Methodology for data integration using SPARQL Constructs in the AEC industry

Gonçal Costa[1], Álvaro Sicilia[1]

[1] La Salle Engineering and Architecture, Ramon Llull University, Barcelona, Spain

Abstract. In the architecture, engineering and construction (AEC) industry, data exchange interoperability in the field of building information modelling (BIM) still presents serious limitations and problems. Many BIM software applications enable data exchange through proprietary formats and open interoperability standards such as IFC. However, the exchange of information across different domains and disciplines using a unique format such as IFC is not always feasible or does not bring about the expected results. Therefore, it is necessary to find alternative solutions to transform the data from one domain to another. Methods to automate data exchange using Semantic Web technologies have been devised in recent decades. However, it is still necessary to achieve data semanticization in a more agile and transparent way, to facilitate the work for end-users as well as for ontology designers and domain experts. In this article, we present a methodology aimed at facilitating the interoperability between a set of data models generated by BIM applications and other related data sources, and domain data models (e.g., energy, acoustics). From the experience gained in its implementation, we highlight some of the critical issues which need to be considered in semantic-based interoperability.

Keywords: Semantic Web, Linked Data, Data Integration, Ontologies, SPARQL.

1 Introduction

Collaboration in the AEC industry requires the exchange of building data between different software applications, systems and platforms. In recent years, most efforts to provide interoperability in this industry through non-proprietary formats have focused on the development of standards such as IFC and CityGML. Their capability to describe buildings designs encompassing the multiplicity of domains involved using common semantics facilitates the exchange of information among a large number of applications. These standards enable to describe a data model with different degrees of definition and levels of detail. Typically, they combine information about building geometry with other features specified through different type of properties, depending on the discipline. However, as a result of the continuous specialization of tasks in the AEC industry, new domains are emerging which also require a common representation to facilitate their exchange. For instance, the data regarding energy conservation measures and the consumption of appliances are increasingly used in the realm of building energy simulations.

Standardization is a key instrument to facilitate data exchange interoperability. However, it is still necessary to make current standards more useful, flexible, combinable and extensible. This is even more relevant today, as software applications are expanding their features and capabilities, requiring data from different sources and domains to perform their tasks more efficiently in order to satisfy the latest industry demands.

The need to both exchange the data from diverse data models and combine data from multiples domains makes it necessary to address the semantic and structural heterogeneity of the data [1]. Semantic Web technologies and, in particular, ontology engineering, are the proper techniques to carry out these tasks. Through ontology alignment it is possible to extract semantic relationships (mappings) between a data source and the target ontologies [2]. The mappings between concepts are key to facilitating the transformation of data from one domain to another. Mapping-based transformations can also be performed to integrate data from different sources in order to represent them according to a target ontology.

During the last two decades, a large number of tools (e.g., Edna, AML, AOTL, LogMap, MaasMtch, OMReasoner, RSDLWB, XMap2) have been developed within the Semantic Web community to help ontology engineers to obtain mappings between ontologies. However, there is currently a lack of methods and tools specifically intended to help experts in the AEC industry to generate the data transformations rules in specific data exchange contexts (e.g., BIM models to energy simulation models). To contribute to filling this gap, we have devised a methodology which combines the "Construct" query form of the SPARQL query language [3] with rule languages designed for the Semantic Web.

2 A methodology to integrate building data

The methodology outlined below aims at addressing data exchange interoperability in a standardized manner. The aim is to facilitate the interoperability between data models generated from BIM applications and other related data sources, and domain data models (e.g., energy, acoustics…). The methodology consists of a series of steps as summarized in **Fig. 1**:

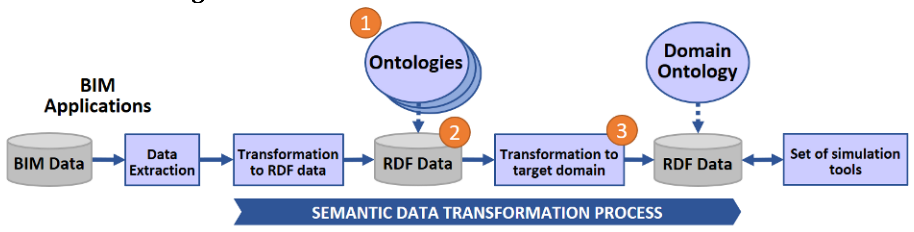

Fig. 1. Overview of the data transformation and integration process (steps are indicated in orange).

The first step is to select the ontologies to represent the data of each possible source. If they do not exist, they must be created; for example, using extraction methods from native schema formats (e.g., XSD, EXPRESS) as implemented in [4]. For example, an ifcOWL ontology [5] can be selected to represent the data of IFC models.

The second step is to transform the input data, described according to the corresponding ontologies selected, into RDF graph models [6]. This step requires software routines to carry out this conversion. In the case of the IFC, an example of the conversion of building data in STEP format into RDF data can be found in [7]. For those cases in which there are no software routines available, these must be developed. It is foreseen that in the next years there will be OWL ontologies available for the main standards of the AEC industry and the corresponding conversion routines to convert the data from native formats into RDF models. If this is the case, the work carried out in these two steps will become unnecessary since we will be able to reuse converters in order to perform the data transformation.

In the third step, the RDF data corresponding to each input data source are integrated into data models that represent a specific domain (e.g., energy, economics). These data models must be generic and representative enough to provide the input required by a set of tools that operate within a same domain. For example, an ontology of the building energy simulation domain should satisfy the input requirements of most of the energy simulation tools.

The data transformation process requires mappings between input and target ontologies. This can be done with ontology matching techniques using a tool such as the one mentioned in Section 1. If the mappings are already defined and published, then they can also be reused to create the transformation rules. The resources spent to manually curate the mappings obtained through ontology alignment tools could be reduced if the ontologies are well documented.

There are several alternative methods to provide the data transformations. One way is to codify them in a compiled program. There are some libraries available to carry out this implementation in programming environments such as C# (e.g., DotNetRdf) and Java (e.g., Apache Jena and OWL API). However, the code must be recompiled every time that a new change or update has to be performed. In addition, the transformation rules remain hidden in the code. To avoid these problems, we turn to Semantic Web languages to decouple the transformation rules from the software applications that will process them. This way, the transformations can be modified without having to recompile the applications. For example, they can be defined in SPARQL language, using the “Construct” query form, and then used as input RDF graphs to generate new ones according to the domain ontologies. This is possible by taking advantage of the logical basis of RDF and SPARQL languages. The use of data transformation rules (i.e., “construct” query forms) to overcome structural and semantic heterogeneity is noteworthy. Although the semantic differences between the input and target ontologies can be partially automated with ontology matching techniques, the participation of domain experts is required to analyse both ontologies and match their structures. The annex includes an example of transformation rule that overcomes the semantic and structural heterogeneity. The knowledge of the domain experts is formalized within the transformation rules. In this regard, it is important to formalize and share the data transformation rules in a standard language such as SPARQL because it enables the AEC community to reuse and modify the rules and to create new ones for other data sources and domains.

3 Applications

The methodology presented has been developed and applied within the OptEEmAL project [8] to provide interoperability between a set of input data sources (BIM, GIS, weather data, energy prices, etc.) and a set of building simulation tools (EnergyPlus, Nest, CitySim, etc.). The developed semantic data model encompasses the building and district scales and facilitates the data integration with other AEC domains. An example in this case is to transform the data from IFC models into instances of SimModel, a building energy data model [9]. A sample of a SPARQL construct query is included in the annex of this paper illustrating how the data are transformed for the particular case of wall elements.

The execution of the queries can be carried out through different implementations. One option is through a service programmed in Java, using the Apache Jena library [11]. In this case, the set of queries necessary to carry out the entire transformation for the given case (IFC to SimModel) is usually quite fast. In total, 40 queries were implemented to carry out the transformation with an average size of approximately 80 lines. Those queries were applied in the case of a IFC model of a real building: 11-story building including 284 rooms, taking less than 2 minutes to complete the transformation on a regular desktop computer (OS: Windows 7 Professional 32-bits, Processor: Intel® Core™2 Quad CPU Q9400 @ 2.66GHz, RAM memory: 4GB) (see the report in the Annex).

Another application of this methodology is to facilitate the transformation of the data described according to ontologies, whose data structures are similar to the native formats (EXPRESS, XSD, SQL, etc.), to ontologies where these data are represented for a more efficient use in the Semantic Web domain [12]. This situation occurs because a typical way to easily generate OWL ontologies from existing data models is through direct transformations. These ontologies facilitate the data transformation from native formats into RDF graphs. However, they often convey an unnecessary complexity that can affect, for example, the performance of using the RDF graphs for reasoning and querying purposes. One way to overcome this problem is to use lightweight ontologies which can better meet the requirements of particular application scenarios. Thus, a RDF-to-RDF transformation between a “native” ontology and a lightweight ontology can be carried out with the methodology proposed in this document.

4 Conclusions

Innovative solutions are necessary to overcome the persisting difficulties concerning interoperability in the AEC industry. We have presented a methodology based on standard language (e.g., RDF and SPARQL) to foster interoperability using Semantic

Web technologies. The ability to manipulate input data through declarative languages such as SPARQL, instead of using procedural programming, provides greater flexibility to update and maintain transformation rules. The creation of these rules is facilitated with the use of ontologies which help to overcome semantic heterogeneity and facilitate mappings discovery.

From the implementation of the methodology, we have identified some issues which still need to be solved. Firstly, the resulting data does not always satisfy the requirements of the target applications. Secondly, even if the data are transformed correctly, the results may not suit the end user's needs. These problems are not new. BuildingSMART has tried to address them in two ways: with Model View Definitions (MVD) and with the ifcDoc tool. However, these techniques are not enough to solve the problems that arise in application scenarios that require complex pre-processing operations. Producing guidelines to explain to users how to export data from the BIM model solves a part of the problem. However, the core of the problem does not lie in the export, but in the modelling with BIM tools.

Finally, it can be expected that as more ontologies and converters from native formats to RDF become freely available, the application of the methodology we have developed will become more feasible. Regarding the reusability of the queries implemented, these can be applied to respond to the needs of new transformations involving the corresponding source and target ontologies. A tool capable of selecting and applying each of the transformations to obtain a final result would make this process much more automated. The development of this tool is presented as the next step in this research line.

References

1. Sheth, A.: Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. In M. Goodchild, M. Egenhofer, R. Fegeas, & C. Kottman (Eds.), Interoperating Geographic Information Systems (pp. 165–180). Norwell, MA: Kluwer Academic Publishers (1999). http://doi.org/10.1007/978-1-4615-5189-8.
2. De Bruijn, J., Ehrig, M., Feier, C., Martíns-Recuerda, F., Scharffe, F., & Weiten, M.: Ontology Mediation, Merging, and Aligning. In Semantic Web Technologies: Trends and Research in Ontology-based Systems (pp. 95–113). John Wiley & Sons, Ltd (2006).
3. Harris, S., & Eaborne, A.: SPARQL Query Language 1.1. (2010). Retrieved from https://www.w3.org/TR/2010/WD-sparql11-query-20100126/#construct.
4. Bohring, H, and Auer, S.: Mapping XML to OWL Ontologies. Leipziger Informatik-Tage 72: 147-156 (2005).
5. Pauwels, P. & Terkaj, W.: EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. Automation in Construction 63: 100-133 (2016).
6. Hayes, J.: A graph model for RDF. Darmstadt University of Technology / University of Chile (2004).
7. Pauwels, P.: Implementation of IFC-to-RDF conversion by Ghent University (Multimedia Lab - SmartLab) and Aalto University, available online: https://github.com/mmlab/IFC-to-RDF-converter (2015), last accessed 2017/09/29.

8. Costa, G., Sicilia, A., Lilis, G. N., Rovas, D. V., & Izkara, J. L.: A comprehensive ontologies-based framework to support retrofitting design of energy-efficient districts. In proc. of European Conference on Product & Process Modelling, Limassol, Cyprus (2016).
9. O'Donnell, J., See, R., Rose, C., Maile, T., Bazjanac, V., & Haves, P.: SimModel: A domain data model for whole building energy simulation. In 12th Conference of International Building Performance Simulation Association (pp. 382–389) (2011)
10. Pauwels, P., & Van Deursen, D.: IFC/RDF: adaptation, aggregation and enrichment. In First international workshop on linked data in architecture and construction (2012).
11. Apache: Apache Jena (2015), https://jena.apache.org/
12. Pauwels, P., & Roxin, A.: SimpleBIM: From full ifcOWL graphs to simplified building graphs. In eWork and eBusiness in Architecture, Engineering and Construction: ECPPM 2016: Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM 2016), Limassol, Cyprus, 7-9 September (p. 11). CRC Press (2016).

Annex

The following is an example of SPARQL construct query developed to transform the data of wall elements from IFC to SimModel. The transformation includes the relation of the walls with their materials – via *ifc:IfcRelAssociatesMaterial* – and the corresponding layers – via *ifc:IfcMaterialLayerSet*. However, aspects such as the transformation of their geometry is implemented separately in another set of queries.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX simcore: <http://d-alchemy.com/schema/simxml/SimModelCore#>
PREFIX simres: <http://d-alchemy.com/schema/simxml/ResourcesGeneral#>
PREFIX simgeom: <http://d-alchemy.com/schema/simxml/ResourcesGeometry#>
PREFIX simbldg: <http://d-alchemy.com/schema/simxml/BuildingModel#>
PREFIX simmep: <http://d-alchemy.com/schema/simxml/MepModel#>
PREFIX simmodel: <http://d-alchemy.com/schema/simxml/Model#>

CONSTRUCT {
   ?SimModel_ID1
      rdf:type                                              simmodel:SimModel ;
      simmodel:simWall_Wall_ExteriorAboveGrade              ?IfcWall ;
      simmodel:simMaterialLayerSet_OpaqueLayerSet_Default   ?IfcMaterialLayerSet .
   ?IfcWall
      rdf:type                              simbldg:SimWall_Wall_Default ;
      simbldg:materialLayerSet              ?IfcMaterialLayerSet ;
      simcore:containingSpatialStructure    ?simBuildingStorey ;
      simcore:ifcGlobalID                   ?IfcGloballyUniqueIdValue ;
      simcore:name                          ?name_IfcRootValue ;
      simcore:simModelName                  ?name_IfcRootValue ;
      simcore:isTemplateObject              "false"^^xsd:boolean  ;
      simcore:objectType                    "Wall"^^xsd:string ;
      simcore:simModelSubtype               "Default"^^xsd:string ;
      simcore:simModelType                  "Wall"^^xsd:string ;
      simcore:simUniqueID                   "not defined" ;
      simcore:sourceModelObjectType         "IfcWall"^^xsd:string ;
      simcore:sourceModelSchema             "IFC4" ;
      simcore:tag                           "not defined" ;
      simcore:refId                         ?simWallRefId .
   ?IfcMaterialLayerSet
      rdf:type                     simres:SimMaterialLayerSet_OpaqueLayerSet_Default ;
      simcore:simUniqueID             "not defined" ;
      simcore:simModelType            "OpaqueLayerSet"^^xsd:string ;
      simcore:sourceModelSchema       "IFC4"^^xsd:string ;
      simcore:simModelSubtype         "Default"^^xsd:string ;
      simcore:sourceModelObjectType   "IfcMaterialLayerSet"^^xsd:string ;
      simcore:isTemplateObject        "false"^^xsd:string ;
      simres:materialLayers           ?MaterialLayerListNode ;
      simres:layerSetName             ?name;
      simcore:refId                   ?RefMatLayerId .
} WHERE {
   ?IfcWall rdf:type ifc:IfcWall .
   ?simBuildingStorey rdf:type ifc:IfcBuildingStorey .
   ?IfcRelContainedInSpatialStructure
      rdf:type ifc:IfcRelContainedInSpatialStructure ;
      ifc:relatingStructure_IfcRelContainedInSpatialStructure  ?simBuildingStorey ;
      ifc:relatedElements_IfcRelContainedInSpatialStructure    ?IfcWall  .
   ?IfcWall
      rdf:type ifc:IfcWall ;
      ifc:globalId_IfcRoot  [express:hasString ?IfcGloballyUniqueIdValue ] ;
      ifc:name_IfcRoot      [express:hasString ?name_IfcRootValue ] .
   bind( strafter( xsd:string(?IfcWall), "IfcWall_" ) as ?simWallRefId ).
   ?IfcRelAssociatesMaterial
      rdf:type  ifc:IfcRelAssociatesMaterial ;
      ifc:relatedObjects_IfcRelAssociates  ?IfcWall ;
         ifc:relatingMaterial_IfcRelAssociatesMaterial
         [ ifc:forLayerSet_IfcMaterialLayerSetUsage ?IfcMaterialLayerSet] .
   ?IfcMaterialLayerSet
      rdf:type  ifc:IfcMaterialLayerSet ;
      ifc:materialLayers_IfcMaterialLayerSet ?MaterialLayerListNode ;
      ifc:layerSetName_IfcMaterialLayerSet [ express:hasString ?name] .
   bind( strafter( xsd:string(?IfcMaterialLayerSet), "IfcMaterialLayerSet_" )
      as ?RefMatLayerId ) .
   bind( URI(CONCAT(STRBEFORE( xsd:string(?IfcWall), "IfcWall_" ), "SimModel_ID1"))
      as ?SimModel_ID1 ) .
}
```

The following is the result obtained when processing the IFC model of the building, described in Section 3, in a Java application implemented according to the proposed methodology:

```
IFC input file:    files/TURv19x4_ENHANCHED.ttl
----------------------------------------------------------------------
Invoking the following sparql CONSTRUCTS (40):

Triples| msec|Objects| Query
----------------------------------------------------------------------
 117173| 2069|  44600|DoubleList_LengthMeasure
   2505|   33|  45540|DoubleList_RealList
    256|   86|     15|SimBuildingStory_BuildingStory_Default
     19|   14|      1|SimBuilding_Building_Default
    105|   16|     13|SimDerivedUnitType_DerivedUnit_Default
    794|  353|     61|SimDoor_GlazedDoor
   3947|  338|    279|SimDoor_OpaqueDoor
     13|    3|      2|SimGeomCurve_CompositeCurve_Default
 174508| 3031|   6387|SimGeomCurve_Polyline_Default
 153981|  940|  17109|SimGeomPoint_Point_CartesianPoint
   8119|   86|    369|SimGeomSolidModel_SweptAreaSolid_ExtrudedAreaSolid
  43198|  292|   3927|SimGeomSurface_BoundedSurface_CurveBoundedPlane
  76341|  702|   7634|SimGeomSurface_ElementarySurface_Plane_Plane
   3151|   20|    315|SimGeomVector_Vector_Direction
  10011|   74|    770|SimIrregularTimeSeries_IrregularTimeSeries
   9895| 1043|    598|SimList_MaterialList
     11|  343|      1|SimLocalPlacement_LocalPlacement_AbsolutePlacement
  42395|  432|   3854|SimLocalPlacement_LocalPlacement_RelativePlacement
    186|    4|     74|SimMaterialLayerLists
     23|   10|      2|SimMaterial_OpaqueMaterial_NoMass
     14|    3|      1|SimModelRepresentationContext_GeometricRepresentationContext_Default
     10|    3|      1|SimOwnerHistory_Default_Default
      0|   17|      0|SimPanel_Glazing
      0|   18|      0|SimPanel_Opaque
  14707|  155|   2451|SimPlacement_Axis2Placement2D_Default
 145129| 1280|  12094|SimPlacement_Axis2Placement3D_Default
     79|   52|      1|SimProject_Project_DesignAlternative
    163|    4|     18|SimSIUnitType_SiUnit_Default
     17|    5|      1|SimSite_BuildingSite_Default
    478|   14|     25|SimSlab_Default_Default_Slab
  73729|  762|   3927|SimSpaceBoundary_SecondLevel_SubTypeA
      0|   43|      0|SimSpaceBoundary_SecondLevel_SubTypeB
      0|    6|      0|SimWall_Wall_Default_StandardWall
  21103| 8015|   1301|SimWall_Wall_Default_Wall
   3486|   40|    205|SimWindow_Window_Default_Window
    451|    6|     50|SimZone
  54982|  897|  18480|Values_List
    140|   93|      1|simMaterial_Glazing
    601|   77|     37|simMaterial_Opaque
   2321| 1811|     80|simSpace_Occupied_Default

Queries completed in 25,269 seconds

Total objects created: 170168

Materializing classes according to: ontology/SimModelComplete.owl
Total triples before materialization: 963875
Materializing inferred types of the new model
Total triples after materialization: 963875
----------------------------------------------------------------------
Storing output/TURv19x4_ENHANCHED_sparql.ttl model with a total of 963875 triples
```