</td>
<td>Extractor and wrapper</td>
<td>66</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions</td>
<td>69</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In the data analysis context, the need to produce valuable information and actionable knowledge from raw data is as old as the existence of data. This need has been the driving force of the evolution and development of computer solutions for obtain, transform, load, store, retrieve, visualize and analyze data. Provide the right information, at the right time, to the right people with an appropriate cost, is critical to take the right decisions in a given domain. This is precisely the ultimate goal of a data warehouse system (DW). A DW provides an integrated view of information from different data sources and facilitates the provision of information (desirably) by following the previous premises. Despite the enormous success of DW systems in the organizations, the implantation, from design to implementation of such a systems involves enormous difficulties. From a classical perspective, the design of a DW system as a whole, means reconciling the analytical needs of an organization with the available data sources, organizing the data under a given conceptual schema, usually the multidimensional, which is based on a logical, usually the relational schema. Nevertheless, in the current context in which organizations operate, the classical approach of data warehousing systems is in many cases unfeasible to meet the general premises of right information at the right time and at the right cost.
This work introduces SETA, an agile analytical framework that proposes an approach for DW systems that combines the robust classical approach to DW with new components and techniques recently developed in this area to meet the challenges of the current analytical environment in organizations. This thesis introduces the term Data LakeHouse. Intuitively, a Data LakeHouse is a solution for the analytical framework in the middle point of the spectrum of solutions between classical DW systems and Data Lakes.

![Figure 1.1: The scope of SETA in the context of the complete analytical pipeline. The base figure has been drawn from the UPC-MIRI Open Data course slides, 2015 [11].](image)

For a first glimpse of SETA and to place it in the context of the complete analytical pipeline, the figure 1.1 graphically illustrates its scope. SETA is a flexible system to generate information to OLAP tools, data mining tools and reporting tools, based on the analytical requirements posed by users in a simple
and intuitive manner using an ontology to modelize the conceptual schema of a given domain. SETA incorporates in its data layer, information from different sources, but unlike classic DW systems, it does not predefine a static logical schema on which the data is materialized to offer a 'fully integrated' view of the sources. SETA combines the 'Schema on read' paradigm from data lakes with the 'Schema on write' paradigm from classic DW systems. Finally, SETA is not intended as a front-end for data analysis (OLAP, data mining, reporting) but to generate information to feed such systems. All this concepts and ideas will be developed in depth along this document.

The proposed framework is based on, what in the context of this work are called pillars, which determines the system design guidelines. These pillars are: 1) user-centric approach, 2) Automation and 3) Model Driven Architectures (MDA). These pillars are supported by another one that is transversal: metadata and semantic aware definitions. As proof of validity, this work presents the instantiation of this framework in a real-world project for the World Health Organization (WHO), The World Information System on Chagas Control (WISCC) [30].

This first chapter includes the contents needed for 1) Listing the main contributions of this work, 2) Characterizing the current analytical environment in which organizations operate, 3) Formulating and justifying the objectives of the system and 4) Enumerating and justifying the pillars. Chapter 2 addresses the state of the art and the relevant research works associated with each of the pillars. Chapter 3 gives an overview of the proposed framework in the context of classical DW systems and Data Lakes to gain both, perspective and better understanding of the fundamentals of the proposed system. Chapter 4 goes into the details of the proposed solution. Chapter 5 is dedicated to the application of the framework in the WHO-WISCC project. Finally, conclusions will be summarized in Chapter 6.

1.1 Contributions

We understand that the processes of Technology Transfer from research centers to markets, play a key role in innovation strategies and that both research centers, companies and even society in general, can obtain clear benefits from them. We believe that the work developed in this thesis provides contributions at both academic and practical level, following the principles of Open Innovation\(^1\) as innovation strategy to move forward with R&D. The practical thinking is one the main

---

drivers of this work, but it also tries to address the exposition of the challenges and proposed solutions with the formalism needed for this type of exercises.

To summarize, the main contributions of this work are listed below:

<table>
<thead>
<tr>
<th>Classical Data Warehouse</th>
<th>Data LakeHouse</th>
<th>Data Lake</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema on write</strong></td>
<td><strong>Schema on write</strong></td>
<td><strong>Schema on read</strong></td>
</tr>
<tr>
<td>Data sources</td>
<td>Data sources</td>
<td>Data sources</td>
</tr>
<tr>
<td><strong>ETL</strong></td>
<td><strong>Wrap</strong></td>
<td><strong>void</strong></td>
</tr>
<tr>
<td><strong>OLAP Cubes</strong></td>
<td><strong>OLAP Cubes</strong></td>
<td><strong>OLAP Cubes</strong></td>
</tr>
<tr>
<td><strong>Data matrices</strong></td>
<td><strong>Data matrices</strong></td>
<td><strong>Data matrices</strong></td>
</tr>
<tr>
<td><strong>Data files</strong></td>
<td><strong>Data files</strong></td>
<td><strong>Data files</strong></td>
</tr>
<tr>
<td><em>Static relational schema</em></td>
<td><em>Flexible schema</em></td>
<td><em>No schema</em></td>
</tr>
</tbody>
</table>

Figure 1.2: ‘Data LakeHouse’ intuition. It combines the ‘schema on Read’ approach from Data Lakes with the ‘Schema on Write’ approach from classical DW in order to simplify both, the ETL and the modelization processes.

1. Definition and refinement of the properties and characteristics of the concept *Data LakeHouse* (DLW) as an intermediate point on the spectrum between classic Data Warehouse systems and systems fully based on the paradigm *Load First/Model Later*. Although the approach *Schema on read* is not new, it is gaining momentum in recent years as it is one of the paradigms that are usually applied in the Big Data arena. Traditionally, DW systems have been designed around the opposite principle, *Schema on write*. This approach requires a pre-defined data model to be implemented as a set of tables. The
data being loaded is then mapped to the existing data model represented by the tables in the database. If the data the user wants to load into the warehouse does not match the existing tables, changes must be made, such as to the ETL process or the table structure. These changes can be expensive and time consuming, especially for domains that deal with complex data or work in a highly dynamic environment. Therefore, an evolution of the classic definition of DW is proposed maintaining its basic structural levels, but adopting at the classical Integration or Global Schema layer the ‘Schema on Read’ approach. Thus, the relational logical scheme ceases to be the reference approach for the construction of the Integration layer and gives way to the use of more flexible models adapted to the structure of the data sources that need to be integrated. In this implementation, the integration level is based on semi-structured schemes, typically implemented through document-store databases. But in the same way that a Data Lake (DL) stores data with different format and schema, the Data LakeHouse uses different physical database engines according the nature of the source datasets. For example, for structured and non changing schema data sources a good option could be a relational database, but for semi-structured and highly changing schema data sources, provably the best choice would be a document-store database. Figure 1.2 shows a graphical intuition about this idea.

2. Definition and development of the metamodel that establishes the concepts implemented by SETA and on which is instantiated the model of a given project / domain. SETA applies the principle of conceptually specifying the domain and the data layers by using ontologies, and defines the artifacts (classes and properties) of the metamodel through three levels of abstraction and the connections (mappings) between these levels: 1) Conceptual Business environment, 2) Conceptual data integration environment and 3) Conceptual data sources environment.

3. Modification and integration of Quarry into the SETA framework. Quarry is a tool developed during last 4 years in the UPC that provides part of the automation functions of multidimensional design and ETL automation that SETA required, which is one of the more complex aspects of the framework. Thus, a multiple objective is achieved 1) Increase the value of the solutions already developed in previous research projects, through its practical application, 2) Reduce costs and development time of SETA and 3) Improve and evolve the Quarry solution solutions for widespread use in real projects.

4. Instantiation of the proposed framework in a real-world project, the WISCC-WHO [30] project.
1.2 The current analytical environment in organizations

Understanding the needs of the current analytical environment requires properly interpret the environment in which organizations operate. Almost all of the articles that address the field of Business Intelligence (BI) agree highlighting the profound changes that the Internet revolution, the social networks, and ultimately, the production of data, his availability and his ubiquity are causing on the analytical needs of the organizations. Proof of this, is the unprecedented evolution of storage technologies, data processing and data analysis over the past 20 years. Below are listed some of the characteristics that determine the environment in which organizations operate today.

![Five Forces Model](image)

Figure 1.3: The Five Forces Model from Michael Porter [21].

1. **Competitive pressure.** One of the best tools available to analyze the environment in which organizations operate is the *Five Force Model* from
Michael Porter [21]. This model systematically establishes an external analysis of the immediate environment where a set of organizations operate (a particular market or industry) by characterizing five main forces as shown in Figure 1.3. These forces determine the competitive pressure to which the market is submitted to, and determine its potential profitability. The greater the pressure, the lower the potential profitability. It is true that the model is not free from criticism, but its simplicity and generality make it to remain widely used. It is also true that many of the criticisms come from its application to a particular company, which is a wrong practice because this model should be applied to determine the competitive structure of an entire sector.

What is relevant to the conclusions that are to be drawn is that as 1) customers and suppliers have greater power (internet, global e-commerce, global supply chains), 2) threats of new entrants is increasing (because business strategies as market crossing or due to the reduction of old entry barriers based on knowledge or technology) and 3) threats due to the growing availability of substitutive products and services, the competitive pressure in most of the markets is increased and their potential profitability is reduced. This reduction encourages organizations to take action either through differentiation strategies or through innovations that modify the structure of the sector in which they compete in their favor.

2. **Market clockspeed.** One consequence of the previous point is that the market dynamics increasingly accelerates. The release cycle of products and services is growing fast and competitive advantages are no longer sustainable. The only sustainable advantage is the superior ability to continuously evaluate sectoral and technological dynamics, the development of capacities to take advantage of current opportunities and anticipate future ones. This is where the capabilities to extract useful knowledge from the vast information available become a key element.

3. **Flexibility and agility.** According to the above, flexibility and agility become necessary qualities for the overall organization in general and for information analysis in particular. In many of today’s organizations the flexibility and agility paradigm is emerging as a necessary condition for its success.

4. **Uncertainty.** Organizations must be flexible and agile in a competitive and rapid cycling environment, but all this determines that uncertainty about its evolution increases. In the field of information analysis, this fact, coupled
with the vast amount of data available implies that the current information needs are often vague and future ones unknown. The base design of the analytical system can no longer incorporate from the time of its initial definition the two main elements that determine it: The requirements and data sources.

In short, all this leads to the characterization of the following properties of the information analysis environment:

- Future analytical requirements can not be predicted with precision at the design time of the information system that should support them.
- The same applies to data sources that the system should incorporate.
- Business users can not be subject to lengthy and costly development processes of the IT department when incorporating new analytical requirements or new sources of information.
- The organization should not have to give up the potential benefits of new technologies and tools for data analysis simply because their reference tools do not allow to take advantage from these benefits.

Therefore, the basis of the analytical system should accommodate such features. Being the DW system the cornerstone of the analytical system, it is necessary to ask whether if its classical definition, despite its enormous success, remains valid or not. In the aim of this work, the answer to this question is assumed to be negative. Therefore, the solution presented in this framework proposes a set of evolutions and different approaches with respect the classical DW framework which are better aligned with the current reality of organizations.

### 1.3 Objectives of the proposed framework

The framework proposed in this paper aims to be a support platform for the analytical pipeline in an environment that has been characterized in the previous section. Therefore, the objectives must be aligned with the analytical needs to be covered in such an environment. These general objectives are formulated as follows:

- **O1**: Maximal independence of the business users from the IT department development processes. Once the system is delivered to the
final user, he/she can not rely completely on the IT department for its operation. In classic DW systems, such reliance is placed sometimes in the ability of the end user to express SQL queries or to have multidimensional algebra skills, which often, is not admissible. The system must interact with the user based on domain concepts and based on analytical requirements that can be intuitively expressed by the user. This objective poses the need to build an advanced GUI that is able to interact with the user in these terms. This objective also means that the user must be able to incorporate new definitions or changes in the domain model.

• O2: Flexibility and agility to incorporate new analytical requirements. In the described environment, time-evolving analytical needs and new needs are assumed to be the norm rather than the exception. The system must be able to incorporate to some extent, this evolution, without adding costly redesign and/or development processes involving the IT resources.

• O3: Flexibility and agility to incorporate new data sources and accommodate their changes. In the same sense as in the previous point, the evolution in data sources initially modelled into the system and the emergence of new data sources relevant to the user are assumed to be frequent and therefore the system must be flexible enough to accommodate these changes nimbly.

• O4: Maximal independence with respect existing BI suites and technologies. The proposed framework, aims a clear separation between the end-user tools for data analysis and the system to generate the information that feeds these tools based on the analytical requirements of the user. OLAP tools, reporting tools or data mining tools used by the business analysts, should not condition decisively the operation of the framework. Just as the rapidly changing environment affects the behaviour of organizations, also it does to BI and data mining solution vendors. From our point of view, betting the entire analytical pipeline to a single 'All-In-One' suite, involves dependence and business continuity risks that in many cases are not assumable.

We recognize that expressions like 'Flexibility enough' and 'As independent as possible' and so on, can be ambiguous and need to be formulated more precisely. At this objective definition level, we propose some relaxation. In Chapter 4 SETA framework details will be defined more specifically these properties when discussing the solutions for the detailed functional requirements of the proposed platform. For example, it will be discussed what kind of data source changes and what
kind of new data sources can be accommodated into the analytical environment without involving developments processes. And if this is not possible for some data source, what kind of components must be developed.

### 1.4 Pillars of the SETA framework

In this paper the word *pillars* is used synonymously to *Design Guidelines*. Design guidelines allow the construction of a given output aligned with a set of general principles or requirements while allowing certain degrees of freedom in the design decisions and its concrete implementation. The design guidelines are related to the construction strategy while concrete decisions belong to tactical or operational level. An example guideline from the sports field would be: 'The team must play in the rival camp as much time as possible'. This directive may have different implementation options that are determined for example by the capacities of the available players, or by depending on the behaviour of the opponent. If the team has very technical players, the directive would be implemented through long ball possessions with short and quick passes. Contrarily, if the team capacities are physical ones, the guideline will be met by great pressure against the possessions of the rival and rapid advances over the opponent’s field.

Design guidelines are especially useful in the field of information systems because usually the options concerning to implementation, tools, methods, etc., are numerous. In fact, in agile methodologies like *Scrum*[^2], the design guidelines are frequently used because they allow accommodate the instructions from the Product Owner while leaving in the hands of the scrum team the most detailed decisions, resulting in a greater development autonomy and therefore more agility. Specifically, as detailed in Chapter 2 *Related work*, the available options in the field of DW systems with respect many of their components, design techniques, procedures, tools, structure, etc. are many.

In addition, the concept of pillars is useful regarding the clarity of the organisation of the document. It is also interesting for future developments in the framework, where there are different improvement paths of the proposal in different perspectives targeted by the pillars. These pillars are the following: 1) user-centric approach, 2) automation, 3) Model Driven Architectures (MDA) and common to the previous three 4) rich metadata and semantic aware definitions. The figure 1.4 outlines the comments in the previous sections regarding the derivation of the

[^2]: Scrum, is part of the Agile movement and an alternative to classical project management paradigms: [http://scrummethodology.com](http://scrummethodology.com).
pillars. The competitive environment in which organizations are embedded and the need to adapt to market with rapid clockspeed, empower the organizational capabilities related to agility and flexibility. These values also determine certain general requirements regarding the data analysis pipeline in the organization and establish the core properties of the information system which must support this pipeline.

Figure 1.4: From organizational environment to analytical framework pillars.
The Figure 1.5 shows an alignment matrix between objectives and pillars. A cell is checked if the corresponding pillar clearly supports the implementation of the objective. Logically, it is possible to discuss about different degrees of support, but for simplicity, only the most obvious associations have been checked. The next paragraphs propose a first introduction and definition of these pillars in SETA.

<table>
<thead>
<tr>
<th>User IT Independence</th>
<th>Automation</th>
<th>MDA Architectures</th>
<th>Metadata and Semantic aware definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Checkmark]</td>
<td>![Checkmark]</td>
<td>![Checkmark]</td>
<td>![Checkmark]</td>
</tr>
<tr>
<td>Adaptable to changing requirements</td>
<td>![Checkmark]</td>
<td>![Checkmark]</td>
<td>![Checkmark]</td>
</tr>
<tr>
<td>Adaptable to changing data sources</td>
<td>![Checkmark]</td>
<td>![Checkmark]</td>
<td>![Checkmark]</td>
</tr>
<tr>
<td>Specific BI suites independence</td>
<td>![Checkmark]</td>
<td>![Checkmark]</td>
<td>![Checkmark]</td>
</tr>
</tbody>
</table>

Figure 1.5: Alignment between framework objectives and design guidelines.

1.4.1 User-centric approach

The user-centric concept in general is ambiguous and it is necessary to set a certain context to define it with specificity. A good general definition can be found on wikipedia: the user-centered design aims to create products and services that meet specific needs of the end user, achieving greater satisfaction and best possible experience with minimal effort on her/his part. This definition contains some important ideas 1) The user-centered design must meet specific user needs 2) dedicating the minimum effort possible and 3) getting a good experience in the use of a product or service.
A key point is that user-centered design must meet the specific needs of the user. However, in the field of design of analytical systems, to define the specific needs can be quite challenging, mainly because different users have different needs, and as often happens, the needs of different users in the same organization may raise conflicts regarding the solution is intended to build into the overall organization. Therefore some relaxation will be assumed in the definition of the user objectives for the definition of what user-centric means: First, it is assumed that there are basically two categories of users: business and technical users. Second, it is assumed a simplification by assigning the objectives of SETA to general objectives of each category of users. Figure 1.6 shows a graphical overview of the users interactions along the analytical pipeline.

![Figure 1.6: General view of the user-system interactions along the complete analytical pipeline.](image)

Concepts directly linked to the term user-centric in the field of DW systems are On-Demand BI, BI 2.0 [7] or Self-Service BI [8]. On-Demand BI coincides with the objectives of interest (reducing the time of deployment, does not require technical skills from business users, ability of the end user to deploy new analytics), but not in the way this approach meets these objectives, i.e. providing into BI solutions embedded business models (or business areas) that already incorpo-
rate predefined analytics for that type of business. BI 2.0 is more applicable to the front-ends of analytical tools such as OLAP because it is based on the recommendation of queries based on ratings and on rewriting queries based on user preferences. Finally, Self-Service BI is associated with the search, identification and integration of data sources outside the organization, autonomously performed by the user. None of these concepts fully agrees with the approach of SETA: It does not recommend or rewrite queries, does not pursue the incorporation of external sources autonomously by the end user and does not provide predefined analytical areas for a particular business. Therefore, our user-centric approach is more related to the capacities of obtain data from the Global Schema quickly and autonomously, using a powerful GUI and using domain concepts. This data is then used to feed the preferred final tools of the business users.

1.4.2 Automation

In general, in scenarios in which the goal is to enhance the end user autonomy with respect costly software development processes, and to enhance the user independence with respect deep technical skills, automation is inevitable. After all, this scenario involves putting more ability to control and operate the systems in the hands of the end user, but at the same time implies to separate away these tools and models from the language that computers understand. To bridge this gap, the system must provide more complex architectures, models and automated software components.

In the specific area of analytical systems, when it is required to incorporate dynamic requirements and new data sources in an agile and flexible manner, the automation is also required. As will be seen in Chapter 2 Related work, the design of the logic model of the DW in a changing environment (with respect both, data sources as requirements) has been, and remains, a very active research field.

However, when the 'distance' between the sources and the final model is very large, automation leads to two undesirable consequences (simultaneous or not) 1) The complexity of processes significantly increases, and 2) The parameters and settings of the automation model that are left to the end user, cause the model no longer be easily manageable, thus no longer flexible nor agile. The consequence of this is that automation has degrees and therefore the choice of the engagement point becomes key. It is true that advances derived from research change the position of that point of commitment, but so is that, this work takes a practical approach according the current technologies.
Specifically, automation in SETA must cover two fundamental elements of the analytical systems: departing from the end user analytical requirements 1) The generation of the target logical model and 2) the generation of the ETL to feed the target model. The commitment point here, is that in order to incorporate a new source into the system, a wrapper must be developed and the source must be explicitly declared into the Definition Layer of the framework, rather than trying to automate the process from end to end. To understand the details of this approach Chapter 3 SETA overview gives an overview of the modules and the architecture of SETA and Chapter 4 SETA framework details enters into the implementation details.

1.4.3 Model Driven Architecture (MDA)

A framework using automation, specification by models and platform independence must necessarily consider the Model Driven Architecture (MDA) [20] from the Object Management Group (OMG). One of its main objectives is to separate the design of the construction technology by defining three architectural levels: domain model, logical model and implementation model, and the transformation processes between these architectural levels.

In the field of DW systems many authors have proposed different strategies for the design and construction of the DW applying in different ways the MDA model. Section 2.3 Model Driven Architectures will review some of the major contributions in this area. Similarly, there are numerous tools, languages and models in the world of Open Software that can be useful.

Nevertheless, we do not want restrict the design of SETA strictly to a particular separation architecture. We are more interested in the fundamentals and strategies than in the level definitions, their tools, languages and transformation models. Our framework include some novel approaches and existing tools that are difficult to fit in a pure MDA architecture.

1.4.4 Metadata and semantic aware definitions

It could be considered that this pillar is transversal to the previous ones. In the case of user-centric approach, a key element is to provide the user with a view of the system in terms of domain concepts that he/she can understand and manipulate. This requires exposing the behavior models and conceptual schemas of the system with high levels of abstraction. In the case of automation departing from models and conceptual schemes, it is necessary that the processes can interpret
such models and schemes and transform them into operations and structures processable by computers. If we want these processes to be flexible and adaptable to changes in the models and schemes, it is necessary to use programming strategies based on metadata. Finally in the case of model driven architectures, the use of definition languages adapted to the abstraction level they represent, is inherent to the model itself.

In the area of DW systems, different metadata classification methods have been proposed. Despite being quite general, one of the most interesting for the purpose of our framework is the classification into Technical metadata, Business metadata and Process / Navigation Information metadata [37]. Technical metadata primarily supports technical staff that must implement and deploy the data warehouse. The technical metadata includes the system metadata, which defines the data structures such as tables, fields, data types, indexes and partitions in a relational engine, as well as databases, dimensions, measures, and data mining models. Business metadata is content from the DW described in more ‘user-friendly’ terms. The business metadata tells the user what data he has, where the data come from, what data mean and how data is related into the DW. The business metadata primarily supports business end users who do not have a technical background, and cannot use the technical metadata to determine what information is stored inside the DW. The Information Navigator metadata is a facility that allows users to browse through both the business metadata and the data inside the DW.

Another fundamental aspect is the way the metadata is specified, i.e., the definitional artefacts and the languages that support them. Perhaps one of the most widespread standards for the exchange of metadata is the Dublin Core model, whose implementations are typically based on the Resource Description Framework (RDF)\(^3\). More than the standard itself, what is interesting, is the simplicity and flexibility to describe any domain concept by using RDF triples subject, predicate, object. Nevertheless, the simplicity of RDF also imposes some major restrictions when modeling schemes and domains. For example, RDF is based on binary relationships and therefore, it does not allow to express restrictions or cardinality properties, besides not to include the concept of classes.

The solution comes from languages such as RDF Schema (RDFS)\(^4\) and Web Ontology Language (OWL)\(^5\). RDFS provides the ability to define classes, taxo-

\(^3\)The RDF language: [https://www.w3.org/RDF/](https://www.w3.org/RDF/).
\(^4\)The RDF Schema language: [http://www.w3.org/TR/rdf-schema/](http://www.w3.org/TR/rdf-schema/).
\(^5\)The OWL: [https://www.w3.org/OWL/](https://www.w3.org/OWL/).
onomic relationships and the declaration of ranges and domains for the properties, so naturally introduces schema representation. OWL incorporates the possibility of defining the cardinality of relationships and inference mechanisms to derive new knowledge from the explicitly stated. OWL is the language of choice for describing ontologies. OWL incorporates the separation that Description Logics (DL) makes between terminological component (TBOX) and Assertive component (ABOX) in an ontology. The TBOX component provides the definition of the set of concepts and relationships of a particular domain modeled using an ontology. The ABOX component provides the statements of individuals belonging to these concepts and their relationships.

In short, this allows us to introduce one of the central elements in SETA, that links the pillars described in the previous sections and articulates the metadata strategy. The definitional component of SETA (Definition Layer) is a set of three ontologies (domain, global schema and sources) and the links between them (mappings). Each of the three ontologies incorporates the concepts and properties associated to the level that each of them represents. At the same time, it allows the articulation of the Model Driven Architecture strategy by separating the TBOX component (SETA definitional metamodel) from the ABOX component (SETA instantiation into a given domain). In 3.2.1 Definition Layer, an overview of the structure and content of the SETA definitional mechanisms is presented. Section 4.1 Definition Layer Formalization presents the formalization of these components.
Chapter 2

Related work

The solution proposed in this work covers different disciplines and components that have been object of research for years. The number of works and references that are somehow related to the system described in this document is very high. For clarity reasons the related work is presented according the pillars presented in section 1.4 Pillars of the SETA framework. Obviously, as this framework uses components developed in previous research projects (for example Quarry [6]), the authors and works directly related to this research are the most interesting in practice (specifically, the references related to the Automation pillar). Nevertheless, we systematically searched for papers related to each of the pillars and found other different and interesting approaches.

2.1 User-centric approach

One of the pillars of the proposed system is the user-centric approach. While we intuitively can take an idea about the significance of this concept, a more thorough analysis of the context in which it appears evidences that this concept has many dimensions and it is necessary to define more concretely what aspects of user-centric approach must incorporate the framework. This section describes some of the properties associated with the concept, discussing their inclusion or not into our definition of user-centric pillar.

In A survey of user-centric data warehouses: From personalization to recommendation[22], the authors perform a review of various proposals where the term user-centric is linked to data warehouse systems, identifying two families of different strategies: 1) Personalization and 2) Recommendation. The former is based on the implicit intervention of the DW system by customizability of the user behavior according to their preferences, while the latter favors the explicit
intervention of the system to assist the user in their decision-making process although the user does not accurately known the DW scheme. The customization can be implemented by creating a user profile and its subsequent exploitation, for example by adding to the OLAP queries, conditions based on the information included in the profile. Thus, different users will get different answers to the same query according to their preferences. The recommendation strategies, require the capture of the user feedback on previous answers (content-based methods) and/or to consider the feedback provided by similar users (collaborative filtering).

Nevertheless, these strategies typically require the existence of a module to process queries (the query processor), which is not contemplated in the proposed framework because its aim is not to answer queries but obtain data to be explored and analyzed with external tools (such as OLAP or data mining tools). In any case, these strategies could be incorporated not in the queries but in the manner in which the user determines and selects the data he wants to analyze. Although strategies based on recommendation open exciting possibilities, it was decided not to include them, for simplicity, in a first design framework, prioritizing the creation of structures, models and languages, which among other things are necessary to properly support a further recommendation system. Regarding customization, is clearly an aspect to include because one of the key aspects of the framework is the interface (GUI) for requirement definition and data set selection.

In many references the user-centric property is directly related with concepts as BI 2.0, Self-Service BI and On-Demand BI.

In Towards Next Generation BI Systems: The Analytical Metadata Challenge [7], authors emphasize that BI systems 2.0 should allow the user to perform tasks of data analysis without having to rely entirely on IT professionals, i.e., the user must be as autonomous as possible once the BI system is delivered to business users. This is fully in line with the objectives of SETA. They defined a set of the metadata artifacts required to support the analytical user activities in the context of next generation BI systems, and they performed a survey on data warehousing user-centric approaches. One of the conclusions is that this ideal scenario is yet far from reality. Another relevant conclusion is the importance of metadata to support the user and to support the automation of many of the processes involved in the data-to-knowledge pipeline. This aspect is revisited in the sections related to Multidimensional design automation and ETL automation.

However, again, the analyzed approaches are based on query recommendation and always receive a query as input. As already mentioned, the proposed framework
does not have this behavior.

In Fusion Cubes: Towards Self-Service Business Intelligence [8], the definition of self-service business intelligence is about enabling non-expert (non-technical) users to make well-informed decisions by enriching the decision process with situational data, i.e. data that is not owned by the decision maker and that is not formally included in the existing BI system. Usually retrieving this situational data involves the whole process of discovering the relevant sources of information, extract the right data, integrate it within the existing BI system data and present it to the user. Ideally, the user should incorporate this data to the analysis process with the minimum support from the IT department and with the maximum independence from the IT development processes.

The vision presented in this work is far from the scope of SETA. Specifically, in the SETA framework (at least in its first definition in this document) ETQ (Extract, Transform and Query) pipelines are not included, nor the possibility of incorporating situational information as proposed in the article. In SETA, the sources should be stated explicitly into the platform through its conceptual model, mappings and links with the three levels of definition: Sources, Global schema and Domain.

2.1.1 Conclusions.

The concept of user-centric in the context of systems DW has multiple perspectives and different strategies. Many are based on the transformation of queries posed by the user on OLAP system using information about the user profile (Personalization), the feedback provided on previous results (Content-based recommendation) or based on the feedback provided in previous results by similar users (collaborative feedback). In any case, to provide a good user support, are required new metadata artifacts associated with all components of the analytical pipeline. In addition, the use of ontologies to represent domain objects and their semantics, provides: 1) The context necessary to enable a query language closer to the natural language and to the domain terminology, and 2) the possibility of including into the analytical process information of external sources semantically annotated which are not defined by the logical model of the DW and therefore are not integrated into the BI system. This opens new possibilities for progress in the new and ambitious paradigms on data analysis like BI 2.0, On-Demand BI and Self-Service-BI.
Nevertheless, SETA is not intended to cover the functionality of front-ends systems like OLAP or data analytics systems, but drawn to these systems the data for further analysis. Therefore, the strategies based on query rewriting are not directly applicable. Indeed, in the case of SETA would be more applicable the term requirement rewriting.

Given that the GUI for manipulating the conceptual schemas through the different components of the framework is a key element, it makes sense to endow it with personalization strategies through user profiles to provide the view of the conceptual schema levels that are appropriated to each user level.

Finally, another common element of many user-centric approaches is the ability of business users to perform the tasks of the analytical pipeline with the minimum technical support. This means providing the user with maximum independence with respect to the IT department development processes. Clearly, the proposed framework should incorporate these capabilities into its design.

2.2 Automation

2.2.1 Multidimensional design automation

In the path to the multidimensional design automation many authors have produced interesting articles addressing the problem of systematic DW design\(^1\). According to Winter et al. [3], multidimensional DW design can be classified by supply-driven or demand-driven frameworks. In pure supply-driven approaches the multidimensional model for the data warehouse is derived from the analysis of data sources. User requirements are used not in the design phase but when the implemented multidimensional model is queried. Oppositely, in pure demand-driven approaches the design is carried out based on the requirements to generate the multidimensional model. The datasource is considered only when populating the data warehouse [13]. Supply-driven approaches simplify the process of properly populating the DW but they do not provide mechanisms to highlight the important parts (from the business requirements point of view) of the data sources, thus wasting resources by specifying unneeded information structures in the MD model because a brute-force search of multidimensional patterns over the sources produces a huge number of MD components [15]. Oppositely, demand-driven approaches lack mechanisms through which to formally match the data sources with

---

\(^1\)A review of existing DW design methodologies can be found in O. Romero, A. Abelló. A survey of multidimensional modeling methodologies. Int. J. Data Warehouse Min. (IJDWM) 5 (2) (2009) 1-23.
information requirements in early stages of the development, thus making it highly complex to populate the DW in a proper manner.

In order to overcome the drawbacks of both pure supply-driven and demand-driven approaches, Mazón and Trujillo proposed a hybrid framework for the multidimensional modeling of Data Warehouses [14]. Additionally, they addressed the problem of the whole development of all the DW components within a common framework based on the Unified Modelling Language (UML) following a Model Driven Architecture (MDA)\(^2\) [19]. In essence, they depart from an information requirement model (Computation Independent Model, CIM) to obtain a conceptual independent model (Platform Independent Model, PIM) which must be reconciled with the data sources to obtain a hybrid PIM. Finally, several logical models of the Data Warehouse can be derived from this hybrid model as Platform Specific Models (PSMs) by considering different deployment platforms (relational, multidimensional, etc). The main drawback of this approach is the way they obtain the CIM (from business objectives to information requirements), because it is difficult to deal with new or changing requirements once the system is defined.

If we assume that business requirements and data sources (both current and futures) are well know and defined, the classical data warehouse\(^3\) approach enriched with some degree of automation is a good solution. Nevertheless, the current reality of most of business domains does not allow the hypothesis that the analytics requirements and data sources are defined exhaustively. Conversely, DW designers must deal with fuzzy and evolving requirements, new and changing data sources and shorter time-to-market deadlines\(^4\). This leads to the need of face the problem of DW design with more flexible procedures and tools, ideally in an automatic manner.

In Automating Multidimensional Design from Ontologies [4] the authors introduced the automation of the design of multidimensional schemas using ontologies (AMDO). The authors proposed a semi-automated method aimed to find multidimensional concepts departing from an ontology representing the data sources of the business domain. They identify in a semi-automatic manner (using the user’s feedback on each step) Facts, potential bases of the facts (a collection

\(^2\)The MDA provides an approach for specifying a system independently of the platform that supports it, specifying platforms and specifying the transformations of the system specification into one for a particular platform [20].

\(^3\)In 3 SETA Overview we enter deeply into this concept to highlight the main differences and contributions of the proposed framework.

\(^4\)See 1.2 Current analytical environment for a discussion about the analytical needs in the current environment.
of concepts characterizing dimensions) and Dimension hierarchies that conform the multidimensional model. The key contribution of this work is the use of ontologies to represent the conceptual schema of the domain at hand (in this case the data sources). Indeed, as ontologies are semantically richer than relational and multidimensional schemas, they are specially appropriated to model non relational domain schemas and data sources overtaking the representation limitations what relational schema has. The automatic part of this method is supply-driven (i.e. from the data sources). The demand-driven part is provided by the user in the form of feedback about the multidimensional components (facts, dimensions and hierarchies) produced on each step. The method assumes that the input ontology provides multiplicities between concepts. This means that it is possible to build a matrix of NxN concepts where each cell is marked if the concept in the row has a to-one relationship with the concept in the column, i.e., the concept in the column is a potential dimension.

Some of the criticisms of this strategy are that 1) it is only valid for a single and small ontology and 2) that information on multiplicity is rarely included in the source ontology. Regarding 2), this information can be provided by manually adjusting the ontology or semi-automatically by exploring the definition of the source by a source-dependent process. With respect 1) being this a model (at least the first phase) supply-driven, suffers from the defects of this kind of approaches when the size of the ontology is big and therefore the volume of potential multidimensional components that are presented to the user can be considerable.

From here, this literature review focuses a sequence of works of progressive enrichment of the AMDO approach which after incorporating the serie of works about ETL automation, culminates into the Quarry solution [6] which is one of the key parts of the proposed architecture. Therefore these references are the most interesting ones for the development of the framework and will be treated in deep in Chapter 4 SETA framework details.

In Automatic validation of requirements to support multidimensional design [24] the multidimensional scheme is derived from a hybrid approach that considers both user requirements and data sources. They introduced MDBE (Multidimensional Design by Examples). In this method the sources considered are relational, while the requirement information are expressed by SQL queries. The model analyzes the queries creating a graph for each one, and is labeled by using the relational schema of the sources and by a set of multidimensional validation rules. Finally, the valid graphs obtained from the different queries (requirements) are joined into a minimum multidimensional model that subsumes all the graphs
and are represented by a logical constellation scheme. MDBE establishes a framework that can be used incrementally to accommodate into the design, new multidimensional requirements facilitating the maintenance of the data warehouse. The weaknesses of this framework (with respect to the general objectives of the SETA framework) are that 1) only contemplates relational sources and 2) the requirements are supplied to the system with SQL queries.

Conversely, in A framework for multidimensional design of data warehouses from ontologies [15], the authors depart from the AMDO framework incorporating some mechanisms for working with larger ontologies, by reducing the multidimensional components obtained in the exploration phase of the source ontology (the supply-driven phase). This is achieved by filtering functions and searching patterns. Specifically, filtering functions implement searching patterns that can be defined by the designer to rank the identified multidimensional components. The searching patterns can be already known heuristics (such as the number of measures of a given fact), or any other determined by the designer. Even so, the automatic phase of the method remains supply-driven. The inclusion of the requirements to derive the final multidimensional scheme, is assisted by the user.

In GEM: Requirement-Driven Generation of ETL and Multidimensional Conceptual Designs [5], converge earlier works on automation of multidimensional design and automation of ETL in a new framework (GEM) whose objective is to generate both the multidimensional schema as the ETL that transports data from data sources to the generated multidimensional model. It is a hybrid approach that considers both, user requirements as available data sources. GEM assumes that data sources are defined by an OWL ontology that captures its structure. The requirements should be provided in a structured way in an XML file that basically indicates facts, measurements, dimensions and aggregation operations. In fact, although the requirements are captured from the ontological domain definition, GEM only supports multidimensional operations and concepts.

Finally, in Quarry: Digging Up the Gems of Your Data Treasury [6], an implementation of the proposed GEM framework is provided by adding a GUI for specifying the requirements departing from the domain ontology in a visual manner (the requirement elicitor module) and a module to deploy the generated multidimensional design and the ETL on the supported physical platforms. Quarry is one of the main components of the proposed framework, therefore Chapter 4 Technical solution will go into the details of its components, operation, and specially, the proposed modifications and new modules for integrate it into the...
SETA framework.

2.2.2 ETL Automation

The main ideas about source treatment are inspired by the work of Dimitrios Skoutas and Alkis Simitsis from the National Technical University of Athens. In [1] they developed a simple method to represent datastores as directed graphs. The nodes of the graph represent elements of two types: 1) elements containing the actual data and 2) elements that contain or refer to others elements. Edges represent the containment and reference relationships between elements. Additionally, edges can be labeled with the corresponding cardinality of the relationship. With this very simple approach it is possible to represent structured and semistructured datastores elegantly and uniformly. Another contribution of this work is the use of an ontology graph (they provide a simple algorithm to transform and ontology into an ontology graph), $G_o$, to describe the domain and use it to annotate the datastore graph, $G_s$. In essence, mappings are pairs of nodes $(v_s,v_o)$ where $v_s \in G_s$ and $v_o \in G_o$. With this formalism, and a set of the most frequent conceptual operations in an ETL scenario, they provided a method to infer the conceptual ETL operations to populate the datastores into the target store. In [2], they improved the process and expressed the ETL using using the graph transformation derivation formalism.

There are some crucial differences between this approach and the strategy developed in SETA. First, our objective is not the automation of the ETL process from the datasource to the target schema because we think that the generalization of this solution for any datasource and any target schema is not possible in real cases. Instead, the SETA framework provides a wrapper for each datasource or family of datasources which encapsulates part of the structural, technical and semantic complexity of the source. This wrapper must be programmed manually. Using a graph representation for both, the datasource and the Global Schema, the process populates the records from source into the Target Schema. In this scenario, the definition-driven datasource processing, isolates both the structural variations of the source and the evolution of the validation rules needed to ensure the data quality of the source. Another crucial difference is that in the framework proposed by [2], the domain ontology is attached and used for annotate both, the datasources and target graphs whereas in our approach the domain ontology is

\[^5\text{In the DW context a wrapper is a program that transforms data from the source native format and schema to some unified format and schema, typically the relational.}\]
not directly attached to the datastore subgraph but to the Global Schema, which in our case can be understood as an Integration Schema. Finally, the automation goal is different. Whereas [1] and [2], follow the automation from the datasource to the domain ontology, the goal of SETA goal is the automation from the Global Schema to a target schema determined by the user’s analytical requirements.

Despite their original ideas, the previous references have some drawbacks. First, they do not propose a clear method to determine what are the concepts of the domain ontology that are really relevant and thus, there is no clear method to determine the relevant components of the datasources that must be propagated to the Global Schema. In other words, the framework do not clearly incorporate the analytical requirements. Second, the method proposed cannot deal with new sources incrementally, i.e., each modification in the sources requires the complete execution of the conceptual ETL derivation. And finally, the automatized ETL is 'conceptual', so cannot be deployed in real systems.

Other authors like Petar Jovanovic, Oscar Romero and Alberto Abelló from the Universitat Politècnica de Catalunya, BarcelonaTech that were investigating the automation of the multidimensional design of DW collaborated with the previous authors to introduce two major contributions to the original works: 1) The analytical requirements and 2) The ETL compactation departing from individual conceptual ETL flows. In Integrating ETL Processes from Information Requirements they developed the CoAl algorithm to consolidate different ETL flows from different analytical requirements into a unique conceptual ETL flow satisfying the individual requirements up to date. From the point of view of data integration, they adopted a different strategies than these based on the existence of an unified view of a global schema. The strategies based on schemas on the fly (Global as view) were not able to deal with complex transformations and data grouping and aggregation, so they proposed a more flexible and powerful procedure based on ETL to populate the target schema. Formalizing ETL as directed acyclic graphs, CoAl integrates a new ETL graph corresponding to a new requirement into the up to date ETL graph belonging to existing requirements in the system.

The output of the algorithm is a conceptual representation of the consolidated ETL expressed with a XML-Like format named xLM. The next steep to the automation goal is to create platform-dependent ETL from this conceptual one. The main contributions in this area were developed in Engine Independence for Logical Analytic Flows [17]. In this work the authors made three main contributions 1) The development of a method to describe analytical flows at logic (platform-independent) level, 2) The conversion of physical analytical flows
into logical ones, and 3) the conversion from logical analytical flows to platform-dependent physical flows.

2.2.3 Conclusions

As result of the difficulty of the DW schema design, many works have faced the problem of creating systematic, semi-automatic and automatic procedures to help the designer in the task of generating the DW schema that must support the multidimensional analytical models required by users. The two main inputs of these procedures are the data sources and the user requirements. The supply-driven approaches focus on the design starting from data sources, while the demand-driven approaches start from user requirements. The scenarios in which the business user ’pulls’ from the analytical pipeline are more flexible than those in which the designer ’push’ a predefined multidimensional model to the user. This is especially obvious in those scenarios where the analytical needs are not fully known in advance, which is nowadays a common situation. However, the pure application of one or another method has drawbacks, so it makes sense to consider that the best approach is to consider both inputs. This is the approach of hybrid methods which combine, with different levels of balance, both approaches.

In the section of current analytical environment it was introduced that the classical approach of DW systems, despite its maturity and robustness does not adequately deal with the analytical needs of today’s organizations. Regarding the logical schema of the DW, the approaches based on a predefined design using the relational model does not provide sufficient flexibility to accommodate new data sources and new analytical requirements. Similarly, relational data sources, must coexist with new data sources with changing structure and represented with much more flexible models. The representation of both, data sources as business domains, by semantically richer models like those based on ontologies, provides great flexibility in the representation of the conceptual models. However, these conceptual models should be able to be instantiated into logical data models to store, manipulate and visualize the data using the widely used and essential tools in any analytical pipeline. In addition, data mining tools are increasingly used in organizations, so the infrastructure that supports the analytical pipeline should also allow the data generation to feed these tools with the same criteria of agility and flexibility than for generating multidimensional models.

Additionally, automate the multidimensional DW design is not enough. A framework that supports the automation of the analytical pipeline should also provide the automation of ETL. This review has identified the most relevant ideas
to be included in the proposed framework and some existing solutions that greatly facilitate its implementation. In Figure 2.1 it is outlined the convergence of works on both topics (MD design automation and ETL automation) into an integrated approach focusing the automation of both topics.

![Diagram](image)

Figure 2.1: Convergence of related works focusing MD design automation (1) and ETL automation (2) into a sequence of collaborations addressing the automation of both, ETL as MD schema, using ontologies.

Nevertheless, we believe that the direct propagation of the data from the sources to the final models that are exploited by the user is not realistic in most of the cases. The SETA framework combines these approaches with the classical DW systems structure creating an Integration or Global schema where part of the complexity of sources integration is solved semi-automatically. Another challenge of the methods reviewed in this section is the automatic identification of multidimensional concepts by exploring the source definition. Again, SETA takes a different approach by explicitly identifying in its Definition Layer these concepts. Additionally, the complexity of the processes for generating the final target schema is separated into two parts: First, to process the sources into the Global Schema (Schema on Write part) using metadata-driven strategies (thus, isolating the code from source and Global Schema changes) and second, to process the schema generation and data extraction from the Global Schema to the final target using a requirement-driven approach (Schema on Read part). This structure is discussed in detail in the next Chapter 3 SETA Overview.
2.3 Model-Driven Architectures (MDA).

Conceptually, MDA[20] from the Object Management Group (OMG), establishes a software development paradigm on which the final system is built by defining a set of models, from the most abstract to the most concrete levels. Ideally, the abstract levels must be expressed through specific terms and concepts of the application domain. In MDA, it is possible move from one abstraction model to the next lower through transformations, which are the main engine of the model. These models are not only documentatives, but constitute the essence of MDA because the ability to automate the transformations is an inherent property of this approach. Figure 2.2 shows a global overview of the MDA framework.

In the design of databases and data warehouses, the essence of separation between models existed long before the definition of MDA by the OMG. The separation of the design in conceptual model, logical model and physical model is already a classic technique in this area. Therefore, it is normal to find works addressing the direct application of the MDA model for designing DW systems. Some of the revised references, fall on the methodological component but either do not provide mechanisms for automation, or focus on concrete aspects of the DW design (ETL or design MD) but not on its whole design. One of the first works to align the DW design with the general framework of MDA is the Model Driven Warehousing (MDDW) [36], that provides a set of comprehensive metamodels to fully model the system DW, including data sources, ETL processes and MD modeling. However, being completely based on CWM6, these metamodels are too complex to be handled by end users and designers[35] and too generic at the conceptual level (PIM). Therefore, this type of purely standards-based approach does not satisfy our user-centric pillar.

![Figure 2.2: MDA general framework. The system is built with models at different abstraction levels. In some cases the production of lower abstraction levels can be automated by transformations.](image)

---

Authors like Jose-Norberto Mazón and Juan Trujillo from the University of Alicante have focused much of his research in the application of MDA to DW design. In *An MDA approach for the development of data warehouses* [19], they develop a MDA-compliant framework for the design of all the DW components. Figure 2.3, shows an overview of his framework. Each DW level (left) is designed according to the three types of viewpoints of MDA. Therefore, they consider that the whole development of a DW can be structured into an integrated framework with five layers and three viewpoints for each layer. Its design is very elaborate and defines both the models and the transformations between the different levels of the MDA framework. They used UML as a basis of modelization language, and added some language extensions to achieve the expressiveness required at each level. Actually, SETA takes a much less ambitious approach in terms of the possibilities of automation, but from our point of view has a better adaptation to situations with evolving data sources and requirements. Perhaps the most ‘weak’ point of his proposal is that it does not contemplate the possibility of defining explicitly the analytical requirements from end-users, that is, it focuses on building a DW system from a more classical approach and therefore, with a more static resulting design.

Figure 2.3: MDA framework for DW design from Trujillo and Mazón. The framework defines three models (MDA levels) for each DW architectural layer (left). Images taken from *An MDA approach for the development of data warehouses* [19].

2.3.1 Conclusions

In short, we think that although MDA is a promising approach in general, its pure application does not match the SETA objectives and pillars. Actually, the
objectives are different, while MDA faces the development of a static DW system with respect to analytical requirements, SETA provides a specific framework for the analytical pipeline addressing frontally the problems of flexibility and agility of the current analytical environment. In SETA, the incremental construction of the analytical system is inherent in its definition, whereas in MDA this is precisely one of its weaknesses. Existing requirement specification models (CIM) do not provide sufficient expressiveness at domain level to define specific analytical requirements, but rather are intended to define more general requirements. At the same time the platform-independent models (PIM) are too complex to be handled directly by end users and designers.

From MDA SETA takes the essence which is the separation between model and implementation but uses different layers and different model specification methods and languages. SETA uses extensively ontologies to represent the sources, the Global Schema and the Domain and takes a semiautomatic approach to these aspects where automation will lead to too complex models and inefficient implementations.
Chapter 3

SETA overview

Chapter 1 introduced the concept Data LakeHouse (DLH) as an approach for the data store on some intermediate point between Data Warehouses (DW) and Data Lakes (DL) strategies. Now it is time to characterize with more precision this engagement point. The first section summarizes the properties and main differences between DW systems and DL systems, introducing the solutions adopted by SETA. The second section goes into the SETA framework architecture, its structural layers and its individual modules.

3.1 Data warehouses, data lakes and data LakeHouses

In Chapter 1, it has been shown that the current analytical environment poses to organizations the need their analytical pipeline to be agile and flexible to adapt to evolving information needs and changing data sources. These cannot be determined at design time of DW systems. The gradual construction of the data store becomes the best alternative. Therefore, the approach of classical DW systems is no longer the most appropriate. However, this does not mean that Big Data technologies are appropriate for all organizations. For example, the successful implementation of a Data Lake also poses significant challenges, as well as resources and capabilities that often are beyond the reach of many organizations\textsuperscript{1}. This section will delve into the characteristics of both to define more concretely the features of a Data LakeHouse.

\textsuperscript{1}Gartner. Gartner Says Beware of the Data Lake Fallacy: \url{http://www.gartner.com/newsroom/id/2809117}. 
The term Data Lake was coined by James Dixon from Pentaho. He wrote: "If you think of a datamart as a store of bottled water - cleansed and packaged and structured for easy consumption - the data lake is a large body of water in a more natural state. The contents of the data lake stream in from a source to fill the lake, and various users of the lake can come to examine, dive in, or take samples". We like this definition because it is very intuitive. In the same line, we would define the term Data LakeHouse as "A set of pools of water with different shapes. Some of them are interconnected with pipes, and some of them are connected to the main output so that the user can easily regulate the composition of the mixture of outgoing water".

Table 3.1 highlights the three proposals with respect some relevant dimensions of an analytical environment. With respect data semantics, in DL is implicit, i.e., it must be interpreted by the data extraction and analysis processes. Of course, this kind of systems provide a set of metadata artifacts to ease the data processing and information interpretation. In DLW systems the idea is to have a Definition Layer with a predefined ontology levels which provide the data structure and semantics. Data in DL does not have any explicit schema, it must be provided on the fly (Schema on read), whereas DWH systems rely on databases systems with

---

flexible schemas like Document Stores and provides a preset definitional mechanism to represent the Schema. Therefore, DLW follow both, Schema on Write and Schema on Read approaches, i.e., data source processing must accommodate the flexible schema provided by the Definition Layer, and data extraction for analysis must accommodate the schema required by the analytical needs dynamically and flexibly.

One of the main challenges in the use of analytical systems in organizations is not its creation, but how to take advantage of its possibilities. One of the key success factors of any analytical framework is to provide a ‘good description’ of the information it contains. In this context, a good description means 1) descriptions in languages close to users and their domains and 2) descriptions that computer systems can process easily to systematize and automate their treatment, i.e. metadata. Since DW systems are intrinsically structured by a known logic data model (relational or multidimensional), point 2) could be relaxed. This is not the case in data lakes and data lakeHouses in which the structure and schema should be modeled on the fly. Any data set that is incorporated into the data lake must be identified and annotated with a set of metadata tags that allow locate, interpret and analyze the data. Perhaps, the main difference between them is that a data lakeHouse system provides a preset definitional mechanism to represent the different schema layers.

The existence or not of an intrinsic schema also determines the processing needs to incorporate data sources into the repository and to obtain data for end-users analysis. In a DL, data is incorporated raw, so the ingestion process is usually limited to the corresponding annotations. In a DW, data should accommodate the DW design. In a DLH data must also accommodate the schemes of the repositories, but being them based on flexible schemes, the processes should be much more flexible. Additionally, in many cases the integration of a source into the repository requires a control process to manage the batch loads and a system to manage physical or logical errors at instance level (alert, fix, relaunch, etc). Consider for example a source that supplies data incrementally. SETA contemplates in its architecture such components (see 4.2 Source Manager and Monitor).

3.2 SETA architecture

This section presents an overview of the framework. To clarify its structure and components, SETA is defined from 4 structural levels: 1) the Definition Layer level brings together the relevant declarations and definitions of the framework, 2) the Automation Layer includes the predefined modules and tools that enable
automation, 3) the Processing Layer groups the processes that move data from sources to destinations and 4) the Data Layer what contains the data. Figure 3.1 schematically shows this structure. We have decided not to include into the figure the lines of flow of data and control structures, to avoid saturating the diagram. The interactions of data and control structures between components will be detailed in Chapter 4 SETA framework details.

![Figure 3.1: SETA Overview. The four structural layers.](image)

### 3.2.1 Definition layer

The definitional level is divided into 4 groups corresponding to 1) sources, 2) integration schema, 3) domain and 4) requirements. The formal definition of each of these levels is included in 4.1 Definition Layer formalization, so this chapter will present the purpose and contents of each group. In order to give context to these definitions, the following paragraphs discuss some aspects related to the use of ontologies applied to domain description, data integration and system modeling.
In the field of information systems modelling, many authors make a clear distinction between models and ontologies [31] [32]. It is said that the models are prescriptive, while ontologies are descriptive, i.e., the models provide instructions on how should be and behave a certain system, while ontologies provide a description 'as is' of a domain, i.e. a commonly accepted representation of a given reality. From the perspective of SETA, the definitional level includes both. For example, the component 'Domain Definition' of the figure includes pure descriptive aspects of the domain (concepts, taxonomies, relationships), while the component 'Analytical Requirements Definition' include definitional artefacts related to the system model (type of extractions, variable transformations, selection of dimensions, etc.). The definition of the OMG for model-driven engineering identifies 4 layers of model abstraction: M0) instance layer, is where objects reside in the real world, M1) model layer, is where models live for a representation of a part of reality, M2) metamodel layer, contains the tools that M1 are built with, and M3) meta-metamodel layer that provides the tools and structures to define meta-models M2.

Regarding the data integration field, the so-called global schema provides an integrated view of data sources. According to [33], there are two approaches to the design of this global schema. In the first approach, the overall schema is expressed in terms of a database model (e.g., the relational model, or a semistructured data model), and represents a unified data structure accommodating the various data sources. In the second approach, the global schema provides a conceptual representation of the application domain, rather than a specification of a data structure. The conceptual representation of the domain is the obvious method to address the data integration problem using a declarative approach. All the advantages deriving from making various aspects of the system explicit (including the mappings to the integration schema) are obtained. By making the explicit representation of the domain, it is possible to gain re-usability of the acquired knowledge. A third advantage has to do with incrementality and extensibility of the system. The conceptual approach to data integration does not impose to fully integrate the data sources at once. Rather, after building the domain model, one can incrementally add new sources or new elements, when they become available, or when needed. SETA again, faces the problem by combining both approaches. In the 'Domain definition' component the strategy is the latter (conceptual representation) while in the 'Integration schema definition' component, the approach is more in line with the former.

Armed with these concepts it is possible to define a first concretion of the characteristics of the Definition Layer in SETA:
Regarding the strategy of system building (model), the SETA models provided by the Definition Layer are positioned in the M2 and M1 levels. In this context, the SETA Definition Layer establishes a metamodel for analytical systems which once instanciated into a given domain, produces a particular model of operation.

SETA uses ontologies for modeling the domain by representing the concepts and the relationships it contains, and distinguishing between taxonomic relationships (hierarchical clustering concepts) and non-taxonomic relationships. Although there is no clear division, we could say that the definition of concepts and taxonomic relationships describe the domain, while non taxonomic relationships have more to do with the model operation providing a prescriptive definition. Thus, from the perspective of system implementation, non-taxonomic relationships must be implemented by the components of the system.

The SETA Definition Layer is composed by three ontologies associated to the definitions of 1) Domain, 2) Global Schema and 3) Sources. In each of them, the descriptive and prescriptive components have different weights. For example, at the level of domain definition, descriptive component dominantes, while at source level prescriptive component dominates. The definitional layer also contains the relationships between these ontologies that establish the information mappings between the different levels.

Finally, SETA uses OWL to define ontologies. In this sense, the meta-model level is described by a TBOX ontology component, while the model level is described by an ABOX ontology component. SETA does not include in its definition real-world instances (M0 level). SETA uses an unique ontology to define the three different schema levels (source, global schema and domain) and uses hierarchies of TBOX classes and properties to organize them. In this document, the terms \textit{ABOX} and \textit{TBOX} are used to avoid confusion with its most widespread definition.

With these considerations it is possible to introduce the contents of each of the components:

- The \textbf{Analytical Requirement Elicitor} component provides the definitional artefacts to obtain from the domain ontology the analytical models required by the user (OLAP cubes, data matrices). Thus, it provides a prescriptive set of statements of the analytical model. It must be able to transform the user requirements into models and formats that could be processed. It is also responsible for storing all requirements incorporated into
the system by different users so that the GUI can present them to the appropriate users in the proper manner.

- The **Domain Definition** component is a descriptive definition of the domain through a domain ontology. His $TBOX^*$ includes the structural and definitional artefacts necessary for its instantiation in a specific application domain (its $ABOX^*$).

- The **Integration Schema Definition** component incorporates both the structural definition of the SETA data integration schema, as prescriptive elements about the operating model. It is defined by an ontology which incorporates in its $TBOX^*$ the artefacts to descriptively represent the domain of integration, and its logical structure according to the schema used to store each dataset in the data layer.

- The **Source Definition** component incorporates the structural definition of the source and prescriptive information necessary for processing it.

### 3.2.2 Automation layer

The automation layer includes the modules that make possible the automation capabilities in SETA.

- **The Schema/ETL Automator** generates the target schema and the ETL to populate data from the Global Schema to the target schema. In the case of MD models this component generates the DDL to create the database structures which supports the corresponding multidimensional logical schema. The inputs of this component are 1) the user requirements posed by the Requirement Elicitor Component of the GUI encoded in a XML-like file which contains the model type to be generated (MD OLAP cube or data matrices to feed Mining Tools), technical information about the target (DB engine, connection identifiers, encoding formats for matrices) and the data requirements (facts/measures, dimensions, additional attributes and filters) and 2) The $ABOX^*$ component of the SETA ontology that includes the Domain concepts and relationships (which are the same data concepts included in the requirements), the Global Schema definition, the source definition and the mappings between them).

- **Target metadata generator**. Is the module responsible for generating the metadata contained in SETA to achieve better integration with the final Data Mining and OLAP tools used by the user. Although these external tools are an essential part of the analytical pipeline (see Figure 1.1), they are not
integrated within the platform. The SETA ontology contains metadata like external codes, helps, descriptions, external links, etc that are interesting to integrate with user tools using standard metadata exchange formats.

- **The Source Manager and Monitor.** It is simply a module that allows to manage and monitor the results of processing the incremental load of the sources into the Integration Schema, i.e. the operation of the source wrappers.

- **The Source Metadata Importer** is a source-dependent module that allows the automatic generation of the *ABOX* definitions of a data source. It is useful when integrating a new source into the framework to generate a default definition and mappings and free the user to manually define them. Once created a default definition, the final user (domain level) and the designer (source and global schema levels) can adjust the definitions and mappings according to their preferences (In 5.2 WISCC Definition Layer there is an example of its use).

### 3.2.3 Processing Layer

The *Processing Layer* contains the processes that actually move the data between the different levels of the *Data Layer*, specifically from the source to the *Data LakeHouse* (wrappers and extractors) and from the Data LakeHouse to the target model automatically generated by the Schema/ETL Automator (*Generated ETL*). Therefore, the processes at this level are domain-specific of a given instantiation of SETA through its *Definition Layer*.

### 3.2.4 Data Layer

Finally, the Data Layer includes the physical database engines and data containers to store the data of the Global Schema and the inbox areas to process the source and the storage structures to keep the raw data when needed.
Chapter 4

SETA framework details

4.1 Definition Layer formalization

4.1.1 SETA Ontology classes and properties

The definition layer includes all the metadata and definitions needed by SETA to work. The framework establishes a meta-model (M2 layer according the model-driven engineering from OMG) which when instantiated into a given domain determines a model for supporting the analytical pipeline (M1 layer). In this context, the definitional components needed to define the three information organization levels (Sources, Global Schema and Domain Schema) are coupled into an ontology terminological set of statements (TBOX) which is instantiated into the target domain by producing a set of TBOX-compliant statements representing respectively the data sources structure and semantics, the Integration Schema structure and logical schema, the conceptual schema of the domain and the mappings between these three levels. SETA uses OWL as specification language. Specifically, the first definition of the framework proposed in this document uses OWL DL\(^1\).

As mentioned, there are three definitional components (Source, Integration Schema and Business Schema) that need to be represented with the ontology. Thus, it is necessary a mechanism to separate the definitional artifacts of these components. One possibility is to separate them in different ontologies, but for compactness and practical reasons the separation will be achieved organizing the three subjects with taxonomic hierarchies. In practice, the ontology will provide three root classes and three root properties from which the artifacts of each definitional component inherits.

\(^1\)OWL provides three increasingly expressive sublanguages: OWL Lite, OWL DL and OWL Full: \url{http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.3}. 

41
Formally, the $TBOX^*$ component of the SETA Definitional Ontology is $T = (T_s, T_g, T_d, A)$, where $T_s$ is the Source Definition Component, $T_g$ is the Integration Schema Component, $T_d$ is the Domain Component and $A$ is a set of annotation properties common to the three components.

- The **Source Definition Component** is a terminological component defined as $T_s = (C_s, P_s)$ where $C_s$ is a set of classes and $P_s$ is a set of properties with the following contents:

  \[
  C_s = \{\text{DataContainer}, \text{KeyContainer}, \text{RefContainer}, \text{DataAttribute}, \\
  \hspace{1cm} \text{SourceContainer}, \text{SourceTechDescriptor}, \text{DataGroup}, \text{SourceClass}\} 
  \]

  \[
  P_s = \{\text{hasSourceComponent}, \text{hasDataContainer}, \text{hasAttribute}, \text{hasKey}, \\
  \hspace{1cm} \text{hasReference}, \text{refDC}, \text{refDCKey}, \text{selfDCKey}, \text{SourceProperty}\} 
  \]

  The classes SourceClass and SourceProperty are the root levels for classes and properties respectively.

- The **Global Schema Definition Component** is a terminological component defined as $T_g = (C_g, P_g)$ where:

  \[
  C_g = \{\text{GlobalSchemaClass}, \text{GS_DataGroup}, \text{GS_DataSet}, \text{GS_Attribute}, \\
  \hspace{1cm} \text{GS_KeyContainer}, \text{GS_RefContainer}, \text{GS_DataSetTechDescriptor}\} 
  \]
\[ P_g = \{ \text{GlobalSchemaProperty}, \text{GS}\_\text{hasDataSet}, \text{GS}\_\text{hasAttribute}, \text{GS}\_\text{hasKey}, \text{GS}\_\text{hasReference}, \text{GS}\_\text{refDS}, \text{GS}\_\text{refDSKey}, \text{GS}\_\text{selfDCKey}, \text{GS}\_\text{hasDataSetDescriptor} \} \]

• The **Domain Definition Component** is a terminological component defined as \( T_d = (C_d, P_d) \) where:

\[
C_d = \{ \text{DomainEntity}, D\_\text{Entity}, D\_\text{Attribute}, D\_\text{KeyContainer}, D\_\text{Taxonomy} \}
\]

\[
P_d = \{ \text{DomainProperty}, D\_\text{hasEntity}, D\_\text{hasAttribute}, D\_\text{hasKey}, D\_\text{hasTaxonomy}, D\_\text{splitEntity}, D\_\text{unionEntity}, D\_\text{joinEntity} \}
\]

• Finally, the **Common Annotation Properties** \( A \), is defined as:

\[
A = \{ \text{hasCode}, \text{hasInternalCode}, \text{hasName}, \text{hasDescription}, \text{SourceAnnotation} \}
\]

Tables 4.1, 4.2 and 4.3 shows the details of classes, properties and annotations properties respectively. The \( ABOX^* \) is \( TBOX^* \)-compliant and is created when the model is instantiated into a given project. Thus, the instances in the *Definition Layer* ontology represents the schema and structure of the corresponding definitional components (Sources, Global Schema, Domain). In the following section, after introducing the mapping relationships, there are some examples to clarify the use of this definitions.
<table>
<thead>
<tr>
<th>Definitional Component</th>
<th>Class</th>
<th>Description</th>
<th>subclassOf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>SourceContainer</td>
<td>Groups all the DataContainer's in a source</td>
<td>SourceClass</td>
</tr>
<tr>
<td>Source</td>
<td>DataContainer</td>
<td>A dataset</td>
<td>SourceClass</td>
</tr>
<tr>
<td>Source</td>
<td>KeyContainer</td>
<td>Simple or composite keys in datasets</td>
<td>SourceClass</td>
</tr>
<tr>
<td>Source</td>
<td>DataAttribute</td>
<td>Any attribute of a dataset</td>
<td>SourceClass</td>
</tr>
<tr>
<td>Source</td>
<td>DataGroup</td>
<td>Class to create data groups</td>
<td>SourceClass</td>
</tr>
<tr>
<td>Source</td>
<td>SourceClass</td>
<td>Root of the source definition taxonomy</td>
<td>SETAClass</td>
</tr>
<tr>
<td>Source</td>
<td>RefContainer</td>
<td>container of references to other datasets</td>
<td>SourceClass</td>
</tr>
<tr>
<td>Source</td>
<td>SourceTechDescriptor</td>
<td>Technical descriptor of a given source</td>
<td>SourceClass</td>
</tr>
<tr>
<td>Global S.</td>
<td>GlobalSchemaClass</td>
<td>Root of the global schema taxonomy</td>
<td>SETAClass</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_DataSet</td>
<td>A dataset</td>
<td>GlobalSchemaClass</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_Attribute</td>
<td>An attribute of a data set</td>
<td>GlobalSchemaClass</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_DataGroup</td>
<td>Class to create data groups</td>
<td>GlobalSchemaClass</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_KeyContainer</td>
<td>Composite key</td>
<td>GlobalSchemaClass</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_RefContainer</td>
<td>References to other datasets</td>
<td>GlobalSchemaClass</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_DataSetTechDescr</td>
<td>Technical descriptor of a dataset</td>
<td>GlobalSchemaClass</td>
</tr>
<tr>
<td>Domain</td>
<td>DomainClass</td>
<td>Root of the domain taxonomy</td>
<td>SETAClass</td>
</tr>
<tr>
<td>Domain</td>
<td>D_Entity</td>
<td>A domain entity</td>
<td>DomainClass</td>
</tr>
<tr>
<td>Domain</td>
<td>D_Attribute</td>
<td>A domain Attribute</td>
<td>DomainClass</td>
</tr>
<tr>
<td>Domain</td>
<td>D_KeyContainer</td>
<td>A domain key container</td>
<td>DomainClass</td>
</tr>
<tr>
<td>Domain</td>
<td>D_Taxonomy</td>
<td>Class to model taxonomies</td>
<td>DomainClass</td>
</tr>
</tbody>
</table>

Table 4.1: Main SETA TBOX* classes.
<table>
<thead>
<tr>
<th>Def. Comp.</th>
<th>Property</th>
<th>Description</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>SourceProperty</td>
<td>Root property for sources</td>
<td>SourceClass</td>
<td>SourceClass</td>
</tr>
<tr>
<td>Source</td>
<td>hasSourceComponent</td>
<td>Groups the datasets of a same source</td>
<td>SourceContainer</td>
<td>DataContainer</td>
</tr>
<tr>
<td>Source</td>
<td>hasDataContainer</td>
<td>A data container including another DataContainer (for example for JSON nested documents)</td>
<td>DataContainer</td>
<td>DataContainer</td>
</tr>
<tr>
<td>Source</td>
<td>hasAttribute</td>
<td>A DataContainer including a single attribute</td>
<td>DataContainer</td>
<td>DataAttribute</td>
</tr>
<tr>
<td>Source</td>
<td>hasKey</td>
<td>DataContainer key definition</td>
<td>DataContainer</td>
<td>KeyContainer, DataAttribute</td>
</tr>
<tr>
<td>Source</td>
<td>hasReference</td>
<td>A DataContainer reference to another DataContainer in this source (for example, for relational sources, or related JSON documents)</td>
<td>DataContainer</td>
<td>RefContainer</td>
</tr>
<tr>
<td>Source</td>
<td>refDC</td>
<td>Reference DataContainer</td>
<td>RefContainer</td>
<td>DataContainer</td>
</tr>
<tr>
<td>Source</td>
<td>refDCKey</td>
<td>Reference DataContainer key (simple or composite)</td>
<td>RefContainer</td>
<td>KeyContainer, DataAttribute</td>
</tr>
<tr>
<td>Source</td>
<td>selfDCKey</td>
<td>Self DataContainer key (Simple or composite)</td>
<td>RefContainer</td>
<td>KeyContainer, DataAttribute</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_Property</td>
<td>Root property for Global S.</td>
<td>GS_Class</td>
<td>GS_Class</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_hasDataSet</td>
<td>A dataset including another dataset</td>
<td>GS_DataSet</td>
<td>GS_DataSet</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_hasAttribute</td>
<td>A dataset including an attribute</td>
<td>GS_DataSet</td>
<td>GS_Attribute</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_hasKey</td>
<td>Dataset key definition</td>
<td>GS_DataSet</td>
<td>GS_KeyContainer, GS_Attribute</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_hasReference</td>
<td>A dataset reference to another dataset</td>
<td>GS_DataSet</td>
<td>GS_DataSet</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_refDC</td>
<td>Reference dataset</td>
<td>GS_DataSet</td>
<td>GS_DataSet</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_refDCKey</td>
<td>Reference dataset key</td>
<td>GS_DataSet</td>
<td>GS_KeyContainer, GS_Attribute</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_selfDCKey</td>
<td>Self dataset key</td>
<td>GS_DataSet</td>
<td>GS_KeyContainer, GS_Attribute</td>
</tr>
<tr>
<td>Global S.</td>
<td>GS_hasDataSetDesc</td>
<td>Dataset tech. descriptor</td>
<td>GS_DataSet</td>
<td>GS_DataSetTechDesc</td>
</tr>
<tr>
<td>Domain</td>
<td>D_hasEntity</td>
<td>A domain entity including another domain entity</td>
<td>D_Entity</td>
<td>D_Entity</td>
</tr>
<tr>
<td>Domain</td>
<td>D_hasAttribute</td>
<td>A domain entity including a domain attribute</td>
<td>D_Entity</td>
<td>D_Attribute</td>
</tr>
<tr>
<td>Domain</td>
<td>D_hasKey</td>
<td>A key of a domain entity</td>
<td>D_Entity</td>
<td>D_Attribute, D_KeyContainer</td>
</tr>
<tr>
<td>Domain</td>
<td>D_splitEntity</td>
<td>A domain entity split relationship</td>
<td>D_Entity</td>
<td>D_Entity</td>
</tr>
<tr>
<td>Domain</td>
<td>D_unionEntity</td>
<td>A domain entity union relationship</td>
<td>D_Entity</td>
<td>D_Entity</td>
</tr>
<tr>
<td>Domain</td>
<td>D_joinEntity</td>
<td>A domain entity join relationship</td>
<td>D_Entity</td>
<td>D_Entity</td>
</tr>
<tr>
<td>Domain</td>
<td>D_hasTaxonomy</td>
<td>A domain entity taxonomy relationship</td>
<td>D_Entity</td>
<td>D_Taxonomy</td>
</tr>
</tbody>
</table>

Table 4.2: Main SETA TBOX\(^*\) properties.
<table>
<thead>
<tr>
<th>Annotation Property</th>
<th>Description</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasCode</td>
<td>External of standard code used for data exchange</td>
<td>SETAClass</td>
<td>rdfs:Literal</td>
</tr>
<tr>
<td>hasInternalCode</td>
<td>Internal code</td>
<td>SETAClass</td>
<td>rdfs:Literal</td>
</tr>
<tr>
<td>hasName</td>
<td>Name</td>
<td>SETAClass</td>
<td>rdfs:Literal</td>
</tr>
<tr>
<td>hasDescription</td>
<td>Description</td>
<td>SETAClass</td>
<td>rdfs:Literal</td>
</tr>
</tbody>
</table>

Table 4.3: Source Definition $TBOX^*$ annotation properties.
4.1.2 Mappings

Mappings establish relationships between data concepts in the three definitional levels and thus, implement the data transformations contemplated by SETA. There are two sets of mappings, one establishes relationships and transformations between the Source Definition Component and the Global Schema Component, and the another between the Global Schema Component and the Domain Component. The selection of transformations in both sets is key to articulate the engagement point that has been mentioned in the overview Chapter: the balance between the complexity of processes ‘Schema on Write’ and processes ‘Schema on Read’.

It is worth mentioning again that at domain level, SETA does not materialize data, it is simply a viewpoint of the Global Schema. The materialization of the data in the target occurs when the user sets the analytical requirements on the view provided by the Domain Component. Therefore, the final materialization of data and its transformations occur through a combination of existing mappings and the requirements posed by the user over the Domain Component. To separate the transformations between the two sets of mappings the following rules are applied:

- Projections (SELECT) are articulated by the existence or not of mappings between attribute components (simple or composed) of a given data set. For example, if there is no mapping from a given DataAttribute to a GS_Attribute, this attribute is not projected to the Global Schema.

- Referential relations and their unfold or compact transformations, are always implemented between the Source and the Global Schema.

- Filters that modify the number of data instances can be implemented both between the Source Schema and the Global Schema and between the Global Schema and the Domain Schema.

- Set operations (SPLIT, UNION) are implemented by the mappings between the Global Schema and the Domain Schema. JOIN operations are also implemented by the mappings between the Global and the Domain schemas.

- Grouping and aggregation constraint: SETA does not directly implement transformations involving aggregation or grouping data. These transformations must be specified into the Requirement Definition Component. This imposes a restriction on the expressivity of the Domain Component which will be unable to represent entities formed by grouping data from Global Schema.
The set of properties of SETA mappings is defined as $M = (M_{sg}, M_{gd}, C_{sg}, C_{gd})$ where $M_{sg}$ is the set of properties mapping between the Source level and the Global Schema level, $M_{gd}$ is the set of properties mapping between the Global Schema level and the Domain level, $C_{sg}$ is a set of classes for annotating $M_{sg}$ properties and $C_{gd}$ is a set of classes for annotating $M_{gd}$ properties, where:

$$M_{sg} = \{\text{SSGS\_directMapDS}, \text{SSGS\_directMapAT}, \text{SSGS\_compactRef}, \text{SSGS\_splitRef}, \text{SSGS\_filterMapDS}\}$$

- **SSGS\_directMapDS** establishes a direct mapping between a **DataContainer** and a **GS\_DataSet**.
- **SSGS\_directMapAT** establishes a direct mapping between a **DataAttribute** and a **GS\_Attribute**.
- **SSGS\_compactRef** compacts two **DataContainer** related by a **RefContainer** into a **GS\_DataSet** (ex: Relational store style to document store style).
- **SSGS\_splitRef** splits a composed **DataContainer** (a **DataContainer** containing another **DataContainer**) into two **GS\_DataSet** related by a **GS\_RefContainer** (ex document store style to relational style).
- **SSGS\_filterMap** establishes a direct mapping between a **DataContainer** and a **GS\_DataSet** but applying a filter attached to the property by annotating it with a class **SSGS\_Filter**.

$$M_{gd} = \{\text{GSDS\_directMapDS}, \text{GSDS\_directMapAT}, \text{GSDS\_filterMapDS}\}$$

- **GSDS\_directMapDS** establishes a direct mapping between a **GS\_DataSet** and a **D\_Entity**.
- **GSDS\_directMapAT** establishes a direct mapping between a **GS\_Attribute** and a **D\_Attribute**.
- **DSDS\_filterMapDS** establishes a direct mapping between a **GS\_DataSet** and a **E\_Entity** but applying a filter attached to the property by annotating it with a class **GSDS\_Filter**.
4.1.3 Definition layer examples

To clarify the formalization of definitions, some modelling examples of the silly data source of the figure 4.2 are included. It is assumed that the source is relational, or is a JSON source in which the Person and Addresses, documents are separated, i.e. Address is not included in Person documents. This example will be useful for the mapping examples transforming this schema to JSON compacted documents with the relations as lists in the document itself or also uncompacted, with documents related following the relational style. For the creation of the ontologies and diagram visualizations we use Protégé\textsuperscript{2}. We used bold text for classes and properties and emphasized text for instances.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{silly_data_source.png}
\caption{Silly data source used for the following examples}
\end{figure}

\footnote{The Stanford open-source Ontology editor \url{http://protege.stanford.edu}.}
Figure 4.3: Graph representation of the $ABOX^*$ corresponding to the dumb data source example

Figure 4.3 shows the Source Definition $ABOX^*$ of the previous example data source. The diagram shows only instances. A tab for decoding properties (edges) is included. The source is grouped into an instance of the class SourceContainer (DumbSource), which includes two DataContainer instances (Person and Addresses) through the hasSourceComponent property. Each DataContainer includes the corresponding attributes through the hasAttribute property and the keys through the hasKey property (in this case, simple keys with only one attribute). The relationship between Person and Addresses is represented by an instance of the RefContainer class (Addresses-Person_ref) related to Person by the hasReference property. This instance, includes the referenced DataContainer (hasDC $\rightarrow$ Addresses) the referent key (selfDCKey $\rightarrow$ Person-id) and the key in the referenced DataContainer (refDCKey $\rightarrow$ Addresses-personId). The SourceTechDescriptor instance is not included in this example.
As an additional example, let’s suppose now that the source is a JSON document in which the addresses are included as a list in the Person document. The figure 4.4 shows the representation in this case.

![Graph representation of the ABOX* corresponding to the dumb data source example. In this case, the source is supposed to be a JSON file with the addresses of a person included in the Person document.](image)

Figure 4.4: Graph representation of the ABOX* corresponding to the dumb data source example. In this case, the source is supposed to be a JSON file with the addresses of a person included in the Person document.

Figure 4.5 illustrates some mappings between the source and the Global Schema. In this example the source is specified according to Figure 4.4, i.e., it is a JSON source in which Addresses documents are included as lists in Person documents. In the case of Global Schema, the designer has decided Person and Addresses to be stored into separated collections with relational-like relationships between them. For clarity, only the mappings between instances of GS_Dataset and DataContainer are shown, the data attributes mappings have been hidden. The mapping between Person and GS_Person is tagged by the SSGS_directMapDS property. As seen in 4.1.2 Mappings this property is implemented by a direct mapping from the source DataContainer class to the GlobalSchema GS_Dataset class. When the processor (wrapper) reaches the embedded DataContainer Addresses, it determines that the mapping is defined by SSGS_splitRef property that tells...
the wrapper that is a split type mapping with the relationship details specified by the corresponding GS_RefContainer. This allows the wrapper to determine the mappings for obtaining the foreign key to be included in the Global Schema GS_DataSet GS_Addresses.

Figure 4.5: Example of mappings transforming a data source with embedded documents into related relational-like documents in the Global Schema.

Figure 4.6 shows the opposite case. A source with relational organization is mapped in the Global Schema compacting the referenced entity Addresses in Person documents. In this case, the mapping between Addresses and GS_Addresses is defined by SSGS_compactRef property, which tells the wrapper Addresses occurrences to be compacted as lists into GS_Person.

Figure 4.7 illustrates an example of mappings between Global and Domain schemas, and the use of D_unionEntity and D_splitEntity properties. In the first case (left) a Person dataset exists in the Global Schema, let’s assume that there is an additional property indicating if the person is a teacher or a student. The Metadata Extractor created an D_Entity D_Person into the Domain Schema connected with the GS_Person with a GSDS_directMapDS and let’s suppose that the user wants to have in the Domain a D_Teacher and a D_Student entities representing students and teachers. He/She must create the D_Entity D_Student
and \textit{D\textunderscore Teacher}, relate them to \textit{D\textunderscore Person} with a \texttt{D\_splitEntity} relationship and create two new \texttt{GSDS\_filterMapDS} mappings annotating them with the corresponding filter class.
4.1.4 Analytical requirements

The Domain Definition Component exposes the user a view of the domain in terms of concepts and business relationships and the mappings link these concepts to the schema stored in the Global Schema. Through the integrated GUI, the user defines directly analytical requirements over the Domain Schema. Different users have different analytical requirements, both regarding the data to be analyzed, as the format adapted to the tools used for analysis. Some users will need to create MD models for analysis using OLAP tools, while others will require data extractions to feed data mining tools. Even the same user may need to obtain different views and models.

Therefore, to organize the presentation, storage and processing of requirements, SETA allows the definition of different Business Views. The analytical requirements must be always defined in the context of a specific Business View. For simplicity, a Business View is always associated with a single analysis model. In its current definition, SETA provides three kinds of models 1) OLAP models, 2) data mining models and 3) data extraction models. Also, for simplicity it will be assumed that the definitional artifacts for extraction and data mining models are the same. Each Business View, stores a set of requirements, independent or not (always associated with the same analysis model type). The task of integrating the different requirements of the same Business View is delegated to the Schema/ETL automation component. For each Business View, the Schema/ETL automation component generates the data and target schema, while the Target Metadata Generator, generates the metadata (descriptions, exchange codes, helps, etc.) obtained from the Domain Component in the format the analysis tool expects. Both the
Figure 4.8: Requirement Elicitor Component. The final user poses analytical requirements over the domain view. Different Business Views can coexist in the system. The user express the analytical model type and the data requirements on each Business View.SETA automates the schema creation and the ETL for data extraction according the final tools used.

Schema/ETL automation, as the Target Metadata Generator, need technical information to determine the encoding formats for metadata, ETL, DLL, DML, etc. Therefore, the Analytical Requirement Component should provide the metadata artifacts to define such technical properties. All the definitions of the requirements are stored in a XML to ease the interconnectivity of the different modules. For example Quarry [6] uses xLM [18][17], an XML-like format to encode the requirements. Figure 4.8 shows a graphical schema of this process.

Finally, for each kind of analytical model the following main definitional artifacts are available:

- OLAP cubes
  - Technical information of the end-user tool used (connection identifiers and JDBC descriptors, Relational engine, etc)
  - Set of MD requirements
    * Measures/facts (Selected over the domain Schema)
    * Analysis dimensions (Selected over the domain Schema)
    * Additional attributes
    * Slicers (Filters over dimensions)
* Additional filters

- OLAP cubes
  - Technical information of the end-user tool used
  - Extraction format (XML, CSV, JSON, etc)
  - Set of variables
  - Filters
  - Variable transformations. Some variable transformations that are useful for data mining. Data Mining tools use to have powerful variable transformation facilities, so the idea is not to overlap with them. Nevertheless there are some transformations that have sense to be disponible at the extraction point. For example to transform a categorical variable into a set of dummy binary variables when the mining models require numeric vectors. As the categories are registered in SETA, it is easy to provide this kind of transformations

4.2 Automation layer

The automation layer includes the predefined modules that make possible the automation. It includes the following components:

- **Schema/ETL automation.** It is one of the fundamental parts of the system and contributes to SETA automation capabilities of the target schema design and the ETL that feeds it. Instead of developing this functionality from scratch, the propose for SETA is to use as starting point Quarry. Quarry is the result of 4 years of previous works in the UPC BarcelonaTech in the field of multidimensional design and ETL automation from authors like Petar Jovanovic, Alberto Abelló, Oscar Romero and Alkis Sitnikis among others [1] [4] [18] [5] [6]. Quarry automates the physical design of a DW system from high-level information requirements. Moreover, Quarry provides tools for efficiently accommodating MD schema and ETL process designs to new or changed information needs of its end-users and facilitates the deployment of the generated DW design over an extensible list of execution engines.

Nevertheless, the explicit declaration of the different levels of data, provided by SETA allows that quarry can greatly be simplified and can work fully automatically. Briefly, the changes proposed to integrate this tool into the SETA framework are the following:
The Quarry component ‘Requirements Interpreter’ maps user requirements to data sources by identifying the multidimensional concepts (facts, dimensions, levels) by exploring the ontological definition of sources. The Requirements Interpreter ‘proposes’ to user the associations of identified concepts. This behavior can be modified and simplified in SETA since all concepts are explicitly stated in the Definition Layer, so the Requirement Interpreter can directly build the logical DDL (Data Definition Language) and ETL identifying concepts in the Domain Definition component (requirements are expressed through concepts at this level) and following the mappings to the Global Schema Definition Component and from here, to the Source component. It is recalled that this work was previously done when sources were included into the framework.

The components that could be maintained with minimal changes are 1) The ‘Design Integrator’ to integrate multiple requirements into a single MD and ETL design. The Design Integrator, should integrate the requirements corresponding to a single Business View (See 4.1.4 Analytical requirements), and 2) The ‘Design Deployer’ to generate DDL and ETL for a particular execution platform.

Quarry does not interpret requirements corresponding to data extractions for data mining tools (data matrices), so a new component must
be developed.

– Finally, the complete redesign of its graphic interface "Requirement Elicitor" integrating it into the SETA GUI is also proposed.

• **Target metadata generator.** Is the module responsible for generating the metadata contained in SETA using metadata exchange standards in order to achieve compatibility with many existing analysis tools on the market. SETA generates the target schema design and extraction processes based on the requirements from the final users. These target schemas will be exploited by external tools (front OLAP reporting tools or data mining tools) outside the scope of SETA. Although these external tools are an essential part of the analytical pipeline (see Figure 1.1), they are not integrated within the platform. Because of its heterogeneity and continuous evolution, this would not be feasible.

Still, there are several metadata exchange standards that could allow coupling the processes of data generation and data analysis. Indeed, SETA contains rich data definitions that are potentially useful for its semantic interpretation. Metadata generation is not currently covered by Quarry, so the proposal is to add a module to the framework to enable the generation of metadata structures tailored to the specific target schema that is to be generated. For example for the generation of multidimensional schemes, a good alternative is the *Warehouse Common Metamodel (CWM)*\(^3\) while for the generation of metadata for data mining tools, a good option is the *Statistical Data and Metadata eXchange standard (SDMX)*\(^4\). Both are widely extended standards, therefore, compatible with many market tools.

• **Source Manager and Monitor.** It is simply a module that allows to manage and monitor the results of processing the incremental load of the sources into the Integration Schema, i.e. the operation of the source wrappers. Wrappers process the data source according to its structural definition in the 'Source Definition' and store the data into the Integration Schema according the 'Integration Schema Definition'. These definitions include different data validation rules. For example, range validations or validations of the references to other datasets in the Integration Schema. The wrapper launches such validations and rejects the records whose validation fails. These errors must be annotated for further review by the user or the system administrator, corrected and annotated for processing again when the prob-

---


lem that violates the validation rule has been fixed (for example, a reference with non-existing target).

Additionally, the source manager annotates into an internal database the load files from the source and manages their status in the pipeline process (process pending, successfully processed, processed with errors, etc). It also provides some statistics dashboards about the processing as well as a front-end to display the processing errors and modify the status of erroneous records so they can be processed again. Figure 4.10 shows a prototyped screen for managing errors of a given source.

4.3 Processing Layer

The Processing Layer contains the processes that actually move the data between the different levels of the Data Layer. The processes at this level are domain-specific of a given instantiation of SETA through its Definition Layer. The processes contained at this layer are: 1) the set of generated ELT processes, 2) the source extractors and 3) the source wrappers.

- The Generated ETL is produced automatically by Quarry. The module ETL Process Integrator generates a conceptual definition of unified and integrated ETL derived from each individual analytical requirement ETL.
Such conceptual definition is encoded in an XML-like format, namely $xLM^5$. From this conceptual definition of unified ETL, the Quarry Deployer Design module, generates the ETL code for specific target platforms$^6$.

- **Extractors** are source-dependent processes that get data from the sources using the mechanisms they provide (API’s, exposed schemas, etc.) through a policy of incremental extraction. Extractors annotate the data load allotment into the control database of the Source Manager and Monitor SETA module for processing.

- **Wrappers**, process the pending data load allotments. They process the data in a metadata-driven way according the definitions of both Source Definition and Integration Schema Definition in the Definition Layer. They perform a transformation of scheme and format driven by the mappings from the source to the schema and format of the Data LakeHouse in such a way that the changes in the format and schema of the source (or the Integration Schema) are accommodated without modifying the wrapper.

### 4.4 Data Layer

Basically, this layer contains the database engines that supports the Global Schema. The current definition of the framework supports a document store (MongoDB) and a relational databases (PostgreSQL). The relational database is used to store data containers that have a know and static schema, while the data containers that are supposed to have a changing schema and structure, are physically stored in the document store engine. The schema is represented by the Definition Layer using the artifacts described in the previous section. This layer also contains the technical information about the physical storage of each data container.

---

$^5$See Integrating ETL processes from information requirements$^{[18]}$ for details.

$^6$See Engine independence for logical analytic flows$^{[17]}$ for details.
Chapter 5

SETA Instantiation

5.1 The WISCC project

The Chagas Disease (American trypanosomiasis) is one of the 17 Neglected Tropical Diseases (NTD) identified by the World Health Organization (WHO) in its First Report on Neglected Tropical Diseases made in October 2010. The first step towards the realization of this project was materialized into the final degree project from Jaume Viñas, who in collaboration with the Department of Neglected Tropical Diseases from WHO, the Department of Service and Information system Engineering and the Centre de Cooperació per al Desenvolupament from the UPC BarcelonaTech, developed the first functional specification of the system. This report has good and detailed contents about the Chagas Disease, the context of NTD, and the objectives, initiatives and programs within the WHO to address the challenges these diseases poses to nowadays societies, both at endemic and non-endemic countries [30].

In June 2013, the Tricycle Strategy of the Programme on Control of Chagas disease, WHO Department of Control of Neglected Tropical Diseases, was presented. The Tricycle Strategy is based on two power wheels: interruption of transmission and care provision of affected population and a steering wheel: an information and surveillance system. The building of this system has an important additional value: to raise awareness on Chagas disease, specially facilitating access to interactive data, disease statistics, maps and diagrams. This milestone, marks the birth of the World Information System on Chagas Disease Control (WISCC). In addition to collecting relevant information associated with the disease worldwide, the system should provide analytical capabilities to analyze it to different types of users, from multiple business perspectives and provide information to other WHO information systems. In this sense, one of the key pieces that should incorporate
the WISCC, was a data warehouse system. Figure 5.1 is a context diagram of the system what shows some of the users and client systems of the WISCC.

![Figure 5.1: Context diagram of the World Information System on Chagas Disease Control (WISCC).](image)

Although during the first phase of the project extensive work was done to identify and formalize the information the system had to manage, soon, the characteristics we have discussed in the section corresponding to the analytical environment, become evident. In addition, being this a long-term project, it was clear that both the analytical needs, as the available data sources could not be precisely determined at the time of design. Therefore, during the early stages of the implementation it was decided to change the design approach, adopting a more flexible strategy for both data capture information, such as for its storage and analysis.

In addition, the number of data sets and variables that should manage the system was very high, and this, only for the information to be captured manually
through the data entry. Although there was an initial definition of these datasets and variables, and a purpose of the protocols for data entry and validation, during the early stages of the implementation, we found that these sets of variables and validation protocols were not at all closed, they constantly evolved and changed. This fact, also determined decisions when building the front-end for manual data entry. Instead of building a system from scratch, we decided to use existing data capture systems in which the modification of the structure and content of the datasets were easily modified without changing much code. For example, we decided to use District Health Information System 2 (DHIS2) as data capture tool. Figure 5.2 shows an overview of the information handled by the data entry subsystem. There are different categories of information according to its origin (official, non official), its nature (individual, collective, estimated), and its domain area (normative, systemic, healthcare, transmission interruption). The bubbles of the right part of the diagram are the datasets that include the variables and measures.

The need to give a satisfactory response to the challenges posed by the construction of the WISCC data warehouse inspired the conception and design of the analytical framework described in this paper. Instead of raising an adhoc solution tailored to the needs of a particular project, it was decided to design a general analytical framework that could be used in analytical projects with similar challenges. Therefore, this chapter focuses on the instantiation of SETA framework in the WISCC project. The first year of the implementation phase of the WISCC focused on the construction of the data entry, defining all entry forms and data flow validation mechanisms, so the only source that has been addressed in depth is the DHIS2 system, which is the application used to capture this data. Go into the details of this tool is not relevant to the purpose of this document, so it include only the most important characteristics. In addition to data entry, DHIS2 provides many other functionalities such as dashboards design, pivot tables development, reports, maps and different types of analytical charts. However, in the context of WISCC what is used is its ability to define datasets, variables, entry forms and validation rules. DHIS2 is a metadata-driven system, i.e., it does not provide functionality particular domain, this is defined by its parametrization. Therefore, in a DHIS2 project, efforts focus on the definition of the data, its organization into datasets, and the definition of the organizational structure that articulates the collaborative data entry at a given geographic area level (in the case of WISCC a global scale).

Unfortunately, we can not show a complete instantiation of the framework because at the present time only the part corresponding to integrate some sources

---

1The District Health Information System 2 (DHIS2) application: https://www.dhis2.org.
Figure 5.2: Information structure of the WISCC data capture subsystem. The information is organized in different levels according to its nature, cardinality, domain area, etc. The bubbles in the right part represent the final datasets that include the variables and measures captured by the system.
into the Global Schema has been developed. So, the next sections show some
details about the creation of the Definition Layer and the creation of the extractors
and wrappers and its control structures.

5.2 The Definition Layer

In its current definition, the information from the data entry consists of about
400 variables, structured in 35 different datasets and 3 different categories grouping
the datasets according to their origin (official, unofficial), their nature (individual,
collective and estimated) and their domain area (normative, systemic, healthcare
and transmission interrupt). As seen in previous chapters, SETA instantiation
requires creating the $ABOX^*$. Since there are three levels of definition, roughly,
we are talking about 105 ($35 \times 3$) data containers or entities, and 1200 ($400 \times 3$)
variables or measures. Clearly, the manual creation of the $ABOX^*$ is not a suitable
option. Also, being the metadata already defined in DHIS2, the logical alternative
is to automate the initial creation of the $ABOX^*$.

Given that DHIS2 allows
the exportation of its defi-
nitions in JSON format, it
has built a program that
reads the JSON definition
file and the SETA $TBOX^*$
OWL file (DHIS2 metadata
extractor), and automatically generates the SETA
$ABOX^*$. This program
generates all three levels of
definition (Source, Global
Schema and Domain) with
the same structure defined in DHIS2 and assuming certain premises by default.
Therefore, once created the $ABOX^*$, it is necessary to manually adjust the def-
inition to fit the Global Schema preferences from the designer and the end-user
customization. The program also generates the mappings between the three lev-
els of definition. By default, each of the datasets is defined to be stored into a
$Document\, Store$ collection ($mongoDB^2$ in this case). It has been implemented in
JAVA and uses the $JSON\_simple$ libraries for the treatment of JSON documents

\begin{figure}[h!]
\centering
\includegraphics[width=\textwidth]{figure5.3.png}
\caption{DHIS2 metadata extractor}
\end{figure}

\footnotesize

\textsuperscript{2}mongoDB, a document store database: \url{https://www.mongodb.com}.
and JENA\(^3\) libraries for the treatment of OWL.

### 5.3 Extractor and wrapper

The flow and structure of operation of the extractor and the wrapper is summarized in the Figure 5.4. Once the definition layer is created, SETA requires the development of the source extractor and wrapper. In the case of the DHIS2 source, the system provides an RESTful Web Services API to extract the data stored in its proprietary database. The API provides the data in JSON format. The extractor is also a RESTful service developed in JAVA that gets the data stored in DHIS2 incrementally, i.e., on each execution extracts the datasets whose creation or update timestamp is posterior to the last extraction timestamp. On each execution, the extractor generates in the WISCC input box a JSON file with the data and annotates the file into the WISCC control database with status ‘process pending’ to be processed later by the wrapper. This operation allow the extractor and the wrapper to be executed separately, i.e., asynchronously. When the wrapper starts, it obtains the pending files annotated in the WISCC control database, it read the file from the input box and processes it. Therefore, SETA stores both, the raw source data files and the processed data into the global schema.

The wrapper processes the DHIS2 JSON datafile and stores it into the document store database according the Definition Layer Components. It reads the source according the structure specified by the Source Definition Component and stores the data according the Global Schema Component and the mapping transformations. The process is completely ABOX\(^*\)-driven in such a way that any change in the source structure (new attributes, new structures, etc) is detected. The wrapper annotates the records with errors in the WISCC control database, creates a new data file with all the records that raised some error and stores it into the input box annotating the original datafile with processing status ‘processed with errors’ and the file with the erroneous records with processing status ‘error fixing pending’. This way, the system administrator can review the errors, fix them and mark the file of erroneous records to be processed again in the next execution of the wrapper.

---

\(^3\)JENA, the open source Java framework for building Semantic Web and Linked Data: [https://jena.apache.org](https://jena.apache.org).
Figure 5.4: Extractor and wrapper of the DHIS2 source. The processes work asynchronously. The extractor gets data from DHIS, store the data file in the input box and annotates the file. When the wrapper starts, processes all the pending files, annotates the results and stores the data into the document store.
Chapter 6

Conclusions

This thesis introduced SETA, a Suite-independent Agile Analytical Framework to support the complete analytical pipeline in business environments where speed and flexibility are key issues for the success of organizations. The working methodology is organized by defining a set of 4 pillars or design guidelines that articulate the framework construction. These pillars are: user-centric approach, automation, Model Driven Architectures and common to all, metadata and semantic aware definitions. These pillars drove the searching of recent works in the field of DW systems and frameworks supporting analytical pipelines. This review provides an overview of the state of the art in the field that has served for the SETA design.

SETA has a fully practical approach. In fact, its conception is a response to the challenges presented by a real-world project: the World Information System on Chagas Control (WISCC) of the World Health Organization (WHO). However, the approach that we have followed is not a specific solution for a concrete project. As seen, SETA has a powerful Definition Layer based on ontologies. In this sense the concepts and properties of the ontological component ($TBOX^*$), establish a metamodel for a general analytical framework that is instantiated into a particular project by creating the ontology assertive component ($ABOX^*$). This document used the superscript '*' to differentiate terminological and assertive SETA components from its most widespread definition: TBOX as concepts and properties and ABOX: instances. In SETA instances are not data instances but schema instances, i.e., entities, attributes and relationships belonging to a particular domain.

The classical structure of DW systems, despite its maturity and robustness does not adequately deal with the analytical needs of today’s organizations. Regarding the logical schema of the DW, the approaches based on a predefined design using the relational model does not provide sufficient flexibility to accommodate new data sources and new analytical requirements. Similarly, relational data sources,
must coexist with new data sources with changing structure and represented with much more flexible schemas. The representation of both, data sources as business domains, by semantically richer models like those based on ontologies, provides great flexibility and expressivity. However, these conceptual models should be able to be instantiated into logical data models to store, manipulate and visualize the data using the widely used and essential tools in any analytical pipeline (OLAP tools). In addition, data mining tools are increasingly used in organizations, so the infrastructure that supports the analytical pipeline should also generate data to feed these tools with the same criteria of agility and flexibility than for generating multidimensional models.

The most recent approaches dealing with MD design and ETL automation using ontologies aim to automate the ETL from the raw sources to the target schema. The user poses the analytical requirements over an ontologic definition of the sources and generates both the ETL and the MD target design. However, this approaches do not contemplate the existence of a Global Schema integrating the sources and do not deal with the practical problem of source treatment and processing. Therefore, we argued that an Global or Integration Schema between the sources and the Domain is need.

Other approaches deal with the flexibility and agility upon changing requirements and data sources with a different strategy. This is the case of Data Lakes, where the sources are stored raw into a schemaless data store. The schema is created on the fly according the analytical requirements. This ‘Schema on Read’ strategy is the opposite of the ‘Schema on Write’ strategy of the classical DW approaches. The approach of SETA to define the data layers and his structure and schema, is positioned into an intermediate point on the spectrum of analytical frameworks between classic DW systems and Data lakes. It applies both approaches ‘Schema on Write’ and ‘Schema on Read’, so the complexity of the source processing and the data extraction processes is balanced. Therefore, this intermediate or engagement point is a key issue.

SETA also generates the target schema and data on the fly but provides a materialized Global Schema to process and integrate the data sources. This Global Schema is stored in flexible database schemas (Document Stores) but it provides Definition Components for sources, Global Schema and domain to expose the schemas with the expressivity required at each level. We call this strategy the Data LakeHouse, a intermediate solution between classical DW and data Lakes.
SETA is based on a Definition Layer that includes three main definitional components 1) the schema components (Source, Global Schema and Domain) are specified by subontologies with concepts and properties adapted to each level and with different purposes. While Domain ontology focus in the description of the concepts and relations of the domain, the Source and Global Schema ontologies focus on the structure and schema of the data, 2) the mappings between the domain and the global schema and between the sources and the global schema. The mappings are a subontology with a set of classes and properties to specify the data and schema transformation between the three schema levels and 3) the Analytical requirements component which provides the high level analytical requirements posed by the user on top of the Domain Schema. The definition of the classes and properties of the SETA ontology determines the engagement point between classic DW systems and Data Lakes mentioned in the previous paragraph.

According to the Model-Driven Architecture pillar, the SETA Definitional System establishes a analytic framework metamodel by providing the $TBOX^*$ (Analytical concepts, classes and properties) of the ontology. The model is created by instantiating the definitional metamodel into a given domain which produces the $ABOX^*$ statements of the ontology. The definition of the OMG (Object Management Group) for model-driven engineering, identifies 4 layers of model abstraction: M0) instance layer, is where objects reside in the real world, M1) model layer, is where models live for a representation of a part of reality, M2) metamodel layer, contains the tools that M1 are built with, and M3) meta-metamodel layer that provides the tools and structures to define meta-models M2. Therefore, in this context SETA models provided by the Definition Layer are positioned in the M2 and M1 levels.
Bibliography


[23] Marie-Aude Aufare et al. An Envisioned Approach for modelling and supporting user-centric query activities on data warehouses. international Journal on Data Warehouse and Mining, 9(2), 89-109, April-June 2013


[31] Diana Marcela Sánchez, José María Cavero, Esperanza Marcos. On models and Ontologies.


