Semantic based Data Integration in Scientific Workflows

M. Abdul Rehman, Jamil Ahmed, Ahmed Waqas, Ajmal Sawand
Department of Computer Science, Sukkur IBA University
Sukkur, Pakistan

***Abstract*—Data Integration has become the most prominent aspect of data management applications, especially in scientific domains like ecology, biology, and geosciences. Today's complex scientific applications and the rise of diverse data generating devices in scientific domains (e.g. sensors) have made data integration a challenging task. In response to these types of challenges, data management applications are providing ground-breaking functionalities which come at the price of high complexity. This paper presents a semantic data integration framework which is based on the exploitation of ontologies. Exploiting a Description Logics formalism and associated reasoning procedures, the framework is able the handle heterogeneous formats and different semantics. Besides an in-depth discussion of the ontology-based integration capability, the paper also discusses a brief overview of the system architecture and its application in a real world scenario taken from ecological research.**

Keywords—Data integration; scientific workflows; ontology; data semantics; data management

I. Introduction

In order to understand complex scientific scenarios such as the world climate or the impact of the decreasing number of one species on others [1] lots of data are required. These data are usually not coming from one institution only but from many heterogeneous sources that need to be integrated [2]. Indeed the problem is not the availability of data but how to relate and interpret data correctly. The rise of data generating devices in scientific applications such as sensor networks has also made integration tasks more challenging. These sensors produce streams of unstructured raw data at different temporal and spatial granularities. Thus, before being used in any climatic application, these data need to undergo several processing steps of transformation and integration since such datasets are highly heterogeneous in terms of format, syntax, structure and semantics.

In such type of scenarios, besides a powerful semantic data integration capability, integration systems must provide comprehensive data management solutions that handle assorted formats and cares about data structures when data flow from a source to a sink. Furthermore besides the detection and resolution of conventional semantic conflicts [3] these systems must deal with issues like:

- *Information transformation* – Based on discovered mappings, information expected by the sink does not correspond to the information provided by the source. For instance, the source stores a *deviceId* while the sink requires *deviceName*; then these data must be transformed such that the same information is dealt with by both source and sink.
- *Missing and incomplete data* – when the target requires information which is not available in the source. For instance the source contains only a *deviceId* whereas the target also expects information about the location of that device (*locationId*, *locationName*).

The integration of data is often implemented in two ways which are sometimes overlapping. First, the use of standards is facilitated. Examples of such standards are the ABCD schema (Access to Biological Collection Data, [1]), the SEEK observation ontology [4], and biomedical ontologies [5]. A common pitfall of standards lies in the standardization process itself. Many stakeholders try to reflect their interests which can either cause the standard to be very general or very specific. In the first case it is hard to apply standards to specific problems because they are getting too complex (hundreds of elements), in the second case many competing standards arise. But standards – and this is their main benefit – fix semantics and syntax of data to a very high extend in a specification.

A second approach towards the data integration issue is the provision of handwritten wrappers which perform integration tasks on their own. This kind of a solution is still widespread even scientists start to realize that these wrappers are only useful as long as the corresponding developers are still available. Wrappers also fix the syntax and semantics of data, but unlike standards syntax and especially semantics are not directly accessible because they are hard-coded into the application. Looking up the syntax is possible by examining data provided from the wrapper. But the meaning of data is still hidden.

Both presented approaches work and provide some kind of a solution – however they share a common pitfall; the syntax and even more the semantics of data often depend on the application or the users' perception. Thus a method for data integration must provide the freedom of choosing one specific semantic and should not require the user to change standards or adapt wrappers permanently.

It is widely recognized that ontologies play a central role in modern scientific applications [5] [6] since their use is considered a possible solution for semantic based integration [2] [7]. Nevertheless a clear methodology for setting up the data integration task (cf. Section 2.2) also plays a vital role since usability and adaptability are determined by it.

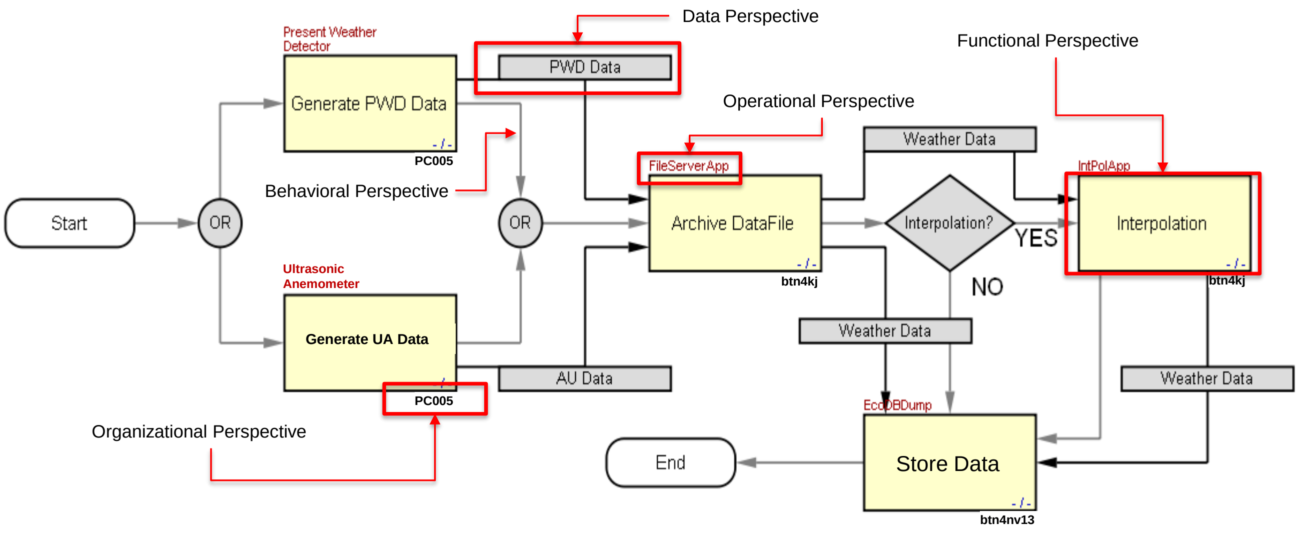


Fig. 1. POPM workflow model for the acquisition of sensor data within BayCEER Lab.

Workflow technology has also contributed a lot towards the integration of scientific data. In recent years many scientific workflow systems like Kepler [8], Taverna [9] and Triana [10] have come up with the capability to integrate scientific data based on ontologies. The lack of a clear method as well as the inability to separate integration steps from domain specific analysis steps (cf. Section 2.1) makes these systems hard to be utilized by a normal domain user. Data integration is implemented in these systems as an atomic step within a workflow – even though it can comprise very complex actions.

The proposed system promises to offer an end-to-end solution (i.e. from data source to data sink) for the data management issues discussed above including powerful and real semantic data integration through the exploitation of ontologies. Furthermore, a well-structured methodology facilitates normal users to manage their specific data integration scenario in a better manner. The main contribution in this paper is to present the semantic integration feature of DaltOn system as well as the methodology for setting up integration tasks.

The remainder of this contribution is structured as follows: Section 2 introduces the overall method which is already applied in several real-world use cases. Section 3 discusses the core algorithm of the semantic integration component of DaltOn together with its foundations. Section 4 relates the work to other approaches and Section 5 finally summarizes and concludes the contribution.

II. Method and Architecture of DaltOn

In the following sections a method for data integration within scientific workflows based on the DaltOn framework is presented.

A. *Overview and Motivating Scenario*

POPM (Perspective Oriented Process Modelling) [11] is a paradigm for modelling processes and/or workflows. Within the POPM paradigm, each modelling construct comprises several orthogonal building blocks, called perspectives. Thus, a modelling construct can be specified by defining different perspectives. There are five main perspectives (shown in Fig. 1) that provide a strong foundation for a process modelling language. The *Functional Perspective* determines the existence and purpose of a process step. The *Operational Perspective* is used to specify the application, service or tool which is required during enactment of a work step. The *Behavioral Perspective* is a mean to determine the execution order of the work steps. The *Organizational Perspective* is used to introduce agents who are eligible or responsible for performing certain work steps. The *Data Perspective* identifies data items and their flow in a process. This latter perspective is of special interest since all data management related issues are contained within this perspective.

i>PM [12] is a graphical tool built upon the POPM paradigm. It allows users to develop process / workflow models by specifying the above discussed perspectives

A **Motivating scenario** is taken from meteorological research and is currently carried out by The Data Group of the Bayreuth Center of Ecology and Environmental Research [13]. The main purpose is to retrieve data from various sensor devices, store it at an intermediate place for archival and finally dump the data into a central database at the institute.

Fig. 1 shows a POPM-based workflow model composed by a domain user for this scenario. The first two steps of the workflow describe the generation of sensor data either in the PWD (Present Weather Detector) or UA (Ultrasonic Anemometer) format depending on the actual sensor. Then these data are transmitted over a serial line to the archival step *Archive DataFile* which is describing the process of storing the sensor data in a file. After a successful data archival, data are usually stored in the central database called *EcoDB* within the *Store Data* work step. The work step *Interpolation* is optional and manipulates data whenever needed. For demonstrating the capabilities of the methodology, the main focus would be on the management of data as it occurs in between the two steps *Archive DataFile* and *Store Data* as shown in Fig. 1.

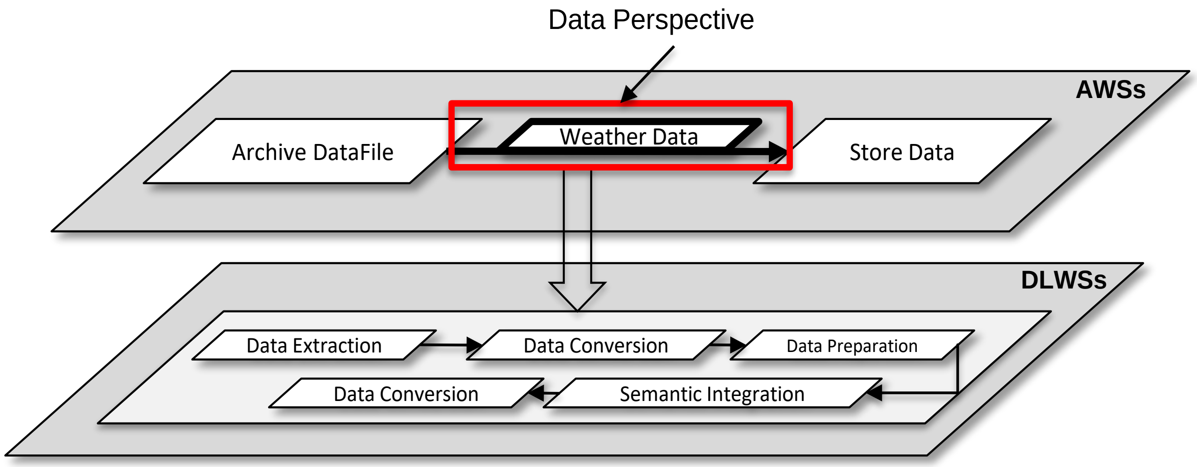


Fig. 2. Classification of work steps in a scientific workflow.

Before storing data in the database, it needs to be passed through some preparation operations since the raw sensor data does not fit in the structure and semantics of the database – sensor devices generate data in their own proprietary formats (here "PWD" or "UA") and also might use different interpretations. Thus the scenario poses many data management related issues:

- Physical data transport from a source (here: file server) to a sink (here: database
- Management of data formats (here PWD and AU
- Validation of data: Sometimes files are truncated due to an interruption in the file transfer from a sensor to the file server.
- Data filtering: This challenge belongs to the process of discarding unwanted values from data files during data transmission.
- Data integration: This is a major challenge since data coming from various devices have completely assorted structures and give different data interpretations.

Separation of workflow steps

In Fig. 2, the upper layer (denoted as 'AWSs') shows an extract of the workflow model of Fig. 1 including only the two work steps 'Archive DataFile' and 'Store Data'.

As stated in the previous section, several steps are needed in order to make the data of 'Archive DataFile' compatible to 'Store Data'. Hence, an application consists of two categories of work steps. First, steps which enact domain operations such as data acquisition, data analysis or data storage. Second, steps which are only used for data integration. Up to now these data related work steps were specified explicitly in the workflow, making it more complex. But separating the steps of a process into two categories assures the productivity of the scientific community since normal domain users (scientists) really do not desire to involve into data specific tasks.

The first layer of the approach is called Application Work Steps (AWSs) layer and contains only those steps of the scientific workflow which are specific to the application but whose nature is not related to pure data management. Examples are 'Archive DataFile' and 'Store Data'. The Data Logistic Work Steps (DLWSs) layer instead contains solely data preparation tasks (such as data integration and conversion). DLWSs are defined in terms of operations provided by DaltOn and incorporated into the scientific workflow instead of being explicitly modelled into it. Together both layers describe a (executable) scientific application. The AWSs layer can be controlled by a normal workflow management system (WfMS); the DaLo-WFs are – due to their nature – also controlled by a WfMS but the execution of these processes is heavily supported by the DaltOn framework.

The data flow perspective of POPM directly corresponds with the definition of DaLo-WFs. It thus wraps up all data management functionality and provides means to define where data resides, where it must be transported to and how it is transformed within scientific workflows.

B. *A Methodology for Data Integration within Scientific Workflows*

The methodology is shown in Fig. 3. It is itself described by a process model. Together, this process describes how a scientific application consisting of AWSs and DLWSs has to be developed. Further, the method assigns clear responsibilities: the application workflow with the scientific analysis is defined by scientists (generally: the domain user) and the DaLo-WFs with data integration tasks are set up by data experts. A short description of each step of the process is given in the following.

Define Application Process: The aim of this step is to define the domain application by developing a workflow which includes only domain specific work steps (as AWSs layer). Domain users, e.g. scientists, are responsible for enacting or at least for supervising the execution of this step. In general every workflow modelling tool can be used to describe an application process; however, i>PM [12] has been used in this work. In the example scenario, the output of this process step is the workflow model shown in Fig. 1.

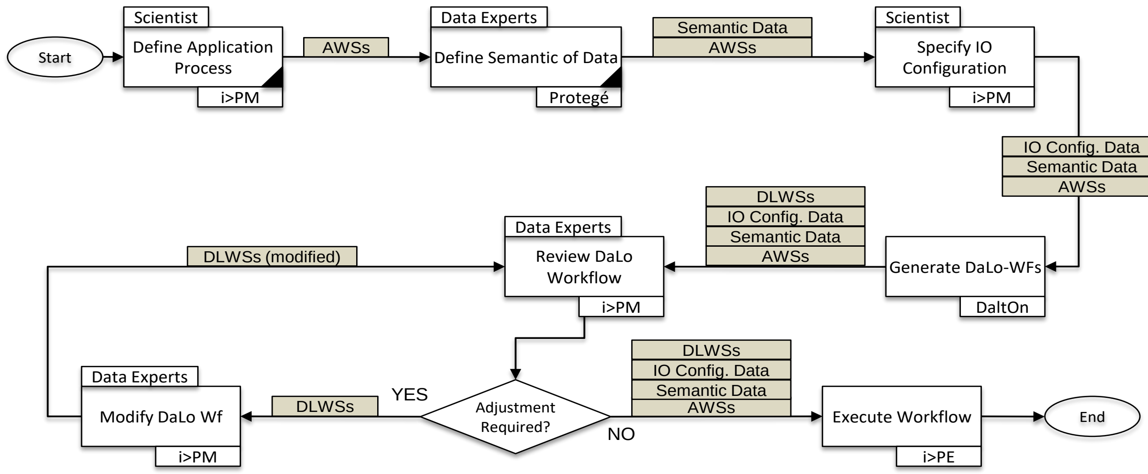


Fig. 3. POPM process of the methodology for the definition of scientific applications.

Define Semantics of Data: Data are described semantically by developing ontologies. Ontologies for the use case are shown in Section III. Both, domain users and data experts are responsible for this step. Domain users because only they can describe a common vocabulary and data experts because only they know specific details about applications, data sources and sinks. However, this step is optional since existing ontologies can be re-used.

Specify IO Configuration: This step supports the generation of DaLo-WFs taking place in the next step of the methodology. The I/O specification may contain information about the exact location and type of data sources/sinks, a description of data formats for input and output data (for instance data schemata), an assignment of the previously defined local ontologies to steps, applications and agents and finally the definition of criteria applied during extraction and insertion of data. Table 1 shows the specification of the 'Weather Data' data container for the example scenario.

Generate DaLo-WFs: In this step each data flow of the application process is transformed into single DaLo-WFs that specify single data integration tasks. A first version of a DaLo-WF can automatically be derived by exploiting the specifications provided in the previous step. The DaLo-WF generation is supervised by a data expert and enacted by DaltOn functions. As an output, this step delivers an executable workflow containing both the AWSs and the DLWSs.

Review DaLo-WFs and *Modify DaLo-WFs*: As stated earlier, the generated DaLo-WFs may not be sufficient for special integration scenarios. Therefore, the DaLo-WFs might have to be adjusted accordingly. Review and modification are again enacted by data experts.

Execute Workflow: Finally, the workflow can be executed by a suitable execution environment [14] for a description of such an execution environment). Both types of steps, AWSs and DLWSs, are executed by the same environment. Involving human actors is possible for both types of steps since some applications might require the interaction of process and scientists.

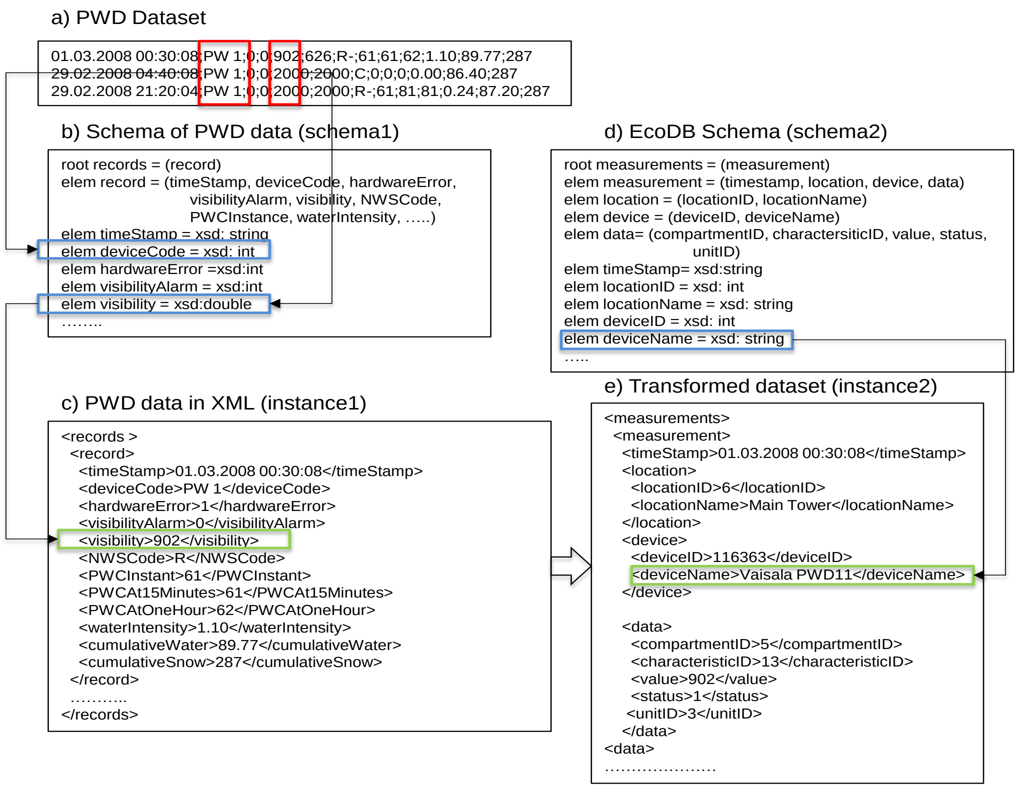
1.1 Architecture of the DaltOn Integration Framework

The architecture of the DaltOn Integration Framework [15][16] follows the approach of separating concerns into single and independent functions. Thus, DaltOn has three major conceptual abstractions, namely Data Provision, Data Operation and Data Integration.

Data Provision bundles components which are used for enacting physical data exchange between data sources (data producing steps) and data sinks (data consuming steps). Each of the sub-components of the Data Provision fulfils a specific task: Data Extraction and Selection (DES) cares about the extraction of a (sub-) set of data from a source based on user- and application-specific criteria, Data Transportation (DT) handles physical data transport and Data Insertion (DI) performs insertion of data.

TABLE I. SPECIFICATION OF 'WEATHER DATA' DATA CONTAINER

Configuration	**Value**	
	Source	Sink
Location	<IP address>:<Path to the File>	<DB Connect String>
Format	Pwd	Xml
Criteria	Null	Visibility=<2000
Schema	URI to Eco1.xsd (schema1)	URI to Eco2.xsd (schema2)
Local Ontology	URI to LocOntoEco1.owl (local ontology1)	URI to LocOntoEco2.owl (local ontology2)
Mapping	URI to mapEco1.rdf (mapping1)	URI to mapEco2.rdf (mapping2)
Reference Ontology	URI to RefOntoEco.owl	



a) PWD Dataset

```
01.03.2008 00:30:08;PW 1;0;0;902;626;R-;61;61;62;1.10;89.77;287
29.02.2008 04:40:08;PW 1;0;0;2000;2000;C;0;0;0;0.00;86.40;287
29.02.2008 21:20:04;PW 1;0;0;2000;2000;R-;61;81;81;0.24;87.20;287
```

b) Schema of PWD data (schema1)

```
root records = (record)
elem record = (timeStamp, deviceCode, hardwareError,
               visibilityAlarm, visibility, NWSCode,
               PWCInstance, waterIntensity, …..)
elem timeStamp = xsd: string
elem deviceCode = xsd: int
elem hardwareError =xsd:int
elem visibilityAlarm = xsd:int
elem visibility = xsd:double
……..
```

c) PWD data in XML (instance1)

```
<records >
 <record>
  <timeStamp>01.03.2008 00:30:08</timeStamp>
  <deviceCode>PW 1</deviceCode>
  <hardwareError>1</hardwareError>
  <visibilityAlarm>0</visibilityAlarm>
  <visibility>902</visibility>
  <NWSCode>R</NWSCode>
  <PWCInstant>61</PWCInstant>
  <PWCAt15Minutes>61</PWCAt15Minutes>
  <PWCAtOneHour>62</PWCAtOneHour>
  <waterIntensity>1.10</waterIntensity>
  <cumulativeWater>89.77</cumulativeWater>
  <cumulativeSnow>287</cumulativeSnow>
 </record>
 ………..
</records>
```

d) EcoDB Schema (schema2)

```
root measurements = (measurement)
elem measurement = (timestamp, location, device, data)
elem location = (locationID, locationName)
elem device = (deviceID, deviceName)
elem data= (compartmentID, charactersiticID, value, status,
            unitID)
elem timeStamp= xsd:string
elem locationID = xsd: int
elem locationName = xsd: string
elem deviceID = xsd: int
elem deviceName = xsd: string
…..
```

e) Transformed dataset (instance2)

```
<measurements>
 <measurement>
  <timeStamp>01.03.2008 00:30:08</timeStamp>
  <location>
   <locationID>6</locationID>
   <locationName>Main Tower</locationName>
  </location>
  <device>
   <deviceID>116363</deviceID>
   <deviceName>Vaisala PWD11</deviceName>
  </device>

  <data>
   <compartmentID>5</compartmentID>
   <characteristicID>13</characteristicID>
   <value>902</value>
   <status>1</status>
   <unitID>3</unitID>
  </data>
<data>
…………………
```

Fig. 4. Excerpt of instance data and schemas for both sides along with actual PWD dataset.

Data Operation encompasses Format Conversion (FC) and Data Preparation (DP); the FC sub-component is carrying out syntactic transformations of data, for instance the conversion of data given in CSV (comma separated values) into a XML representation and back. DP contains functions which can be applied to data such as unit conversions or simple arithmetic operations but is not meant to replace scientific analysis steps.

Data Integration finally is the heart of the DaltOn system [17] that aims at the semantic integration of data. It comprises only one sub-component so far, the Semantic Integration (SI).

Beside these main three abstractions, DaltOn is using wrappers for accessing data sources through a unified interface (but not for executing data integration tasks) and a RDF based (triple) data store as repository.

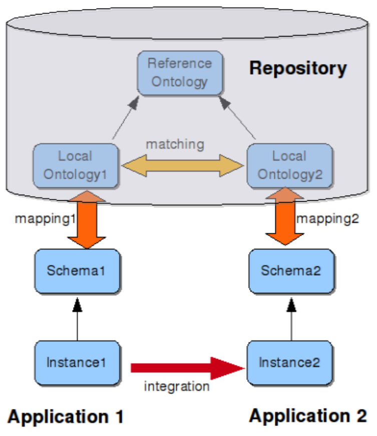


Fig. 5. The architecture of the SI component

III. Semantic Based Integration

Schema mapping is a specification describing how data from a source schema can be mapped to a sink (or target) schema. This is usually considered an essential building block of data exchange and integration solutions.

Schema mappings are usually discovered (semi-) automatically using a match operation that can either corresponds to a structure-based or a semantic-based approach. The feasibility of later approach is supported by the design of DLWSs which require that AWSs share a common domain of discourse. In DaltOn, this aspect is represented with ontologies. A first issue is that several applications may interpret elements of the same domain differently. In such situations, an alignment between the different interpretations needs to be discovered. Based on discovered mappings, DaltOn solves related issues. One of these concerns is information integration. That is, the information expected from the source document does not correspond to the information the source is able to provide. Another important issue deals with missing or incomplete information in the source instance document. Both situations generally prevent the fulfilment of the generation of a source instance document. In order to pursue the integration, the paper proposes a solution based on the exploitation of a repository. The repository is a central place which stores and proposes query facilities to retrieve information related to the domain of discourse.

The example scenario is interesting as it exploits an important set of functionalities available in the SI component, e.g. different kinds of mapping correspondences and repository exploitation.

A. Basic Notions

Basically Ontologies are used to represent the knowledge of a domain in a common way, enabling these to be shared among machines and human beings [18]. Thus, an ontology consists of concepts with relationships between them, which provides a common vocabulary for knowledge to be exchanged between machines and human beings. In order to support reasoning within ontologies, a logical formalism is used, such as Description Logics (DL) [19], as a mean to represent ontologies. This family of formalisms allows the representation and reasoning over domain knowledge in a formally and well-understood way. Central DL notions are concepts (unary predicates) and relationships, also called properties or roles (binary predicates). A key notion in DL is the separation of the terminological (or intensional) knowledge, called TBox, from the assertional (or extensional) knowledge, called ABox. The TBox contains the descriptions of concepts and their relationships in the following form:

$$\mathrm{Device} \sqsubseteq \forall \mathrm{situatedAt.Location} \sqcap$$

$$\exists \mathrm{situatedAt.Location} \sqcap$$

$$\forall \mathrm{hasDeviceName.String} \sqcap$$

$$\exists \mathrm{hasDeviceName.String}$$

This description states that the Device concept is defined as being situated in at least one location, and locations only, and has at least one name which must be string of characters (in an OWL serialization this is supported by XML Schema data types). In contrast, ABoxes contain assertions of concepts and their roles in the following form:

Device(device_116366),

Location(location_3),

hasDeviceName(device_116366,

“Vaisala QLi50 2”)

situatedAt(device_116366, location_3)*

These assertions state that objects with identifiers ‘device_116366’ and ‘location_3’ are instances of respectively the 'Device' and 'Location' concepts. These two objects are related by the 'situatedAt' object property. Finally, the ‘device_116366’ object is related to the value ‘Vaisala QLi50 2’ via the ‘hasDeviceName’ data type property.

A TBox and an ABox together denote a Knowledge Base (KB), denoted as KB = < TBox, ABox >.

B. DaltOn SI Component Architecture

The main objective of SI is to generate a valid input document for the target step of an Application Workflow. The Fig. 5 presents the details of this component's architecture. This architecture is based on the set of documents each DaLo-WF application can access. This set comprises four kinds of documents:

1) An instance document which corresponds to the output document of Application1 (produced by the source step), respectively the input document of Application2 (consumed by the sink step).

2) A schema associated to each instance document.

3) A mapping between elements of the schema to elements of a local ontology.

4) A local ontology which supports the particular interpretation of each concept in an application.

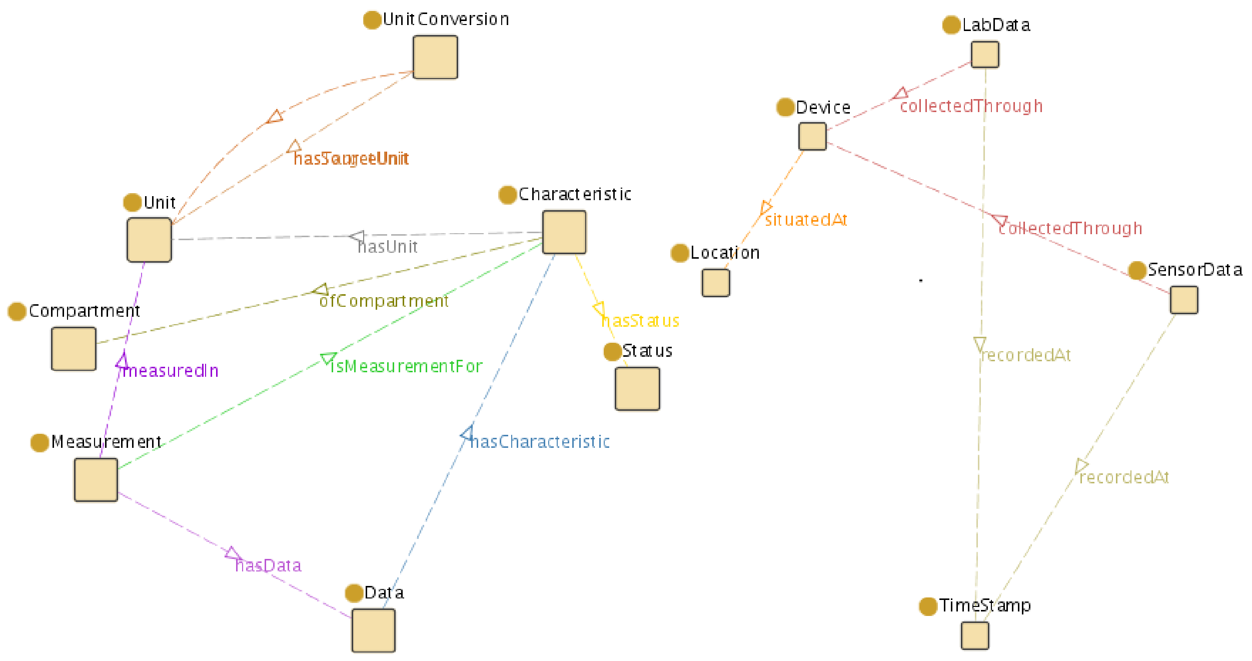


Fig. 6. Concepts and roles of the reference ontology in the meteorological example.

The upper part of Fig. 5 emphasizes two other components:

1) A reference ontology which provides a common vocabulary to the local ontology. This approach makes the local ontologies comparable and enables to process matches between concepts.

2) The repository is responsible for the storage of the knowledge bases associated to the application domain. It also stores the mappings that are being discovered by the matching solution.

The role of an application instance and schema document is obvious in the context of a DaLo-WF. They are usually created by the application developer and come at no extra cost. These documents are expressed in XML and XML Schema (henceforth XSD) respectively. Fig. 4 presents an extract from instance1, instance2, schema1 and schema2.

The mappings, ontologies and ABox assertions contained in the repository impose extra work from the application designers. Nevertheless, the task of developing these documents is limited due to the following: (i) generation of a single reference ontology is generally sufficient, (ii) reuse of local ontologies among several DaLo-WFs is generally possible, (iii) low expressivity of the reference ontology, (iv) use of adapted tools which simplify the creation of these documents.

Concerning aspects (i) and (ii), as per experiences in using DaltOn in medicine, biological and ecological domains emphasize that usually one unique reference ontology is sufficient for all DaLo-WFs of an application. The design of this reference ontology can be facilitated by exploiting existing domain ontologies. The reference ontology does not need all the expressiveness proposed by some well-known ontologies in scientific domains.

Concerning (iii), the expressive power of the local and reference ontologies are not the same. The reference ontology provides a common vocabulary on the domain of discourse. This common vocabulary enables schema mapping to be generated. Fig. 6 presents a graph of the reference ontology in the meteorological use case, developed using Protégé tool [25]. This graph presents concepts as nodes, roles as labeled edges. For readability reasons, subsumption relationships are not depicted.

A local ontology implements the local interpretation to the concepts of the reference ontology and also provides the way to include new concepts defined with respect to the concepts and roles of the reference ontology. Example 1 presents an extract of the concept definitions of local ontology1 in the meteorological use case.

Example 1 Concept definitions of local ontology1

$loc1.HardwareErrorCode \sqsubseteq Status$
$loc1.TimeStamp \equiv TimeStamp$
$loc1.PresentWeatherData \sqsubseteq SensorData$
$loc1.SensorData \sqsubseteq Data \sqcap$
$\forall recordedAt.TimeStamp \sqcap$
$\exists recordedAt.TimeStamp$
$\sqcap \forall collectedThrough.Sensor \sqcap$
$\exists collectedThrough.Sensor$
$\forall hasStatus.Status \sqcap \exists hasStatus.Status$
$loc1.CumulativeSnow \sqsubseteq Characteristic$
$loc1.Status \equiv Status$
$loc1.NWS \equiv NWS$
$loc1.Sensor \equiv Sensor$

Concerning (iv), the design of the different ontologies (reference and local ones) as well as the generation of reference ontology concept and role assertions, stored in the repository, are facilitated by the use of a Protégé plug-in named DBOM [20], [21]. This plug-in eases the creation of knowledge bases expressed in DL from relational databases. For instance, in the meteorological use case, the meteorologists provided us with databases containing domain specific information about location of sensors, devices, units used by these devices, etc. Using the DBOM plug-in the system is able to create a reference ontology serialized in OWL and at the same time to generate a valid ABox which was later integrated in the repository.

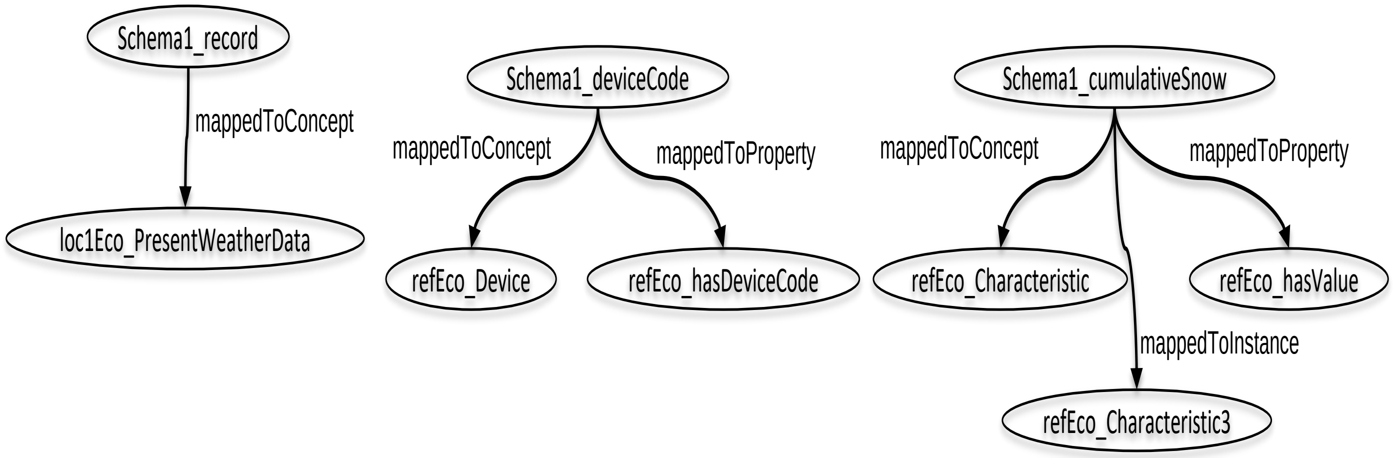


Fig. 7. Excerpt of the graph of mapping1 in use case.

C. Schema to Ontology Mapping

The mapping relates elements from schema1 (respectively schema2) to concepts and roles of local ontology1 (resp. ontology2). A schema mapping is generally represented as a triple consisting of a schema, a local ontology, and a mapping specifying relationships between them. The system uses this representation and restricts the set of mapping relationships to:

- a mapping to an ontology concept (denoted *'mappedToConcept'*)
- a mapping to an ontology role (denoted *'mappedToRole'*)
- a mapping to a concept instance which is stored in the repository (denoted *'mappedToIndividual'*).

The syntax of the mapping solution is restricted such that not all combinations of the mapping relationships are accepted. The restrictions and their associated semantics are characterized in Table 2.

The simplest abstraction of an XML document is a labeled ordered tree, possibly with data values associated to the leaves. But for the mapping approach, the system takes advantage of the object model view which can also be applied to an XML document. Starting from this view, the system assumes that any XML element is at least mapped to a DL concept or DL individual. This first assumption enables the system to disallow the mapping #1 and #3 which do not inform about an associated DL concept nor DL individual. The purpose of mapping #4 is to inform the system about the absence of mapping for a given XML element. In fact, this most effectively and rapidly performed by users by omitting such a mapping for this element.

In cases where an XML element is not empty, i.e. it contains a data value, it is necessary to map it to a DL property. This is the case of mapping #5, #7 and #8 in Table 2 Mapping #8 is a specialization of mapping #5 where extra information about an associated concept individual is provided. Mapping #7 can be viewed as being equivalent to #8 where the type instance is not specified. In cases of an empty XML element, no DL property needs to be attached to the mapping. Hence, it corresponds to mappings #2 or #6. The latter being a specialization of the former where extra information about a DL concept instance is provided.

Finally mapping #4 is considered as a shortcut of mapping #6 where the DL concept is omitted. This kind of mapping is supported if the processing of the DL realization reasoning procedure, i.e. providing the most specific concept an individual is an instance of, returns a single concept. Thus there cannot be any ambiguities about the type of this individual. Fig. 7 displays an extract of a graph representing the mapping between the schema and the local ontology associated to application1 in the meteorological use case. In this figure, relations start from the XML element of a given schema (pattern is “schemaName_elementName”) and points to an ontology element, i.e. concept, property or individual, of a given ontology (pattern is “ontologyName_elementName”).

This figure emphasizes mappings related to mapping #2, #5 and #8 of Table 2. For instance, the mapping of the 'record' XML element, which is empty and root of the document, is mapped, via 'mappedToConcept' to the 'PresentWeatherData' DL concept. The 'deviceCode' XML element is non-empty and mapped to the 'Device' DL concept and its 'hasDeviceCode' property. Finally the 'cumulativeSnow' element, again a non-empty element, is mapped to a concept ('Characteristic'), a property ('hasValue') and a individual ('characteristic/3'). In order to explain the integration methodology and present the matching issues, it is necessary to present the mapping associated to application2 as well (Fig. 8). Notably, only two elements are mapped to a DL concept: 'measurement', the root element of schema2 and 'data', an empty and nesting element. All other elements are mapped to DL concepts and properties. Finally, the mapping language is the same for the output and input applications of a DaLo-WF.

D. Methodology and Heuristics

Generating the input document of a DaLo-WF's Application2 is a multi-step process. These steps correspond to (i) matching the local ontologies, (ii) matching the (XML) schemata and (iii) generating the target instance document.

Matching local ontologies

This matching step searches for correspondences between the DL concepts of both local ontologies. This operation is supported by the existence of a common vocabulary, the reference ontology. In order to discover as many matches as possible, two techniques are considered to find correspondences: DL-based and navigation-based mappings.

TABLE II. MAPPING POSSIBILITIES IN SI

#	mappedTo Concept	mappedTo Role	mappedTo Individual	Semantics
1				Not accepted
2	X			Empty XML element is mapped to an ontology concept
3		X		Not accepted
4			X	Equivalence to a concept instance
5	X	X		Non empty XML element is mapped to a concept and a role
6	X		X	Empty XML element is mapped to an ontology concept and an individual
7		X	X	Not empty XML element mapped to a role and a concept instance
8	X	X	X	Non empty XML element is mapped to a concept and role as well as a concept instance

The DL-based approach is performed using a DL reasoner and particularly its concept subsumption inference procedure.

In the navigation-based approach, an ontology is taken in terms of a directed acyclic graph where nodes correspond to DL concepts and the edges correspond to DL properties. Basically, it searches for navigation paths between two concepts. This is performed by exploiting the (SPARQL) query facilities of the (triple store) repository. The navigation-based approach also exploits a DL reasoner with its concept subsumption, instance checking and realization inference procedures. This approach is non-deterministic and may return several different paths. So it is important for the algorithm to qualify paths and to select the most appropriate one. This qualification is based on several factors: the length (L) of each path (i.e. the number of properties along a path) and the characteristics of the properties used along a path, i.e. functionality, inverse functionality.

As the implementation formalizes ontologies using decidable species of OWL, i.e. OWL Lite and OWL DL, it is possible to distinguish properties based on their functional characteristics. As a functional property, denoted as ‘prop’, is defined as:

$$\forall x, y_1, y_2 \mid \exists prop(x, y_1) \cap \exists prop(x, y_2) \Longrightarrow y_1 = y_2$$

The decidability issue of DL reasoning tasks is a main concern in the solution. For this reason, inverse functional properties are not considered, which are supported in OWL, but are only associated to decidable inferences for object properties. Thus inverse functional properties on data type properties yield an OWL Full ontology which is not decidable.

The system distinguishes between several navigation approaches:

- L=1 and the property is functional: 'functionalNavigation'.
- L=1 and the property is not functional: 'nonFunctionalNavigation'.
- L>1: 'pathNavigation'.

The match operator applied in the DaLo context is able to find several correspondences, usually belonging to the two presented categories, between a given pair of DL concepts. In order to deal with this issue, the system propose a heuristic to select a preferred correspondence. This heuristic is based on a total order of the DL-based and navigation-based categories.

Definition: For a given pair of DL concepts C1 and C2, respectively from local ontologies 1 and 2, if a set of correspondences are found between these two concepts: the system knows that there must be at most one DL-based correspondence between C1 and C2 but several navigation-based mappings can coexist with it. For this reason, the system ranks the navigation-based correspondences according to a preference total order: functional Navigation > non-functional Navigation > path Navigation.

Concerning navigation-based relationships, setting a property to be functional is an important commitment for the knowledge engineer. The system thus considers that a functionalNavigation is preferred to a nonFunctionalNavigation. Finally the system considers that navigation with a single edge is more trustable than a path made of several edges.

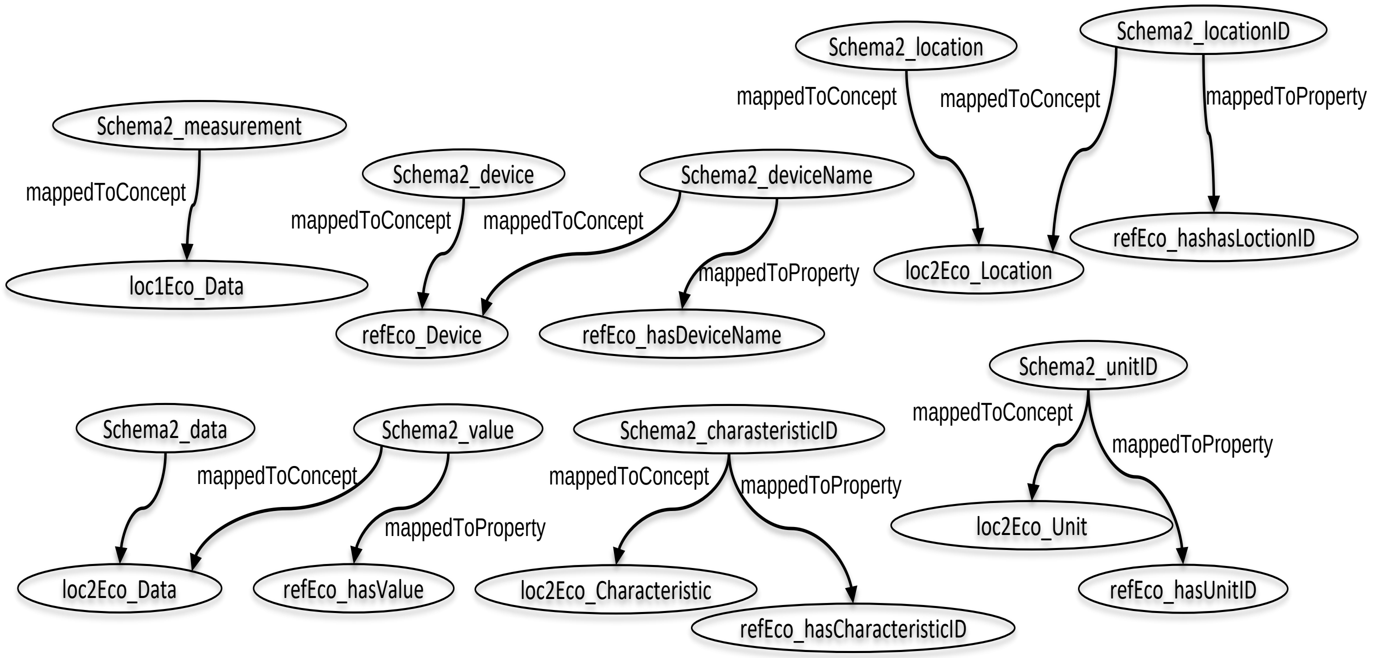


Fig. 8. Excerpt of the graph of mapping2 in the use case.

Thus the system obtains a partial order on the total set of discovered correspondences. On the use cases which are implemented with DaltOn so far, a heuristic has been added, stating that functionalNavigation is preferred to concept generalization which is preferred to nonFunctionalNavigation, thus obtaining a total order on correspondence preferences: concept equivalence > concept specialization > functionalNavigation > concept generalization > nonFunctionalNavigation > pathNavigation.

Other heuristics could also be applied, e.g. generalization > functionalNavigation, and SI supports the definition of specific preference orders.

Matching schemata

The purpose of this step is to discover mappings between Schema1 and Schema2 from the mappings discovered in the previous step, i.e. between local ontology1 and Local ontology2 [Fig. 5]. This step can be easily performed using the schema to ontology mapping, i.e. from schema 1 to local ontology 1, respectively schema 2 and local ontology 2.

Finally, due to the dual matching solution (logic-based and navigational-based), the accuracy of data stored in the repository and the possibilities to adjust heuristics. These false matches generally occur when local ontologies are modified due to replacement or configuration modifications at the sensors. In these cases, the adjustments need to be performed on the local ontologies and, possibly in non-monotonicity situations of the local ontologies, to the reference ontology [Fig. 6].

Target instance generation

Starting from these mappings, it is possible to consider the generation of data values for (non-empty) target elements. For navigation-based correspondences, the processing is relatively obvious as it is sufficient to follow the selected paths between two concepts. This navigation is performed starting from a specific node of the ontology graph. For instance, in the case of the location sink element, the mapped source element is 'deviceCode' and the 'hasDeviceCode' DL property (Fig. 7). For a given source instance which has 'PW1' as a value for 'deviceCode', SI will use methods of reference reconciliation [22] to identify the associated graph node. Starting from this node, it is possible to follow the path to the searched value.

For DL-based correspondences, it is required to inspect the DL properties associated to each mapping in order to detect possible transformations. For instance in Table 3, the sink element 'deviceName' is related, via Concept equivalence, to the 'deviceCode'. But the 'deviceCode' element is mapped to the property 'hasDeviceCode' (Fig. 7) while 'deviceName' is mapped to 'hasDeviceName'. Thus a transformation needs to be performed.

A final step consists in enabling the integration of data from application1's instance onto application2's instance document. Different forms of mappings are available, e.g. relational queries, relational view definitions, XQuery queries or XSLT transformations, to perform this task. The system opted for XSLT transformation since it does not need the expressiveness and complexity of relation queries and views.

By selecting XSLT, the system also benefits from procedural attachment possibilities when performing transformations. That is SI includes a set of procedures, developed in the Java language, to enable the retrieval of values stored in the repository at runtime. Most of these procedures generate, from predefined templates, SPARQL queries and execute them on the repository's ABox.

IV. Related Work

Kepler [8] is an open-source scientific workflow system which is evolved from Ptolemy system [23]. Kepler's data integration approach is based on a semantic mediation system and utilizes the automated integration services from a middleware called SEEK [4]. SEEK exploits ontological information to support structural data transformation for scientific workflow composition. The prerequisite of the system is to define the structural and semantic type of input and output ports of actors and services they represent. A user then defines registration mappings to associate contextual paths on the ontologies to data objects generated in response to the queries on the service input/output, named ports. Then these input and output registration mapping rules are composed to construct correspondence mappings between structural types of the source and those of the target; DaltOn's SI implementation also generates correspondence mappings and stores them for future use as well. The approaches of Kepler and DaltOn are quite similar as they both aim at transforming data semantically based on a semantic mediation system. Although the objectives are almost identical, the design of the solution is different as Kepler does not use the semantic of the ontologies to generate mappings. Another notable difference between Kepler and DaltOn is that SEEK does not consider format conversion (syntax incompatibility) and data transportation implicitly – which is beneficial for the normal scientific user.

Triana [10][24] is a workflow-based, graphical problem solving environment. Like Kepler, Triana also provides a rich library of pre-configured and built-in tools. As far as semantic integration is concerned, unlike Kepler, Triana does not support semantic data integration. In Triana, domain users need some pre-developed tools which can perform schema transformation, generate mappings and correspondences and finally integrate data. Also the users need to know how such tools are developed, how these tools are used and in which sequence they must be applied. With the approach, a standard schema (DaLo-WF) is provided must fits most use cases but which can be adjusted in case it does not fit the requirements of an application.

TABLE III. MAPPING POSSIBILITIES IN SI

Elements of XSD1	Elements of XSD2	Preferred correspondence
record	measurement	Concept specialization
deviceCode	device	Concept equivalence
	deviceName	Concept equivalence
	location	nonFunctionalNavigation
	locationID	nonFunctionalNavigation
cumulativeSnow	data	Concept generalization
	value	Concept generalization
	characteristicID	Concept generalization
	unitID	nonFunctionalNavigation

Taverna [9], a scientific workflow management system, is part of the myGrid project. In order to convert data formats, Taverna provides "Shims", which are used as web services. DaltOn differs from Taverna in the way that it handles format conversions (syntactical conversions) dynamically. In Taverna, the domain user required some sort of specialized services that convert schemas and performs mappings as well. DaltOn instead provides a transparent way to deal with semantic integration issues.

V. CONCLUSION

This contribution discussed in detail a method for developing scientific applications. One of the main messages is that separation of concerns can help to ease handling complex application scenarios as they often occur in scientific domains. This is achieved by applying POPM which already introduces a separation of concerns and by further separating data integration tasks from domain related tasks. Thus the readability of a process is increased and domain users can focus on their expertise – the scientific analysis.

The other main contribution is an ontology based data integration framework called DaltOn; instead of fixing transformation semantics in code, it is specified as a mapping between ontologies. Since data transformation is specified on a conceptual level, changing and adjusting these transformations whenever schemas or ontologies evolve is rather easy.

REFERENCES

[1] ABCD Schema – Task Group on Access to Biological Collection, Website: http://www.bgbm.org/TDWG/CODATA/, [May 2017].

[2] Cheatham, Michelle, and Catia Pesquita. "Semantic Data Integration." In Handbook of Big Data Technologies, pp. 263-305. Springer International Publishing, 2017.

[3] Ning, Z. H. A. O. "Semantic Conflict Resolution Scheme Based on Ontology." DEStech Transactions on Engineering and Technology Research sste (2016).

[4] SEEK Observation Ontology, draft design document, Website: http://lists.admb-project.org/seek.ecoinformatics.org/,[May 2017].

[5] Groß, Anika, Cédric Pruski, and Erhard Rahm. "Evolution of biomedical ontologies and mappings: Overview of recent approaches." Computational and structural biotechnology journal 14 : 333-340, 2016

[6] Liu, Jia, et al. "Grid workflow validation using ontology-based tacit knowledge: A case study for quantitative remote sensing applications." Computers & Geosciences, 98 : 46-54, 2017

[7] Hitzler, P. "Ontology Design Patterns for Data Integration: The GeoLink Experience." Ontology Engineering with Ontology Design Patterns: Foundations and Applications, 25 : 267, 2016

[8] Crawl, Daniel, Alok Singh, and Ilkay Altintas. "Kepler WebView: A Lightweight, Portable Framework for Constructing Real-time Web Interfaces of Scientific Workflows." Procedia Computer Science, 80 : 673-679, 2016

[9] Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, Paul Fisher, Jiten Bhagat, Khalid Belhajjame, Finn Bacall, Alex Hardisty, Abraham Nieva de la Hidalga, Maria P. Balcazar Vargas, Shoaib Sufi, and Carole Goble: "The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud", Nucleic Acids Research, 41(W1): W557-W561, 2013.

[10] Vahi, Karan, et al. "A case study into using common real-time workflow monitoring infrastructure for scientific workflows." Journal of grid computing 11.3 : 381-406, 2013

[11] Jablonski, S.; Bussler, C.: Workflow Management – Modeling Concepts, Architecture and Implementation. London: Int. Thomson Computer Press, 1996

[12] ProDatO Integration Technology GmbH: Handbuch iPM Integrated Process Manager. Softwaredocumentation (in German), Erlangen, Germany, www.prodato.de [May 2017]

[13] Website BayCEER, http://www.bayceer.uni-bayreuth.de/, [May 2017].

[14] Jablonski, S.; Faerber, M.; Götz, M.; Volz, B.; Dornstauder, S.; Müller, S.: Configurable Execution Environments for Medical Processes, 4th Int'l Conference on Business Process Management (BPM), Vienna Austria, 2006.

[15] Jablonski, S.; Rehman, M.A.; Volz, B.; Curé, O.: Architecture of the DaltOn Data Integration System for Scientific Applications. 3rd International Workshop on Workflow Systems in e-Science (WSES 08), Lyon, France, 2008

[16] Jablonski, S.; Rehman, M.A.; Volz, B.; Curé, O.: DaltOn: An Infrastructure for Scientific Data Management. Proc. of the Int'l Workshop on Applications of Workflows in Computational Science (AWCS), pp.520-529, Krakow, Poland, 2008.

[17] Jablonski S.; Curé O.; Jochaud F.; Rehman M. A.; Volz B. : Semantic Data Integration in the DaltOn System. Workshop Information Integration Methods, Architectures and Systems (IIMAS) at 24th Int'l Conference on Data Engineering, Cancún, México, 2008.

[18] Staab, S.; Studer, R.: Handbook of Ontologies. Springer, Germany. 2004.

[19] Baader, F.; Calvanese, D.; McGuinness, D.L.; Nardi, D.; Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications, New York, USA: Cambridge University Press, 2003.

[20] Cure, O.; Jochaud, F.: Preference-Based Integration of Relational Databases into Description Logic. in Proc.DEXA'07, 2007, pp. 854-863.

[21] Cure, O.; Bensaid, J.D.: Integration of relational databases into OWL knowledge bases: demonstration of the DBOM system in Proc. ICDE Workshops pp 230-233, 2008

[22] Saïs, Fatiha, and Rallou Thomopoulos. "Ontology-aware prediction from rules: A reconciliation-based approach." Knowledge-Based Systems 67 : 117-130, 2014.

[23] Ptolemaeus, Claudius: System design, modeling, and simulation: using Ptolemy II, Vol. 1. Berkeley: Ptolemy.org, 2014.

[24] Website myGrid, http://www.mygrid.org.uk, [May 2017]

[25] Website Protégé, http://protege.stanford.edu/, [May 2017]