Project Number:	7PM- 266588
Project Title:	SISOB (An Observatorium for Science in Society based in Social Models)
Deliverable Type:	Report
Deliverable Number:	3.2
WP:	3
Version:	1
Date:	Month Day, Year
Title of Deliverable:	General Purpose Tool and API for Accessing Different Databases
Editor:	Eduardo Guzmán
Authors:	Eduardo Guzmán (UMA), Antonio Gallardo (CICE), Beatriz Barros (UMA), Daniel López (UMA).
Dissemination level:	PU / PP
Keywords:	Information extraction, bibliographic information, data sources, CVN
Abstract:	This deliverable presents the API developed in the project to access, retrieve, and manage information from the different sources of data that are currently being used, as well as from any other sources that could be used in the future. As mentioned in other deliverables, the interchange format chosen in the project is the CVN. Consequently, this API is able to interpret and process information structured according to that format, and can easily transform any data stored in the objects of the API into a set of CVN files. The deliverable also shows how this API has been used in different components of the SISOB system.

Capacities. Science in Society. Collaborative Project

Version history

Version	Date	Description
0.3	31/07/2012	API description.
0.5	30/08/2012	Addition of the connections with other WPs.
0.8	03/10/2012	Review of the contents
0.9	05/10/2012	English writing review

Executive Summary

Research activity information extraction is an essential first stage in the process of making inferences about the impact of Science in Society. The two former deliverables of WP 6 and the one of WP 5 reveal how this process has been managed in the project framework. Next step is the process of unifying all data from different sources in a single format able to be understood and processed by the rest of the components of the SISOB system. All these components are the technological implementation of the contributions made by the different partners of the project. Accordingly, we first needed a common format. After an in-depth study, the CVN was selected as the best choice for representing the curricular information and products of the researcher.

Translation from the data collected in the extraction process to an intermediate format (the CVN) requires an API able to manage and populate this CVN. Additionally, conceptual model defined in WP 2 (see deliverables 1 and 2 for more information) needs a technological representation and the proper mechanisms to turn information from CVN to the conceptual model and vice versa. Precisely, this deliverable introduces this API, showing the way in which it has been designed, i.e. a multilevel API comprising different libraries each one with a specific purpose.

Finally, this deliverable also describes how this API is used in the SISOB system. As shown in deliverable 6.2, this system has been designed, following the good practices recommended in software engineering, as a service-oriented architecture where each one of the components is decoupled from the rest. In this architecture, information flows through a bus and, for this reason, communication and thus data exchange is made using the API presented in this report.

Table of contents

Figures

1. Introduction

This deliverable presents the API developed in the project to access, retrieve and manage information from the different sources of data that are currently being used, as well as from any other sources that could be used in the future. As mentioned in other deliverables, the interchange format chosen in the project is the CVN. Consequently, we have constructed an API which is able to interpret and process information structured according to the CVN. Likewise, through this API one can easily transform any data stored in the objects of the API into a set of CVN files.

The CVN project was launched in 2006 by the Spanish Foundation for Science and Technology (FECYT). CVN is a strategic project for the creation of a common space for the integration and exchange of researchers' curricular information. Initially, CVN has been created to give solutions to these scenarios through the generation of a standardized format capable of updating, modeling, and integrating every curricular design that exists currently. As a result, we avoid effort duplication that would mean collecting new information for the creation of a new curriculum, in order to achieve cooperation based on the exchange of information among different agents and systems of R&D. For more information about CVN, see deliverable 3.1.

This deliverable will also show how the CVN API can be connected to the other components of the SISOB architecture (for more information see the released deliverables of WP 6) and how this API can generate data in the format of the Conceptual Model defined in WP 2 (more information in the released deliverables of WP 2).

Next section aims to sketch the workflow of information in the SISOB architecture, from the data sources to the representation, following the guidelines of the conceptual model that facilitate the data manipulation. Next, the CVN API is explained in detail focusing on its structure and different components. Section 4 connects the API with the components that use it in the SISOB system. Finally, Section 5 presents the conclusions.

2. Information Extraction and Access in the SISOB System

Information about the scientific production of researchers can be extracted from different data sources. Some of them are directly provided by different organizations such as universities, Ministries of Science, or any other research entities. However, others have to be extracted from external sources, such as the Web of Knowledge, Scopus, Citeseer, etc. Finally, additional sources of information are the web pages containing information about the scientific production such as the researchers' personal web pages, containing relevant information about their career and achievements.

Deliverable 6.1 explained how the process of extracting information is carried out in the SISOB system. As shown in the information workflow of the SISOB system

represented in Figure 1, once the data have been extracted, they are first managed using the CVN API classes. From the CVN format, information can be turned into the Conceptual Model API, which provides the data containers and operations needed to properly process these data. The information in the Conceptual Model format could be temporarily stored in databases to facilitate its manipulation and analysis.

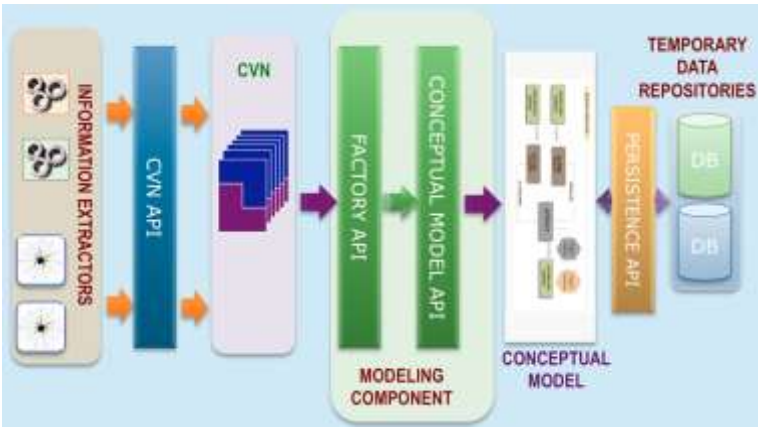

Figure 1. Information Extraction and Management in the SISOB system.

3. CVN API

This section describes the CVN API, which has been designed as a layered set of libraries. Figure 2 summarizes graphically its structure, which is made of seven different layers, or packages. . In general, the elements of this API are accessed through the `Factory` layer. It brings access to these elements whose functionalities are specified in the `Interface` layer. The rest of layers are mainly organized in two general classes, i.e. `Cvn` and `CvnItem`, and another three layers of complementary classes (`Entities`, `Auxiliary Entities` and `Input Data`). Finally, a layer of views, `Curricular Item Views`, has been defined to cook the data according to the needs of the different components of the SISOB system.

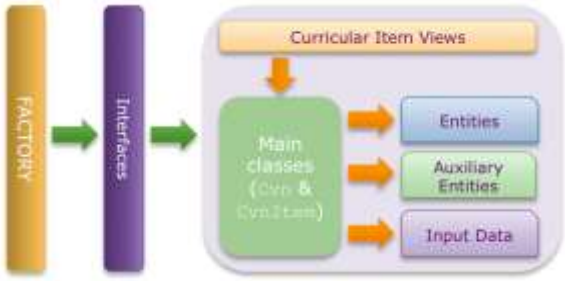

Figure 2. CVN API Structure.

The following subsections explain in detail the main classes and interfaces, and their methods.

3.1. The Main Classes: `Cvn` and `CvnItem`

Figure 3. CVN API main classes.

The principal components of the curriculum, i.e. the researcher and the curricular item, are modeled mainly through four classes: `Cvn`, `CvnBean`, `CvnItem`, and `CvnItemBean`. As can be seen in Figure 3, both `CvnBean` and `CvnItemBean` are data containers for storing curricular information. The `CvnBean` summarizes all information from a researcher, while `CvnItemBean` is more specific since it contains only data about one curricular element. Accordingly, the `CvnBean` contains collection of `CVNItemBean` instances. The other two classes, i.e. the `Cvn` and `CvnItem` inherit from `CvnBean` and `CvnItemBean`, respectively, adding to them functions that process the data of stored by their antecessors.

Below, each one of the classes included in Figure 3 are explained in detail:

- **CvnBean** class is a data container storing all the information related to a researcher. Accordingly, it contains a collection with all the curricular items of the researcher and also his/her personal information through an instance of a class called `Agent`.
- **Cvn** class has a set of operations that allow accessing the curricular items in different ways, depending on the type of item. Consequently, this class makes the access to the information easier, providing specific operations to obtain the data directly. For instance, it offers a method that returns the set of a researcher's patents, and another one that lists the conferences which (s)he attended , etc. It also provides methods for managing curricular items and, finally, methods for saving the researcher's data.
- **CvnItemBean** class models an item or element of the CVN, for example, a paper, publication, project, etc.
- **CvnItem** provides methods for adding properties to the `CvnItem` (for example, add a description, language, etc.), and also methods that return a specific view of a curricular item (for example `getStay` returns a view of a research stay made by the researcher). Thus, `CvnItem` is a generalization that is valid to represent any kind of curricular element, and the particularization of each type of item is made through viewing classes, which will be explained later in this document.

3.2. Entities of CVN

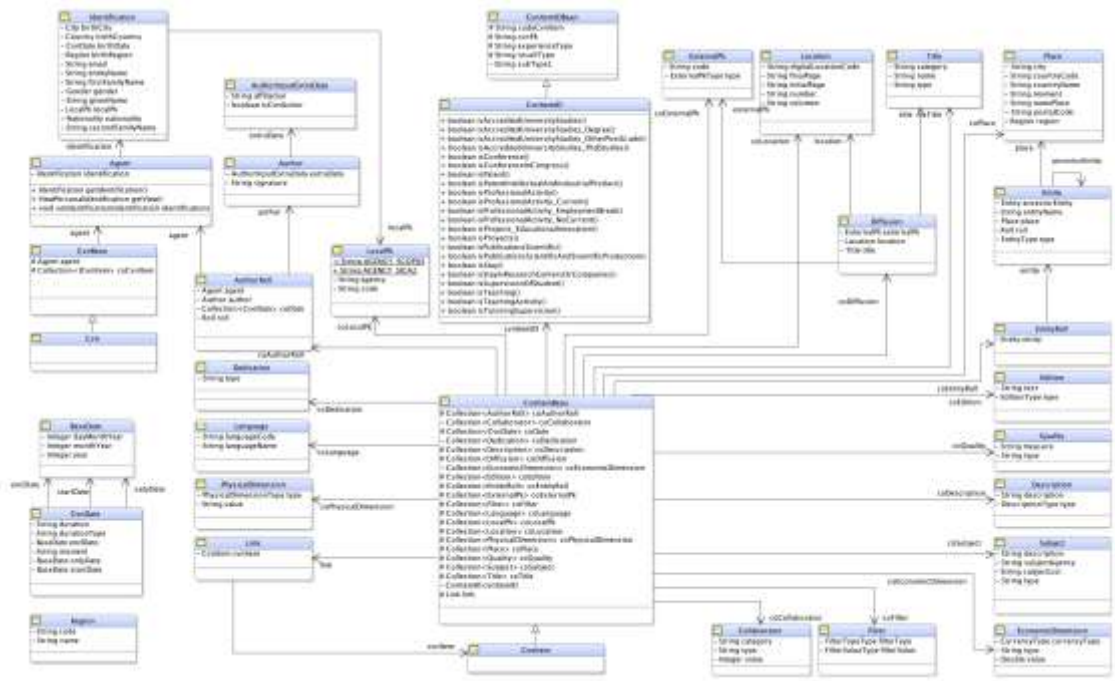

Figure 4. CVN entities.

As mentioned before, CVN is a specification designed to be flexible enough to be able to represent any kind of item from a researcher's CV. For this reason, the `CvnItemBean` class needs to be supported by other classes to complement the information stored in an instance of `CvnItemBean` class. This layer has been developed precisely for this purpose. As a consequence, when a curricular item is stored using the CVN API, most of its information will be stored in an instance of the `CvnItemBean` class, but the rest of these data will be kept in instances of this layer classes.

Figure 4 depicts the CVN entity layer and the relationship between its classes and the main class they support, i.e. the `CvnItemBean`. In addition, the other classes of the former layer are represented and connected to the components of this new layer. All these entities are explained in detail below:

- **CVN** is a superclass with the CVN structure.
- **Agent** class models the researcher as an actor.
- **Identification** class is stored in an Agent class as a property. The instances of this class store all the personal data of a researcher, e.g. name, nationality, gender, birth date, etc.
- **AuthorRole** is the role of the researcher in the curricular item. For instance, a researcher can participate in a conference as an author, reviewer, member of the program committee, or chair, etc.
- **Author** class relates the curricular item to the author's role in it, supplying new data such as the way the author signed that curricular item, that is, his/her signature.
- **AuthorInputExtraData**: objects of this class are usually included as properties of Author entities, containing additional data, such as the affiliations for searching the researcher data with the signature and affiliations.
- **CvnItemId** class models the unique identification data of curricular item, and provides methods indicating the specific type that corresponds to this curricular item.
- **CvnItemIdBean** contains all the fields that define a type of curricular item.
- **Language** of the curricular item.
- **PhysicalDimension** has information about the dimension of the curricular item, such as the number of pages (in the case of paper), number of hours (for a course), etc.
- **Link** class allows connecting one curricular item to another.
- **Dedication** class is the time spent on the curricular item. This information makes sense in items such as contracts, where the time is relevant. Accordingly, the values of dedication could be full-time, part-time, etc.
- **LocalPk** is an identification code of the curricular item in an external agency.
- **ExternalPk** contains the information data of the external agency and therefore, its information is closely related to the previous entity.
- **Location** class instances contain data about the situation of the curricular item

inside their context. For instance, in the case of a paper, its initial page, end page, volume, etc.
- **Title:** There are many curricular items that have a title, for example papers.
- **Diffusion** class models the external references made to this curricular item. For instance, regarding a paper, the citations of the paper in question in other authors' papers.
- **Place**, is the geographical location of the curricular item: city, region, country, etc.
- **Entity** contains the name of the organization under which the curricular item was developed or took place. For instance, the name of a university, or a research center.
- **EntityRole** is the role of the previous entity in the elaboration of the curricular item.
- **Edition**: is the edition of the curricular item. For example, in the case of a book, its edition number.
- **Quality** class models the quality indicators of the curricular item.
- **Description** contains a narrative description of the curricular item.
- **Subject** models the set of keywords identifying the curricular item.
- **EconomicDimension** is an economic value given to the curricular item. For instance, in the case of a project, the funding received to accomplish it.
- **Filter** adds specific information of the curricular item.
- **Collaborator**: Some curricular items have been made in collaboration with other authors whose data is not included in CVN format. Therefore, this new entity was developed to include some information regarding these authors.
- **CvnDate** class contains the date of the curricular item.

3.3. Interfaces

The interfaces make independent the implementation of the functionalities from their use, facilitating their update and their reusability. Accordingly, these interfaces allow us to group certain functionalities provided by the API, without taking into account the way they are implemented. Our API provides the following interfaces:

Figure 5. API Interfaces.

- **ICvn** interface specifies the operations required to add curricular items (such as, for instance, `addConferenceInCongress`), operations for saving researcher data (such as `saveAgentIdentification`), and operations that return all the stays, patents, etc.
- **ICvnBean** interface specifies getter and setter methods for the `Cvn` classes properties.
- **ICvnItem** interface specifies operations for adding properties in the `CvnItem` (`addDate`, `addTitle`, ...).
- **ICvnItemBean** interface specifies getter and setter methods for the `CvnItem` properties.
- **ICvnItemView** interface specifies methods that return views from the different types of the `CvnItem`, for example, `getPatent`, `getStay`, etc.
- **IBaseBean** interface is the base interface of all the other beans of this API layer beans and, therefore, all other interfaces of this layer. It also extends to

`Serializable` interface that is mandatory.

3.4 Factory

Construction and instantiating of the component of this API are made through a layer that implements the factory pattern, i.e. a design pattern for programming languages which delegates the decision about the type of class to be instanced precisely to the factory layer.
Figure 6 shows a diagram with the classes involved in this API layer:

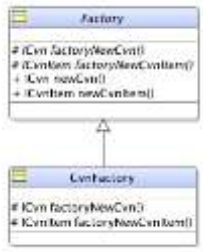

Figure 6. API Factory layer.

As can be seen in Figure 6, `Factory` is an abstract class and `CvnFactory` is a class that inherits from `Factory`. `CvnFactory` is responsible for creating objects from classes, implementing the `ICvn` interface.

3.5 CVN Auxiliary Entities

This layer provides a collection of entities useful for storing additional information, which complements those kept by the other layers of this API. More concretely, the auxiliary entities are: `City`, `Region`, `Country`, `Nationality`, `Gender`, `Entity Type`, `Description Type`, `Edition Type`, `Role`, `PhysicalDimensionType`, `External Pk Type`, `Currency Type`, `Filter Value Type`, and `Filter Type`. Figure 7 illustrates these entities. As can be seen in the Figure, all these auxiliary entities inherit from the `GenericAuxBean` class having, therefore, the same fields, which are summarized below:

- **Id**: Identification of the entity.
- **Code**: For example, for a country the code would be ES.
- **Name**: For instance, in the case of a country, the name would be Spain.

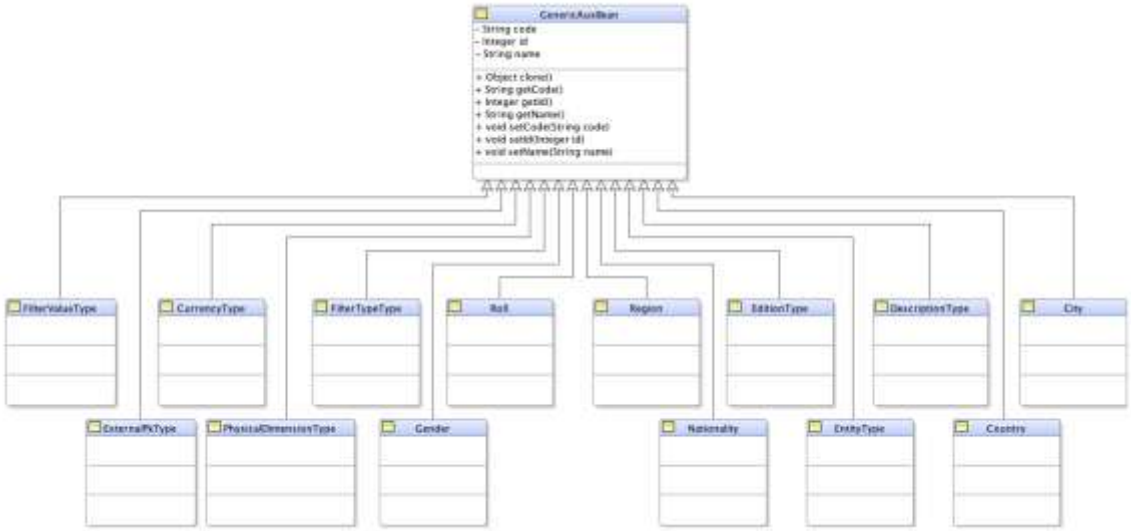

Figure 7. CVN Auxiliary Entities.

3.6. Input data

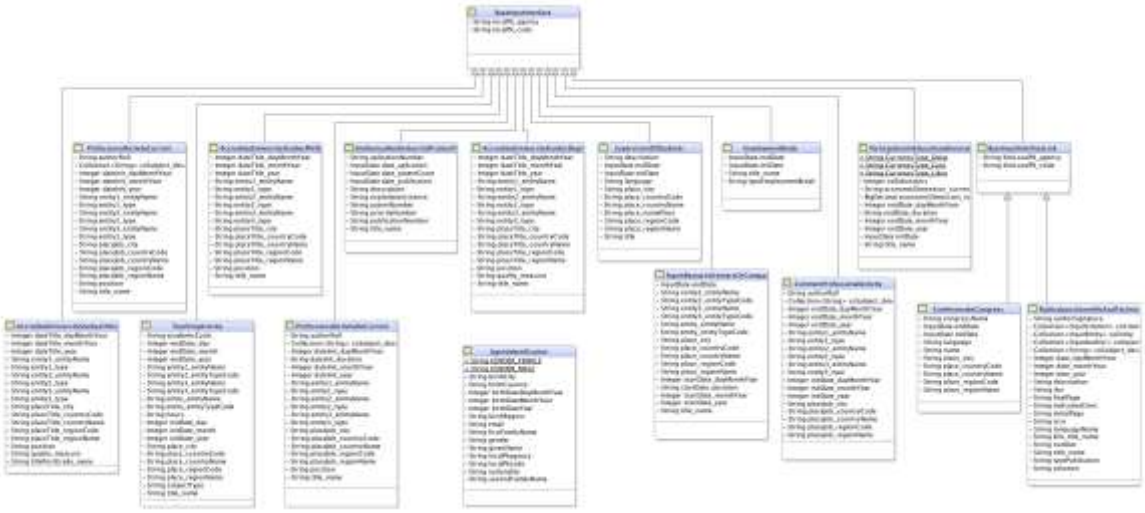

Figure 8. Input Data Entities.

Previous entity layers allow storing the curricular information in a raw format. The generality of CVN has revealed to be flexible enough to store any kind of curricular element. However, this generality feature makes it difficult to add information to a researcher's CV following directly the CVN specification and, therefore, the patterns defined by the former layers of this API. Precisely for this reason, this new API layer tries to use the entities which closer to the common understanding of a researcher's CV in order to express the curricular information, using the rest of the API layers.

Consequently, these new entities are used to add curricular item into CVN format by instantiating them. There is an entity for each curricular item type: `AccreditedUniversityStudiesDegree`

- `AccreditedUniversityStudiesOtherPostGrade`
- `AccreditedUniversityStudiesPhDStudies`
- `ConferenceInCongress`
- `EmploymentBreak`
- `IntellectualAndIndustrialProductPatent`
- `ParticipationInEducationalInnovationProjects`
- `ProfessionalActivityCurrent`
- `ProfessionalActivityNoCurrent`
- `PublicationsScientificAndTechnicalDocuments`
- `StayInResearchCentersOrCompanies`
- `SupervisionOfStudents`
- `TeachingActivity`

3.7. Curricular Item Views

Analogously to the former section, the access to the curricular items, with the aim of extracting information about them, is made through a set of classes, representing views on the data. These entities or classes are, therefore, different in terms of the information we want to consult. All these views are represented in Figure 9, and enumerated below:

- `ViewPersonalIdentification`
- `ViewAccreditedUniversityStudy`
- `ViewConference`
- `ViewEmploymentBreak`
- `ViewPatent`
- `ViewProfessionalActivity`
- `ViewProject`
- `ViewPublication`
- `ViewStay`
- `ViewTeaching`
- `ViewTutoringSupervision`

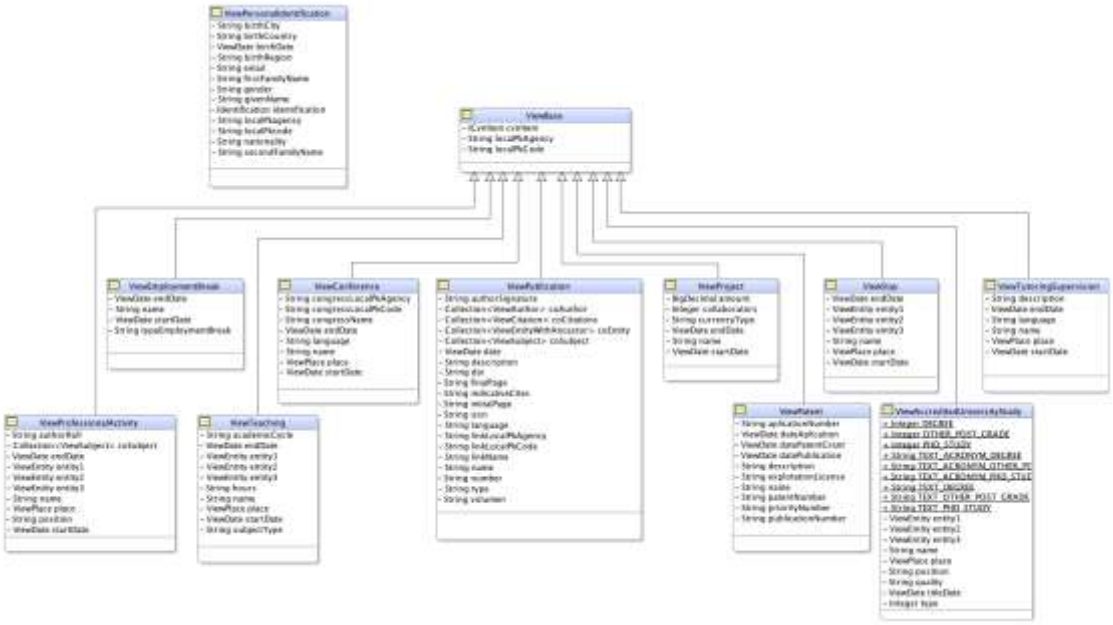

Figure 9. Curricular Item Views.

4. The Use of the API in SISOB System

The API explained in this deliverable is currently used in different components of the system. More concretely, it is used in the following components:

- **Gate Data Extractor:** This component is in charge of extracting information from semi-structured and non-structured data sources (see Figure 10). As a result, it returns the data in CVN format using the API.

Figure 10. The Gate Data Extractor Component of the SISOB system.

- **CVN to Conceptual Model:** Translation between formats is needed in the SISOB system to allow the different components to manage the information more efficiently. For this reason, several converters have been developed to make the required transformations. In this case, the conversion is made from the CVN XML-based format to an ontological format closer to the reality, that is, the conceptual model (see Figure 11).

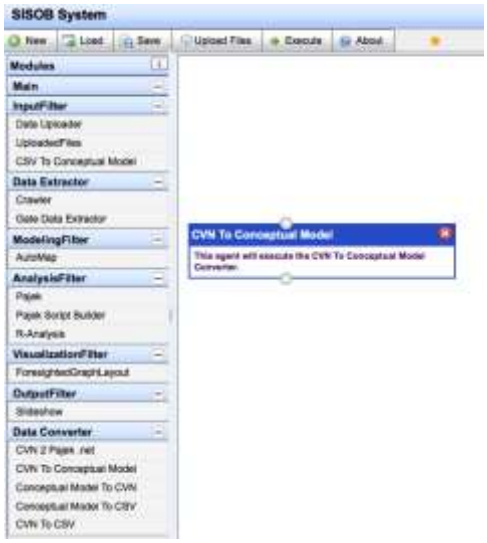

Figure 11. The CVN to Conceptual Model Component of the SISOB system.

- **Conceptual Model to CVN:** This converter is analogous to the former, but it makes the opposite transformation (see Figure 12).

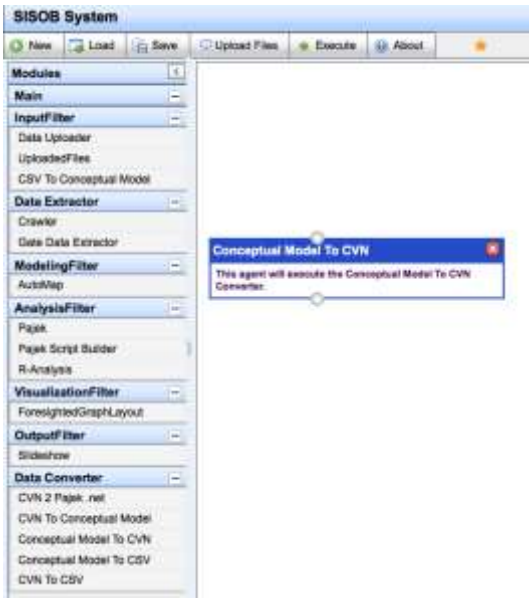

Figure 12. The Conceptual Model to CVN Component of the SISOB system.

- **CVN to CSV:** Other types of conversions are available in the SISOB system. Accordingly, this component turns the curricular information contained in a CVN-based format to a raw format, that is CSV (see Figure 13).

Figure 13. The CNV to CSV Component of the SISOB system.

- **CSV to CVN:** This component makes the opposite conversion regarding the former

component.

5. Conclusions

This deliverable has presented the structure of the CVN API designed as an artefact to access and manage the curricular information stored in the different data sources (databases, web sites, etc.). Furthermore this API provides a way of homogenizing the data managed in the different components of the SISOB system, as has been shown in this deliverable.