DEFINITION OF A HYBRID PROGRAMMING LANGUAGE FOR INTEROPERABILITY OF HETEROGENEOUS SOFTWARE SYSTEMS AT THE SEMANTIC LEVEL

GUILLERMO GONZÁLEZ-CALDERÓN

Thesis submitted as partial requirement to obtain the degree of Doctor of Engineering.

Advisor:
Ph.D. Carlos Mario Zapata Jaramillo – National University of Colombia

Doctoral committee:
Ph.D. Andrian Marcus – University of Texas at Dallas, USA
Ph.D. Gloria Lucía Giraldo. – National University of Colombia, COLOMBIA
Ph.D. Oscar Pastor López – Polytechnic University of Valencia, SPAIN.

GRADUATE PROGRAM IN SYSTEMS (COMPUTER SCIENCE)
FACULTY OF MINES
NATIONAL UNIVERSITY OF COLOMBIA
MEDELLIN CAMPUS
2015

DEDICATION

To my father, who has always been there supporting me

Guillermo

ACKNOWLEDGMENTS

First, I want to thank my Father, Mother, and Brother for their support and advice in this chapter of my life. I also want to highlight all the help, guidance, dedication, and support from my Advisor during all these years.

I also want to thank those friends and coworkers who supported and helped me somehow: Juan David Velásquez, Gloria Giraldo, Kate Fisher, Paula Berrío, Roberto Manjarrés, Gloria Gasca, Bell Manrique, María Clara Gómez, Gloria Carvajal, Jovani Jiménez, John William Branch, Juan Fernando Vélez, Germán Sánchez, Jairo Ortíz, and all those people who one way or another contributed to the achievement of this goal.

TABLE OF CONTENTS

TABLE OF FIGURES

ABSTRACT

In Information Technology, interoperability is the ability of different software systems to communicate, exchange data, and work together. At the semantic level, interoperability can be used for analyzing, sharing, and comparing data with intended meaning, from different data sources in different domains. Several models, architectures, data structures, and programming languages may be used for developing software applications for a company; such diversity usually leads those heterogeneous systems to be incompatible for carrying out some tasks related to semantic interoperability, such as data comparison. In order to complete those tasks, several applications, standards, and languages are used. However, some problems still remain: the need to build one-to-one system solutions, the impediment of converting existing systems into new standardized ones, the complexity of the existing solutions, and the difficulties of the reading/writing process into both the source and the target systems guaranteeing data consistency. For these reasons, in this Thesis we propose the definition and formalization of a hybrid programming language which includes the

main operations for achieving interoperability of heterogeneous software systems at the semantic level.

The main contributions of this Thesis are summarized as follows:

- The definition of a set of rules for managing common data from different systems, in particular domains, from XML documents.
- The capability of defining correspondences of records from different data sources, based on specific domains, considering the data semantics.
- The simplicity of analyzing similar data for determining inconsistencies among heterogeneous software systems, by using a domain-specific programing language.
- The specification of a hybrid programming language for achieving interoperability at the semantic level.
- The implementation of the above defined elements into the Semantic Interoperability Language—SIL—a new interoperability programming language applicable to different domains at the semantic level.

We make these contributions in order to:

- Define equivalencies of records in different data sets within specific domains, to compare and analyze common data among heterogeneous software systems from XML documents.
- Facilitate inconsistency identification among common data sets stored in different information systems, taking into account the data semantics.
- Give the user the possibility of using a hybrid programming language to interoperate between two heterogeneous software systems at the semantic level. The hybrid programming language is intended to be used instead of the combination of several programming languages to achieve the desired results, making the process easier.

The results of this Thesis lead to the following future work:

- Automated identification of data inconsistencies, made by means of SIL.
- Addition of new data structures and rules to the SIL specification, in order to make it widely used.
- Direct modification of data sources in heterogeneous software systems.

1. INTRODUCTION

Interoperability can be defined as "the capability of the software product to interact with one or more specified systems." (ISO/IEC, 2001). Interaction is only a facet of interoperability. In fact, this capability is also linked to business processes like sharing, analyzing, and modifying information of a set of systems.

Software systems can be heterogeneous in nature, because they can be built by using different models, architectures, data structures, operating systems, and programming languages, among other factors. Organizations need solutions to exchange and update their data, which may be spread over different information systems and are specific for different domains. Thus, organizations need interoperability for sharing, analyzing, and modifying the information stored in their software systems.

When we talk about managing data from different repositories of common domains, we are referring to the semantic level of interoperability, in both the metadata (structure) and the data themselves. Data can be stored in several formats and within several systems, and sometimes such data can be duplicated or missing. Semantic interoperability is, consequently, the ability of dealing with common data by intercommunicating software systems with the intention of reaching data consistency in specific domains, guaranteeing no missing or extra data, and equivalent data sets. For analytical purposes, this process is complex and overwhelming.

Some work has been devoted to solve the main problems related to semantic interoperability: middleware systems and ontologies have been the most common artifacts used for data comparison and interaction between systems, in order to define common data sets. In addition, data warehouses and international standards have contributed to the process of gathering data into repositories and new software systems for data analysis (Widom, 1995). Web services and query languages have promoted the joint use of data among heterogeneous software systems for achieving data consistency. However, some problems still remain:

- Some solutions may need to be built into a one-to-one basis for all pairwise systems, and this can lead to a burden for the analysts. For example, if we need to use either a middleware system or a web service to interoperate a specific pair of applications, such a solution might not be suitable to other pairs of systems, because of the structure and data models of the applications.
- There are some approaches that guarantee semantic interoperability, but in practice this is not always the case; for example some methods are recommended for guaranteeing semantic interoperability when developing new software systems (*e.g.,* international standards such as Model Driven Architecture—MDA and ontologies) or when merging information from several software systems into a single repository (*e.g.,* data

warehouses) (Manconi & Rodriguez-Tomé, 2011). However, heterogeneous legacy software systems, who might not been developed using such approaches, are typically needed and used as independent applications by companies, and replacing such systems with new systems developed using the new standards, might not be a feasible option.

- Some solutions are not capable of writing data into the software systems we intend to interoperate; updating data in the source systems might be necessary to guarantee data consistency. Data warehouses are examples of this situation, since they are capable to retrieve information from the source software systems, but they are unable of writing information directly into such systems (Theodoratos & Sellis, 1997).
- There are no simple approaches for the semantic analysis of heterogeneous data sources in common domains, where you can define both data equivalences and correspondences, and use them to interoperate such systems.

The mentioned problems lead us to propose a way to achieve semantic interoperability between two heterogeneous systems, where the user doesn't have to be an expert on computer science. We tackle this by defining a new programming language with hybrid properties, making it more intuitive for the user. With a domain-specific programming language we can provide specific functionality for semantic interoperability. We need to use both declarative and procedural paradigms for defining the hybrid programming language, because we want to exploit the expressive power of query languages (declarative in nature), and we need the ability of using sequences of commands (procedural languages) for repetitive work. The definition and formalization of such hybrid programming language include the specification of the main operations for achieving interoperability at the semantic level at specific domains.

This Ph.D. Thesis is structured in the following way: Chapter 2 is devoted to explain the main concepts needed to understand the problem; a state-of-the-art review related to semantic interoperability is presented in Chapter 3; the proposed solution is presented in Chapter 4; in Chapter 5 we present a case study and, finally, in Chapter 6 we show the results and future work.

2. THEORETICAL FRAMEWORK

2.1. Heterogeneous Software Systems

Organizations tend to change over time: personnel come and go; companies open new departments and branches to reach other markets, close the ones that are not needed / profitable anymore, and upgrade existing ones to meet the current needs and trends. Those variations demand changes in the way companies process their information; they need to create/upgrade their software systems to deal with those new requirements, and also need to integrate the new systems with the existing ones. Thus, it is expected for organizations to have heterogeneous software systems for managing different business processes. Typically, those applications are developed at different times, depending on the specific needs of the company at each moment. Therefore, companies tend to have heterogeneous software systems, as developers change over time and their products may vary from the existing ones; they develop applications according to their perceptions and preferences, including the current trends in software development, being very likely for the companies to have heterogeneous systems at all times.

When users perform transactions, they usually access several computers and systems distributed across networks with heterogeneous technology and support systems.

Software systems can differ in several aspects like platforms, operating systems, architecture, and even in the versions of the required libraries for running the programs (Young, Berzins, & Luqi, 2002). For example, applications compiled on a 64-bit system should exhibit problems when they are executed on a 32-bit system. If an application was developed to be executed on machines using a Linux Operating System, it is unable to run on a computer using Microsoft Windows®, unless there is a middleware solution. The version of the operating system may also affect the compatibility of software applications. For example, software systems developed to run on Microsoft Windows 7® could be unable to run on a previous version of the same operating system family such as Windows XP®. The versions of the programming languages could differ and its libraries could be incompatible with prior or future versions of the same programming languages as well. This fact leads to problems in compiling the source code or executing existing programs.

Software developed to be run on mobile devices is another example. Native applications have their own architecture, storage systems, programming languages, libraries, etc. Usually, different platforms are not compatible with each other: an application developed for IOS® is incompatible with an Android® device and vice versa.

With such a variety of systems and their data resources, there are several factors that could make software systems heterogeneous: incompatibilities between operating systems and hardware (*system*), differences in representation and encodings (*syntactic*), variances in data structures, models, and schemas (*structural*), and inconsistencies in terminology and meanings (*semantic*) (Ouksel & Sheth, 1999).

2.2. Semantic Interoperability

Typically, enterprises have several heterogeneous software systems sharing common information in specific domains. Personnel and automated processes need to interact with such data, no matter what systems they come from, guaranteeing their consistency. Such requirement is essential to perform day-to-day tasks, to ensure data quality, to make good decisions, and to improve the company workflows.

In order to guarantee data consistency, different information systems need to be consistent with the meaning of their intended data in their domains. Semantic interoperability is defined then as the capability of software systems to exchange information and interpret it properly with their intended meaning, based on specific domains, avoiding ambiguities in interpretation of data (Manconi & Rodriguez-Tomé, 2011).

Data is classified into different domains. For example, the payroll system manages different information than the customer services system. However, they could have common information related to the same subjects, but with different name conventions, making data processing among systems even more difficult. Think about the information of employees. In one system there could be a database table with a "salary" field, and in another system that field could be named "pay". They refer to the same thing but have different names. In this case, it is important to take into account the semantics for data comparison and analyses.

Software systems could have different information too, even when they refer to the same data structure and semantics: what happens if the personal information of the employees of a company is stored in two different systems? If one record is updated on a system and the other record—which contains data about the same employee—is not updated, inconsistency arises. If there is a version control system, you can easily determine which record was updated last, and then you might know which system holds the current record. However, if there is no such a version control system, you would not know for sure which record is the "correct one".

Semantic interoperability can be used to integrate different software applications by exchanging data, and modifying either the source or the target software system after finding inconsistencies in common domains. If two software systems were developed to work on the same platform with the same architecture, storage structure, and operating system, communicating them is trivial. But if the applications were built on different platforms or operating systems, with specific data structures, compatibility problems arise. In this case, we need to find a way to interact with these heterogeneous systems. The need for interoperability among software systems is supported by the exchange of information as a key to achieve the goals of the companies and by the need of consistency of the information in specific domains, taking into account semantically related information.

Syntactic interoperability is the ability of two systems to exchange data and communicate. To achieve this, it is required to have common data formats and to adopt communication protocols, being then a pre-requisite to semantic interoperability (Manconi & Rodriguez-Tomé, 2011). The way applications store data should be considered before exchanging information. Many proprietary formats in the information systems differ from each other, making difficult the way to retrieve information from them, because they are very likely to be incompatible. A common data format or data structure is required to guarantee the information to be interpretable from the defined structure. For example, SQL and XML standards provide syntactic interoperability allowing detection of syntactic errors, offering features for dealing with structured data.

2.3. Extensible Markup Language—XML

The Extensible Markup Language–XML is derived from the Standard Generalized Markup Language—SGML (ISO/IEC, 2006) with flexible text format. It is defined by the XML 1.0 Specification (World Wide Web Consortium, 2008) produced by the World Wide Web Consortium—W3C, and other standards (Bikakis, Tsinaraki, Gioldasis, Stavrakantonakis, & Christodoulakis, 2013)

XML gives the user the possibility to define hierarchies with their own elements and attributes. This standard was developed in a structured way with the idea of making it be both human and machine readable (Hendler & Pardo, 2012)

XML is widely used for the representation of data structures independent of the platform or operating system used, for example in web services or configuration files (Fennell, 2013). Currently, most software systems have the possibility of exporting and importing its data in XML format, trying to standardize the data structures for managing information, because XML plays an important role in the exchange of a data on the web (Kyu & Nyunt, 2009).

2.4. Programming Languages

A programming language is a set of valid sentences. It has semantics and syntax. The semantics provide the meaning of every construction that is possible in that programming language. The syntax defines its structure. A grammar is a formal language specification, a way to describe the semantics and syntax of a programming language (Gabbrielli & Martini, 2010) .

2.4.1 Programming Paradigms

Programming paradigms are different ways to create source code. A programming paradigm is not better or worse than other. In some cases, a paradigm is better than others and in other occasions may not. It depends on the particular case where used.

In Procedural programming the programmer creates procedures and functions, and invokes them when necessary. Procedures have variables, iteration, and modularization. They use "records", a collection of items typically indexed, for storing data. The user defines what to do (goal) and how he/she is going to achieve it (sequence of actions). For example, if you have an array holding the salaries of the employees and you want to print the values, you should do a loop and iterate all over the array accessing each record using its index and then printing its value. Some examples of programming languages that are based using this paradigm are C, FORTRAN, and Basic (Ishida, Sasaki, & Fukuhara, 1991).

Object-Oriented Programming is a well-known and popular paradigm. It is based on the concept of "Objects" and the interaction between them, using messages. Objects are instances of Classes, which define their type. Each object has attributes and methods. This paradigm also supports procedural programming characteristics such as variable definition, functions, iteration, among others (Jacobsen, Christerson, Jonsson, & Overgaard, 1992) . For the salaries example, in this case the employees would be objects based on the same class, with similar attributes. You would have an array of objects (employees) and could iterate over them, accessing the salary for each employee, through a method to get the desired value. Popular object-oriented languages include C++, Java, Objective-C, Ruby, and PHP.

Declarative programming is used in a different way. The idea is to define what to do but not how to do it. The user declares functions and uses recursion to obtain the desired results. Structured Query Language–SQL is an example of a language that uses this paradigm (Lloyd, 1994). In this case, if you want to know the salaries of the employees you do not iterate over the records, you just select the field (salary) from the table (employees) with an instruction. Note that you do not need to describe how to obtain the results, just specify what you need. In procedural or object-oriented programming you should do it with a loop, specifying how to achieve it. Other examples of declarative languages are regular expressions, CSS, and Prolog.

2.4.2 Interpreters

In order to implement programming languages there are two approaches: compilers and interpreters. The interpreter takes as input the source code of a program and its argument values, and it produces the output directly, generating its executable.

On the other hand, a compiler involves more steps; it takes as input the source code and then it produces an executable. This executable represents a program in object code or machine code, executed by a computer, like assembly language or *bytecode* that can be run separately on the input data and that will produce the output. The compiler pre-processes the program first (Chauhan, 2013).

We are going to focus on interpreters, since we need our program to take some code as input an immediately execute it. We don't want the user to compile the code every time, the main purpose of the language will be to provide an easy way to obtain results.

A typical interpreter has four major phases: lexical analysis, parsing, semantic analysis, and evaluation.

2.4.2.1 Lexical Analysis

The goal of lexical analysis is to divide the program text into words, also known as "tokens". The goal of a lexer is to obtain a sequence of tokens: It receives a string of characters as input, and returns a list of tokens as output. Each token has two attributes: a token type (symbol category) and the text (value) associated with it (Aiken, 2014).

For example, in the piece of program text

if x==y then z=1; else z=2;

the tokens are:

Keywords: *if*, *then* and *else*.

Variable names: *x*, *y* and *z*.

Constants: *1* and *2*

Operators: double equal (==) and the assignment operator (=)

Punctuation: semicolon (;)

Separators: blanks ( )

2.4.2.2 Parsing

Once words are classified, the next step is to understand the structure of the sentences. For each sentence, the parser checks whether it conforms to the syntax of the language: It takes as input the list of tokens returned by the lexer, parses it according to the syntax rules, and produces a representation of the syntactic structure using trees (Aho, Sethi, & Ullman, 1986). For the code of the example, the parse tree is going to be *if-then-else*. The root of the diagram consists of three parts: a predicate, a *then* statement and an *else* statement. The predicate consists of three pieces: a variable, a comparison operator, and another variable; and together they form a relation ($x == y$). The *then* statement consists of an assignment where *z* gets the value of 1. The *else* statement has the form of an assignment where z gets the value of 2. All together form a parse tree of the i*f-then-else*, showing its structure, breaking it up into its basic pieces. This is shown in Figure 1.

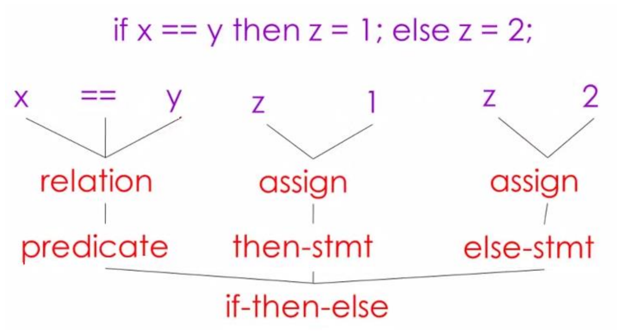
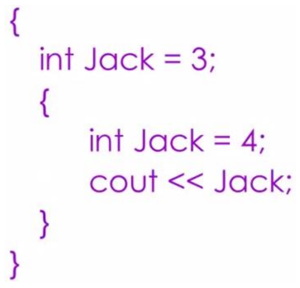


Figure 1. Parse Tree of the if-then-else statement (Aiken, 2014)

2.4.2.3 Semantic Analysis

After understanding the sentence structure, the next step is to try to understand the meaning of what has been written. Typically, semantic analysis consists of tracking declarations of variables, functions, and types, as well as type checking (Bennett, 1990). In programming languages the semantic analysis may be a problem with variable bindings. To prevent this, as shown in Figure 2, we have two variables called Jack, and the programming language has rules to prevent ambiguities. In the example shown in Figure 2, the value printed by the output statement is 4 because the use of the inner variable Jack binds to the definition in the inner section, and the outer definition is hidden. The outer definition is not active in this scope because it is hidden by the inner definition.



```
{
   int Jack = 3;
   {
        int Jack = 4;
        cout << Jack;
   }
}
```

Figure 2. Variable bindings (Aiken, 2014)

Interpreters perform many semantic validations besides variable bindings, for example the do type checking, to verify that the values assigned to variables are the same type.

2.4.2.4 Evaluation

Finally, the evaluator takes the parse tree produced by the parser, evaluates each item, and returns the output. Evaluators usually traverse the parse tree recursively. It is recommended to optimize the evaluation process for allowing calculations to be performed more quickly (Romer, et al., 1986).

2.4.3 Domain Specific Languages—DSL

A domain-specific language—DSL is created to solve problems in a specific domain. On the other hand, general-purpose languages—GPL are created to solve problems in many domains (Mernik, Heering, & Sloane, 2005). Examples of DSL are query languages such as SQL, XPath, and XQuery; UNIX shell scripts, OCL, and HTML among others. Examples of GPL are C++, Python, and Java.

The DSL life cycle consists of five development phases: decision, analysis, design, implementation, and deployment (Mernik, Heering, & Sloane, 2005).

2.4.3.1 Decision

Depending on the specific needs, you should use an existing DSL or GPL, or develop a new DSL. Creating a new domain-specific language rather than using another language should be worthwhile if the DSL facilitates to express a solution in a more clear and concise way than existing languages would do, and if the type of problem to solve is common (Freudenthal & Cybernetica AS, 2009).

2.4.3.2 Analysis

In this phase the problem domain is identified with the domain knowledge. The domain model should take into account (Mernik, Heering, & Sloane, 2005):

- Domain definition including its scope
- Terminology: ontology and vocabulary
- Descriptions of domains concepts
- Identification of the parts that differ and the parts that are the same for each code base.

2.4.3.3 Design

A DSL can be designed based on another language, or it can be created from scratch. It also can be classified as (Wile, 2004):

- Informal: specified in natural language with examples
- Formal: specified using regular expressions, grammar, etc.

2.4.3.4 Implementation

There are two main implementation patterns (Evans, 2004):

- Interpreter: there are no transformations, it is directly executable
- Compiler / Application generator: Generates source code from the model.

2.4.3.5 Deployment

Developers use DSLs to specify programs. Those models are implemented with one of the patterns (interpreter/compiler), creating working software for the end-users (Visser, 2008).

3. STATE-OF-THE-ART REVIEW

There are several approaches to address semantic interoperability, hybrid programming languages, and specific solutions for dealing with heterogeneous systems.

Typically, Semantic Heterogeneity occurs when there are differences on the interpretation, meaning, or intent of use of a set of data. On the other hand, Structural Heterogeneity occurs when the same concepts are modeled in different systems with different logical structures (Arch-int & Arch-int, 2013).

There are two major problems in exchanging and sharing information in a semantically consistent way (Bergamaschi, Castano, & Vincini, 1999):

- How to determine if sources contain information which is related to the same or similar concept(s)?
- How to handle semantic heterogeneity to support the process of information integration?

Ontologies and Metadata Vocabularies are considered ways for providing semantic context when determining the relevance of resources. Ontologies are usually created to define the meaning of the terms and concepts used in a specific domain. Ontological commitment is defined as choosing and sharing vocabulary elements across applications in a coherent and consistent way (Guarino, Carrara, & Giaretta, 1994), and it is a good basis for semantic interoperability in heterogeneous systems.

Although ontologies have been considered the answer to semantic interoperability, there are concerns about the way static ontologies resolve semantic conflicts when there are dynamic semantics. For example, database schemas could change over time and then, the original semantics as they were defined, would not be longer valid. This could lead to semantic conflicts, affecting the interpretations that are performed when answering queries and then obtaining unexpected results (Cui, Jones, & O'Brien, 2002). To address this, the SMILE system (Arch-int & Arch-int, 2013) proposes the ontology mapping process, where they have to discover a term defined in one ontology corresponding to the same or similar term defined in another ontology to resolve the semantic and structural conflicts, and enable the interoperability of heterogeneous information systems by applying it to the e-learning domain. However, this system applies only for the learning domain and it is not usable for other systems.

Companies are building information systems by integrating previously independent applications, together with new developments. This integration process has to deal with legacy applications. Typically, a legacy application can only be used through its specific interface, and cannot be modified. However, an option for developers is to create wrappers, like web services to provide other ways to access the data in such legacy systems. In many cases, the cost of rewriting a legacy application would be prohibitive. These heterogeneous software systems need to communicate each other. One way to achieve this is building specific software (middleware) to communicate a pair of such systems (Hadim &

Mohamed, 2006). For example, if you manage your invoices in Microsoft Excel® and have common information in a MySQL database, you may use specific middleware software to import/export data from one system to another (Curry & Mossire, 2006). However, if you try to exchange information with a different version of the software, for example Excel 2010 instead of Excel 2003, this could not work and it would be necessary to use another middleware that supports the new version of the spreadsheet.

There are many middleware systems in the market but only for the most known applications. If you want to interoperate custom software applications you should buy or build specific middleware software for each pair of systems (Bouguettaya, Benatallah, & Elmagarmid, 2012).

Middleware systems have their own interfaces to communicate and translate data from one system to another, and this is done through specific tools and languages from the middleware providers. An example of a middleware is *OpenESB* (Bouguettaya, Benatallah, & Elmagarmid, 2012), which offers a Java implementation of web service technologies to integrate applications. Another example is *Oracle Fusion Middleware* (Bouguettaya, Benatallah, & Elmagarmid, 2012), which offers multiple technologies, including development tools, to make applications interoperate. In these middleware systems, interoperability is achieved through the orchestration of web services, which include interfaces definition, data transformation, routing, etc. This is a common way to integrate and interoperate heterogeneous software systems, nowadays. The implementation in these middleware systems represents the semantic interoperability rules, which unambiguously define the data structures and exchange protocols among systems.

Other solutions are data warehouses, which are databases that integrate information from several data sources and are used for reporting, as they focus on data storage. At the level of semantic interoperability, this is a good option to hold common information because all data are coupled, but this information is read-only and then cannot be modified, leading a gap into data consistency. Other approaches are web services, which are software applications that exchange data with other web-based applications independent of hardware, software, and operating systems.

Web services are useful in linking applications operating on heterogeneous software systems performing critical functions for many businesses. However, there is a need for the systems to be accessible by them, using specific platforms (middleware systems) to develop and execute such web services. Regarding query languages, SQL has the possibility of modifying data, but neither XPath nor XQuery have.

This issue restricts the ability of having interoperability at the semantic level, because is possible to identify the inconsistencies in the information, but is not possible to correct them.

There are some remaining problems to take into account:

• Web services are recommended to negotiate and retrieve information from a source system. However, every web service must be programmed only for single tasks, instead of defining a general process. Also, the amount of retrieved/sent information is restricted to little packages of data. Finally, every interoperability task needs the construction of an atomic web service.

• Several query languages are defined for read-only purposes and the interoperability process needs capabilities of writing and modifying data in the target system.

• Middleware systems must be built for every pair of heterogeneous systems.

• Data warehouses need to merge the information from several sources into a common read-only repository.

• Standards are only recommended for building new systems. Legacy systems are sometimes difficult to adapt to standards. In addition, standards are version-sensitive.

4. SEMANTIC INTEROPERABILITY LANGUAGE—SIL

Heterogeneous software systems need to guarantee semantic interoperability and the existing solutions still exhibit some drawbacks. In order to address the problems discussed in the previous chapter, we defined the Semantic Interoperability Language—SIL.

4.1 Language Definition

In order to have an easy-to-use language, with no need to install additional software, we created SIL as an interpreted language. We used Regular Expressions for parsing purposes, and the Lexer was made from scratch. SIL is based on JavaScript, so you only need a browser to run SIL code. A simple interface (web page) was created using HTML5, CSS3, JQuery and Bootstrap 3.

The language is composed of two files: regularExpressions.js for the queries and string management, and sil.js with the lexer, the functions and procedures to handle of the operations of the language.

4.2 Characteristics

In order to guarantee syntactic interoperability we need a common data format among the systems we want to interoperate (Manconi & Rodriguez-Tomé, 2011). We chose XML because it is widely used and nowadays many software systems have the option of exporting their data is such format (Fennell, 2013).

We take into account that there are many legacy systems that need to interoperate, and with the option of exporting their data to XML, we can use the language on those formatted data, making SIL able to be used on a countless group of software systems.

With respect to semantic interoperability, we give the user the possibility of defining equivalences in both the elements and its values. The user can also define collections of similar values.

SIL was designed to achieve the semantic interoperability between heterogeneous software systems, facilitating the analysis, comparison, and interaction of similar XML data sets.

In SIL you can create objects and collections of objects. The concept of object is different than the traditional Object-Oriented Programming—OOP concept, where an object is an instance of a class. Objects in SIL represent the structure of something, similar to a class in OOP, being first class entities, and primary form of decomposition.

SIL offers a set of features for achieving semantic interoperability, offering the user easy methods for operating different data sets.

4.3 Basic Notation and syntax

- All objects (or collections of them) are declared using the $ sign before the name of the object/collection
- The -> sign is used to get/set a property of an object/collection or to execute an action (method).
- The **:=** sign is used to assign a value to a collection of objects.
- Methods always have parentheses at the end of their names

In order to illustrate the usage of the language, as an example, imagine you have two systems for managing common information about the students of a course. One system uses a MySQL database, and the other system is an Excel spreadsheet. You can export their data into XML files, and then use SIL to iterate over the data, define equivalences, find inconsistencies, etc.

After exporting the data, you have two files: Mysql.xml and Excel.xml.

4.4 Object definition

To create an object, you should use the *isAnObject()* method. For example, to declare an object called student you should use:

$student->isAnObject();

To define the attributes of an object there are two methods: *hasA(attrib)* and *hasAUnique(uniqueAttrib)*. The *hasAUnique(uniqueAttrib)* method is used to set up the o primary key of the object.

Continuing with the example of the students, you can say a student has a unique id, has a first name, a last name, an email, and a gender:

$student->hasAUnique('id');

$student->hasA('firstName');

$student->hasA('lastName');

$student->hasA('email');

$student->hasA('gender');

4.5 Values and equivalences

The *value(attrib)* method is used to specify a value for an attribute of an object.

The *isEquivalentTo(value)* method is used to assign a equivalence to an existing value.

For each attribute of the object, you can define the equivalences of values (if there is any).

For example, you can define that for the gender attribute the 'Male' value is equivalent to the 'M' value. So if in a set a record has the 'Male' value and in another set, the matching record has the 'M' value, there are going to be considered equivalent. This is used when comparing different records, to not report inconsistencies between similar values.

$student->gender->value('Male')->isEquivalentTo('M');

You can assign other equivalences as well:

$student->gender->value('Male')->isEquivalentTo('1');

$student->gender->value('Female')->isEquivalentTo('F');

$student->gender->value('Female')->isEquivalentTo('0');

4.6 Collections

For creating a collection of objects, it is necessary to use the *isBasedOn(object)* method.

If you want to create two collections of students, each one referring to a different data set, you proceed as follows:

$excelStudents->isBasedOn($student);

$mysqlStudents->isBasedOn($student);

4.7 Equivalences in field names

The *isDefinedAs(field)* method is used to specify a equivalent field name for an attribute of an object. For example, if you want to specify that the first name of the student is defined as *fname* in the *$mysqlStudents* collection, you can do it as follows:

$mysqlStudents->firstName->isDefinedAs('fname');

You can define as many equivalences as you need:

$mysqlStudents->id->isDefinedAs('identification');

$mysqlStudents->lastName->isDefinedAs('lname');

4.8 Data sources and locations

The *isAccessedFrom(file)* method is used to specify the source file where the XML data is for a collection of objects. For the *$excelStudents* and *$mysqlStudents* collections, you can specify their data sources as follows:

$excelStudents->isAccessedFrom('excel.xml');

$mysqlStudents->isAccessedFrom('mysql.xml');

The *hasAsRecordPath(path)* method is used to specify the base path (the parent route) of all the objects in the XML file to access the records.

Based on the XML structure of each file, you can define the parent path of the students for each case. For example:

$excelStudents->hasAsRecordPath('/Workbook/Worksheet/@ss:Name="Sheet1"/Table/Row');

$mysqlStudents->hasAsRecordPath('/DATAPACKET/ROWDATA/ROW');

The *isAt(path, [position])* method specifies the location of the attribute in the XML file (the route to access the object data). This uses the record path as a base, so you don't need to enter the record route again (base path))

For example, to specify that the ids of the *$excelStudents* are in */Workbook/Worksheet/@ss:Name="Sheet1"/Table/Row/Cell/Data*, in the first row, you do:

$excelStudents->id->isAt('Cell/Data',1);

For the other attributes, you can do it in a similar way:

$excelStudents->lastName->isAt('Cell/Data',2);

$excelStudents->firstName->isAt('Cell/Data',3);

$excelStudents->gender->isAt('Cell/Data',4);

$excelStudents->email->isAt('Cell/Data',5);

$mysqlStudents->id->isAt('@identification');

$mysqlStudents->firstName->isAt('@fname');

$mysqlStudents->lastName->isAt('@lname');

$mysqlStudents->email->isAt('@email');

$mysqlStudents->gender->isAt('@gender');

If you need to specify a section of a string composed of several words (separated by commas), just use the name of the field followed by [index], where index is the position of the word in the sentence (index starts at 0). For example:

<field>this, is, a, text</field>

field[0] would refer to "*this*", *field[1]* to "*is*", *field[2]* to "*a*", and *field[3]* to "*text*".

Note: Currently, SIL only supports the comma as a list separator.

4.9 Query methods

The *in(collection)* method returns the common records in the collections, for example to find the excel students with corresponding records in the MySQL database:

$otherStudents:=$excelStudents->in($mysqlStudents);

The *notIn(collection)* method returns the records that are not in the other collection, for example the excel students that are not listed in the MySQL database:

$otherStudents:=$excelStudents->notIn($mysqlStudents);

If you need to list a specific field, for example all ids:

$result:=$excelStudents->id

$result:=$mysqlStudents->id

If you need to concatenate text, use the *concatWith(otherString)* method, for example to get the full names of the *$mysqlStudents* collection:

$mysqlStudents->firstName->concatWith($mysqlStudents->lastName);

4.10 Data management methods

The *add(collection)* method add records to an existing collection. For example to add a set of students to an existing collection of students:

$mysqlStudents->add($otherStudents);

Add to the *$excelStudents* collection the *$mysqlStudents*' student with id=1:

$excelStudents->add($mysqlStudents->id='1');

The *remove(collection)* method removes a set of records from a collection of objects. For example to remove from the *$excelStudents* collection the corresponding students that exist in the *$mysqlStudents* collection:

$excelStudents->remove($mysqlStudents);

Remove from the *$excelStudents* collection the student with id=1:

$excelStudents->remove($excelStudents->id='1');

The *updateWith(collection)* method updates all the existing records of an existing collection with the ones of a given collection.

For example, to update all the records of the *$mysqlStudents* collection with the records of the *$excelStudents* collection, you can do the following:

$mysqlStudents->updateWith($excelStudents);

Update all the fields of a record:

$excelStudents->id='1'->updateWith($mysqlStudents->id='1');

Update a specific field of a record:

$excelStudents->id='1'->firstName->updateWith($mysqlStudents->id='1'->firstName);

The *syncWith(collection)* method synchronizes all the existing records of an existing collection with the ones of a given collection.

For example, to synchronize both sets (if there are inconsistencies on existing data, the MySQL records will remain untouched):

$excelStudents->syncWith($mysqlStudents);

Synchronize both sets (if there are inconsistencies on existing data, the excel records will remain untouched):

$mysqlStudents->syncWith($excelStudents);

4.11 Comparison methods

The *compareTo(collection)* method is used for comparing two collections of objects. It returns a new collection with all the differences.

Compare both student collections:

$result:=$excelStudents->compareTo($mysqlStudents);

Compare a specific record:

$result:=$excelStudents->id='1'->compareTo($mysqlStudents->id='1');

5. CASE STUDY

In order to test SIL, in the first semester of 2015, we worked with two groups of undergraduate students of the computer science programs from two universities in Medellín, Colombia. The first group of students was from the *National University of Colombia* and the second, from *University of Medellín*.

5.1 First Phase

The first phase of the case study was to select the best students of each group, in order to have competitive programmers. Once we selected the students, we asked them to solve the following problem, using any programming language. They chose the desired programming language to work with. They also were asked to report how long they took to solve the problem.

The description of the problem is as follows:

Hospital ABC has two software systems for managing their patients' information. The first one is used for accounting. The second one is used for scheduling appointments.

Both systems are supposed to have the same patients' information (demographics); however, sometimes the ABC's staff noted some inconsistencies in the data. For analysis purposes, imagine they exported the data of each system into a separate xml file (Patients.xml and Patients2.xml).

Based on that, write a program (in any programming language) to:

1. Find all the data of a given patient in a particular file. You should search using a primary key.

2. Determine if all the data of a given patient in a file corresponds to its matching one in the other file.

3. If a patient is missing in one system (file), copy his/her info to the other system (file)

4. Update a patient's info in one or both files ([specific field]/ all fields) -- name, date of birth, social security number, address

5. Delete a patient from one file (or from both)

6. List the patients who are in a file and are not in the other file

7. List the patients who are in both files

5.1.1 Results

Regarding the results, just for comparison purposes, for the first student who turned the exercise in, it took the student around 14 hours to solve the problem, and around 1500 lines of code. He used Java. He created the following files:

Main.java (25 lines), *hospital.java* (51 lines), *patient.java* (110 lines), *Patient2.java* (121 lines), *patients.java* (33 lines), *Physician.java* (36 lines), *SingletonDB.java* (108 lines), and *MainForm.java* (1031 lines).

After reviewing the code, it is clear that the student created several classes with their associated attributes, the *mutators* and *accessors* methods (getters and setters), data collections, conditionals, loops, etc. This approach met the requirements; however is not an optimal solution: it took the student a lot of time to implement it (around 14 hours) and the total number of lines (1515) is too large for a simple problem. If for any reason there is a need to work with different data sets, this approach would be useless, because it is a system specific solution, making the user to create the whole code again.

5.2 Second Phase

The second step of the exercise was, giving the information of Chapter 4 to the students, to solve the same problem using SIL. First, the students read the syntax and examples of SIL, and then developed the solution.

5.2.1 Results

The same student, who solved the problem using Java in the previous phase, spent about eight hours to solve the same problem using SIL. Studying the SIL syntax and examples took him around two hours, and writing the solution in SIL, the remaining six hours.

As you can tell, there is a huge difference in times and number of lines of code. The Java approach took the student around 14 hours compared to around six hours of the SIL approach (about half the time), and the number of lines in the SIL approach was 93 compared to 1515 in the Java approach, that is just the 6%.

The results from the other students were similar. For these kind of problems, there are disadvantages by using general-purpose languages such as Java or C#, because for each different case you have to create all the structures (classes, objects, relationships), methods, use external libraries for XML parsing, and make specific validations; and it will take more time than with a domain-specific language as SIL, because it was designed specifically for the interoperability of heterogeneous software systems.

6. CONCLUSIONS AND FUTURE WORK

Based on the problem of semantic interoperability, and considering advantages and drawbacks of the existing solutions, we built a domain-specific hybrid language to deal with the main difficulties of guaranteeing semantic interoperability between heterogeneous software systems. We offered the needed features to identify and to correct inconsistent data between two heterogeneous software systems, to guarantee the quality of the information that is shared, facilitating the process.

SIL is a domain-specific language with specific features for dealing with semantic interoperability. The language is interpreted and has an interpreter, and there is no need to compile the instructions for getting the results. This allows SIL to be used in any browser without the necessity of installing additional software.

We specified the rules and operations of a DSL to guarantee the data interoperability among heterogeneous software systems, achieving significant improvements in the data management of the companies. We created a solution where the user can define his/her own equivalences for the data sources, can query documents, and can modify information based on other data without incurring in using difficult solutions, spending a lot of time.

The results of this Thesis can generate the following future work:

- Automated identification of data inconsistencies, made by using SIL.
- Addition of new data structures to the SIL specification, in order to make it widely used.
- Direct modification of data sources in the heterogeneous software systems.

7. APPENDICES

7.1 XML Representation

A representation in XML of a bibliography document is as follows:

```
<bib>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
        <author><last>Stevens</last><first>W.</first></author>
        <publisher>Addison-Wesley</publisher>
        <price>65.95</price>
    </book>

    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
        <author><last>Stevens</last><first>W.</first></author>
        <publisher>Addison-Wesley</publisher>
        <price>65.95</price>
    </book>

    <book year="2000">
        <title>Data on the Web</title>
        <author><last>Abiteboul</last><first>Serge</first></author>
        <author><last>Buneman</last><first>Peter</first></author>
        <author><last>Suciu</last><first>Dan</first></author>
        <publisher>Morgan Kaufmann Publishers</publisher>
        <price>39.95</price>
    </book>

    <book year="1999">
        <title>The Economics of Technology and Content for Digital
TV</title>
        <editor>
               <last>Gerbarg</last><first>Darcy</first>
                <affiliation>CITI</affiliation>
        </editor>
            <publisher>Kluwer Academic Publishers</publisher>
        <price>129.95</price>
    </book>
</bib>
```

We can identify the next information from the XML document:

1. There is only one root element (bib), which is the parent of the rest of the elements.
2. Each element may have other elements. For example the first “book” element has four elements: title, author, publisher, and price. However, the “price” element does not have children elements.
3. An element may have attributes. For example the second “book” element has an attribute “year” with the value of “1992”.
4. Each element may have text (a value) associated to it. For example the title of the last book is “The Economics of Technology and Content for Digital TV”.
5. It is required for every element to have both an opening and a closing tag. For example <price> and </price>, or if the element has no text, it should be <element/>.

7.2 Example XML Files

As an example, imagine you have two applications for managing common information about the students of a course. One system uses a MySQL database, and the other system is an Excel spreadsheet. After exporting their data into XML files, the following is the XML representation.

7.2.1 Mysql.xml

```
<?xml version="1.0" standalone="yes"?>
<DATAPACKET Version="2.0">
<METADATA>
<FIELDS>
<FIELD FieldName="identification" DisplayLabel="identification"
FieldType="String" FieldClass="TField"/>
<FIELD FieldName="fname" DisplayLabel="fname" FieldType="String"
FieldClass="TField"/>
<FIELD FieldName="lname" DisplayLabel="lname" FieldType="String"
FieldClass="TField"/>
<FIELD FieldName="email" DisplayLabel="email" FieldType="String"
FieldClass="TField"/>
<FIELD FieldName="gender" DisplayLabel="gender" FieldType="String"
FieldClass="TField"/>

</FIELDS>
</METADATA>
<ROWDATA>
<ROW identification="1090909" fname="Servio Tulio  " lname="Benítez Arco      "
email="stbenit@unal.edu.co  " gender="Male"/>
<ROW identification="1090901" fname="Oscar Darío   " lname="Botero Vargas     "
email="odbotero@unal.edu.co " gender="Male"/>
<ROW identification="1090902" fname="Holman A.     " lname="Buenaventura Ochoa"
email="holman@hackermail.com" gender="Male"/>
<ROW identification="1090903" fname="César Daniel  " lname="Builes Suaza      "
email="cdbuiles@unal.edu.co " gender="Male"/>
<ROW identification="1090904" fname="José Luis     " lname="Carrascal Rojas   "
email="jlcarras@unal.edu.co " gender="Male"/>
<ROW identification="1090905" fname="Julián Alfredo" lname="Castro Orozco     "
email="jacastr1@unal.edu.co " gender="Male"/>
<ROW identification="1090906" fname="Mary Inés     " lname="Duarte Herrera    "
email="miduarte@unal.edu.co " gender="Female"/>
<ROW identification="1090907" fname="Daniel        " lname="Gaviria Giraldo   "
email="dgaviria@unal.edu.co " gender="Male"/>
<ROW identification="1090908" fname="Didier Fabián " lname="Granados M.       "
email="dfgranad@unal.edu.co " gender="Male"/>
```

```
<ROW identification="1090913" fname="Elvis Fernando" lname="Higuita Carvajal  "
email="ehiguita@unal.edu.co " gender="Male"/>
<ROW identification="1090924" fname="Alejandra     " lname="Lopera Velosa     "
email="aloperav@unal.edu.co " gender="Female"/>
<ROW identification="1090935" fname="John Fredy    " lname="Medina Eusse      "
email="jfmedina@unal.edu.co " gender="Male"/>
<ROW identification="1090946" fname="Andrea        " lname="Mesa Múnera       "
email="amesam@unal.edu.co   " gender="Female"/>
<ROW identification="1090947" fname="Juan Esteban  " lname="Muñoz Rendón      "
email="jemunoz@unal.edu.co  " gender="Male"/>
</ROWDATA>
</DATAPACKET>
```

7.2.2 Excel.xml

```
<?xml version="1.0"?>
<?mso-application progid="Excel.Sheet"?>
<Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet"
 xmlns:o="urn:schemas-microsoft-com:office:office"
 xmlns:x="urn:schemas-microsoft-com:office:excel"
 xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet"
 xmlns:html="http://www.w3.org/TR/REC-html40">
 <DocumentProperties xmlns="urn:schemas-microsoft-com:office:office">
  <Author>Carlos Mario Zapata J.</Author>
  <LastAuthor>Chaverra</LastAuthor>
  <LastPrinted>2004-02-26T21:08:00Z</LastPrinted>
  <Created>2003-10-14T21:35:03Z</Created>
  <LastSaved>2008-10-13T15:26:45Z</LastSaved>
  <Company>FAC. MINAS - UN de COLOMBIA</Company>
  <Version>14.00</Version>
 </DocumentProperties>
 <OfficeDocumentSettings xmlns="urn:schemas-microsoft-com:office:office">
  <AllowPNG/>
 </OfficeDocumentSettings>
 <ExcelWorkbook xmlns="urn:schemas-microsoft-com:office:excel">
  <WindowHeight>8835</WindowHeight>
  <WindowWidth>15180</WindowWidth>
  <WindowTopX>120</WindowTopX>
  <WindowTopY>120</WindowTopY>
  <ProtectStructure>False</ProtectStructure>
  <ProtectWindows>False</ProtectWindows>
 </ExcelWorkbook>
 <Styles>
  <Style ss:ID="Default" ss:Name="Normal">
   <Alignment ss:Vertical="Bottom"/>
   <Borders/>
   <Font ss:FontName="Arial"/>
   <Interior/>
   <NumberFormat/>
   <Protection/>
  </Style>
  <Style ss:ID="s70" ss:Name="Hyperlink">
   <Font ss:FontName="Arial" ss:Color="#0000FF" ss:Underline="Single"/>
  </Style>
  <Style ss:ID="s63">
  <Font ss:FontName="Arial" x:Family="Swiss" ss:Size="14"/>
 </Style>
 <Style ss:ID="s64">
  <Borders>
   <Border ss:Position="Bottom" ss:LineStyle="Continuous" ss:Weight="1"/>
   <Border ss:Position="Left" ss:LineStyle="Double" ss:Weight="3"/>
   <Border ss:Position="Right" ss:LineStyle="Continuous" ss:Weight="1"/>
   <Border ss:Position="Top" ss:LineStyle="Double" ss:Weight="3"/>
  </Borders>
  <Font ss:FontName="Arial" x:Family="Swiss" ss:Bold="1"/>
 </Style>
 <Style ss:ID="s65">
  <Borders>
   <Border ss:Position="Bottom" ss:LineStyle="Continuous" ss:Weight="1"/>
   <Border ss:Position="Left" ss:LineStyle="Continuous" ss:Weight="1"/>
   <Border ss:Position="Right" ss:LineStyle="Continuous" ss:Weight="1"/>
   <Border ss:Position="Top" ss:LineStyle="Double" ss:Weight="3"/>
  </Borders>
  <Font ss:FontName="Arial" x:Family="Swiss" ss:Bold="1"/>
 </Style>
 <Style ss:ID="s66">
  <Borders>
   <Border ss:Position="Bottom" ss:LineStyle="Continuous" ss:Weight="1"/>
   <Border ss:Position="Left" ss:LineStyle="Double" ss:Weight="3"/>
   <Border ss:Position="Right" ss:LineStyle="Continuous" ss:Weight="1"/>
   <Border ss:Position="Top" ss:LineStyle="Continuous" ss:Weight="1"/>
  </Borders>
 </Style>
 <Style ss:ID="s67">
  <Borders>
   <Border ss:Position="Bottom" ss:LineStyle="Continuous" ss:Weight="1"/>
   <Border ss:Position="Left" ss:LineStyle="Continuous" ss:Weight="1"/>
   <Border ss:Position="Right" ss:LineStyle="Continuous" ss:Weight="1"/>
   <Border ss:Position="Top" ss:LineStyle="Continuous" ss:Weight="1"/>
  </Borders>
 </Style>
 <Style ss:ID="s71">
  <Font ss:FontName="Arial" ss:Underline="Single"/>
 </Style>
 <Style ss:ID="s72">
  <Font ss:FontName="Arial" x:Family="Swiss" ss:Underline="Single"/>
 </Style>
 <Style ss:ID="s73" ss:Parent="s70">
  <Alignment ss:Vertical="Bottom"/>
  <Borders>
   <Border ss:Position="Bottom" ss:LineStyle="Continuous" ss:Weight="1"/>
   <Border ss:Position="Left" ss:LineStyle="Continuous" ss:Weight="1"/>
   <Border ss:Position="Right" ss:LineStyle="Continuous" ss:Weight="1"/>
   <Border ss:Position="Top" ss:LineStyle="Continuous" ss:Weight="1"/>
  </Borders>
  <Protection/>
 </Style>
</Styles>
<Worksheet ss:Name="Hoja1">
 <Table ss:ExpandedColumnCount="11" ss:ExpandedRowCount="24" x:FullColumns="1"
  x:FullRows="1" ss:DefaultColumnWidth="60">
```

```
   <Column ss:Index="2" ss:AutoFitWidth="0" ss:Width="99.75"/>
   <Column ss:AutoFitWidth="0" ss:Width="75.75"/>
   <Column ss:Index="5" ss:AutoFitWidth="0" ss:Width="115.5"/>
   <Column ss:Width="40.5"/>
   <Row ss:AutoFitHeight="0" ss:Height="18">
    <Cell ss:StyleID="s63"><Data ss:Type="String">INGENIERÍA DEL
SOFTWARE</Data></Cell>
   </Row>
   <Row ss:AutoFitHeight="0" ss:Height="13.5"/>
   <Row ss:AutoFitHeight="0" ss:Height="13.5">
    <Cell ss:StyleID="s64"><Data ss:Type="String">ID</Data></Cell>
    <Cell ss:StyleID="s65"><Data ss:Type="String">LAST NAME</Data></Cell>
    <Cell ss:StyleID="s65"><Data ss:Type="String">FIRST NAME</Data></Cell>
    <Cell ss:StyleID="s65"><Data ss:Type="String">GENDER</Data></Cell>
    <Cell ss:StyleID="s65"><Data ss:Type="String">E-MAIL</Data></Cell>
    <Cell ss:StyleID="s65"><Data ss:Type="String">GROUP</Data></Cell>
    <Cell ss:StyleID="s65"><Data ss:Type="String">GRADE</Data></Cell>
   </Row>
   <Row>
    <Cell ss:StyleID="s66"><Data ss:Type="Number">1090909</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Benítez Arco</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Servio Tulio</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">M</Data></Cell>
    <Cell ss:StyleID="s67"/>
    <Cell ss:StyleID="s67"><Data ss:Type="Number">3</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="Number">5</Data></Cell>
   </Row>
   <Row>
    <Cell ss:StyleID="s66"><Data ss:Type="Number">1090901</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Botero Vargas</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Oscar Darío</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">M</Data></Cell>
    <Cell ss:StyleID="s67"/>
    <Cell ss:StyleID="s67"><Data ss:Type="Number">9</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="Number">3.2</Data></Cell>
   </Row>
   <Row>
    <Cell ss:StyleID="s66"><Data ss:Type="Number">1090902</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Buenaventura
Ochoa</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Holman A.</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">M</Data></Cell>
    <Cell ss:StyleID="s73" ss:HRef="mailto:holman@hackermail.com"><Data
      ss:Type="String">holman@hackermail.com</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="Number">9</Data></Cell>
    <Cell ss:StyleID="s67"><Data
ss:Type="Number">4.0999999999999996</Data></Cell>
   </Row>
   <Row>
    <Cell ss:StyleID="s66"><Data ss:Type="Number">1090903</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Builes Suaza</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">César Daniel</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">M</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">cdbuiles</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="Number">2</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="Number">2</Data></Cell>
```

```
</Row>
<Row>
 <Cell ss:StyleID="s66"><Data ss:Type="Number">1090904</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">Carrascal Rojas</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">José Luis</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">M</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">jlcarras</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="Number">4</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="Number">1.2</Data></Cell>
</Row>
<Row>
 <Cell ss:StyleID="s66"><Data ss:Type="Number">1090905</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">Castro Orozco</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">Julián Alfredo</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">M</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">jacastr1</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="Number">6</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="Number">3.5</Data></Cell>
</Row>
<Row>
 <Cell ss:StyleID="s66"><Data ss:Type="Number">1090906</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">Duarte Herrera</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">Mary Inés</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">F</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">miduarte</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="Number">1</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="Number">4.3</Data></Cell>
</Row>
<Row>
 <Cell ss:StyleID="s66"><Data ss:Type="Number">1090907</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">Gaviria Giraldo</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">Daniel</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">M</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">dgaviria</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="Number">5</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="Number">2.8</Data></Cell>
</Row>
<Row>
 <Cell ss:StyleID="s66"><Data ss:Type="Number">1090908</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">Granados M.</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">Didier Fabián</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">M</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">dfgranad</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="Number">1</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="Number">1.9</Data></Cell>
</Row>
<Row>
 <Cell ss:StyleID="s66"><Data ss:Type="Number">1090913</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">Higuita Carvajal</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">Elvis Fernando</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">M</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="String">ehiguita</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="Number">3</Data></Cell>
 <Cell ss:StyleID="s67"><Data ss:Type="Number">4.5</Data></Cell>
 <Cell ss:Index="11" ss:StyleID="s71"/>
</Row>
```

```
   <Row>
    <Cell ss:StyleID="s66"><Data ss:Type="Number">1090924</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Lopera Velosa</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Alejandra</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">F</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">aloperav</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="Number">2</Data></Cell>
    <Cell ss:StyleID="s67"><Data
ss:Type="Number">2.2999999999999998</Data></Cell>
   </Row>
   <Row>
    <Cell ss:StyleID="s66"><Data ss:Type="Number">1090935</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Medina Eusse</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">John Fredy</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">M</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">jfmedina</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="Number">5</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="Number">3.5</Data></Cell>
   </Row>
   <Row>
    <Cell ss:StyleID="s66"><Data ss:Type="Number">1090946</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Mesa Múnera</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Andrea</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">F</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">amesam</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="Number">4</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="Number">2.6</Data></Cell>
   </Row>
   <Row>
    <Cell ss:StyleID="s66"><Data ss:Type="Number">1090947</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Muñoz Rendón</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">Juan Esteban</Data></Cell>
    <Cell ss:StyleID="s67"><Data ss:Type="String">M</Data></Cell>
    <Cell ss:StyleID="s67"/>
    <Cell ss:StyleID="s67"><Data ss:Type="Number">8</Data></Cell>
    <Cell ss:StyleID="s67"><Data
ss:Type="Number">4.0999999999999996</Data></Cell>
   </Row>
   <Row ss:Index="24">
    <Cell ss:Index="4" ss:StyleID="s72"/>
   </Row>
  </Table>
  <WorksheetOptions xmlns="urn:schemas-microsoft-com:office:excel">
   <PageSetup>
    <Layout x:Orientation="Landscape" x:CenterHorizontal="1"
x:CenterVertical="1"/>
    <Header x:Margin="0"/>
    <Footer x:Margin="0"/>
   </PageSetup>
   <Print>
    <ValidPrinterInfo/>
    <HorizontalResolution>-2</HorizontalResolution>
    <VerticalResolution>600</VerticalResolution>
   </Print>
   <Selected/>
   <TopRowVisible>3</TopRowVisible>
```

```
    <Panes>
     <Pane>
      <Number>3</Number>
      <ActiveRow>15</ActiveRow>
      <ActiveCol>4</ActiveCol>
     </Pane>
    </Panes>
    <ProtectObjects>False</ProtectObjects>
    <ProtectScenarios>False</ProtectScenarios>
   </WorksheetOptions>
   <Sorting xmlns="urn:schemas-microsoft-com:office:excel">
    <Sort>APELLIDOS</Sort>
    <Sort>NOMBRES</Sort>
   </Sorting>
  </Worksheet>
</Workbook>
```

7.3 SIL Implementation

7.3.1 regularExpressions.js

```
var commentRegExp = /^\s*(\/\/)+/;
var isAnObjectRegExp = /^\s*\$([\w\d]+)->isAnObject\(\);/;
var hasAUniqueRegExp = /^\s*\$([\w\d]+)->hasAUnique\(['"](\w+)['"]\);/;
var hasARegExp = /^\s*\$([\w\d]+)->hasA\(['"](\w+)['"]\);/;
var isEquivalentToRegExp =  /^\s*\$([\w\d]+)->(\w+)-
>value\(['"]([\w\s,\/]+)['"]\)->isEquivalentTo\(['"]([\w\s,\/]+)['"]\);/;
var isBasedOnRegExp = /^\s*\$([\w\d]+)->isBasedOn\(\$(\w+)\);/;
//var isDefinedAsRegExp =  /^\s*\$([\w\d]+)->(\w+)-
>isDefinedAs\(['"](\w+)['"]\);/;
var isDefinedAsRegExp =  /^\s*\$([\w\d]+)->(\w+)-
>isDefinedAs\(['"](\w+)['"](?:,['"]([\/\w\s@:='"\-
,\.\\]+)['"],?(\d?))?\);/;
var isAccessedFromRegExp =  /^\s*\$([\w\d]+)-
>isAccessedFrom\(['|"]([\w\.]+)['"]\);/;
var hasAsRecordPathRegExp = /^\s*\$([\w\d]+)-
>hasAsRecordPath\(['"]([\/\w|@:='"]+)['"]\);/;
var isAtRegExp =  /^\s*\$([\w\d]+)->(\w+)-
>isAt\(['"]([\/\w|@:='"]+)['"],?(\d)?\);/;

var allCollectionItemsRegExp =  /^\s*\$([\w\d]+)->all\(\);/;
var collectionAttributeEqualToRegExp =  /^\s*\$([\w\d]+)-
>(\w+)=['"]([\w\/\s,\-]+)['"];/;
var collectionAttributeRegExp =  /^\s*\$([\w\d]+)->(\w+);/;

var regularExpression = function(idDataSource){
    var regularExpression = new RegExp($("#inputTextArea").val(), "gim");
    //console.log(regularExpression.source);
    //console.log($("#dataSource1TextArea").val());
    var text =
$('#'+idDataSource).val().replace(/[\n|\r|\t|]/g,'').trim();

    var matches = text.match(regularExpression);

    //console.log(matches);
    //$('#outputTextArea').html(matches);
    var output='';
    if(matches!==null){
        for(var i=0;i<matches.length;i++){
            output+=matches[i]+ '\n';
        }
    }
    var newstr = output.replace(regularExpression, "$1");
    $('#outputTextArea').html(newstr);
};

var queryRegularExpression = function(sentence,text, regExp){
    console.log(regExp);
    var regularExpression = new RegExp(regExp, "gim");
    //console.log(regularExpression.source);
    //console.log($("#dataSource1TextArea").val());
    //var regularExpression = regExp;
    //var text =
$('#'+idDataSource).val().replace(/[\n\r\t]/g,'').trim();
    var matches = text.match(regularExpression);
    //console.log('matches',matches);
    var output='';
    if(matches!==null){
        for(var i=0;i<matches.length;i++){
            output+=matches[i]+ '\n';
        }
    }
    //var newstr = output.replace(regularExpression, "$1");
    //$('#outputTextArea').text(newstr);
    var newstr =  $('#outputTextArea').text() + sentence + '\n' +
output.replace(regularExpression, "$1") + '\n----------------------------
-------------------------------\n' ;
    $('#outputTextArea').text(newstr);
};

var matchValueRegularExpression = function(text, regExp, value,
delimiter, position){
    var regularExpression = new RegExp(regExp, "gim");

    var matches = text.match(regularExpression);

    //console.log('matches',matches);

    var output='';
    var values = [];
    //console.log('matches', matches);
    if(matches!==null && matches.length>1){
        for(var i=0;i<matches.length;i++){
            //output+=matches[i]+ '\n';
            values[i]=matches[i].replace(regularExpression,'$2');
            if(delimiter!==''){
                var tokens = values[i].split(delimiter);
                if (tokens[position]){
                    //console.log('token',tokens[position].trim());
                    if(tokens[position].trim()===value){
                        output+=matches[i]+ '\n';
                    }
                }
            }

            //if(values[i].spl){
            //    output+=matches[i]+ '\n';
            //}
        }
        return output.replace(regularExpression, "$1");
    }
    return null;
    //var values = output.replace(regularExpression, "$2");
    //console.log('values', values);

};

//var matchesRegularExpression = function(text, regExp){
//    var regularExpression = new RegExp(regExp, "gim");
//
//    var matches = text.match(regularExpression);
//
//    //console.log('matches',matches);
//
//    var output='';
//    if(matches!==null){
//        for(var i=0;i<matches.length;i++){
//            output+=matches[i]+ '\n';
//        }
//    }
//    var newstr = output.replace(regularExpression, "$1");
//    return newstr;
//};
```

7.3.2 sil.js

```
var loadFileContentsInTextArea = function(inputFileId,textAreaId){
    var fileObj =  $('#'+inputFileId)[0].files[0];
    var objectURL = null;
    if(fileObj){
        objectURL = window.URL.createObjectURL(fileObj);
    }
    if(objectURL){
        $.ajax({
            url : objectURL,
            //type: "GET",
            dataType: "text",
            success : function (data) {
                $("#"+textAreaId).val(data);
            },
            fail: function (data){
                console.log('error',data);
            }
        });
    }
};



$(document).ready(function() {

    $.ajax({
        url : "xml/patients.xml",
        //url : "xml/excel.xml",
        dataType: "text",
        success : function (data) {
            $("#dataSource1TextArea").val(data);
        }
    });

    $.ajax({
        url : "xml/patients2.xml",
        //url : "xml/mysql.xml",
        dataType: "text",
        success : function (data) {
            $("#dataSource2TextArea").val(data);
        }
    });

    $("#dataSource1InputFile").on('change', function(){

loadFileContentsInTextArea($(this).prop('id'),'dataSource1TextArea');
    });

    $("#dataSource2InputFile").on('change', function(){

loadFileContentsInTextArea($(this).prop('id'),'dataSource2TextArea');
    });

    $("#dataSource2InputFile").on('change', function(){
```

```
loadFileContentsInTextArea($(this).prop('id'),'dataSource2TextArea');
    });


    $("#runButton").on('click', function(){
        run($('#inputTextArea').val());
    });



    //$("#inputTextArea").on('keyup', function(){
    //    regularExpression('dataSource1TextArea');
    //});


});

var run = function(text){
    //var scriptText = $('#inputTextArea').val();
    //var oldScript = document.getElementById('scriptContainer');
    //if (oldScript) {
    //    oldScript.parentNode.removeChild(oldScript);
    //}
    //var newScript = document.createElement('script');
    //newScript.id = 'scriptContainer';
    //newScript.text = scriptText;
    //document.body.appendChild(newScript);

    $('#outputTextArea').text('');
    var sil = new Sil();
    sil.run(text);
    //$('#outputTextArea').html(sil.objects);

};

function SilObject(){
    var name;
    this.equivalences = [];
    this.uniqueAttributes = [];
    this.attributes = [];

    this.addEquivalence = function(equivalence){
        var exists = false;
        this.equivalences.forEach(function (item, index, array) {
            if (item.attributeName === equivalence.attributeName &&
item.value1 === equivalence.value1 && item.value2 ===
equivalence.value2){
                exists = true;
                return;
            }
        });
        if(!exists){
            this.equivalences.push(equivalence);
        }
    };

    this.getEquivalences = function (attribute,value){
        var equivalentValues = [];
        this.equivalences.forEach(function (item, index, array) {
            if (item.attributeName === attribute && item.value1 ===
value){
                if (equivalentValues.indexOf(value)<0)
                    equivalentValues.push(value);
                if (equivalentValues.indexOf(item.value2)<0)
                    equivalentValues.push(item.value2);
            }
            if (item.attributeName === attribute && item.value2 ===
value){
                if (equivalentValues.indexOf(value)<0)
                    equivalentValues.push(value);
                if (equivalentValues.indexOf(item.value1)<0)
                    equivalentValues.push(item.value1);
            }
        });
        return equivalentValues;
    };

    this.addUniqueAttribute = function(attribute){
        if (this.uniqueAttributes.indexOf(attribute)<0){
            this.uniqueAttributes.push(attribute);
        }
    };

    this.addAttribute = function(attribute){
        if (this.attributes.indexOf(attribute)<0){
            this.attributes.push(attribute);
        }
    };
}

function SilCollection(){
    var name;
    var basedOn;
    var accessedFrom;
    var recordPath;
    this.definitions = [];
    this.paths = [];
    this.objects = [];

    this.queryRegularExpression = function (sentence,regularExpression,
text) {
        if (text==null)
            text = $('#'+this.accessedFrom).val();
        queryRegularExpression(sentence,text,regularExpression);
    };

    this.addDefinition = function(definition){
        var exists = false;
        this.definitions.forEach(function (item, index, array) {
            //if (item.attributeName === definition.attributeName &&
item.value === definition.value && item.delimiter ===
definition.delimiter && item.position === definition.position){
            if (item.attributeName === definition.attributeName){
                exists = true;
                return;
            }
        });
        if(!exists){
            this.definitions.push(definition);
        }
    };

    this.addPath = function(path){
        var exists = false;
        this.paths.forEach(function (item, index, array) {
            //if (item.attributeName === path.attributeName && item.value
=== path.value && item.position === path.position){
            if (item.attributeName === path.attributeName){
                exists = true;
                return;
            }
        });
        if(!exists){
            this.paths.push(path);
        }
    };

    //this.addObject = function(object){
    //    var exists = false;
    //    this.objects.forEach(function (item, index, array) {
    //        var propertiesTally = 0;
    //        var equalPropertiesTally = 0;
    //        for (var property in object) {
    //            if (object.hasOwnProperty(property)) {
    //                propertiesTally++;
    //                if(item.property===object.property){
    //                    equalPropertiesTally++;
    //                }
    //            }
    //        }
    //        if(propertiesTally===equalPropertiesTally){
    //            exists = true;
    //              return;
    //          }
    //      });
    //      if(!exists){
    //          this.objects.push(object);
    //      }
    //};

    this.addObject = function(instance,silObject){
        var exists = false;
        this.objects.forEach(function (item, index, array) {
            //if(item._id === object._id){
            //    exists = true;
            //    return;
            //}
            silObject.uniqueAttributes.forEach(function (attributeItem,
attributeIndex, attributeArray){
                if(item[attributeItem]===instance[attributeItem]){
                    exists = true;
                    return;
                }
            });
        });
        if(!exists){
            this.objects.push(instance);
        }
    };

    this.updateObject = function(object){
        this.objects.forEach(function (item, index, array) {
            if(item._id === object._id){
                for (var property in object) {
                    if (object.hasOwnProperty(property)) {
                        item.property = object.property;
                    }
                }
                return;
            }
        });
    };

    this.getContainer  = function(){
        if(this.recordPath){
            var tokens = this.recordPath.substr(1).split(/\//);
            return tokens[tokens.length-1];
        }
        return '';
    };

    this.getDefinition = function(attribute){
        for (var i=0; i< this.definitions.length; i++){
            if (this.definitions[i].attributeName===attribute)
                return this.definitions[i];
        }
        return null;
    };

    this.getDefinitionAttribute = function(attribute){
        var definition = this.getDefinition(attribute);
        if (definition!=null) return definition.value;
        return attribute;
    };

    this.getDefinitionDelimiter = function(delimiter){
        var definition = this.getDefinition(delimiter);
        if (definition!=null) return definition.delimiter;
        return '';
    };

    this.getDefinitionPosition = function(position){
        var definition = this.getDefinition(position);
        if (definition!=null) return definition.position;
        return '';
    };
}

function SilLexer(text) {
    //var words = text.split(/\s+/);
    //var words =
text.trim().split(/\s+(?=(?:[^'"]*['"][^'"]*['"])*[^'"]*$)/);
    var words = text.trim().split(/[\n\r]+/);
    console.log('words',words);
    var next = 0;
    this.nextWord = function () {
        if (next >= words.length) return null;
        return words[next++];
    };
    this.hasNextWord = function () {
        return next < words.length;
    };
}

function Sil () {
    var dictionary = {};
    var objects = {};
    var collections = {};
    this.stack = [];

    this.addWords = function (new_dict) {
        for (var word in new_dict)
            dictionary[word.toUpperCase()] = new_dict[word];
    };
    this.run = function (text) {
        this.lexer = new SilLexer(text);
        var word;


        while (this.lexer.hasNextWord()) {
            word = this.lexer.nextWord();
            //console.log(word);
            //console.log(word.match(isEquivalentToRegExp));
            num_val = parseFloat(word);
            if (dictionary[word]) {
                dictionary[word](this);
            }
            else if (word.match(commentRegExp)) {
                console.log('comment',word);
            }
            else if (word.match(isAnObjectRegExp)) {
                var objectName = word.replace(isAnObjectRegExp,'$1');
                objects[objectName] =  new SilObject();
                objects[objectName].name = objectName;
            }
            else if (word.match(hasAUniqueRegExp)) {
                var objectName = word.replace(hasAUniqueRegExp,'$1');
                var uniqueAttributeName =
word.replace(hasAUniqueRegExp,'$2');
                if(objects[objectName]){

objects[objectName].addUniqueAttribute(uniqueAttributeName);
                }
            }
            else if (word.match(hasARegExp)) {
                var objectName = word.replace(hasARegExp,'$1');
                var attributeName = word.replace(hasARegExp,'$2');
                if(objects[objectName]){
                    objects[objectName].addAttribute(attributeName);
                }
            }
            else if (word.match(isEquivalentToRegExp)) {
                var objectName = word.replace(isEquivalentToRegExp,'$1');
                var attributeName =
word.replace(isEquivalentToRegExp,'$2');
                var value1 = word.replace(isEquivalentToRegExp,'$3');
                var value2 = word.replace(isEquivalentToRegExp,'$4');
                if(objects[objectName]){

if(objects[objectName].uniqueAttributes.indexOf(attributeName)>=0 ||
objects[objectName].attributes.indexOf(attributeName)>=0) {

objects[objectName].addEquivalence({'attributeName': attributeName,
'value1': value1, 'value2': value2});
```

```
                    }
                }
            }
            else if (word.match(isBasedOnRegExp)) {
                var collectionName = word.replace(isBasedOnRegExp,'$1');
                var objectName = word.replace(isBasedOnRegExp,'$2');
                if(objects[objectName]){
                    collections[collectionName] =  new SilCollection();
                    collections[collectionName].name = collectionName;
                    collections[collectionName].basedOn = objectName;
                }
            }
            else if (word.match(isDefinedAsRegExp)) {
                var collectionName =
word.replace(isDefinedAsRegExp,'$1');
                var attributeName = word.replace(isDefinedAsRegExp,'$2');
                var value = word.replace(isDefinedAsRegExp,'$3');
                var delimiter = word.replace(isDefinedAsRegExp,'$4');
                var position = word.replace(isDefinedAsRegExp,'$5');

                if(collections[collectionName]){

if(objects[collections[collectionName].basedOn].uniqueAttributes.indexOf(
attributeName)>=0 ||
objects[collections[collectionName].basedOn].attributes.indexOf(attribute
Name)>=0){

collections[collectionName].addDefinition({'attributeName':
attributeName, 'value': value, 'delimiter': delimiter,
'position':position});
                    }
                }
            }
            else if (word.match(isAccessedFromRegExp)) {
                var collectionName =
word.replace(isAccessedFromRegExp,'$1');
                var dataSource = word.replace(isAccessedFromRegExp,'$2');
                if(collections[collectionName]){
                    collections[collectionName].accessedFrom =
dataSource;
                }
            }
            else if (word.match(hasAsRecordPathRegExp)) {
                var collectionName =
word.replace(hasAsRecordPathRegExp,'$1');
                var path = word.replace(hasAsRecordPathRegExp,'$2');
                if(collections[collectionName]){
                    collections[collectionName].recordPath = path;
                    var obj = {ssn:'123', name:'john'};
                    var obj2 = {ssn:'1233', name:'john'};
```

```
collections[collectionName].addObject(obj,objects[objectName]);

collections[collectionName].addObject(obj2,objects[objectName]);
                }
            }
            else if (word.match(isAtRegExp)) {
                var collectionName = word.replace(isAtRegExp,'$1');
                var attributeName = word.replace(isAtRegExp,'$2');
                var value = word.replace(isAtRegExp,'$3');
                var position = word.replace(isAtRegExp,'$4');
                if(collections[collectionName]){

if(objects[collections[collectionName].basedOn].uniqueAttributes.indexOf(
attributeName)>=0 ||
objects[collections[collectionName].basedOn].attributes.indexOf(attribute
Name)>=0){

collections[collectionName].addPath({'attributeName': attributeName,
'value': value, 'position': position});
                    }
                }
            }
            else if (word.match(allCollectionItemsRegExp)) {
                var collectionName =
word.replace(allCollectionItemsRegExp,'$1');
                if(collections[collectionName]){

collections[collectionName].queryRegularExpression(word,"(<"+collections[
collectionName].getContainer()+">(?:\\s*<\\w+>[\\w\\s\\/,\\-
]*<\\/\\w+>\\s*)*<\\/"+collections[collectionName].getContainer()+">)",nu
ll);
                }
            }
            else if (word.match(collectionAttributeEqualToRegExp)) {
                var collectionName =
word.replace(collectionAttributeEqualToRegExp,'$1');
                var attributeName =
word.replace(collectionAttributeEqualToRegExp,'$2');
                var value =
word.replace(collectionAttributeEqualToRegExp,'$3');

                if(collections[collectionName]){

if(objects[collections[collectionName].basedOn].uniqueAttributes.indexOf(
attributeName)>=0 ||
objects[collections[collectionName].basedOn].attributes.indexOf(attribute
Name)>=0){
                        var rootElement =
collections[collectionName].getContainer();
```

```
                        var attribute =
collections[collectionName].getDefinitionAttribute(attributeName);
                        var delimiter =
collections[collectionName].getDefinitionDelimiter(attributeName);
                        var position =
collections[collectionName].getDefinitionPosition(attributeName);
                        var values ='(?:';
                        var equivalentValues =
objects[collections[collectionName].basedOn].getEquivalences(attributeNam
e,value);
                        if(equivalentValues.length===0)
                            values+= value;
                        for(var i=0;i<equivalentValues.length;i++)
                            values+=equivalentValues[i]+'|';
                        if(values.lastIndexOf('|')===values.length-1)
                            values = values.substr(0,values.length-1);
                        values+=')';
                        //console.log('values',values);

                        //console.log('values',values);

//console.log('equivalentValues',objects[collections[collectionName].base
dOn].getEquivalences(attributeName,value));
                        //console.log('delimiter',delimiter, 'position',
position);
                        var regExp =
"(<"+rootElement+">\\s*(?:\\s*<\\w+>[\\w\\s\\/,\\-
]*<\\/\\w+>\\s*)*<"+attribute+">([\\w\\s\\/\\-,]*"+values+"[\\w\\s\\/\\-
,]*)<\\/"+attribute+">(?:\\s*<\\w+>[\\w\\s\\/,\\-
]*<\\/\\w+>\\s*)*\\s*<\\/"+rootElement+">)";
                        var initialSet = null;
                        if (delimiter!==null && delimiter!=='' &&
position!=null && position!==''){
                            initialSet =
matchValueRegularExpression($('#'+collections[collectionName].accessedFro
m).val(),regExp,value, delimiter, position);
                        }

collections[collectionName].queryRegularExpression(word,regExp,initialSet
);
                    }
                }
            }

            else if (word.match(collectionAttributeRegExp)) {
                var collectionName =
word.replace(collectionAttributeRegExp,'$1');
                var attributeName =
word.replace(collectionAttributeRegExp,'$2');

                if(collections[collectionName]){
```

```
if(objects[collections[collectionName].basedOn].uniqueAttributes.indexOf(
attributeName)>=0 ||
objects[collections[collectionName].basedOn].attributes.indexOf(attribute
Name)>=0){
                        var rootElement =
collections[collectionName].getContainer();
                        var attribute =
collections[collectionName].getDefinitionAttribute(attributeName);

collections[collectionName].queryRegularExpression(word,"<"+rootElement+"
>\\s*(?:\\s*<\\w+>[\\w\\s\\/,\\-
]*<\\/\\w+>\\s*)*(<"+attribute+">[\\w\\s,\\/\\-
]+<\\/"+attribute+">)(?:\\s*<\\w+>[\\w\\s\\/,\\-
]*<\\/\\w+>\\s*)*\\s*<\\/"+rootElement+">");
                    }
                }
            }
            else {
                //throw "Unknown word";
                console.error("Unknown word",word);
            }
        }
        console.log(objects,collections);
    };
}
```

7.3.3 index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Semantic Interoperability Language - SIL</title>
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" href="css/main.css" />
    <script src="js/jquery-1.11.3.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <script src="js/regularExpressions.js"></script>
    <script src="js/sil.js"></script>

</head>
<body>
<br/>
    <div class="row">
        <div id="dataSourcesDiv" class="col-xs-12">
            <div class="col-xs-6" id="dataSource1Div">
                <div class="form-group">
```

```
                    <label class="control-label" for="dataSource1InputFile">XML
File 1 input</label>
                    <input type="file" id="dataSource1InputFile">
                    <p class="help-block">Please choose a file or copy the
text.</p>
                </div>
                <textarea class="form-control" name="dataSource1TextArea"
id="dataSource1TextArea" cols=305" rows="20" placeholder="Data Source
1"></textarea>
            </div>
            <div class="col-xs-6" id="dataSource2Div">
                <div class="form-group">
                    <label class="control-label" for="dataSource2InputFile">XML
File 2 input</label>
                    <input type="file" id="dataSource2InputFile">
                    <p class="help-block">Please choose a file or copy the
text..</p>
                </div>
                <textarea class="form-control" name="dataSource2TextArea"
id="dataSource2TextArea" cols="30" rows="20" placeholder="Data Source
2"></textarea>
            </div>
        </div>
    </div>
    <hr/>
    <div class="row">
        <div id="languageDiv" class="col-xs-12">
            <div class="col-xs-6" id="inputDiv">
                <textarea class="form-control" name="inputTextArea"
id="inputTextArea" cols="30" rows="20" placeholder="Input">
//Create the $patient object:
$patient->isAnObject();

//Set up the patient's primary key:
$patient->hasAUnique('ssn');

// Define a $patient's attributes:
$patient->hasA('firstName');
$patient->hasA('lastName');
$patient->hasA('dateOfBirth');
$patient->hasA('dayOfBirth');
$patient->hasA('monthOfBirth');
$patient->hasA('yearOfBirth');
$patient->hasA('gender');
$patient->hasA('address');
$patient->hasA('streetAddress');
$patient->hasA('zipCode');
$patient->hasA('city');
$patient->hasA('state');

// Assign equivalences to existing values:
$patient->state->value('NY')->isEquivalentTo('New York');
$patient->state->value('Florida')->isEquivalentTo('FL');
$patient->gender->value('M')->isEquivalentTo('Male');
$patient->gender->value('1')->isEquivalentTo('Male');
$patient->gender->value('F')->isEquivalentTo('Female');
```

```
$patient->gender->value('0')->isEquivalentTo('Female');

// Creating a collection of patients:
$patientSet1->isBasedOn($patient);
$patientSet2->isBasedOn($patient);

//specify equivalent field names for patient's attributes:
$patientSet1->firstName->isDefinedAs('name',' ',0);
$patientSet2->firstName->isDefinedAs('first_name');

$patientSet1->lastName->isDefinedAs('name',' ',1);
$patientSet2->lastName->isDefinedAs('last_name');

$patientSet2->ssn->isDefinedAs('social_security');

$patientSet1->dateOfBirth->isDefinedAs('date_of_birth');
$patientSet1->dayOfBirth->isDefinedAs('date_of_birth','/',1);
$patientSet1->monthOfBirth->isDefinedAs('date_of_birth','/',0);
$patientSet1->yearOfBirth->isDefinedAs('date_of_birth','/',2);

$patientSet2->dateOfBirth->isDefinedAs('dob');
$patientSet2->dayOfBirth->isDefinedAs('dob','/',0);
$patientSet2->monthOfBirth->isDefinedAs('dob','/',1);
$patientSet2->yearOfBirth->isDefinedAs('dob','/',2);

$patientSet1->streetAddress->isDefinedAs('address');
$patientSet2->streetAddress->isDefinedAs('address',',',0);

$patientSet1->zipCode->isDefinedAs('zip_code');
$patientSet2->zipCode->isDefinedAs('address',',',3);

$patientSet2->city->isDefinedAs('address',',',1);

$patientSet2->state->isDefinedAs('address',',',2);



/////specifies the location of the attribute in the XML file
//(the route to access the object data)
$patientSet1->ssn->isAt('social_security');
$patientSet1->firstName->isAt('name');
$patientSet1->lastName->isAt('name');
$patientSet1->dateOfBirth->isAt('date_of_birth');
$patientSet1->gender->isAt('social_security');
$patientSet1->address->isAt('social_security');
$patientSet1->zipCode->isAt('social_security');
$patientSet1->city->isAt('social_security');
$patientSet1->state->isAt('social_security');

$patientSet2->ssn->isAt('social_security');
$patientSet2->firstName->isAt('name');
$patientSet2->lastName->isAt('name');
$patientSet2->dateOfBirth->isAt('date_of_birth');
$patientSet2->gender->isAt('social_security');
$patientSet2->address->isAt('social_security');
$patientSet2->zipCode->isAt('social_security');
```

```
$patientSet2->city->isAt('social_security');
$patientSet2->state->isAt('social_security');

//Specify the data source for the patient's collections:
$patientSet1->isAccessedFrom('dataSource1TextArea');
$patientSet2->isAccessedFrom('dataSource2TextArea');

//specify the base path (the parent route):
$patientSet1->hasAsRecordPath('/patients/patient');
$patientSet2->hasAsRecordPath('/hospital/physician/patient');


$patientSet1->state='New York';
$patientSet2->state='New York';
$patientSet1->state='NY';
$patientSet2->state='NY';
$patientSet1->state='FL';
$patientSet2->state='FL';
$patientSet1->state='Florida';
$patientSet2->state='Florida';
$patientSet1->streetAddress='1223 abc street';
$patientSet2->streetAddress='1223 abc street';
$patientSet1->dayOfBirth='05';
$patientSet2->dayOfBirth='03';
$patientSet1->firstName='John';
$patientSet2->firstName='John';
$patientSet1->lastName='Doe';
$patientSet2->lastName='Doe';
$patientSet1->firstName='Doe';
$patientSet1->lastName='John';
$patientSet2->ssn='123456';
$patientSet1->ssn='123456';
$patientSet1->all();
$patientSet1->firstName;
$patientSet2->firstName;
$patientSet1->dateOfBirth;
$patientSet2->dateOfBirth;
$patientSet2->dateOfBirth='03-05-1957';
$patientSet2->address='1223 abc street, new york city, NY, 12345';
                </textarea>
            </div>
            <div class="col-xs-6" id="outputDiv">
                <textarea class="form-control" name="inputTextArea"
id="outputTextArea" cols="30" rows="20" placeholder="Output"></textarea>
            </div>
        </div>
    </div>
    <br/>
    <div class="row">
        <div class="col-xs-2 col-xs-offset-5">
            <button id="runButton">Run</button>
        </div>
    </div>

</body>
</html>
```

7.4 XML Files for case Study

7.4.1 Patients.xml

```
<patients>
        <patient>
                <name>John Doe</name>
                <gender>Male</gender>
                <ssn>123456</ssn>
                <date_of_birth>03/05/1957</date_of_birth>
                <address>1223 abc street</address>
                <zip_code>12345</zip_code>
                <city>New York City</city>
                <state>New York</state>
        </patient>
        <patient>
                <name>Sarah Smith</name>
                <gender>Female</gender>
                <ssn>111111</ssn>
                <date_of_birth>05/12/1942</date_of_birth>
                <address>23 palmetto dr</address>
                <zip_code>13245</zip_code>
                <city>Los Angeles</city>
                <state>California</state>
        </patient>
        <patient>
                <name>Brad Pitt</name>
                <gender>Male</gender>
                <ssn>854675</ssn>
                <date_of_birth>08/23/1967</date_of_birth>
                <address>879 d street</address>
                <zip_code>56478</zip_code>
                <city>Buffalo</city>
                <state>New York</state>
        </patient>
        <patient>
                <name>Diana Shell</name>
                <gender>Female</gender>
                <ssn>456548</ssn>
                <date_of_birth>01/12/1976</date_of_birth>
                <address>12 25th av</address>
                <zip_code>47857</zip_code>
                <city>Orlando</city>
                <state>Florida</state>
        </patient>
                <patient>
                <name>Ann Taylor</name>
                <gender>Female</gender>
                <ssn>244321</ssn>
                <date_of_birth>12/12/1987</date_of_birth>
                <address>233 clinton av</address>
                <zip_code>77777</zip_code>
                <city>Miami</city>
                <state>Florida</state>
        </patient>
</patients>
```

7.4.2 Patients2.xml

```
<hospital>
        <city>Orlando</city>
        <name>Orlando General Hospital</name>
        <physician>
                <name>Eric Moore</name>
                <patient>
                        <first_name>John</first_name>
                        <middle_initial></middle_initial>
                        <last_name>Doe</last_name>
                        <dob>03/05/1957</dob>
                        <social_security>123456</social_security>
                        <address>1223 abc street, new york city, NY
12345</address>
                </patient>
                <patient>
                        <first_name>Diana</first_name>
                        <middle_initial>J</middle_initial>
                        <last_name>Shell</last_name>
                        <dob>01/12/1976</dob>
                        <social_security>456548</social_security>
                        <address>12 25th av, orlando, FL 47857</address>
                </patient>
        </physician>
        <physician>
                <name>Ann Bloom</name>
                <patient>
                        <first_name>Sara</first_name>
                        <middle_initial></middle_initial>
                        <last_name>Smith</last_name>
                        <dob>05/12/1942</dob>
                        <social_security>111111</social_security>
                        <address>123 palmeto rd, los Angeles, CA 13245</address>
                </patient>
                <patient>
                        <first_name>William</first_name>
                        <middle_initial>J</middle_initial>
                        <last_name>Oconnel</last_name>
                        <dob>01/27/1946</dob>
                        <social_security>22222</social_security>
                        <address>57s adaq blv, dallas, TX 54741</address>
                </patient>
                <patient>
                        <first_name>Jack</first_name>
                        <middle_initial></middle_initial>
                        <last_name>Johnson</last_name>
                        <dob>09/12/1969</dob>
                        <social_security>33333</social_security>
                        <address>999 abc blv, fairview, NJ 07425</address>
```

```
                 </patient>
        </physician>
</hospital>
```

7.5 Java Files for case Study

7.5.1 Main.java

```
import GUI.MainForm;

/**
 * Created by carlos on 24/04/2015.
 */
public class Main {

    public static void main(String Args[]) {

        MainForm form = new MainForm();

       /* System.out.println("hola al mundo");
        System.out.println(SingletonDB.getDB1().get(1).getName());
        System.out.println(SingletonDB.getDB1().get(1).getGender());
        System.out.println(SingletonDB.getDB1().get(1).getSsn());
        System.out.println(SingletonDB.getDB1().get(1).getDate_of_birth());
        System.out.println(SingletonDB.getDB1().get(1).getAddress());
        System.out.println(SingletonDB.getDB1().get(1).getZip_code());
        System.out.println(SingletonDB.getDB1().get(1).getCity());
        System.out.println(SingletonDB.getDB1().get(1).getState());*/

        //System.out.println(SingletonDB.getDB2().getCity());
    }

}
```

7.5.2 hospital.java

```
package controller;

import java.util.List;

import javax.xml.bind.annotation.XmlAttribute;

import javax.xml.bind.annotation.XmlElement;

import javax.xml.bind.annotation.XmlRootElement;

/**
```

```
 * Created by carlos1 on 4/24/15.
 */
@XmlRootElement
public class hospital {
    private String city;
    private String name;
  private List<Physician> physician;
    public hospital() {
    }
    public hospital(String city, String name, List<Physician> physician) {
        this.city = city;
        this.name = name;
        this.physician = physician;
    }
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
   public List<Physician> getPhysician() {
        return physician;
    }

    public void setPhysician(List<Physician> physician) {

        this.physician = physician;

    }

}
```

7.5.3 patient.java

```
package controller;

/**

 * Created by carlos on 24/04/2015.

 */

public class patient {

    private String name;

    private String gender;

    private String ssn;

    private String date_of_birth;

    private String address;

    private String zip_code;

    private String city;

    private String state;

    public patient(String name, String gender, String ssn, String date_of_birth,
String address, String zip_code, String city, String state) {

        this.name = name;

        this.gender = gender;

        this.ssn = ssn;

        this.date_of_birth = date_of_birth;

        this.address = address;
        this.zip_code = zip_code;

        this.city = city;

        this.state = state;

    }

    public patient() {

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getGender() {

        return gender;

    }

    public void setGender(String gender) {

        this.gender = gender;

    }

    public String getSsn() {

        return ssn;

    }

    public void setSsn(String ssn) {

        this.ssn = ssn;

    }

    public String getDate_of_birth() {

        return date_of_birth;

    }

    public void setDate_of_birth(String date_of_birth) {
        this.date_of_birth = date_of_birth;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public String getZip_code() {
        return zip_code;
    }
    public void setZip_code(String zip_code) {
        this.zip_code = zip_code;
    }
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public String getState() {
        return state;
    }
    public void setState(String state) {
        this.state = state;
    }
    public String getStateInits() {
        String[] str3A = getState().trim().split(" ");
        if(str3A.length == 1) {
            return str3A[0].substring(0,2).toUpperCase();
        }
        return               str3A[0].substring(0,1).toUpperCase()            +
str3A[1].substring(0,1).toUpperCase();
    }
}
```

7.5.4 Patient2.java

```
package controller;
/**
 * Created by carlos1 on 4/24/15.
 */
public class Patient2 {
    private String first_name;
    private String middle_initial;
    private String last_name;
    private String dob;
    private String social_security;
    private String address;
    public Patient2() {
    }
    public Patient2(String first_name, String middle_initial, String last_name,
String dob, String social_security, String address) {
        this.first_name = first_name;
        this.middle_initial = middle_initial;
        this.last_name = last_name;
    this.dob = dob;

    this.social_security = social_security;

    this.address = address;

}

public String getFirst_name() {

    return first_name;

}

public void setFirst_name(String first_name) {

    this.first_name = first_name;

}

public String getMiddle_initial() {

    return middle_initial;

}

public void setMiddle_initial(String middle_initial) {

    this.middle_initial = middle_initial;

}

public String getLast_name() {

    return last_name;

}

public void setLast_name(String last_name) {

    this.last_name = last_name;

}

public String getDob() {

    return dob;

}

public void setDob(String dob) {

    this.dob = dob;

}
```

```
    public String getSocial_security() {

        return social_security;

    }

    public void setSocial_security(String social_security) {

        this.social_security = social_security;

    }

    public String getAddress() {

        return address;

    }

    public void setAddress(String address) {

        this.address = address;

    }

    public String getFullName() {

        if(getMiddle_initial().isEmpty()) {

            return getFirst_name() + " " + getLast_name();

        } else {

            return  getFirst_name()  +  "  "  +  getMiddle_initial()  +  "  "  +
getLast_name();

        }

    }

    public String getState() {

        String[] str1A = getAddress().split(",");

        String[] str2A = str1A[2].trim().split(" ");

        return str2A[0];

    }

    public String getSingleAdress() {

        String[] str1A = getAddress().split(",");

        return str1A[0].trim();
    }

    public String getCity() {

        String[] str1A = getAddress().split(",");

        return str1A[1];

    }

    public String getZipCode() {

        String[] str1A = getAddress().split(",");

        String[] str2A = str1A[2].trim().split(" ");

        return str2A[1].trim();

    }

}
```

7.5.5 patients.java

```
package controller;

import java.util.List;

import javax.xml.bind.annotation.XmlAttribute;

import javax.xml.bind.annotation.XmlElement;

import javax.xml.bind.annotation.XmlRootElement;

/**

 * Created by carlos on 24/04/2015.

 */

@XmlRootElement

public class patients {

    private List<patient> patient;

    public patients(List<patient> patient) {

        this.patient = patient;

    }

    public patients() {
    }

    public List<patient> getPatient() {

        return patient;

    }

    public void setPatient(List<patient> patient) {

        this.patient = patient;

    }

}
```

7.5.6 Physician.java

```
package controller;

import java.util.List;

/**

 * Created by carlos1 on 4/24/15.

 */

public class Physician {

    private String name;

    private List<Patient2> patient;

    public Physician() {

    }

    public Physician(String name, List<Patient2> patient) {

        this.name = name;

        this.patient = patient;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {
        this.name = name;

    }

    public List<Patient2> getPatient() {

        return patient;

    }

    public void setPatient(List<Patient2> patient) {

        this.patient = patient;

    }

}
```

7.5.7 SingletonDB.java

```
package controller; /**

 * Created by carlos on 24/04/2015.

 */

import controller.hospital;

import controller.patients;

import controller.patient;

import javax.xml.bind.JAXBContext;

import javax.xml.bind.JAXBException;

import javax.xml.bind.Marshaller;

import javax.xml.bind.Unmarshaller;

import java.io.File;

import java.util.List;

public abstract class SingletonDB {

    //private            static            String            file1Path            =

"C:\\Users\\carlos\\Documents\\t1\\patients.xml";

    private final static String file1Path = "xml/patients.xml";
    private final static String file2Path = "xml/patients2.xml";

    private static patients patientsList;

    private static hospital hospitalDB;

    private static List<patient> DB1;

    static {

        loadDB();

    }

    public static void loadDB() {

        try {

            {

                File file1 = new File(file1Path);

                JAXBContext                     jaxbContext                     =
JAXBContext.newInstance(patients.class);

                Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();

                patientsList = (patients) jaxbUnmarshaller.unmarshal(file1);

            }

            {

                File file2 = new File(file2Path);

                JAXBContext                     jaxbContext                     =
JAXBContext.newInstance(hospital.class);

                Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();

                hospitalDB = (hospital) jaxbUnmarshaller.unmarshal(file2);

            }

        } catch (JAXBException e) {

            e.printStackTrace();

        }

    }

    public static List<patient>  getDB1() {
        return patientsList.getPatient();

    }

    public static hospital getDB2() {

        return hospitalDB;

    }

    public static void saveDB() throws JAXBException {

        {

            File file = new File(file1Path);

            JAXBContext jaxbContext = JAXBContext.newInstance(patients.class);

            Marshaller jaxbMarshaller = jaxbContext.createMarshaller();

            jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,true);

            jaxbMarshaller.marshal(patientsList,file);

        }

        {

            File file = new File(file2Path);

            JAXBContext jaxbContext = JAXBContext.newInstance(hospital.class);

            Marshaller jaxbMarshaller = jaxbContext.createMarshaller();

            jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,true);

            jaxbMarshaller.marshal(hospitalDB,file);

        }

        loadDB();

    }

}
```

7.5.8 MainForm.java

```
package GUI;

import controller.Patient2;
import controller.Physician;

import controller.SingletonDB;

import controller.patient;

import javax.swing.*;

import javax.xml.bind.JAXBException;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

/**

 * Created by carlos on 24/04/2015.

 */

public class MainForm extends JFrame{

    private JTabbedPane tabs;

    private JButton buscarButton;

    private JTextField PKtexField;

    private JComboBox FilecomboBox;

    private JTextArea OputtextArea;

    private JTextField pktextFieldCC;

    private JButton Compararbutton;

    private JTextArea OputtextArea2;

    private JButton BuscarFaltantesbutton;

    private JTextField ssntextField;

    private JTextField nametextField;

    private JTextField dobtextField;

    private JTextField adresstextField;

    private JButton BuscarbuttonAP;

    private JCheckBox archivo1CheckBox;

    private JCheckBox archivo2CheckBox;
    private JButton Updatebutton;
    private JLabel archivosLabel;
    private JTextField codigoZipField;
    private JTextField ciudadield;
    private JTextField estadoField;
    private JTextArea FaltantestextArea;
    private JButton AgregarFalantesbutton;
    private JComboBox ArchivoscomboBox;
    private JButton BuscarFbutton;
    private JTextArea ArchivostextArea;
    private JButton borrarDeArchivoSelecionadoButton;
    private JButton borrarDeAmbosArchivosButton;
    private JFrame self;
    patient p1 = null;
    Patient2 p2 = null;
    ArrayList<patient> faltantes1;
    ArrayList<Patient2> faltantes2;
    public String generateStateIt(String str) {
        //String[] strA = str.trim().split(" ");
        switch(str.trim().toUpperCase()) {
            case "CA" : return "California";
            case "LF" : return "Florida";
            case "NY" : return "New York";
            case "TX" : return "Texas";
            case "NJ" : return "New Jersey";
            default : return str;
        }
    }
```

```
    public String generateState(String str) {
        String[] str3A = str.trim().split(" ");
        if(str3A.length == 1) {
            return str3A[0].substring(0,2).toUpperCase();
        }
        return               str3A[0].substring(0,1).toUpperCase()               +
str3A[1].substring(0,1).toUpperCase();
    }
    public patient S2toS1(Patient2 elem) {
        patient temp = new patient();
        temp.setName(elem.getFullName().trim());
        temp.setAddress(elem.getSingleAdress().trim());
        temp.setDate_of_birth(elem.getDob());
        temp.setCity(elem.getCity().trim());
        temp.setGender("Indefinido");
        temp.setSsn(elem.getSocial_security());
        temp.setZip_code(elem.getZipCode().trim());
        temp.setState(generateStateIt(elem.getState().trim()));
        return temp;
    }
    public ArrayList<patient> faltantesSitema2() {
        ArrayList<patient> faltantes1 = new ArrayList<>();
        for (patient elem : SingletonDB.getDB1()) {
            boolean falta = true;
            for (Physician elem2 : SingletonDB.getDB2().getPhysician()) {
                for (Patient2 elem3 : elem2.getPatient()) {
                    if (elem.getSsn().equals(elem3.getSocial_security())) {
                        falta = false;
                        break;
                    }
                }
            }
            if (falta) faltantes1.add(elem);
        }
        return  faltantes1;
    }
    public ArrayList<Patient2> faltatnesSistema1() {
        ArrayList<Patient2> faltantes2 = new ArrayList<Patient2>();
        for (Physician elem : SingletonDB.getDB2().getPhysician()) {
            for (Patient2 elem2 : elem.getPatient()) {
                boolean falta = true;
                for (patient elem3 : SingletonDB.getDB1()) {
                    if (elem3.getSsn().equals(elem2.getSocial_security())) {
                        falta = false;
                        break;
                    }
                }
                if (falta) faltantes2.add(elem2);
            }
        }
        return faltantes2;
    }
    public ArrayList<patient> enAmbos() {
        ArrayList<patient> ambos = new ArrayList<patient>();
        for(patient elem : SingletonDB.getDB1()) {
            for(Physician elem2 : SingletonDB.getDB2().getPhysician()) {
                for(Patient2 elem3 : elem2.getPatient()) {

                    if(elem.getSsn().equals(elem3.getSocial_security())) {

                        ambos.add(elem);

                    }

                }

            }

        }

        return ambos;

    }

    public Patient2 S1toS2(patient elem) {

        Patient2 temp = new Patient2();

        String str = elem.getAddress() + ", " + elem.getCity().toLowerCase() + ",
" + generateState(elem.getState().trim()) + " " + elem.getZip_code().trim();

        temp.setAddress(str);

        temp.setDob(elem.getDate_of_birth());

        String[] temp2 = elem.getName().trim().split(" ");

        if(temp2.length == 3) {

            temp.setFirst_name(temp2[0]);

            temp.setMiddle_initial(temp2[1]);

            temp.setLast_name(temp2[2]);

        } else {

            temp.setFirst_name(temp2[0]);

            temp.setMiddle_initial("");

            temp.setLast_name(temp2[1]);

        }

        temp.setSocial_security(elem.getSsn());

        return temp;

    }
```

```
/*public String generateName(String str) {
}*/
public  MainForm() {
    super("ABC Hospital");
    self  = this;
    FilecomboBox.addItem("archivo 1");
    FilecomboBox.addItem("archivo 2");
    ArchivoscomboBox.addItem("pacientes en el sistema 1 y no el 2");
    ArchivoscomboBox.addItem("pacientes en el sistema 2 y no el 1");
    ArchivoscomboBox.addItem("pacientes en ambos sistemas");
    setContentPane(tabs);
    pack();
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(832, 624);
    setVisible(true);
    buscarButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            OputtextArea.setText("");
            if (FilecomboBox.getSelectedIndex() == 0) {
                patient p = null;
                for (patient elem : controller.SingletonDB.getDB1()) {
                    if (elem.getSsn().equals(PKtexField.getText().trim())) {
                        p = elem;
                        break;
                    }
                }
                if (p == null) {
                        JOptionPane.showMessageDialog(self,
                                "El numero no se ecuentra ningun paciente con la
nuemro de seguro social ingresado en la base de datos selecionada",
                                "Error de busqueda",
                                JOptionPane.ERROR_MESSAGE);
                        return;
                    }
                    OputtextArea.append("Nombre : " + p.getName() + "\n");
                    OputtextArea.append("Genero : " + p.getGender() + "\n");
                    OputtextArea.append("numero de seguro social : " + p.getSsn()
+ "\n");
                    OputtextArea.append("fecha    de    nacimiento    :    "    +
p.getDate_of_birth() + "\n");
                    OputtextArea.append("direccion : " + p.getAddress() + "\n");
                    OputtextArea.append("codigo  zip  :  "  +  p.getZip_code()  +
"\n");
                    OputtextArea.append("ciudad : " + p.getCity() + "\n");
                    OputtextArea.append("estado : " + p.getState());
                }
                if (FilecomboBox.getSelectedIndex() == 1) {
                    Patient2 p = null;
                    init:
                    for (Physician elem : SingletonDB.getDB2().getPhysician()) {
                        for (Patient2 elem2 : elem.getPatient()) {
                            if
(elem2.getSocial_security().equals(PKtexField.getText().trim())) {
                                p = elem2;
                                break init;
                            }
                        }
                    }
                    if (p == null) {
                        JOptionPane.showMessageDialog(self,
                                "El numero no se ecuentra ningun paciente con la
nuemro de seguro social ingresado en la base de datos selecionada",
                                "Error de busqueda",
                                JOptionPane.ERROR_MESSAGE);
                        return;
                    }
                    OputtextArea.append("Name : " + p.getFirst_name() + " " +
p.getMiddle_initial() + " " + p.getLast_name() + "\n");
                    OputtextArea.append("Fecha de nacimiento : " + p.getDob() +
"\n");
                    OputtextArea.append("Numero de sguro social : " +
p.getSocial_security() + "\n");
                    OputtextArea.append("Direcion : " + p.getAddress() + "\n");
                }
            }
        });
        Compararbutton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                patient p1 = null;
                Patient2 p2 = null;
                OputtextArea2.setText("");
                init2:
```

```
                for (patient elem : controller.SingletonDB.getDB1()) {
                    if (elem.getSsn().equals(pktextFieldCC.getText().trim())) {
                        for              (Physician              elem2              :
SingletonDB.getDB2().getPhysician()) {
                            for (Patient2 elem3 : elem2.getPatient()) {
                                //if
(elem2.getSocial_security().equals(PKtexField.getText().trim())) {
                                //}
                                if
(elem3.getSocial_security().equals(elem.getSsn())) {
                                    p1 = elem;
                                    p2 = elem3;
                                    break init2;
                                }
                            }
                        }
                    }
                }
                if (p2 == null || p1 == null) {
                    JOptionPane.showMessageDialog(self,
                            "El  numero  de  seguro  social  no  se  encutra  en  uno  o
mas sitemas",
                            "Error de busqueda",
                            JOptionPane.ERROR_MESSAGE);
                    return;
                }
                boolean consistente = true;
                {
                    /*String str;
                    if(p2.getMiddle_initial().isEmpty()) {
                        str = p2.getFirst_name() + " " + p2.getLast_name();
                    } else {
                        str = p2.getFirst_name() + " " + p2.getMiddle_initial() +
" " + p2.getLast_name();
                    }*/
                    if (!p2.getFullName().equals(p1.getName())) {
                        OputtextArea2.append("Los nombres no conciden \n");
                        OputtextArea2.append(p2.getFullName() + " es difenrente
de " + p1.getName() + "\n\n");
                        consistente = false;
                    }
                }
                {
                    if (!p1.getDate_of_birth().equals(p2.getDob())) {
                        OputtextArea2.append("Las fechas de nacimiento no
concienden \n");
                        OputtextArea2.append(p1.getDate_of_birth() + " es
difenrente de " + p2.getDob() + "\n\n");
                        consistente = false;
                    }
                }
                {
                    /*String[] str1A = p2.getAddress().split(",");
                    String[] str2A = str1A[2].trim().split(" ");//contiene la
direccion y el estado
                    String[] str3A = p1.getState().trim().split(" ");*/
                    if (!p1.getStateInits().equals(p2.getState())) {
                        OputtextArea2.append("Los estados no conciden \n");
                        OputtextArea2.append(p1.getStateInits()  +  "  es  diferente
de " + p2.getState());
                        consistente = false;
                    }
                    if (!p1.getAddress().equals(p2.getSingleAdress())) {
                        OputtextArea2.append("las direccionesn conciden \n");
                        OputtextArea2.append(p1.getAddress() + " es diferente " +
p2.getSingleAdress());
                        consistente = false;
                    }
                    if
(!p1.getCity().toUpperCase().equals(p2.getCity().trim().toUpperCase())) {
                        OputtextArea2.append("las ciudades no conciden \n");
                        OputtextArea2.append(p1.getCity()  +  "  es  diferente  "  +
p2.getCity());
                        consistente = false;
                    }
                    if (!p1.getZip_code().equals(p2.getZipCode())) {
                        OputtextArea2.append("los codigos zip no conciden \n");
                        OputtextArea2.append(p1.getZip_code()  +  "  es  diferente  "
+ p2.getZipCode());
                        consistente = false;
                    }
                }
                if (consistente) {
                    OputtextArea2.append("todo es concistente");
                }

            }

        });

        BuscarFaltantesbutton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                FaltantestextArea.setText("");

                faltantes1 = faltantesSitema2();

                faltantes2 = faltatnesSistema1();

                /*for (patient elem : SingletonDB.getDB1()) {

                    boolean falta = true;

                    for (Physician elem2 : SingletonDB.getDB2().getPhysician()) {

                        for (Patient2 elem3 : elem2.getPatient()) {

                            if  (elem.getSsn().equals(elem3.getSocial_security()))
{

                                falta = false;

                                break;

                            }

                        }

                    }

                    if (falta) faltantes1.add(elem);

                }*/

                /*for (Physician elem : SingletonDB.getDB2().getPhysician()) {

                    for (Patient2 elem2 : elem.getPatient()) {

                        boolean falta = true;

                        for (patient elem3 : SingletonDB.getDB1()) {

                            if
(elem3.getSsn().equals(elem2.getSocial_security())) {
```

```
                            falta = false;
                            break;
                        }
                    }
                    if (falta) faltantes2.add(elem2);
                }
            }*/
            if (faltantes1.isEmpty() && faltantes2.isEmpty()) {
                JOptionPane.showMessageDialog(self,
                        "No hay pacientes faltantes");
            }
            for (patient elem : faltantes1) {
                FaltantestextArea.append(elem.getSsn() + "(" + elem.getName()
+ ") Falta en el sistema 2\n");
            }
            FaltantestextArea.append("\n\n");
            for (Patient2 elem : faltantes2) {
                FaltantestextArea.append(elem.getSocial_security()  +  "("  +
elem.getFullName() + ") Falta en el sistema 1\n");
            }
        }
    });
    BuscarbuttonAP.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            p1 = null;
            p2 = null;
            archivo1CheckBox.setEnabled(false);
```

```
                archivo2CheckBox.setEnabled(false);
                for(patient elem : SingletonDB.getDB1()) {
                    if(elem.getSsn().equals(ssntextField.getText().trim())) {
                        p1 = elem;
                        break;
                    }
                }
                init3 :
                for(Physician elem : SingletonDB.getDB2().getPhysician()) {
                    for(Patient2      elem2      :      elem.getPatient())      {
if(elem2.getSocial_security().equals(ssntextField.getText().trim())) {
                            p2 = elem2;
                            break init3;
                        }
                    }
                }
                if(p1 == null && p2 == null) {
                    JOptionPane.showMessageDialog(self,
                            "El numero de seguridad social ingresado no se enctra
registrado en el sistema",
                            "Error de busqueda",
                            JOptionPane.ERROR_MESSAGE);
                    return;
                }
                if(p1 != null) {
                    archivo1CheckBox.setEnabled(true);
                    ssntextField.setText(p1.getSsn());
                    nametextField.setText(p1.getName());
                    dobtextField.setText(p1.getDate_of_birth());

                    adresstextField.setText(p1.getAddress());

                    codigoZipField.setText(p1.getZip_code());

                    ciudadield.setText(p1.getCity());

                    estadoField.setText(p1.getState());

                }

                if(p2 != null) {

                    archivo2CheckBox.setEnabled(true);

                    if(p1 == null) {

                        ssntextField.setText(p2.getSocial_security());

                        nametextField.setText(p2.getFullName());

                        dobtextField.setText(p2.getDob());

                        adresstextField.setText(p2.getSingleAdress());

                        codigoZipField.setText(p2.getZipCode());

                        ciudadield.setText(p2.getCity());

                        estadoField.setText(generateStateIt(p2.getState()));

                    }

                }

            }

        });

        Updatebutton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                boolean update = false;

                if(archivo1CheckBox.isSelected()) {

                    p1.setSsn(ssntextField.getText().trim());

                    p1.setName(nametextField.getText().trim());

                    p1.setDate_of_birth(dobtextField.getText().trim());
```

```
                    p1.setAddress(adresstextField.getText().trim());
                    p1.setZip_code(codigoZipField.getText().trim());
                    p1.setState(estadoField.getText().trim());
                    p1.setCity(ciudadield.getText().trim());
                    update = true;
                }
                if(archivo2CheckBox.isSelected()) {
                    p2.setSocial_security(ssntextField.getText().trim());
                    String[] temp = nametextField.getText().trim().split(" ");
                    if(temp.length == 3) {
                        p2.setFirst_name(temp[0]);
                        p2.setMiddle_initial(temp[1]);
                        p2.setLast_name(temp[2]);
                    } else {
                        p2.setFirst_name(temp[0]);
                        p2.setMiddle_initial("");
                        p2.setLast_name(temp[1]);
                    }
                    p2.setDob(dobtextField.getText().trim());
                    String str = adresstextField.getText().trim() + ", " +
ciudadield.getText().trim().toLowerCase() + ", " +
generateState(estadoField.getText().trim()) + " " +
codigoZipField.getText().trim();
                    p2.setAddress(str);
                    update = true;
                }
                if(update) {
                    try {
                        SingletonDB.saveDB();
                        JOptionPane.showMessageDialog(self,
                                "La    base    de    datos    ha    sido    actulizada
correctamente");
                    } catch (JAXBException e1) {
                        JOptionPane.showMessageDialog(self,
                                "Error al guardar la base de datos",
                                "Error grave",
                                JOptionPane.ERROR_MESSAGE);
                        e1.printStackTrace();
                    }
                }
            }
        });
        AgregarFalantesbutton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if(!faltantes1.isEmpty()) {
                    Physician temp = new Physician();
                    temp.setName(" ");
                    ArrayList<Patient2> lista = new ArrayList<Patient2>();
                    for(patient elem : faltantes1) {
                        lista.add(S1toS2(elem));
                    }
                    temp.setPatient(lista);
                    SingletonDB.getDB2().getPhysician().add(temp);
                }
                for(Patient2 elem : faltantes2) {
                    //ArrayList<p>
                    SingletonDB.getDB1().add(S2toS1(elem));
                }
                try {
                    SingletonDB.saveDB();
                    JOptionPane.showMessageDialog(self,
                            "La base de datos ha sido actulizada correctamente");
                } catch (JAXBException e1) {
                    JOptionPane.showMessageDialog(self,
                            "Error al guardar la base de datos",
                            "Error grave",
                            JOptionPane.ERROR_MESSAGE);
                    e1.printStackTrace();
                }
                SingletonDB.loadDB();
            }
        });
        BuscarFbutton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                ArchivostextArea.setText("");
                switch (ArchivoscomboBox.getSelectedIndex()) {
                    case 0 : {
                        for(patient elem : faltantesSitema2()) {
                            ArchivostextArea.append("Nombre  :  "  +  elem.getName()
+ "\n");
                            ArchivostextArea.append("Genero       :       "       +
elem.getGender() + "\n");
```

```
                            ArchivostextArea.append("numero de seguro social : "
+ elem.getSsn() + "\n");
                            ArchivostextArea.append("fecha de nacimiento : " +
elem.getDate_of_birth() + "\n");
                            ArchivostextArea.append("direccion : " +
elem.getAddress() + "\n");
                            ArchivostextArea.append("codigo zip : " +
elem.getZip_code() + "\n");
                            ArchivostextArea.append("ciudad : " + elem.getCity()
+ "\n");
                            ArchivostextArea.append("estado : " + elem.getState()
+ "\n\n");
                        }
                        break;
                    }
                    case 1 : {
                        for(Patient2 elem : faltatnesSistema1()) {
                            ArchivostextArea.append("Name : " +
elem.getFirst_name() + " " + elem.getMiddle_initial() + " " + elem.getLast_name()
+ "\n");
                            ArchivostextArea.append("Fecha de nacimiento : " +
elem.getDob() + "\n");
                            ArchivostextArea.append("Numero de sguro social : " +
elem.getSocial_security() + "\n");
                            ArchivostextArea.append("Direcion : " +
elem.getAddress() + "\n\n");
                        }
                        break;
                    }
                    default: {
                        for(patient elem : enAmbos()) {
                            ArchivostextArea.append("Nombre : " + elem.getName()
+ "\n");
                            ArchivostextArea.append("Genero : " +
elem.getGender() + "\n");
                            ArchivostextArea.append("numero de seguro social : "
+ elem.getSsn() + "\n");
                            ArchivostextArea.append("fecha de nacimiento : " +
elem.getDate_of_birth() + "\n");
                            ArchivostextArea.append("direccion : " +
elem.getAddress() + "\n");
                            ArchivostextArea.append("codigo zip : " +
elem.getZip_code() + "\n");
                            ArchivostextArea.append("ciudad : " + elem.getCity()
+ "\n");
                            ArchivostextArea.append("estado : " + elem.getState()
+ "\n\n");
                        }
                        break;
                    }
                }
            }
        });
        borrarDeArchivoSelecionadoButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int index = -1;
                int indexi = -1;
                int indexj = -1;
                if(FilecomboBox.getSelectedIndex() == 0) {
                    for(int  i  =  0;  i  <  SingletonDB.getDB1().size();  i++)  {
if(SingletonDB.getDB1().get(i).getSsn().equals(PKtexField.getText().trim())) {
                            index = i;
                            indexi = 0;
                            indexj = 0;
                            break;
                        }
                    }
                    if(index != -1) SingletonDB.getDB1().remove(index);
                }
                if(FilecomboBox.getSelectedIndex() == 1) {
                    for(int          i          =          0;          i          <
SingletonDB.getDB2().getPhysician().size(); i++) {

                        for(int          j          =          0;          j          <
SingletonDB.getDB2().getPhysician().get(i).getPatient().size();        j++)        {
if(SingletonDB.getDB2().getPhysician().get(i).getPatient().get(j).getSocial_secur
ity().equals(PKtexField.getText().trim())) {
                                index = 0;
                                indexi = i;
                                indexj = j;
                                break;
                            }
                        }
                    }
                    if(indexi      !=      -1      &&      indexj      !=      -1)
SingletonDB.getDB2().getPhysician().get(indexi).getPatient().remove(indexj);
                }
                if(index == -1 || indexi == -1 || indexj == -1) {
                    JOptionPane.showMessageDialog(self,
                            "No se ecuntra el paciente con el ide selecionado",
                            "Error",
                            JOptionPane.ERROR_MESSAGE);
                    return;
                }
                try {
                    SingletonDB.saveDB();
                    JOptionPane.showMessageDialog(self,
                            "La base de datos ha sido actulizada correctamente");
                } catch (JAXBException e1) {
                    JOptionPane.showMessageDialog(self,
                            "Error al guardar la base de datos",
                            "Error grave",
                            JOptionPane.ERROR_MESSAGE);
                    e1.printStackTrace();
                }
            }
        });
        borrarDeAmbosArchivosButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int index = -1;
                int indexi = -1;

                int indexj = -1;

                for(int i = 0; i < SingletonDB.getDB1().size(); i++) {
if(SingletonDB.getDB1().get(i).getSsn().equals(PKtexField.getText().trim())) {
                        index = i;
                        //indexi = 0;
                        //indexj = 0;
                        break;
                    }
                }
                for(int i = 0; i < SingletonDB.getDB2().getPhysician().size();
i++) {
                    for(int j = 0; j <
SingletonDB.getDB2().getPhysician().get(i).getPatient().size(); j++) {
if(SingletonDB.getDB2().getPhysician().get(i).getPatient().get(j).getSocial_secur
ity().equals(PKtexField.getText().trim())) {
                            //index = 0;
                            indexi = i;
                            indexj = j;
                            break;
                        }
                    }
                }
                if(index == -1 || indexi == -1 || indexj == -1) {
                    JOptionPane.showMessageDialog(self,
                            "El paciente no se cuentra en uno o mas archivos",
                            "Error",
                            JOptionPane.ERROR_MESSAGE);
                    return;
                }
SingletonDB.getDB2().getPhysician().get(indexi).getPatient().remove(indexj);
                SingletonDB.getDB1().remove(index);
                try {
                    SingletonDB.saveDB();
                    JOptionPane.showMessageDialog(self,
                            "La base de datos ha sido actulizada correctamente");
                } catch (JAXBException e1) {
                    JOptionPane.showMessageDialog(self,
                            "Error al guardar la base de datos",
                            "Error grave",
                            JOptionPane.ERROR_MESSAGE);
                    e1.printStackTrace();
                }
            }
        });
    }
}
```

7.6 SIL Approach for the case Study

```
$patient->isAnObject();
$patient2->isAnObject();
$patient2->hasA('first_name');
$patient2->hasA('middle_initial');
$patient2->hasA('last_name');
$patient2->hasA('dob');
```

```
$patient2->hasA('social_security');

$patient2->hasA('address');

$patient->hasA('name');

$patient->hasA('gender');

$patient->hasA('ssn');

$patient->hasA('date_of_birth');

$patient->hasA('address');

$patient->hasA('zip_code');

$patient->hasA('city');

$patient->hasA('state');

$patient->gender->value('Male')->isEquivalentTo('undefined');

$patient->gender->value('Female')->isEquivalentTo('undefined');

$patient->state->value('FL')->isEquivalentTo('Florida');

$patient->state->value('NY')->isEquivalentTo('New York');

$patient->state->value('CA')->isEquivalentTo('California');

$patient->state->value('TX')->isEquivalentTo('Texas');

$patient->state->value('NJ')->isEquivalentTo('New Jersey');

$patient->city->value('New York City')->isEquivalentTo('new york city');

$patient->city->value('Los Angeles')->isEquivalentTo('los Angeles');

$patient->city->value('Orlando')->isEquivalentTo('orlando');

$patients2->isBasedOn($patient2);

$patients->isBasedOn($patient);

$tempS1->isBasedOn($patients2);

$patients2->isAccessedFrom('patients2.xml');

$patients2->hasAsRecordPath('/hospital/physician');

$patients->isAccessedFrom('patients.xml');

$patients->hasAsRecordPath('/patients/patient');

$patients2->first_name->isAt('patient/first_name');
```

```
$patients2->middle_initial->isAt('patient/middle_initial');

$patients2->last_name->isAt('patient/last_name');

$patients2->dob->isAt('patient/dob');

$patients2->social_security->isAt('patient/social_security');

$patients2->address->isAt('patient/address');

$patients->name->isAt('name');

$patients->gender->isAt('gender');

$patients->social_security->isAt('ssn');

$patients->date_of_birth->isAt('date_of_birth');

$patients->address->isAt('address');

$patients->zip_code->isAt('zip_code');

$patients->city->isAt('city');

$patients->state->isAt('state');

$patients->ssn='ponerSSNAqui'->name->updateWith('nuevo nombre aqui');

$patients->ssn='ponerSSNAqui'->gender->updateWith('nuevo nombre aqui');

$patients->ssn='ponerSSNAqui'->ssn->updateWith('nuevo ssn aqui');

$patients->ssn='ponerSSNAqui'->date_of_birth->updateWith('nueva fecha de
nacimiento aqui');

$patients->ssn='ponerSSNAqui'->address->updateWith('nueva direccion aqui');

$patients->ssn='ponerSSNAqui'->zip_code->updateWith('nuevo codigo zip aqui');

$patients->ssn='ponerSSNAqui'->city->updateWith('nuevo city aqui');

$patients->ssn='ponerSSNAqui'->state->updateWith('nuevo state aqui');

$tempS1->first_name := $patients->name[0];

$tempS1->middle_initial := '';

$tempS1->last_name := $patients->name[1];

$tempS1->dob := $patients->date_of_birth;

$tempS1->social_security := $patients->ssn;
```

```
$tempS1->address := $patients->address->concatWith($patients->city->concatWith($patients->state->concatWith($patients->zip_code)));
$patients2->social_security='ponerSSNAqui'->updateWith($tempS1->social_security='ponerSSNAqui');
$tempS2->name := $patients2->first_name->concatWith($patients2->last_name);
$tempS2->gender := 'undefined';
$tempS2->ssn := $patients2->social_security;
$tempS2->date_of_birth := $patients2->dob;
$tempS2->address := $patients2->address[0];
$tempS2->zip_code := $patients2->address[3];
$tempS2->city := $patients2->address[1];
$tempS2->state := $patients2->address[2];
$tempS1->first_name := $patients->name[0];
$tempS1->middle_initial := '';
$tempS1->last_name := $patients->name[1];
$tempS1->dob := $patients->date_of_birth;
$tempS1->social_security := $patients->ssn;
$tempS1->address := $patients->address->concatWith($patients->city->concatWith($patients->state->concatWith($patients->zip_code)));
$patients->add($tempS2->notIn($patients));
$patients2->add($tempS1->notIn($patients2));
$patients->remove($patients->ssn='ponerSSNAqui');
$patients2->remove($patients2->social_security='ponerSSNAqui');
$tempS2->name := $patients2->first_name->concatWith($patients2->last_name);
$tempS2->gender := 'undefined';
$tempS2->ssn := $patients2->social_security;
$tempS2->date_of_birth := $patients2->dob;
$tempS2->address := $patients2->address[0];
```

```
$tempS2->zip_code := $patients2->address[3];

$tempS2->city := $patients2->address[1];

$tempS2->state := $patients2->address[2];

$result := $patients->noIn($tempS2);
```

REFERENCES

Aho, A. V., Sethi, R., & Ullman, J. D. (1986). *Compilers: principles, techniques, and tools.* Boston, MA: Addison-Wesley Longman.

Aiken, A. (2014). *Stanford - Compilers*. Retrieved from Coursera: https://www.coursera.org/course/compilers

Arch-int, N., & Arch-int, S. (2013, December 15). Semantic Ontology Mapping for Interoperability of Learning Resource Systems using a rule-based reasoning approach. *Expert Systems with Applications, 40*(18), 7428–7443. doi:10.1016/j.eswa.2013.07.027

Bennett, J. P. (1990). *Introduction to Compiling Techniques.* Berkshire, England: McGraw-Hill.

Bergamaschi, S., Castano, S., & Vincini, M. (1999). Semantic Integration of Semistructured and Structured Data Sources. *ACM Sigmod Record*, *28*, 54-59.

Bikakis, N., Tsinaraki, C., Gioldasis, N., Stavrakantonakis, I., & Christodoulakis, S. (2013). *The XML and Semantic Web Worlds: Technologies, Interoperability and Integration. A survey of the State of the Art.* Springer.

Bouguettaya, A., Benatallah, B., & Elmagarmid, A. K. (2012). Interconnecting heterogeneous information systems. *Springer Science & Business Media*.

Chauhan, S. (2013). *Programming Languages - Design and Constructs.* Laxmi Publications.

Cui, Z., Jones, D., & O'Brien, P. (2002, March). Semantic B2B Integration: Issues in Ontology-based Approaches. *ACM SIGMOD Record, 31*, 43-48.

Curry, E., & Mossire, J. (2006). Works in Progress: The 2nd International Middleware Doctoral Symposium. *iEEE Distributed Systems Online, 7*(3).

Deus, H. F., Correa, M. C., Stanislaus, R., Miragaia, M., Maass, W., De Lencastre, H., . . . Almeida, J. S. (2011). S3QL: A distributed domain specific language for controlled semantic integration of life sciences data. *BMC bioinformatics 12, no. 1 (2011),* 285.

Dolin, R. H., & Alschuler, L. (2011). Approaching semantic interoperability in Health Level Seven. *Journal of the American Medical Informatics Association 18, no. 1,* 99-103.

Evans, E. (2004). *Domain Driven Design: Tackling Complexity in the Heart of Software.* Addison-Wesley.

Fennell, P. (2013, June 15-16). Extremes of XML. *XML London 2013*, 80-86. doi:10.14337/XMLLondon13.Fennell01

Freudenthal, M., & Cybernetica AS, T. (2009, September 04). Domain Specific Languages in a Customs Information System. *Software, IEEE, PP*(99). doi:10.1109/MS.2009.152

Gabbrielli, M., & Martini, S. (2010). How to Describe a Programming Language. In *Programming Languages: Principles and Paradigms* (pp. 27-55). London: Springer-Verlag. doi:10.1007/978-1-84882-914-5_2

Guarino, N., Carrara, M., & Giaretta, P. (1994). Formalizing Ontological Committment. *National Conference on Artificial Intelligence (AAAI-94)* (pp. 560-567). Seattle: Morgan-Kaufman.

Gustavo, A., Casati, F., Kuno, H., & Machiraju, V. (2004). *Web services: concepts, architectures and applications.*

Hadim, S., & Mohamed, N. (2006). Middleware challenges and approaches for wireless sensor networks. *IEEE Distributed*.

Haslhofer, B., & Klas, W. (2010). A survey of techniques for achieving metadata interoperability. *ACM Computing Surveys (CSUR) 42, no. 2 (2010),* 7.

Hendler, J., & Pardo, T. A. (2012, September 24). *A Primer on Machine Readability for Online Documents and Data*. Retrieved from https://www.data.gov/developers/blog/primer-machine-readability-online-documents-and-data

Ishida, T., Sasaki, Y., & Fukuhara, Y. (1991). Use of procedural programming languages for controlling production systems. *Seventh IEEE Conference on Artificial Intelligence Applications* (pp. 71-75). Miami Beach, FL: IEEE.

ISO/IEC. (2001). *ISO/IEC 9126-1: Software Engineering - Product Quality - Part 1: Quality Model, International Organization for Standardization.* Geneva.

ISO/IEC. (2006). *ISO/IEC 19757-3 Information technology -- Document Schema Definition Languages (DSDL) -- Part 3: Rule-based validation -- Schematron.* ISO/IEC, Switzerland.

Jacobsen, I., Christerson, M., Jonsson, P., & Overgaard, G. (1992). *Object Oriented Software Engineering.* Addison-Wesley ACM Press.

Kyu, Z. M., & Nyunt, T. T. (2009, October 4-6). Storing DTD-independent XML data in relational database. *IEEE Symposium on Industrial Electronics & Applications*, 197 - 202. doi:10.1109/ISIEA.2009.5356465

Linthicum, D. S. (2000). *Enterprise application integration.* Addison-Wesley Professional.

Lloyd, J. W. (1994). Practical Advantages of Declarative Programming. *Joint Conference on Declarative Programming*.

Manconi, A., & Rodriguez-Tomé, P. (2011). A Survey on Integrating Data in Bioinformatics. In M. Biba, & F. Xhafa (Eds), *Learning Structure and Schemas from Documents* (pp. 413-432). Berlin: Springer-Verlag.

Mernik, M., Heering, J., & Sloane, A. M. (2005, December). When and how to develop domain-specific languages. *ACM Computing Surveys, 37*(4), 316-344. doi:10.1145/1118890.1118892

Ouksel, A., & Sheth, A. (1999, March). Semantic Interoperability in Global Information Systems. *ACM SIGMOD Record*, 5-12.

Romer, T. H., Lee, D., Voelker, G. M., Wolman, A., Wong, W. A., Baer, J.-L., . . . Levy, H. M. (1986). The Structure and Performance of Interpreter. *Architectural Support for Programming Languages and Operating Systems*, 150-159.

Sangwhan, C., Abusharekh, A., & Abidi, S. S. (2015). Towards a'Big'Health Data Analytics Platform. *Big Data Computing Service and Applications (BigDataService), 2015 IEEE First International Conference on* (pp. 233-241). IEEE.

Theodoratos, D., & Sellis, T. (1997). Data Warehouse Configuration. *Proceedings of 23rd International Conference on Very Large Data Bases*, (pp. 126-135).

Visser, E. (2008). WebDSL: A case study in domain-specic language engineering. *Generative and Transformational Techniques in Software Engineering (GTTSE 2007).* Lecture Notes in Computer Science. Springer.

Widom, J. (1995). Research Problems in Data Warehousing. *4th International Conference Information and Knowledge Management*, (pp. 25-30).

Wile, D. (2004). Lessons learned from real DSL experiments. *Science of Computer Programming, 51*(3), 265-290.

World Wide Web Consortium. (2008, November 26). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Retrieved from http://www.w3.org/TR/xml/

Young, P., Berzins, V. G., & Luqi, J. (2002). Using an Object Oriented Model for Resolving Representational Differences between Heterogeneous Systems. *The 17th ACM Symposium on Applied Computing.* Madrid, Spain.