A Method to generate a Modular ifcOWL Ontology

Walter TERKAJ [a,1], and Pieter PAUWELS [b]
[a] *Institute of Industrial Technologies and Automation (ITIA-CNR), Milan, Italy*
[b] *University of Ghent, Ghent, Belgium*

Abstract. Building Information Modeling (BIM) and Semantic Web technologies are becoming more and more popular in the Architecture Engineering Construction (AEC) and Facilities Management (FM) industry to support information management, information exchange and data interoperability. One of the key integration gateways between BIM and Semantic Web is represented by the ifcOWL ontology, i.e. the Web Ontology Language (OWL) version of the IFC standard, being one of reference technical standard for AEC/FM. Previous studies have shown how a recommended ifcOWL ontology can be automatically generated by converting the IFC standard from the official EXPRESS schema. However, the resulting ifcOWL is a large monolithic ontology that presents serious limitations for real industrial applications in terms of usability and performance (i.e. querying and reasoning). Possible enhancements to reduce the complexity and the data size consist in (1) modularization of ifcOWL making it easier to use subsets of the entire ontology, and (2) rethinking the contents and structure of an ontology for AEC/FM to better fit in the semantic web scope and make its usage more efficient. The second approach can be enabled by the first one, since it would make it easier to replace some of the ifcOWL modules with new optimized ontologies for the AEC-FM industry. This paper focuses on the first approach presenting a method to automatically generate a modular ifcOWL ontology. The method aims at minimizing the dependencies between modules to better exploit the modularization. The results are compared with simpler and more straight-forward solutions.

Keywords. IFC, ifcOWL, Ontology, Modularization, EXPRESS

1. Introduction

BIM (Building Information Modeling) is gaining more and more relevance in the Architecture Engineering Construction (AEC) and Facilities Management (FM) industry to support the digitalization of the business process. Industry Foundation Classes (IFC) [16] is one of the standards in the BIM domain and it is widely used in industrial applications. However, there are barriers limiting its semantic interoperability and adoption on a larger scale [23]. Indeed, the IFC standard is provided as single schema written in EXPRESS language [14] that is extremely large and complex, being characterized by an almost monolithic structure. For instance, the IFC4_ADD1 EXPRESS schema contains 768 En-

[1]Corresponding Author: Institute of Industrial Technologies and Automation (ITIA-CNR), Milan, Italy; E-mail: walter.terkaj@itia.cnr.it

tity data types, 206 Enumeration data types, 60 Select data types, 131 defined data types, 46 FUNCTION declarations, and 2 RULE declarations. The complex structure of IFC jeopardizes its exploitation by industrial domains outside the core AEC applications that may need a simple model of building, spaces, elements and their relations with geometry, topology, monitoring, automation and control, safety, etc.

Semantic Web offers opportunities to provide more effective solutions also for the BIM domain, by exploiting its typical enablers in terms of formal modeling language, data distribution, extensibility, and automatic reasoning. Possible BIM solutions based on Semantic Web technologies include:

1. an OWL version of IFC, named ifcOWL. Previous works [22,21] demonstrated how the ifcOWL can be automatically generated by converting the IFC EXPRESS schema to OWL. For example, this conversion leads to an ifcOWL ontology [21] for IFC4_ADD1 with 1313 classes, 1580 object properties, 13867 logical axioms, and 1158 individuals.
2. the development of novel ontologies for BIM that are based on the semantic web principles and designed exploiting modularity and extendability since the beginning. Such approach is currently investigated by the World Wide Web Consortium (W3C) with the Linked Building Data (LBD) Community Group that is working on a set of loosely related ontologies for Building Topology (BOT) [24], Product, Geometry, Automation and Control [28], etc.

Given the original complexity of the IFC schema, also the resulting ifcOWL ontology is considerably large and complex to load and use. The ifcOWL ontology has many interdependencies that it becomes a huge challenge to exploit data distribution both at Tbox and Abox level. The ontology takes full advantage of OWL2 DL expressivity ($\mathcal{SHIQ}(\mathcal{D})$), which can lead to a high number of assertions when handed to OWL reasoning engines because all axioms are loaded when the ontology is referenced by an RDF graph.

This paper will investigate how the ifcOWL ontology can be split into separate ontology modules, so that end users and applications only need to select the modules that are actually going to be used. The modularization is expected to reduce the complexity and provide enablers also for future extensions and integrations. Section 2 briefly presents related works on ontology modularization, whereas Section 3 addresses the specific problem of modularizing the ifcOWL ontology. Section 4 presents the modularization algorithm and Section 5 shows the results of the application of the algorithm to generate a modular ifcOWL ontology. Finally, conclusions are drawn in Section 6.

2. Related Works

As defined by d'Aquin et al. [10], the task of partitioning an ontology is "the process of splitting up the set of axioms into a set of modules $\{M_1, ..., M_k\}$ such that each M_i is an ontology and the union of all modules is semantically equivalent to the original ontology O". The topic of ontology modularization has been largely addressed in the literature [27]. Indeed, modularity can be beneficial both during the design phase and during the deployment and usage. Some of the benefits of modularity can be mentioned as follows [19]:

- scalability for querying data and reasoning on ontologies
- scalability for evolution and maintenance
- complexity management
- understandability
- context-awareness and personalization
- reuse

Modularity can be applied to pursue different goals [7] while using different strategies for modularity [19], some times also in a concurrent way:

- disjoint or overlapping modules
- semantics-driven strategies
- structure-driven strategies (e.g. using graph decomposition algorithms)
- machine learning strategies
- monitoring modularization and making it evolve

Various techniques have been proposed mainly to process large ontologies and extract modules from them, e.g. [5,9,11,13,15,18]. Modularization has been applied in various knowledge domains, for instance architectural design [6,3] and biomedical domain [20,26,29]. In some cases the ontologies were designed since the beginning in a modular way, for example the TOVE ontologies [12] supporting the enterprise integration. Furthermore, when addressing a modularization problem, the definition of the evaluation criteria [10] plays a key role and it must be consistent with the overall goals.

3. ifcOWL Ontology and Modularity

The modularization of ifcOWL is an important step in updating the ontology so that it can be more efficiently used in a web context. Indeed, a modular ifcOWL is expected to improve:

- usability
- performance (e.g. query and reasoning)
- ease of alignment with other ontologies, also reducing overlapping

At least two strategies can be envisioned to generate a modular ifcOWL:

1. modularization by content, i.e. the definition of classes and properties are separated based on the knowledge domain they are related to, e.g. geometry, units of measurement, building components, HVAC, etc.
2. modularization by axiom type, i.e. separating the different axioms that are included in ifcOWL, such as definition of classes, subsumption, data/object properties, domain/range of properties, equivalent classes, cardinality restrictions apart, etc. This option might even align with the idea of being able to load an ifcOWL in specific OWL profiles (OWL2 EL, OWL2 QL, OWL2 RL - see [17]). For example, an ifcOWL version not containing cardinality restrictions could be used to conform with the OWL2 EL profile. On the other hand, most OWL reasoners allow to specify to which level of expressiveness (RDFS, OWL2 EL, OWL2 QL, OWL2RL) an ontology should be loaded. Thus, when reasoning is concerned, this first option is already supported by using an OWL reasoner with appropriate settings.

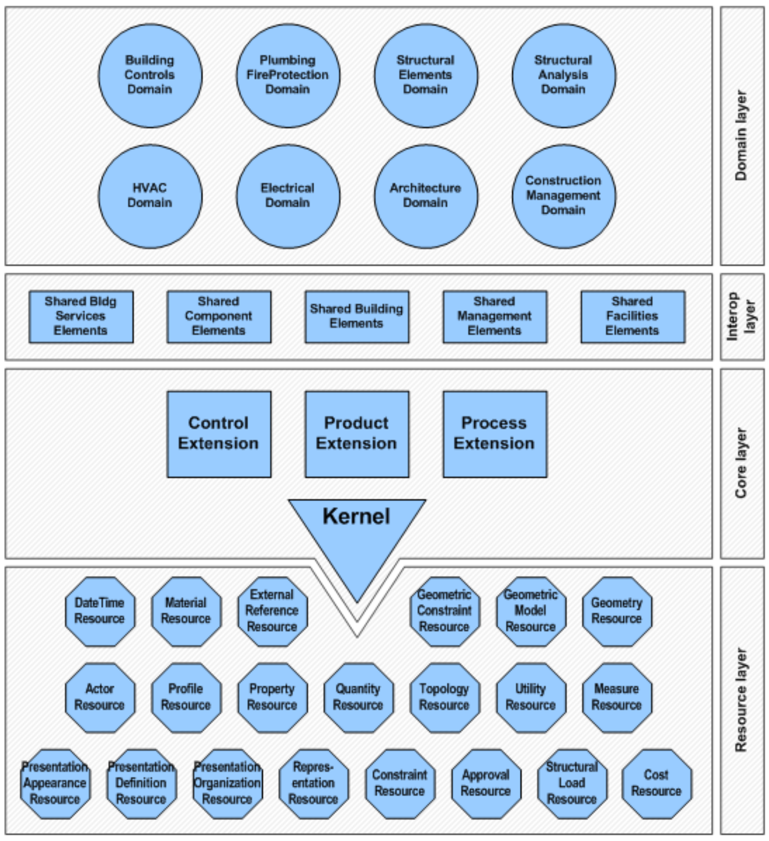


Figure 1. The IFC data schema architecture with conceptual layers, as displayed in the introduction of the IFC specification (IFC4 ADD1) [16]

Herein the attention is focused on the first strategy that is supported by the fact that the IFC standard was developed in a modular way and each data type (i.e. entity, enumeration, select, defined) in the EXPRESS schema belongs to a specific sub-schema, as reported in its documentation [16]. Indeed, the IFC schema consists of four layers, each containing sub-schemas (see Figure 1) that define a part of all EXPRESS data types. For example, the `IfcActorResource` schema (bottom left in Figure 1) contains 3 enumerations and 8 entities. The corresponding OWL definitions could in theory be kept in a separate `IfcActorResource` ontology module, which would be significantly smaller than the complete ifcOWL ontology, thus resulting in better usability. However, many of the data types in the IFC schema are tightly interconnected with each other, not only within a sub-schema but also between different sub-schemas. In addition, in several cases there are reciprocal dependencies between sub-schemas, even belonging to separate layers. For example, `IfcApprovalResource` imports `IfcControlExtension` and vice versa. Hence, in order to make a useful modularization, a full investigation of the schema needs to be made, and the relation between the different sub-schemas (and therefore modules) would need to be reconsidered to a significant level and detail.

Beetz et al. 2009 [4] already proposed a modular ontology for an earlier version of ifcOWL. The authors addressed the problem of interwoven interdependencies by moving some axioms to additional modules, named *pivot ontologies*, that include a set of independent semantic clusters. However, the problem of cyclic references was not completely solved. Furthermore, the author addressed the problem of modularization of the Abox ontologies.

4. Modularization Algorithm

The proposed modularization algorithm can be applied to convert any EXPRESS schema (e.g. IFC [16], but also ISO 15531, ISO 14649, etc.) to a modular OWL ontology. A novel algorithm was developed to exploit the peculiar problem settings, since the modularization takes place while converting an EXPRESS schema (e.g. IFC schema) to an OWL ontology (e.g. ifcOWL), instead of being executed on an already existing large monolithic ontology (e.g. the already generated full ifcOWL). Once the algorithm assigns an EXPRESS definition to a module, then the conversion to the corresponding OWL axiom is executed as stated in [21] and described with more details in [22]. It must be noted that an automatic conversion of a technical standard from an EXPRESS schema to an OWL ontology may lead to problems related to the lack of a precise definition and meaning of some concepts, thus hindering semantic interoperability. Therefore, a proper ontological analysis of the original standard should be carried out, as addressed in the works [2,25,8]. However, such analysis goes beyond the scope of this work.

The goal of the algorithm consists in finding the best way of implementing a given input modularization by minimizing the number of direct import relations between modules. Moreover, the algorithm must avoid to create reciprocal dependencies between modules because it would lead to circular import paths. Even though circular import is not forbidden according to OWL2, still it is not desirable because it would actually weaken the modularization. Indeed, a direct import of any node in a circular path will lead to indirectly importing all the nodes in the circle; thus the final effect is that all the modules in a circular path are merged.

In summary, the algorithm receives as input the following pieces of information:

- content of a parsed EXPRESS schema in terms of data types (i.e. defined data, entity, select, enumeration), subsumption relationships and attributes of each entity data type.
- input modularization in terms of mapping between EXPRESS data types and modules. This mapping can be the results of more or less sophisticated methodologies, or it can be provided in a technical documentation (as in the case of IFC [16]), or it can be simply set by the user based on his/her needs.
- priority level associated with each module. This priority is used to set import relations between modules. *Ceteris paribus*, the module with lower priority will import the module with higher priority. For instance, the priority may be associated with the layer in the whole IFC schema, giving highest priority to the modules in the Resource Layer and the lowest to the modules in Domain Layer.

The modularization algorithm is decomposed into two routines Algorithm 1 and Algorithm 2. Algorithm 1, via the function `GenModularETO`, elaborates the various EX-

PRESS definitions that must be converted to a corresponding OWL axiom. The OWL axiom is serialized as a set of triples that are added to a specific module based on the result of the function `SetModule` in Algorithm 2. Thus, the function `SetModule` incrementally adds import relationships between modules based on the actual needs derived from the inter-module dependencies between EXPRESS data types. After STEP 4 of Algorithm 1 all the OWL axioms required to convert the EXPRESS schema are assigned to a specific module. Moreover, the full set of dependencies (i.e. import relations involving the term `owl:import`) between modules is available and can be represented as a directed graph, where the modules are nodes and the import relations are arcs. With reference to the notation adopted in the algorithm, the graph can be defined as $G = (M, I)$, where M is the set of modules (i.e. nodes) and I is the set of direct import relations (i.e. arcs). If $(w, z) \in I$, then it means that module $w \in M$ directly imports module $z \in M$.

Algorithm 1 Modularization Algorithm

```
Input: set $Ent$ of EXPRESS entities
       set $Enu$ of EXPRESS enumerations
       set $S$ of EXPRESS selects
       set $D$ of EXPRESS defined data types
       set of supertypes $sup(t)$ of EXPRESS data type $t \in (Ent \cup Enu \cup S \cup D)$
       set of items $it(s)$ belonging to the EXPRESS select $s \in S$
       set $attr(e)$ of attributes of entity $e \in Ent$
       data type $ran(e,a) \in (Ent \cup Enu \cup S \cup D)$ being the range of attribute $a \in attr(e)$
       set $M$ of modules
       module $mod(t) \in M$ to which the data type $t \in (Ent \cup Enu \cup S \cup D)$ is assigned
       set $I$ of ordered pairs of modules defining direct import relations
  function GENMODULARETO($Ent, Enu, S, D, sup, it, attr, ran, M, mod$)
    for all $t \in (Ent \cup Enu \cup S \cup D)$ do                                    ▷ STEP 1
      add the OWL axiom defining $c$ to module $mod(t)$
    for all $t \in (Ent \cup Enu \cup S \cup D)$ do                                    ▷ STEP 2
      for all $a \in sup(t)$ do
        add the OWL axiom defining the subsumption realtionship to the module returned
        by SETMODULE($mod(t), mod(a), I$)
    for all $s \in S$ do                                                               ▷ STEP 3
      for all $a \in it(s)$ do
        add the OWL axiom defining the subsumption relation between $s$ and $a$ to the
        module returned by SETMODULE($mod(s), mod(a), I$)
    for all $e \in Ent$ do                                                             ▷ STEP 4
      for all $a \in attr(e)$ do
        add the OWL axiom defining the attribute relation (i.e. property defini-
        tion and restrictions) between $e$ and $ran(e,a)$ to the module returned by
        SETMODULE($mod(e), mod(ran(e,a)), I$)
    Apply the transitive reduction to the graph $G = (M, I)$                           ▷ STEP 5
```

The result of Algorithm 1) and 2 is a directed acyclic graph (DAG), i.e. cycles in the graph are avoided. It can be demonstrated that the resulting graph is a DAG by considering that a topological ordering is possible if and only if the graph has no directed cycles. A topological ordering can be generated from the resulting graph because each pair of nodes (i.e. modules) can be ordered, since Algorithm 2 guarantees that there is only one import direction (direct or indirect) between them. Axioms involving atoms belonging to two different modules are added always to the same module, thus solving the problem of circular imports without needing to merge modules.

The resulting graph can be further optimized by applying a transitive reduction [1] that allows to obtain a graph with fewer arcs but the same reachability (cf. STEP 5 of Al-

Algorithm 2 Set Module Algorithm

```
Input: set $M$ of modules
        priority $p(m)$ of module $m \in M$
        set $I$ of ordered pairs of modules defining direct import relations
Output: selected module
        updated set $I$
  function SETMODULE($x, y, I$)
    if $x = y$ then
      return $x$
    else
      Calculate the transitive closure of graph $G = (M, I)$ to obtain the set of reachability relations $R$
      if $(x, y) \in R$ then
        return $x$
      else if $(y, x) \in R$ then
        return $y$
      else if $p(x) > p(y)$ then
        add $(y, x)$ to the set $I$, return $y$
      else
        add $(x, y)$ to the set $I$, return $x$
```

gorithm 1). In case of a DAG the transitive reduction is unique and consists in a subgraph of the original graph that minimizes the number of arcs, i.e. the number of the imports.

5. Experiments

This section presents the experiments related to the generation of a modular ifcOWL ontology from the IFC4 EXPRESS schema[2]. As reported in Table 1, the input modularization is based on the 38 IFC sub-schemas (Figure 1), plus the ontology modules `express`[3] and `list`[4] that are automatically included during the EXPRESS to OWL conversion. Table 1 reports also the priority level associated with each module, as required to execute the algorithm. Three different versions of modularization algorithm have been tested to demonstrate the benefits of the full version presented in Section 4:

1. *Simple* version, i.e. the modularization algorithm consisting of Algorithm 1 and Algorithm 3 that represents a simplification of Algorithm 2.
2. *Basic* version, the modularization algorithm consisting of Algorithm 1 and Algorithm 2, but without STEP 5 in Algorithm 1.
3. *Full* version, i.e. the modularization algorithm consisting of Algorithm 1 and Algorithm 2.

Algorithm 3, used in the *Simple* version, implements the selection of the module where the OWL axioms are added by looking at the incumbent need, without considering the already set module dependencies. This simplification leads to a higher number of direct import relations (189) compared to the *Basic* version (95). Moreover, the *Simple* version causes the realization of circular import patterns (e.g. modules 10 and 11 import each other), thus disabling the chance to execute a straightforward and deterministic transitive reduction.

[2]`http://www.ontoeng.com/modularIfcOWL/`

[3]`https://w3id.org/express`

[4]`https://w3id.org/list`

Table 1. Modules of the ifcOWL ontology with definition of *id* and *priority* level.

Module	IFC Layer	Label	Priority
list	N/A	1	5
express	N/A	2	5
IFCACTORRESOURCE	Resource	3	4
IFCAPPROVALRESOURCE	Resource	4	4
IFCCONSTRAINTRESOURCE	Resource	5	4
IFCCOSTRESOURCE	Resource	6	4
IFCDATETIMERESOURCE	Resource	7	4
IFCEXTERNALREFERENCERESOURCE	Resource	8	4
IFCGEOMETRICCONSTRAINTRESOURCE	Resource	9	4
IFCGEOMETRICMODELRESOURCE	Resource	10	4
IFCGEOMETRYRESOURCE	Resource	11	4
IFCMATERIALRESOURCE	Resource	12	4
IFCMEASURERESOURCE	Resource	13	4
IFCPRESENTATIONAPPEARANCERESOURCE	Resource	14	4
IFCPRESENTATIONDEFINITIONRESOURCE	Resource	15	4
IFCPRESENTATIONORGANIZATIONRESOURCE	Resource	16	4
IFCPROFILERESOURCE	Resource	17	4
IFCPROPERTYRESOURCE	Resource	18	4
IFCQUANTITYRESOURCE	Resource	19	4
IFCREPRESENTATIONRESOURCE	Resource	20	4
IFCSTRUCTURALLOADRESOURCE	Resource	21	4
IFCTOPOLOGYRESOURCE	Resource	22	4
IFCUTILITYRESOURCE	Resource	23	4
IFCKERNEL	Core	25	3
IFCCONTROLEXTENSION	Core	24	2
IFCPROCESSEXTENSION	Core	26	2
IFCPRODUCTEXTENSION	Core	27	2
IFCSHAREDBLDGELEMENTS	Interoperability	28	1
IFCSHAREDBLDGSERVICEELEMENTS	Interoperability	29	1
IFCSHAREDCOMPONENTELEMENTS	Interoperability	30	1
IFCSHAREDFACILITIESELEMENTS	Interoperability	31	1
IFCSHAREDMGMTELEMENTS	Interoperability	32	1
IFCARCHITECTUREDOMAIN	Domain	33	0
IFCBUILDINGCONTROLSDOMAIN	Domain	34	0
IFCCONSTRUCTIONMGMTDOMAIN	Domain	35	0
IFCELECTRICALDOMAIN	Domain	36	0
IFCHVACDOMAIN	Domain	37	0
IFCPLUMBINGFIREPROTECTIONDOMAIN	Domain	38	0
IFCSTRUCTURALANALYSISDOMAIN	Domain	39	0
IFCSTRUCTURALELEMENTSDOMAIN	Domain	40	0

Algorithm 3 Simple version of Set Module Algorithm

```
Input: set $M$ of modules
       priority $p(m)$ of module $m \in M$
       set $I$ of ordered pairs of modules defining direct import relations
Output: selected module
        updated set $I$
1: function SETMODULE($x, y, I$)
2:     if $(x,y) \notin I$ then
3:         add $(x,y)$ to the set $I$
4:     return $x$
```

The comparison between the *Basic* and *Full* versions show the impact of the transitive reduction, since the total number of import relations becomes 46. A synthetic comparison of the three algorithm versions is reported in Table 2, showing that a great deal of unnecessary imports can be eliminated. The graph-based representation of the three modular ifcOWL solutions are shown in Figures 2 and 3 highlighting how strongly interconnected are the IFC sub-schemas. Analyzing Figure 3b, it can be noted that:

- there are modules in the Core layer (i.e. IfcControlExtension and IfcProcessExtension) that are not actually used in any of the modules in the upper levels;
- just two modules in the Resource layer are not imported (directly or indirectly) by IfcKernel, i.e. IfcStructuralLoadResource and IfcMaterialResource;
- just two modules in the Interoperability layer are imported by modules in the Domain layer, i.e. IfcSharedBldgServiceElements and IfcSharedComponentElements.

Table 2. Synthetic results of the three versions of modularization algorithm for the ifcOWL ontology, considering modules 3-40 defined in Table 1

	Simple	Basic	Full
n. modules	38	38	38
total n. imports	189	96	47
max n. imports per module	12	6	2
avg n. imports per module	4.97	2.5	1.21
circular imports	yes	no	no

The experiments demonstrate how the proposed algorithm enables to optimize the number of import relations. This result is important because it leads to a decomposed ifcOWL ontology with a minimal number of inter-dependencies, thus easing the selection and extraction of a subset of modules that may better fit the requirements of a user.

Finally, even if the same *Full* version of the algorithm is adopted, different solutions can be obtained based on the priority assigned to the modules. Indeed, the final result of the algorithm is deterministic only if all modules have a different priority. On the other hand, if an import relation is required between two modules having the same priority, then the direction of the import depends on the order of the definitions that are elaborated by Algorithm 1. For example, since modules 14 and 15 need each other (see Figure 2) and have the same priority, the final solution could include that module 15 imports module 14, instead of vice versa as in the experiment (see Figures 3a and 3b).

6. Conclusions

This paper presented an approach to generate a modular version of an OWL ontology that is automatically converted from an EXPRESS schema. The attention was focused on the case of the IFC schema given the large size of the resulting ifcOWL ontology. A modular version of ifcOWL can help to solve practical problems related to its usability and the scalability of software applications based on it. Moreover, the modularization algorithm can be used also to extract fragments of the ifcOWL that are relevant for the specific applications. This can be achieved by missing to assign some EXPRESS data types to any module. Further developments will address:

- the generation of additional OWL modules to convert the IFC Property Sets that are currently not included in the IFC EXPRESS schema
- the investigation of other modularization strategies, e.g. the second one presented in Section 3, and the introduction of criteria to at least partially control the definition of dependencies between modules, e.g. by optimizing their priorities
- testing the benefits of working with a subset of ifcOWL modules from a computational perspectives

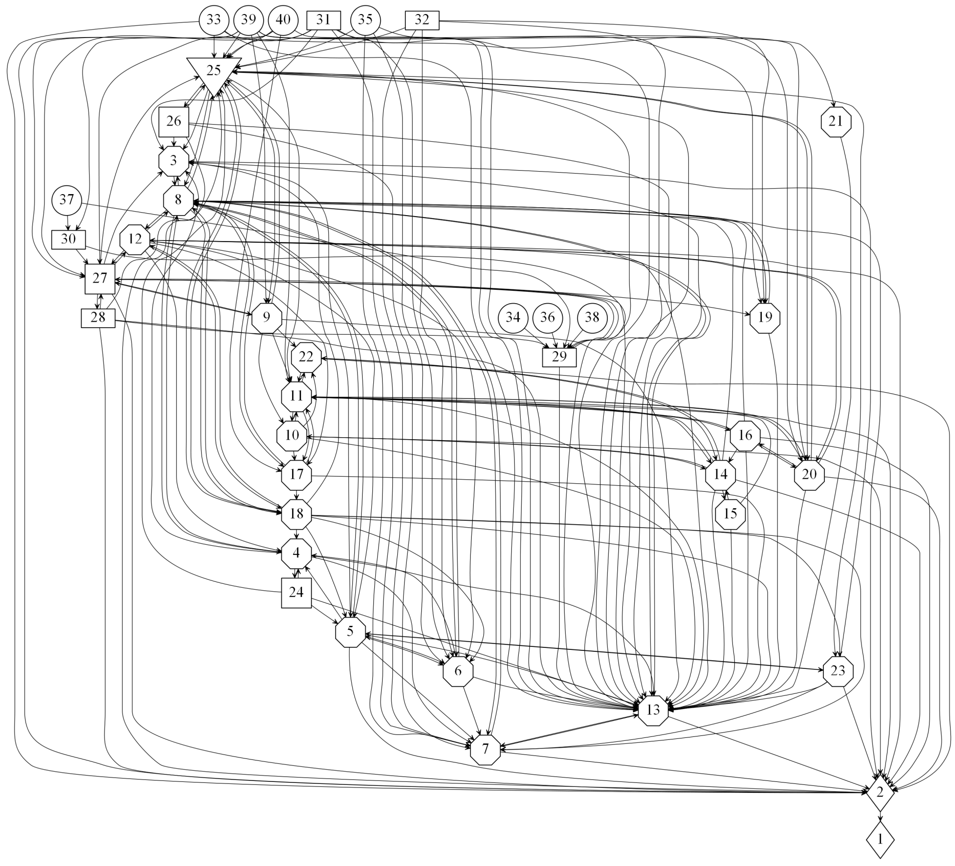


Figure 2. Modular ifcOWL ontology resulting from the *Simple* version of the algorithm. The labels of the nodes are defined in Table 1

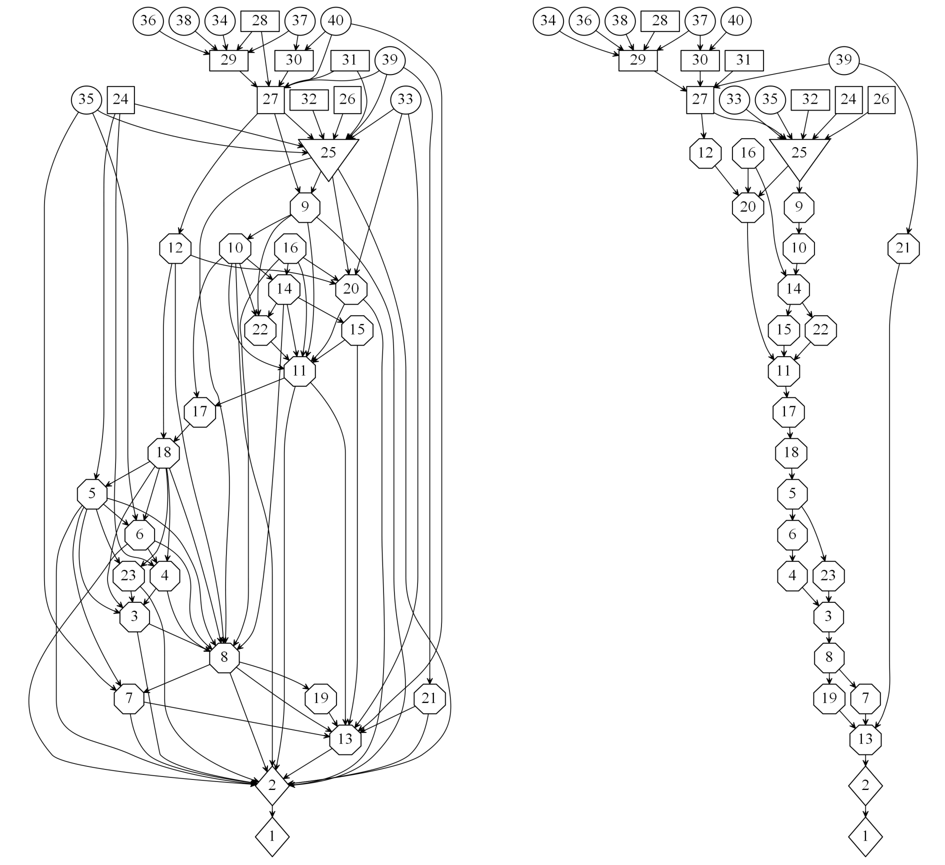
- the integration of a fragment of the ifcOWL ontology with other ontologies
- the comparison of the modular ifcOWL with other ontologies for BIM that are designed to be modular since the beginning [28]
- modularization strategies for Abox ontologies [4].

Acknowledgments

This work has been partially funded by the Italian research project Smart Manufacturing 2020 within the Cluster Tecnologico Nazionale Fabbrica Intelligente.

References

[1] A. V. Aho, M. R. Garey, and J. D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.



(a) *Basic* version (b) *Full* version

Figure 3. Modular ifcOWL ontology resulting from the *Basic* (a) and *Full* (b) versions of the algorithm. The labels of the nodes are defined in Table 1

[2] P. P. F. Barcelos, G. Guizzardi, A. S. Garcia, and M. E. Monteiro. Ontological evaluation of the itu-t recommendation g.805. In *2011 18th International Conference on Telecommunications*, pages 232–237, May 2011.

[3] J. Bateman, S. Borgo, K. Lüttich, C. Masolo, and T. Mossakowski. Ontological modularity and spatial diversity. *Spatial Cognition and Computation*, 7(1):97–128, 2007.

[4] J. Beetz, J. van Leeuwen, and B. de Vries. Ifcowl: A case of transforming express schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23(1):89101, 2009.

[5] C. Bezerra, F. Freitas, J. Euzenat, and A. Zimmermann. ModOnto: A tool for modularizing ontologies. In *Proc. 3rd workshop on ontologies and their applications (Wonto)*, page No pagination., Salvador de Bahia, Brazil, Oct. 2008. bezerra2008a.

[6] M. Bhatt, J. Hois, and O. Kutz. Ontological modelling of form and function for architectural design. *Applied Ontology*, 7(3):233–267, 2012.

[7] S. Borgo. Goals of modularity: A voice from the foundational viewpoint. In *WoMO*, pages 1–6, 2011.

[8] S. Borgo, E. M. Sanfilippo, A. Šojić, and W. Terkaj. *Ontological Analysis and Engineering Standards: An Initial Study of IFC*, pages 17–43. Springer International Publishing, Cham, 2015.

[9] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Extracting modules from ontologies: A logic-based approach. In H. Stuckenschmidt, C. Parent, and S. Spaccapietra, editors, *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, pages 159–186, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[10] M. d'Aquin, A. Schlicht, H. Stuckenschmidt, and M. Sabou. Criteria and evaluation for ontology modularization techniques. In H. Stuckenschmidt, C. Parent, and S. Spaccapietra, editors, *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, pages 67–89, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[11] P. Doran, V. Tamma, and L. Iannone. Ontology module extraction for ontology reuse: An ontology engineering perspective. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 61–70, New York, NY, USA, 2007. ACM.

[12] M. S. Fox and M. Gruninger. Enterprise modeling. *AI magazine*, 19(3):109, 1998.

[13] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: Extracting modules from ontologies. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 717–726, New York, NY, USA, 2007. ACM.

[14] International Organization for Standardization. ISO 10303-11: Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual, 2004. Available online: http://www.iso.org/iso/iso_catalogue/catalogue_tc/-catalogue_detail.htm?csnumber=38047 (Last accessed on 14 September 2017).

[15] R. Kontchakov, L. Pulina, U. Sattler, T. Schneider, P. Selmer, F. Wolter, and M. Zakharyaschev. Minimal module extraction from dl-lite ontologies using qbf solvers. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 836–841, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[16] T. Liebich, Y. Adachi, J. Forester, J. Hyvarinen, S. Richter, T. Chipman, M. Weise, and J. Wix. Industry Foundation Classes IFC4 official release, 2013. Available online: http://www.buildingsmart-tech.org/-ifc/IFC4/final/html/index.htm. Last accessed on 14 January 2015.

[17] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. Owl 2 web ontology language profiles (second edition), W3C Recommendation 11 December 2012. Available online: https://www.w3.org/TR/owl2-profiles/ (Last accessed on 12 July 2017).

[18] S. Oh and H. Y. Yeom. Evaluation criteria ontology modularization tools. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 365–368, Aug 2011.

[19] C. Parent and S. Spaccapietra. An overview of modularity. In H. Stuckenschmidt, C. Parent, and S. Spaccapietra, editors, *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, pages 5–23, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[20] J. Pathak, T. M. Johnson, and C. G. Chute. Survey of modular ontology techniques and their applications in the biomedical domain. *Integr. Comput.-Aided Eng.*, 16(3):225–242, Aug. 2009.

[21] P. Pauwels, T. Krijnen, W. Terkaj, and J. Beetz. Enhancing the ifcowl ontology with an alternative representation for geometric data. *Automation in Construction*, 80:77 – 94, 2017.

[22] P. Pauwels and W. Terkaj. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, 63:100–133, 2016.

[23] P. Pauwels, S. Zhang, and Y.-C. Lee. Semantic web technologies in aec industry: A literature overview. *Automation in Construction*, 73(Supplement C):145 – 165, 2017.

[24] M. H. Rasmussen, P. Pauwels, C. A. Hvid, and J. Karlshøj. Proposing a Central AEC Ontology that allows for Domain Specific Extensions. In *LC3 2017: Volume I Proceedings of the Joint Conference on Computing in Construction (JC3)*, pages 237–244, Heraklion, Greece, July 2017.

[25] F. Ruy, R. Falbo, M. Barcellos, and G. Guizzardi. An ontological analysis of the iso/iec 24744 metamodel. *Frontiers in Artificial Intelligence and Applications*, 267:330–343, 2014. cited By 4.

[26] A. Sojic, W. Terkaj, G. Contini, and M. Sacco. Modularising ontology and designing inference patterns to personalise health condition assessment: the case of obesity. *Journal of Biomedical Semantics*, 7(1):12, May 2016.

[27] H. Stuckenschmidt, C. Parent, and S. Spaccapietra. *Modular ontologies: concepts, theories and techniques for knowledge modularization*, volume 5445. Springer, 2009.

[28] W. Terkaj, G. F. Schneider, and P. Pauwels. Reusing domain ontologies in linked building data: the case of building automation and control. In *Proceedings of the 8th International Workshop on Formal Ontologies Meet Industry*, 2017.

[29] P. Wennerberg, K. Schulz, and P. Buitelaar. Ontology modularization to improve semantic medical image annotation. *Journal of Biomedical Informatics*, 44(1):155 – 162, 2011. Ontologies for Clinical and Translational Research.