A METHODOLOGY FOR EXTENDING MUNICIPAL COUNCIL PROPERTY TAX INFORMATION SYSTEM (MCPTIS) INTEROPERABILITY TO GEO-SPATIAL DOMAIN

R.D. Kene[1] and P.N. Mulkalwar[2]

[1]Dept. of Computer Science, Adarsh Education Society's Arts, Commerce & Science College, Hingoli

[2]Department of Computer Science, Amolakchand Mahavidyalaya, Yavatmal

rvkene@gmail.com, pnm_amv@rediffmail.com

ABSTRACT

This paper presents a methodology for extending MCPTIS interoperability to geo-spatial domain. It consists of five stages. First, an IFC-compliant ontology describing the hierarchy structure of MCPTIS objects, their relationships and their properties is developed. The emphasis is on semantic indexing and retrieval of taxable property information from an Industry Foundation Classes (IFC) model. Second, ontology mapping is used to link similar relationships or concepts between the source (e.g. MCPTIS) and target (e.g. Geographic Information System GIS) ontologies. The output is an extended ontology that contains all classes and properties from both MCPTIS and GIS domains, which are relevant to the case study and use cases examples. Then, taxable property's elements and GIS data are translated into Resource Description Framework / Web Ontology Language format, thus can be processed by semantic web applications. Once the information has been gathered from different sources and transformed into an appropriate semantic web format, the SPARQL query language is used in the fourth section to retrieve this information from a dataset. Also, a new process is developed to translate semantic web query results into the XML representations of the IFC schema and data.

Keywords: *Interoperability, MCPTIS, GIS, SPARQL, IFC, XML*

Introduction

In this paper, we concentrate on developing a platform to promote the implementation of applications on distributed cloud systems. Further aim is to research that how minimal human involvement can be derived from an acceptable model for deploying an application. Our framework must be able to include, beginning with an implementation overview, how the various pieces of the applications must be implemented on distributed platforms. The approach proposed and tested in this paper suggests a technique to infer the required implementation model that applies some of the Semantic Web field ideas. Since the purpose of this study is to bridge the gap between MCPTIS and GIS models at the semantic level by employing semantic web technology, we describe how semantic web technology can be used to connect the MCPTIS and GIS data together in meaningful ways. This study adopts ontology mapping methods that are being increasingly used to map MCPTIS and GIS ontologies.

Perceptualization

Ontologies may be divided into upper, domain, and application ontologies, based on the elements and the level of information. An upper ontology defines broad concepts that are appropriate across a broad range of domains, for example, time, complexity, position, entity, action, etc. The most popular upper ontologies include DOLCE, GFO, BFO, PROTON, Sowa's ontology, and SUMO. In order to formalize and describe shared concepts in a particular field of interest, domain ontology is developed. For illustration, a rule of a particular position may be represented: a landscaper utilizes a measuring tape to obtain a measurement; therefore "landscaper" is an example of the concept landscaper, "measuring tape" is an example of the measuring instrument of the concept, and "utilizes" and "take a measurement" are often used to establish the relationships between these perceptual items. The e-COGNOS seems to be a primary manifestation of those kinds of ontologies (as well as several other forms following subsections discuss). Application ontology seems to be a description of such semantics of a general, centered field of interest that describes the applicable principles for a specific application (e.g. MCPTIS or GIS). This phase explores how far this level of ontology is being used to establish semantic

interoperability among both MCPTIS and GIS services, considering the growing significance among application ontologies throughout accelerating the convergence of various types of knowledge. Previous efforts to establish ontology in the Architecture Engineering Construction (AEC) have certainly paved the way for the smooth unification of data relevant to real estate and municipal council / corporation property tax information system, but there is no framework ontology for the domain of the real estate and municipal council / corporation property tax information system that covers all IFC classes with distinctly different attributes. Therefore, based on the EXPRESS schema, we create a novel ontology. The emphasis has been on elements of the IFC schema, such as attributes, data types of attributes, entities, relationship, individuals.

The semantic web society tends to demonstrate an increasing intent to embrace description logic as the formal structure tool to define the application domain in an organized way. In brief, description logic models the application domain by defining the basic concepts of the domain and simply using those concepts to describe the properties of an object and the domain users. In terms of definitions, roles (e.g. relationships and properties) and entities, description logic often defines the domain in simple terms (or instances). The suggested methodology integrates the real estate and municipal council / corporation property tax information system for conceptual modeling by using description logic definitions.

There seem to be two kinds of concepts: in this analysis, basic concepts are used to describe the basic classes of the IFC domain in which only the essential conditions are defined and their description will identify them. To characterize subclasses of the primitive ones, given concepts have been used (i.e. built using primitive concepts and properties). We thus describe as "primitive" concepts the IFC classes there at highest point of the ontology's hierarchical structure. Want us to presume, for instance, that an entity x is just an instance of IfcWindow (as something of a primitive concept), so x has IfcWindow's properties which including 'overall height' and 'overall width'. IfcWindowStandardCase describes the standard window that is integrated through an opening as well as its profile describes a rectangle from the inside of the 2D plane including its opening.This IFC entity is characterized as a "defined concept", and therefore corresponding IfcWindowStandardCase properties are essential and adequate. Afterwards, let us say that maybe an individual y would be an instance of IfcWindowStandardCase (as just a given concept), so that y has the properties of IfcWindowStandardCase and that the individual y that has the set of IfcWindowStandardCase related properties (e.g. embedded in an opening, etc.) is sufficient to infer an instance of IfcWindowStandardCase.

We describe the IFC classes (or conceptual frameworks in description logic) mostly by their supertype entities and even their relations with the other classes for the purpose of defining the ontology. IfcConstructionType can be described, proceeding from the above examples, as follows:

(defprimclass IfcConstructionType (?be IfcCapitalValue)

:=> (and (exists (?oh)

(and (OverallHeight ?oh)

(>= (OverallHeight ?oh) 0)))

A primitive IfcConstructionType concept, which is a subtype of IfcCapitalValue, is defined. To define the primitive concepts and add a set of essential but not adequate conditions, the keyword 'defprimclass' is used. The expression above also notes that there is at least one "OverallHeight" in all "IfcConstructionType" classes, which is a positive unit of measure, greater than zero. The mentioned definition 'IfcConstructionResidentialType' is characterized as 'defconcept IfcConstructionResidentialType' where 'defconcept' generates named descriptions that classify entity sets or classes.

Modern languages of ontology, which including Web Ontology Language, are based on description logic. Web Ontology Language defines it in forms of classes (instead of concepts), properties (instead of roles) and

individuals, as description logics characterize the domain in terms of concepts, roles, and individuals. In general, description logics have driven the formal specification of the Web Ontology Language and its Resource Description Framework / XML exchange syntax has been affected by an upward compatibility criterion with the Resource Description Framework. We use these ontologies from the Web Ontology Language to construct the application ontology. Through specifying appropriate and/or adequate properties to a class, the Web Ontology Language axioms offer semantics regarding classes and properties. The SubClassOf axiom depicts the subclass/superclass relationship, because even though IfcConstructionType is a subclass of IfcCapitalValue, all IfcCapitalValue attributes are inherently inherited, but not any other manner. The basic concepts presented by "defprimconcept" with "subclassOf" axioms are converted into Web Ontology Language. The axiom of EquivalentClasses says that the two or even more class expressions serve the purpose of the same group of individuals since they are identical to one another and it is assumed that the subclass relationship goes in both directions. In description logic, the expressions using "defconcepts" associate to "equivalentClasses" in Web Ontology Language.

To categorize the IFC classes in taxonomical hierarchy, in this proposed methodology we describe owl:Things containing individuals or we can say that every class is rdfs:subclassOf owl:Thing. A corresponding Web Ontology Language class is created for each IFC component within EXPRESS. The attributes and properties are then translated into the required attributes and properties of the Web Ontology Language. To every ENTITY definition inside this EXPRESS schema, we produce a related owl:Class in ontology. In addition, we are using a hierarchical entity structure to describe the IFC entities, whereby each entity is connected by subtype/supertype relationships with every other entity. These relationships "Subtype of" and "Supertype of" are translated into relationships with rdfs:SubClassOf and rdfs:SuperClassOf. To define subjects and objects, URI references are also used in the Resource Description Framwork / Web Ontology Language models. To transcribe the EXPRESS entities and relationships within them, this proposed research needs to take advantage of the various URIs. In order to explain the relationships between individuals and literal data (string, numbers, data types) and object properties, the suggested approach uses object properties to connect individuals to other individuals. For example, the "CompositionType" property connects string values to "IfcTaxbleProperty". We add the "hasAttribute" property that corresponds to "CompositionType" with "IfcTaxbleProperty". Any properties should be described as an IFC object. Therefore, "rdfs:isDefinedBy" is used to state that an IFC object identifies a property (e.g. BuildingAddress) (e.g. IfcPostalAddress). Phase 3 specifies how, using description logic including Web Ontology Language syntax, the IFC schema can even be elevated to an ontological level. Figure 2 depicts IFC ontology and its Resource Description Framework graph.

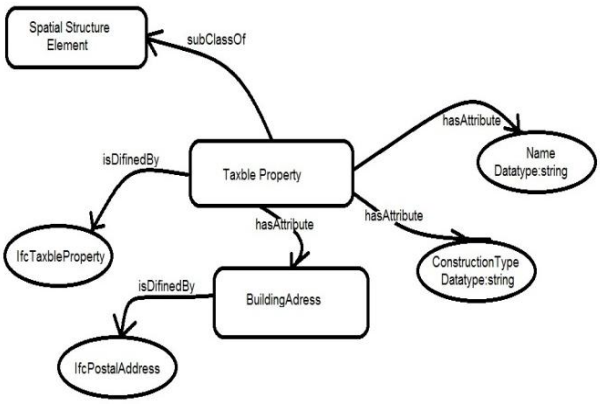

Figure 2: Characterization of an EXPRESS entity (i.e. TaxbleProperty) as Resource Description Framework graphs

Unification

In this stage, ontology mapping is being used to describe semantically comparable definitions between MCPTIS and GIS ontologies, e.g. terms that are semantically identical and comparable. While a full analysis of different methods for representing multiple ontologies is outside the reach of this study, an example offers a brief summary of the method. Presume how users (as MCPTIS users) would

like to combine latitude, longitude and altitude data with location of the building on the site regarding current workplace environmental conditions. The geomorphologic information was analysed using a GIS software product and manipulated. We therefore accept the graph structure paradigm to reflect components of both the two ontologies throughout this analysis in order to convert GIS or reference ontology objects and instance through the MCPTIS or destination ontology. The Resource Description Framework Graphs of GIS ontology, implemented by the Center of Excellence for Geospatial Information Science (CEGIS) (USGS 2013) and MCPTIS ontology, focused mostly on IFC schema, are seen in Figure 3 as just an illustration. Resource Description Framework(s) and Web Ontology Language are represented by these two distinct (and though comparable) ontologies.

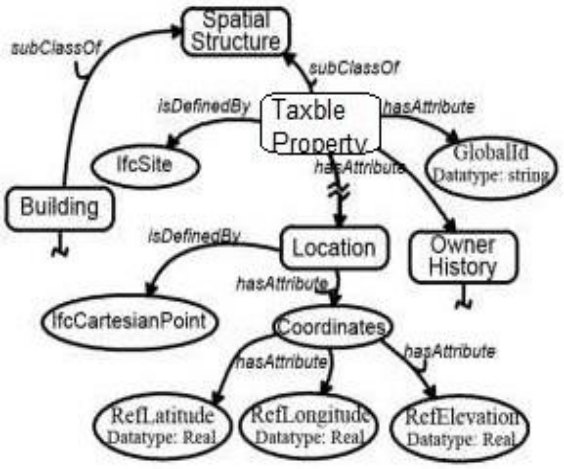

Figure 3(a): The Resource Description Framework graph of MCPTIS ontologies

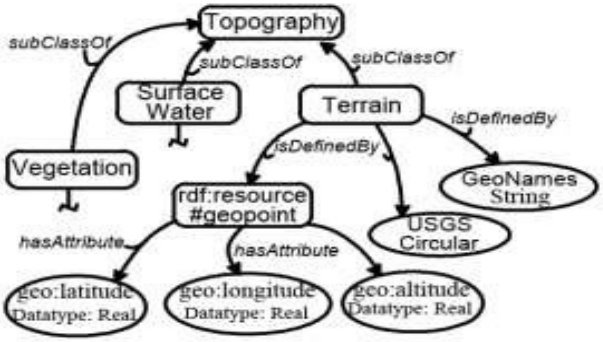

Figure 3(b): The Resource Description Framework graph of GIS ontologies

To measure the degree of correlation of triples such as subject, predicate and object, we equate the architectures of entities of interest. It is capable of calculating the similar characteristics amongst MCPTIS and GIS ontologies by following the method of Graph Matching for Ontologies (GMO). This ontology-matching technique employs the bipartite graph model of the resource description framework to describe ontologies. The Resource Description Framework bipartite graph including its GIS ontology is seen in Figure 4 in which properties nodes have been described through circles, class expressions have been expressed by circular rectangles and edge symbols S, P, O denote subject, predicate and object.

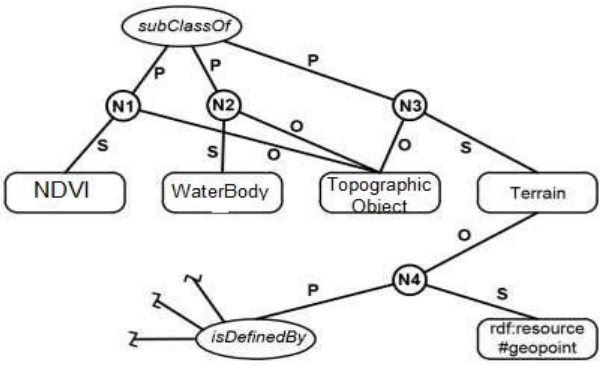

Figure 4: The Resource Description Framework bipartite graph of the GIS ontology.

There would be the following block structure in the adjacency matrix of its directed bipartite ontology graph, denoted by A:

$$A = \begin{pmatrix} 0 & 0 & A_{ES} \\ 0 & 0 & A_S \\ A_E & A_{OP} & 0 \end{pmatrix}$$

A_{ES} is a matrix describing references with external entities (such as rdfs:subClassOf) to statements; A_S is a matrix describing references with ontological entities (internal entities) to statements; A_E is a matrix describing references with statements to external ontological entities; and A_{OP} is a matrix describing references with statements to internal entities. The external entities usually involve a few similar ones (e.g. rdfs:subClassOf, rdfs: isDefinedBy) there in example as shown in Figure 3(a) included in two ontologies. Besides that, AES seems to be

a zero matrix whether such entities were not included through the as subjects of Ontology (as shown in Figure 3(b)). In Figure 5, the matrix depiction of GIS ontology seems to be as follows:

subClassOf											
isDefinedBy											
hasAttribute											
NDVI									1	0	0	0
Water Body									0	1	0	0
Topographic Object									0	0	0	0
Terrain									0	0	1	0
#Geo:Point									0	0	0	1
N1	1	0	0	0	0	1	0	0				
N2	1	0	0	0	0	1	0	0				
N3	1	0	0	0	0	1	0	0				
N4	0	1	0	0	0	0	0	1				

Figure 5: Matrix representation of GIS ontology

In Figure 5, as illustration, researchers categorize entities as properties (e.g. rdf:type and rdfs:subClassOf), classes (e.g. Terrain), and instances (e.g. individuals and data literals). It will also increase the probability towards establishing a comparable relationship or definition by adopting a particular classification (e.g. RefLatitude and geo:latitude are class entities) and role (e.g. subject or predicate). Even before mapping across two domain ontologies, the built-in properties, datatypes, and URIs utilized within all ontologies should have been considered. As one and the same, it tends to result in similar semantics for almost any two adjacent URIs. Then, the matrix of similarities of MCPTIS ontology entities to GIS ontology entities and the external GIS ontology entities to the external MCPTIS ontology entities may be represented. The structural similarity matrix of MCPTIS and GIS ontologies is constructed, predicated on the formulation. We considered these same aforementioned entities to be comparable concepts with regard to the structural similarity between some of the described Resource Description Framework graphs:

geopoint and CartesianPoint, GeoNames and GlobalId, geo:latitude and

RefLatitude, geo:longitude and Ref Longitude, and geo:altitude and RefElevation.

The resulting ontology specifies the relationship among both MCPTIS and GIS ontologies, along with correspondences.

Denotation

In this stage, developers transmit the components of real estate construction site and Geographic Information Systems information further into languages of formal ontology, Resource Description Framework / Web Ontology Language. Since the Resource Description Framework seems to be an outstanding alternative to XML in some kind of a way that would allow through dynamic data sharing among programs, this becomes easier to translate MCPTIS and Geographical Information Systems data to XML-like specifications, including such IFC for BasePropertyValue and GML for those GIS data formats. Figure 6 indicates the EXPRESS entity from the IFC class where certain 'Subject' is often utilized to model physical items which including ConstructionYear or BasePropertyValue, 'Property' is used to allocate the subject property (i.e. ConstructionYear) to the year and 'Value' or 'Unit' that is utilized to determine the year property. GML specifies characteristics of physical entities (e.g. Peth, Road) utilizing basic properties like strings, float values, integer values, boolean values and geometric primitive entities which includes Points, Lines, and Polygons. Since the classic GML framework has been founded mostly on

Resource Description Framework, it has several Resource Description Framework characteristics, along with concept of expressing knowledge in "striped form" by defining property values on objects such that nodes and edges are additionally described by components.

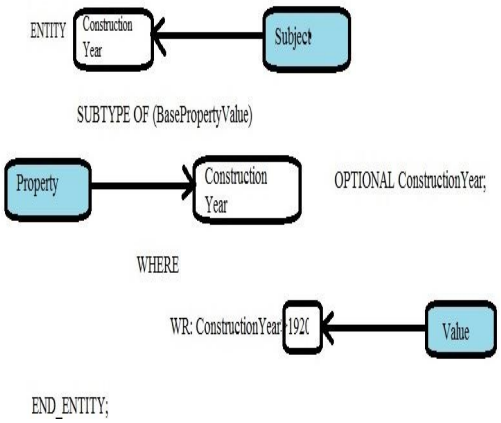

Figure 6: Example of EXPRESS entity

The header of an IFC file indicates its name, definition, version of the translator (if it is used) and version of the schema. In the context of every IFC code, the IFC entities with some of their own attributes are defined (both standard and entirely optional). Any IFC entity begins only with a particular character '#' trailed with any integer. Because each customer requires an unique id with in Web Ontology Language, designers is usingsuch unique numbers to identify the most specific instances or individuals in various classes. Every other additional IFC entity has been specified only after '=' symbol, as well as its attributes are described in brackets by a series of comma-separated properties. Standard attributes often obtain non-null values about an IFC entity, however optional attributes which have null values implied by just a '$' symbol.

Although every taxable property information (e.g. capital value, tax rate, base property value, units, built up area, type of building, type of property, age of property, floor of property, construction type and etc.) is described as a collection of further IFC entities, for all IFC entity definition, we first describe a Web Ontology Language class called a "Entity"
(i.e. <owl:Class rdf:about="…Entity"/>). After which, mostly as subtype of such a generalized class developers describe the standardized supertypes of all the other IFC entities. On equal hierarchical level, , IFC entities which including "IfcCapitalValue", "IfcTaxRate", "IfcBasePropertyValue" and "IfcBuiltUpArea" are the highly generalized IFC classes. IfcCapitalValue, for example, is stated as shown below:

<owl:Class rdf:about="…IfcCapitalValue">

<rdfs:subClassOf rdf:resource="…Entity"/>

</owl:Class>

A association (i.e. subClassOf) among each subject (i.e. IfcCapitalValue) described through 'rdf:about' as well as an object (— for example Entity) described through 'rdf:resource' has been expressed by a Resource Description Framework Data model. We proceed with that of the subtype of every IFC class until either the ontology cannot be joined by any further IFC entities. For eg, the supertype of only about two entities is "IfcCapitalValue"; "IfcBasePropertyValue" and "IfcBuiltUpArea" all seem to have zero child nodes; (i.e. There are almost no subtype entities within that ontology).

We categorized these IFC attributes further into three categories as per respective properties to accomplish the transformation among IFC and Resource Description Framework / Web Ontology Language transcripts; (1) leaf node, (2) simple type, and (3) complex type. The attribute of a leaf node would be an IFC hierarchal structure attribute which has zero child nodes or attributes. The "value" seems to be the only parameter needed to describe an attribute of every leaf node. A IfcOwner entity, for instance, includes four attributes for its leaf node; AdharNumber (ID) (optional), Name (normal), Description (optional) and Addresses (optional). The values (e.g. Shri. Ajay Patil as a string value) should be the only parameters that needed to define exactly the properties of all these leaf nodes. It is thus possible to describe this IFC entity as #1= IFCOWNER ($, 'Shri. Ajay Patil 604520143452 (ENU)',$,$,$);. We need an owl:Class to describe the IFC entity as its subject of the Resource Description Framework declaration irrespective of the nature of attributes by using the "rdf:about"

statement to define the subject. Consequently, to say that this IFC entity is often a subtype of yet another entity or resource, the rdfs:subClassOf property has been utilized. Each Web Ontology Language Class is therefore a subclass of owl:Thing. The entity for IfcTaxableProperty is described as shown below:

<owl:Class rdf:about=".. IfcTaxableProperty">

<rdfs:subClassOf rdf:resource="…IfcEntity"/>

</owl:Class>

As a Web Ontology Language data type property, researchers describe each leaf node attribute since it represents literal data (for example, characters, integers, boolean data, and so on) with an IFC entity. The literal data type as specified through 'rdfs:range' and 'rdfs:domain' is being utilized to show that an instance including its IFC entity is perhaps the leaf node attribute. The attribute 'Name' for this with the IfcOwner entity is specified as shown below:

<owl:DatatypeProperty rdf:about="…Name">

<rdfs:domain rdf:resource="…IfcOwner"/>

<rdfs:range rdf:resource="&xsd;string"/>

</owl:DatatypeProperty>

The same as XML, among its open and closed brackets (< .. >), certain properties for all the Web Ontology Language class must be declared. Whereas if "Name" attribute string value does seem to be "Shri. Ajay Patil' should have been described as shown below:

<owl:NamedIndividual rdf:about="…Name">

<…Name rdf:datatype="&xsd;string">

Shri. Ajay Patil 604520143452 (ENU)

</…Name>

</owl:NamedIndividual>

An IFC entity describes a basic form of attribute. In that same way, the IFC entity serves as somewhat of an attribute value. IfcSoftware, for example, contains a single basic form attribute, SoftwareDeveloper, described by the entity of IfcSoftwareDeveloper, with three attributes of the leaf node; Version, SoftwareFullName, and SoftwareIdentifier.

We utilise "rdfs:isDefinedBy" throughout the Web Ontology Language file for describing the basic form attribute. In addition, "rdfs:domain" is often utilized to declare that the basic form attribute is just an IFC entity instance. The attribute 'SoftwareDeveloper' for IfcSoftware is described as below:

<owl:DatatypeProperty rdf:about="…SoftwareDeveloper">

<rdfs:domain rdf:resource="…IfcSoftware"/>

<rdfs:isDefinedBy rdf:resource="…IfcOwner"/>

</owl:DatatypeProperty>

There are so many IFC entities which can be described with just a single attribute without any kind of reference to yet another IFC entity, which includes IfcRatioMeasure_NORM, IfcRatio_Measure, and IfcSpecular_EXPN. Thus, we do not describe such distinct entities as a class of Web Ontology Language, but rather as a property including its Web Ontology Language data type. The taxonomy of properties is constructed by rdfs:subPropertyOf, analogous to the role of 'rdfs:subClassOf' in defining their taxonomy of IFC classes. Even though the different IFC entities are being used to define an IFC attribute, other IFC attributes are specified as 'subPropertyOf'.

For example, the entity IfcMeasure WithUnit includes two basic forms of attributes: the value specified by the separate entity IfcRatio_Measure and the Unit described by the entity IfcUnit. The attribute 'ValueComponent' of IfcMeasureWithUnit is described as shown below:

<owl:DatatypeProperty rdf:about="…IfcRatio_Measure">

<rdfs:subPropertyOf rdf:resource="…Valuet"/>

<rdfs:range rdf:resource="&owl;real"/>

</owl:DatatypeProperty>

One and sometimes more sub-attributes defining the properties and values are used in a complex form of attribute. Furthermore, "Concept Type" or "C-Type" defines a separate list or ordered collection of sub-attributes.

IfcCartesian_Coordinate_Point, for example, uses a single complex form attribute, Coordinates, described with more than just a list of 1 to 3 sub-attributes (such as IfcLength). Researchers describes the concept type properties (e.g. matrix, range, and group) through property axioms or through further precisely range axioms to articulate this restriction. The range of values for its 'Coordinate' attribute is often restricted to either a list of the first, second and/or third sub-attributes of IfcLength. Additionally, either specific IFC entities or a group of individuals describe those sub-attributes. Thus, to describe certain distinct IFC classes, researchers utilize 'subPropertyOf' tag. The IfcCartesian Coordinate Point attribute 'Coordinates' is described as shown below:

```
<owl:DatatypeProperty rdf:about="…-Coordinates">
        <rdfs:domain rdf:resource="…IfcCartesian_Coordinate_Point"/>
        <rdfs:range>
                <rdfs:Datatype>
                        <owl:oneOf>
                                …
                                        <rdf:type rdf:resource="&rdf;Vector"/>
                                        <rdf:first>pos=0</rdf:first>
                                        …
                        </owl:oneOf>
                </rdfs:Datatype>
        </rdfs:range>
</owl:DatatypeProperty>
…
<owl:DatatypeProperty rdf:about="…IfcLength">
        <rdfs:subPropertyOf rdf:resource="…Coordinates"/>
        <rdfs:range rdf:resource="&owl;real"/>
</owl:DatatypeProperty>
```

The IFC to Resource Description Framework / Web Ontology Language transformation method flowchart for various kinds of IFC attributes is as seen in Figure, abovementioned method proceeds till no further attributes remain for yet more transformations. Figure provides a good example of that kind of translation, in which an IFC feature throughout the EXPRESS schema has been translated into such an ontology format composed in the Web Ontology Language. As seen in Figure including its EXPRESS data model, 'Subject' should be utilized to characterize a physical entity including a taxable property, 'Property' should be utilized to define a attribute (such us Name, CompositionType) as the subject (that is a taxable property) as well as 'Value' or 'Individual' is utilized to determine the value for this same property. In this phase, various ontology builder applications have been utilized to manipulate and simulate the language of ontology and then donate to other how classes as well as entities are correlated. Protégé has become one of the earliest and yet quite commonly utilised ontology scripting application to mapping the Resource Description Framework, which is still extensively utilized. The W3C has built up effective semantic web applications which including CWM which can be utilized in the Resource Description Framework format to parse IFC documents. CWM is a generalized data processor including compiling, parsing, testing and transforming semantic web content (W3C 2013).

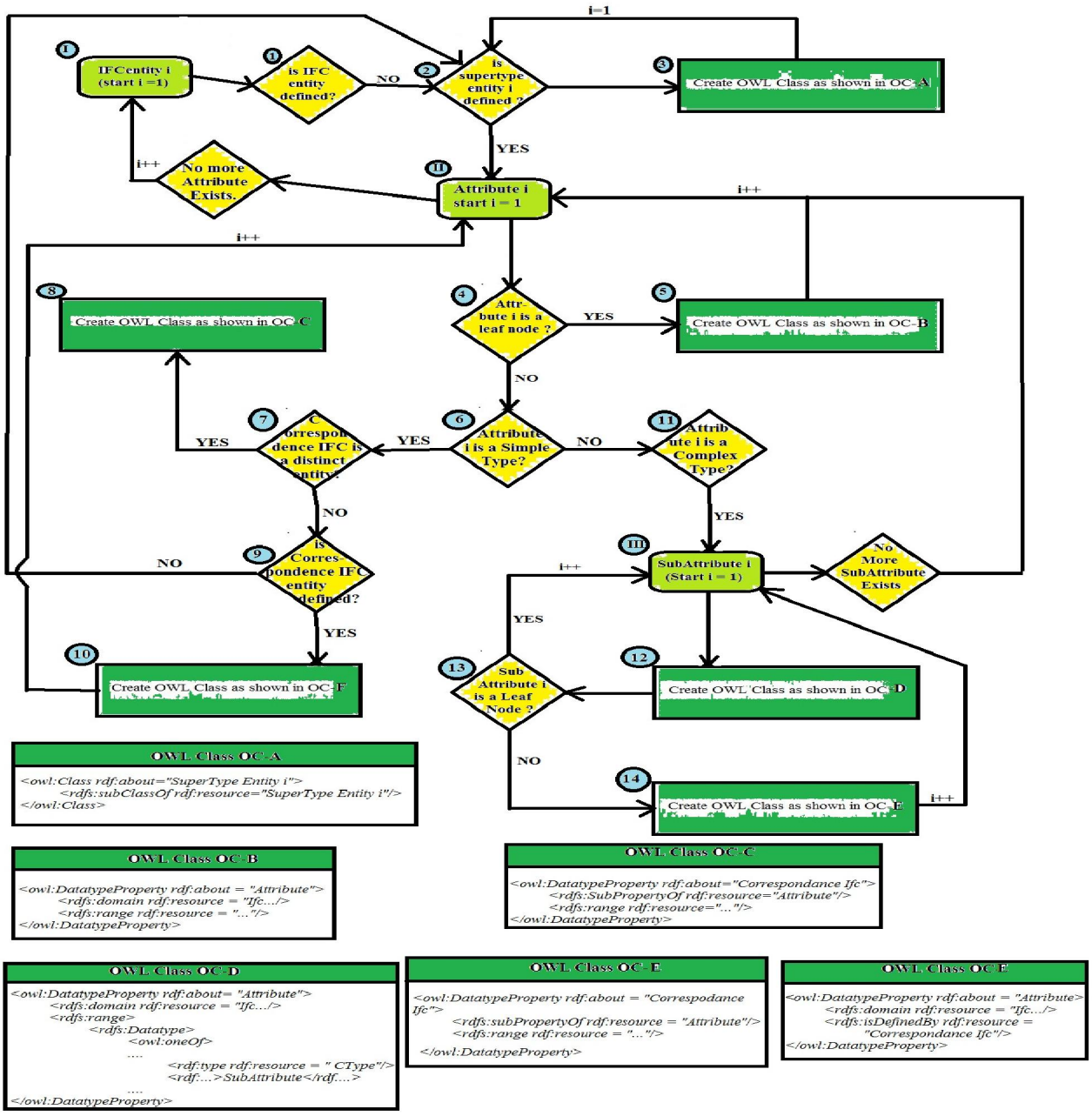

Figure 7: Workflow Diagram of IFC to Resource Description Framework / Ontology Web Language transformation

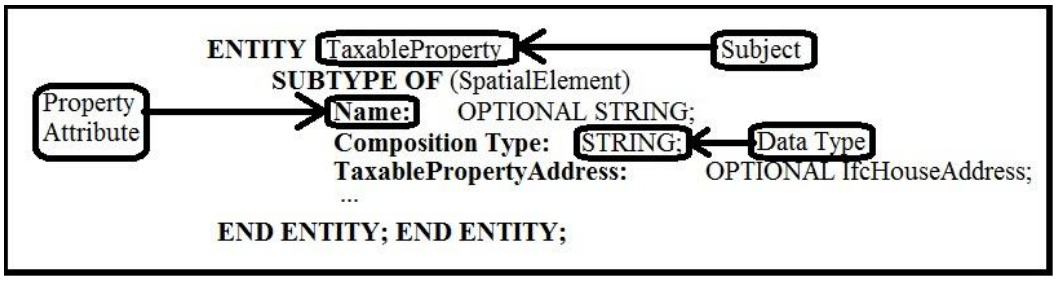

Figure 8: EXPRESS entity

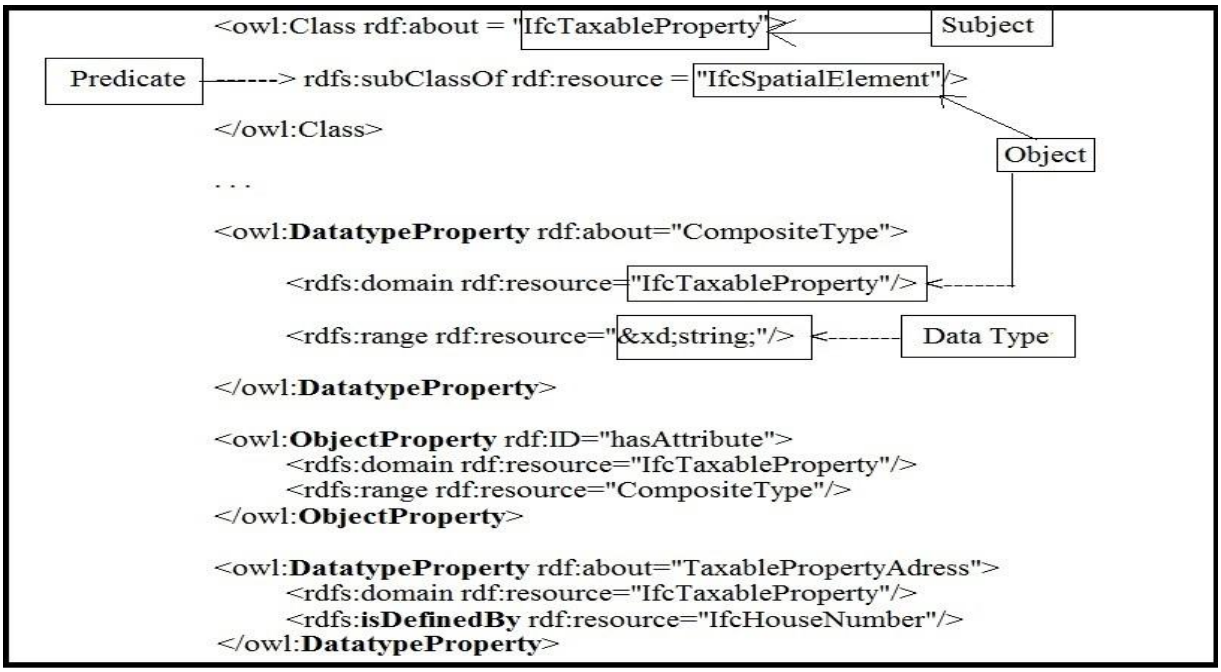

Figure 9: Web Ontology Language Entity Transformation for Figure 8

The majority of Geographic Information Systems have always been focused on relational models which organize information as per data or relationship tables. A Geographic Information Systems relational database seems to be a collection of relationships (tables) that included a fixed number of variables or attributes, and that to describe predicates among various classes a semantic web approach utilizes description logic. Defining relational terminologies as well as recognizing their ontological equivalents must be the baseline for translating a GIS relational database itself into Resource Description Framework. A relation (or table) is grouped into tuples (or rows) with the same attributes (or columns). Every one of the relations has been described as kind of a class (a group of things one that associate with those properties.). However if the terms haven't already been exist inside this Resource Description Framework schema, a class has been specified as owl:Class (a subclass of rdf:Class). Every attributes correlated with such a relation may be described as properties by this specification. A Resource Description Framework property specifies a relationship among subjects and objects, as stated previously. When defining relationships among instances of two classes, object type properties have been utilized. Instead of that, each relationships among a class instance with values are described using several properties including the data type. Within that relational database, constraints are yet another significant aspect. For a particular attribute they permit that to restrict the available values. For example, an altitude attribute may be constrained by a constraint to values within 300 to 350 meters. Any of these constraints can be represented in the Resource Description Framework schema by rules.

A primary key of a relational table distinctly determines the feature of every other tuple within the table in relational database terminology. During this research, the primary key is defined as a functional property axiom, since for each case, a functional property should only have one (distinctive) value.

Geographical Information Systems information can be retrieved from the database through Quantum GIS, a free and open - source GIS application which facilitates the use of GML as well as other data formats used to translate data. The translation stage uses the GeoTools Application Programming Interface (APIs), which itself is applicable, to manipulate as well as parse GML data further into Resource Description Framework. GeoTools is a java based Application Programming Interface designed and supported by the Open Source Geospatial Foundation, that offers guidelines complaints methodology for manipulating GIS

information. It should always be remembered that those same tools merely allow the exchanging and manipulating the data and also that the approach suggested is irrespective of some particular set of applications. An instance of this kind of transformation is seen in Figure 10, in which a relational database is first transformed to GML and afterwards translated.

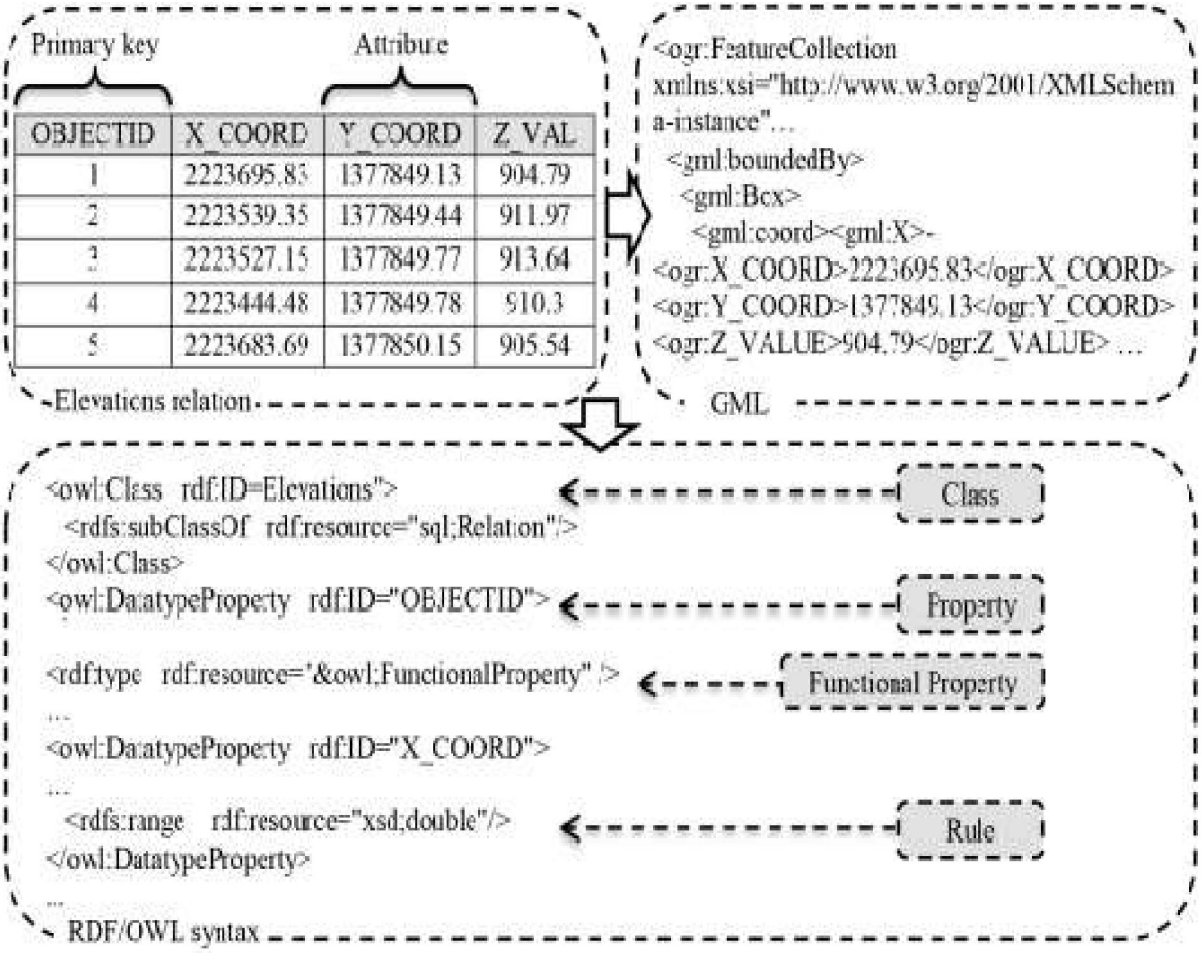

Figure 10: Transformation Example of Relational database to ontology database

Translation

Through constructing an interoperable interface which converts SPARQL queries to semantically identical IFC entities, this phase seeks to tackle the heterogeneity disparity across semantic data accessible as Web Ontology Language or Resource Description Framework Graphs and MCPTIS software. There may be significant variations in context (or semantics) between Resource Description Framework or Web Ontology Language data and IFC data models, in addition to the basic to syntax and maybe even structure. Furthermore, in order to utilize themselves within an MCPTIS framework, decision-makers should have domain awareness as well as be capable to understand semantics from the data provider. For example, multiple schemas may include an object called "Normalized Difference Vegetation Index," but the definition of "Normalized Difference Vegetation Index" by the data provider can sometimes vary itself from value within that target schema (for example, IFC schema). Those who seem to have the same definition, however different names like the Height and Elevation, in reference to the terms of identical names and somehow multiple meanings. Semantics employed by data suppliers do not generally complement those mostly used by MCPTIS application in certain situations. To accomplish the same, we describe a set of mappings seen between ontologies of the Web Ontology Language / Resource Description Framework and even the IFC EXPRESS schema to

translate the actual SPARQL query results into a number of outputs that are semantically relevant in order to determine the validity of MCPTIS. Maybe the most understandable format that seems to be compatible mostly with MCPTIS applications is XML regarding SPARQL query results. In order to guarantee consistency of data across database processing and transmit accurate results to MCPTIS customers, this move recognises the importance of integrity constraints in the query translation technique. For example, as somewhat of a functional property axiom, a "PRIMARY key" constraint is described, since a functional property may have distinct value for each instance. A NOT NULL constraint has been used to make absolutely sure that normal attributes cannot have a NULL value in an ifcXML entity. A UNIQUE constraint has been used to make absolutely sure that the values (or individuals) of all properties are distinct.

A SPARQL query includes a number of triples, such that every one of the subject, predicate, and object seems to be a variable, such as Resource Description Framework triples. A standard SPARQL query looks almost like “select DISTINCT ?subject ?property ?value WHERE {?subject rdfs:subClassOf SpatialElement. ?subject ?property ?value.}. This query returns the properties and value elements of every subjects identified in the dataset as subclasses of every SpatialElement class. The WHERE clause of a SPARQL query specifies that data to somehow be retrieved from a dataset, and triples of the URIs in the Resource Description Framework are essential to define that what data to be retrieve. The DISTINCT keyword avoids duplicate responses from being displayed by the SPARQL processor. It is also possible to use other SPARQL query keywords, such as CONSTRUCT, FILTER, and OPTIONAL, to allow the query, the versatility to extract information which might or might not follow each triple sequence. A interpretation of the process created to translate query results through SPARQL into ifcXML is presented in this section.

The ifcXML specification must include header details and based on pre - defined information structure (e.g. serialization component(s), EXPRESS objects and parameters for ifcXML specification, etc.) independent of that same manner wherein query results can be provided. Thereby also, at just the top of the hierarchy of an ifcXML specification that includes other components, we describe a single root element (i.e. <ifc:uos...>). The XML namespace, schema as well as other configuration are specified by this root element. Almost every element must be composed only between opening and closing (<>), indicating the element's start and end point. For eg, IfcTaxableProperty should be written as <IfcTaxableProperty> <Name>...</Name> </IfcTaxableProperty>, if it can be specified by a 'Name' attribute (here as usual attribute). XML elements should have attributes which give the elements extra context. In such an IFC text, each one of the XML attributes may be described as a property. IFC attributes do seem to be, however, interpreted in XML as objects. Throughout the comparison purposes in above example, the <Name> element is then an attribute for this with the IfcTaxableProperty object. Another approach to distinguish an object as well as its attributes in ifcXML is to designate the XML element a specific identifier code.Tbhis is achieved by utilizing an id parameter. In quotations, each attribute does have a label and a value. It is also necessary to characterize the IfcTaxableProperty object as <IfcTaxableProperty id="…"> <Name>...</Name> </IfcTaxableProperty>. An even more variable which is already declared somewhere else may be referenced using such an attribute ref or href. Within that particular instance, the element would then have the attribute xsi:nil="true" to indicate there was no content with in element (e.g.<IfcTaxableProperty xsi:nil="true" ref="..."/>).). From the same document, the certain id value mentioned by an attribute ref (or even href) should appear. Every value about an IFC object must be defined to be the content of the XML variable. Whereas if string value is Ajay Patil for its Name attribute, then this must be defined as <Name>Ajay Patil</Name>. The data conversion phase flowchart for various forms of IFC attributes as seen in Figure 10. The implementation of this method is demonstrated in the following illustration of a use case. eAs many of the additional attributes

were being omitted from those in the example for reasons of convenience and because of efficient use of space.

The description including its interaction seen between data sources and the destinations framework (e.g. MCPTIS tools) schema is a core feature of query processing (called as mapping). So when an analogous IFC entity is defined, a pre - specified container structure containing header information and serialization unit(s) is applied to its necessary attributes and associated entities. So that each set of query results has been translated to an XML equivalents, entitled ifcXML to that same EXPRESS-based description. It designs and records a mapping method between both the outputs of the XML query and the ifcXML data structure. The suggested mapping model defines, in a computer understandable format, the associations within XML and ifcXML data models.

The IFCOwner object includes dual (FirstName and Surname) leaf node properties and that is not declared. Therefore, inside this output, we construct an XML entity like <IfcOwnerid='i123000'>...</IfcOwner>.
Although the entity has not been explicitly defined, a unique identifier (i.e. id= "i123000") is provided. The conversion phase begins from the first attribute, which is a leaf node attribute (i.e. FirstName). The XML variable with in output is then filled in as <IfcOwner id="i123000"> <FirstName> Ajay</FirstName> </IfcOwner>, in which "FirstName" was its name of the attribute, and "Ajay" is the value of the attribute. The IFCOwner object may be defined including all attributes until this procedure is replicated. It is possible to transform and apply the Surname attribute to the IFCOwner object in a similar manner.

The IfcOwnerAndTaxableProperty entity contains basically two forms of attributes (IFCOwner defines TheOwner and IfcTaxableProperty defines TheTaxableProperty) and that is not defined somewhere else. Therefore, in the output, we construct an XML element like IfcOwnerAndTaxableProperty id= "i123100"> ..... Although the entity has not been declared previously, a unique identifier (i.e. id= "i123100") is provided. The conversion phase begins with the first attribute, which is a basic type attribute (i.e. TheOwner). Besides that, with in output, we finalize the XML element as ... where TheOwnern should be the name of the attribute. The IFCOwner object has already been defined, so that all the already declared id (i.e. true" ref="i1000"/>) is returned." It is possible to transform and apply the TheTaxableProperty attribute to that of the IfcOwnerAndTaxableProperty entity in such an equivalent manner. The main distinction seems to be that the IfcTaxableProperty entity and its two leaf node attributes (name and description) in the TheTaxableProperty attribute must be described inside this TheTaxableProperty attribute on its own, since they have not been explicitly defined.

The IfcLatLong entity contains a single complex attribute type, LatLongCoordinates, that is also described by a list of 2 to 3 sub-attributes. We construct an XML variable in the output as <IfcLatLong id="i1200"> ...</IfcLatLong> and give a unique identifier (i.e. id="i1200") because the IFC object is not defined anywhere.

The attribute LatLongCoordinates seems to be a complex form attribute with that of the list's Concept Type (cType), therefore we insert through the output XML element as <IfcLatLong id="i1200"> <LatLongCoordinates exp:cType="list">... </LatLongCoordinates> </IfcLatLong>. <LatLongCoordinates>. It does have two sub-attributes, none of these are IFC entities, and "double" is the primitive type. The XML element with in output was further completed as <IfcLatLong id="i1200"> <LatLongCoordinates exp:cType="list"> <exp:double pos="0">5.17E-16</exp:double> <exp:double pos="1">1. </exp:double> </LatLongCoordinates> </IfcLatLong>, for which primitive type of sub-attribute is double as well as the sub-attribute value has been 5.17E-16 (or 1.). Each sub-attribute can also have an attribute with a position. IfcCartesianPoint is yet another example, which mentioned previously, of an IFC object including a complex type attribute that should be translated and applied to the ifcXML output

similarly. The first and only distinction is that the sub-attributes in the Coordinates attribute are leaf node attributes identified by an entity called IfcLengthMeasure. The resulting ifcXML documents must be checked by the parser validating an XML schema. Consequently, it is possible to correct all the syntactic errors and incomplete elements or incorrect order of elements.

XML Query result and its equivalent ifcXML entity

XML Query Result (Example)

```
<result>
        <binding name = "value">
                <literal>Ajay</literal>
        </binding>
        <binding name = "subject">

<literal>FirstName</literal>
        </binding>
        <binding name = "predicate">

<literal>rdfs:domain</literal>
        </binding>
        <binding name = "object">
                <uri>…#IfcOwner</uri>
        </binding>
</result>
```

Its equivalent ifcXML entity is as given below:

```
<ifcOwner id="i123000">
        <FirstName>Ajay</FirstName>
        <Surname>Patil</Surname>
</ifcOwner>
```

Conclusion

In the first phase of this study, an IFC-compliant ontology describing the hierarchy structure of MCPTIS objects, their relationships and their properties is developed. The emphasis is on semantic indexing and retrieval of taxable property information from an Industry Foundation Classes (IFC) model. Second, ontology mapping is used to find extended ontology that contains all classes and properties from both MCPTIS and GIS domains. Then, taxable property's elements and GIS data are translated into Resource Description Framework / Web Ontology Language format that can be processed by semantic web applications. Once the information has been gathered from different sources and transformed into an appropriate semantic web format, the SPARQL query language is used in the last section to retrieve this information from a dataset.

References

1. Bishr, Y., Goodchild M. F., (1999). "Probing the Concept of Information Communities: A First Step Toward Semantic Interoperability in Interoperating Geographic Information Systems". Proceedings of Interop'97, Kluwer: pp. 55-71.
2. Sheth, A.P., (1999). "Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics, in Interoperating Geographic Information Systems". Kluwer: pp. 530.
3. Harvey F., (1999). "Semantic Interoperability: A Central Issue for Sharing Geographic Information". Annals of Regional Science, 3 (2) (Geospatial data sharing and standardization): pp. 213-232.
4. Bansal, V. K. (2011). "Use of GIS and Topology in the Identification and Resolution of Space Conflicts". Journal of Computing in Civil Engineering, 25(2), 159-171.
5. Anumba, C. J., Issa, R. R., Pan, J., and Mutis, I. (2008). "Ontology-based information and knowledge management in construction." Construction Innovation:Information, Process, Management, 8(3), 218-239.
6. Lima, C., Diraby, T., Fies, B., Zarli, A., and Ferneley, E. (2003). "The E-Cognos

Project: Current Status and Future Directions of an Ontology-Enabled IT Solution Infrastructure Supporting Knowledge Management in Construction." Construction Research Congress, 1-8

7. El-Diraby, T., and Osman, H. (2011). "A domain ontology for construction concepts in urban infrastructure products." Automation in Construction, 20(8), 1120-1132.
8. Kene R. D., Mulkalwar P. N., (2018). "Interoperability between Architectural Engineering Construction (AEC) Domain and Geographical Information Systems (GIS) Cloud: A Scientometric Review". EDU WORLD, A Multidisciplinary International Journal, Vol. IX, Number – 3 (Special Issue), 337-346.
9. Lee, S.-K., Kim, K.-R., and Yu, J.-H. (2013). "BIM and ontology-based approach for building cost estimation." Automation in Construction, 41, 96-105.
10. Peachavanish, R., Karimi, H. A., Akinci, B., and Boukamp, F. (2006). "An ontological engineering approach for integrating CAD and GIS in support of infrastructure management." Advanced Engineering Informatics, 20(1), 71-88.