Towards an Ontological Grounding of IFC

Stefano Borgo[1]
stefano.borgo@cnr.it

Emilio M. Sanfilippo[1,2]
emilio.sanfilippo@itia.cnr.it

Aleksandra Šojić[2]
aleksandra.sojic@itia.cnr.it

Walter Terkaj[2]
walter.terkaj@itia.cnr.it

[1]Laboratory for Applied Ontology (ISTC-CNR), Trento, Italy;
[2]Institute of Industrial Technologies and Automation (ITIA-CNR), Milan, Italy

Abstract

The Industry Foundation Classes (IFC) is a standard providing an open vendor-independent file format and data model for data interoperability and exchange for Architecture/Engineering/Construction and Facility Management. Some works in the literature addressed the conversion of the standard to the Web Ontology Language, but there is still need of an in depth ontological analysis of its constructs. With this work we start such an analysis focusing on the IFC type/occurrence distinction. The goal is to increase the correct understanding and use of the standard while ensuring logical coherence, ontological soundness and conceptual clarity.

1 Introduction

Information and Communication Technologies (ICT) play a central role in supporting various engineering tasks in the field of manufacturing, building and construction industry. Nevertheless, the use of heterogeneous application tools supporting industrial activities, the lack of a common conceptualisation of the terms used by various actors across different communities, and the lack of formal representations threaten the quality of process and product modelling as well as the effective sharing of data between the stakeholders [25, 30]. In this paper, we focus on the Industry Foundation Classes (IFC) [6], an information modelling standard supported by several Computer Aided Design (CAD) systems. According to the U.S. National Building Information Modeling Standard [17], IFC is the most mature and widespread schema for the building industry.

In order to overcome some drawbacks related to the native language specification of IFC, namely EXPRESS, and benefit from the use of Semantic Web based approaches and technologies, different communities have been working on the conversion of the standard into the Web Ontology Language (OWL) [33]. Nevertheless, the development of IFC-like ontologies has not yet delved into the ontological grounding of the standard, its assumptions and rules that are not always explicitly formalized (e.g. the distinction and relation between type and occurrence entities; the inheritance and overriding of property sets). A simple conversion of IFC into OWL is not enough because ontologies should attempt at making explicit the implicit ontological commitments and conceptualisations of the world laying behind information systems terminologies. When concepts used for knowledge representation and data sharing are not analysed and clearly defined, the different information systems using

In: V. Chaudhri and A. Rademaker (eds.): Proceedings of the 6th Workshop on Formal Ontologies meet Industry (FOMI), Rio de Janeiro, Brasil, 22-September-2014, published at http://ceur-ws.org

them cannot be rigorously (thus reliably) aligned for automated information sharing and exploitation. Indeed, a rigorous ontological perspective, as suggested modern theories of Ontology Engineering [9, 10, 34, 5, 23], is crucial to take full advantage of modern ontological tools.

We focus hereby on the IFC type/occurrence distinction, which plays an important role in the standard, aiming at exploiting the re-use of data and minimising the replication of information. Thus it is important to correctly understand to what types and occurrences refer, especially if one aims at developing IFC-driven ontology-based applications. In the next section we introduce IFC and its general structure; Section 3 presents the state of the art about the conversion of IFC into OWL; Section 4 analyses the IFC type/occurrence distinction, from both a terminological and ontological perspective. We conclude with some remarks about future work.

2 Industry Foundation Classes

The Industry Foundation Classes (IFC) is a standard providing an open vendor-independent file format and data model for data interoperability and exchange for Architecture/Engineering/Construction and Facility Management (AEC/FM). It is released by buildingSMART International and its current release (IFC 4) is registered as ISO 16739. IFC supports interoperability across different applications used to design, construct and operate construction facilities by capturing information about all aspects of a building throughout its lifecycle [14, 30, 2, 21].

The IFC data model is defined in the EXPRESS modelling language, the dedicated formal language developed within the ISO 10303 STEP standard [11]. The current IFC release is built on a modular structure that distinguishes among four conceptual layers, which are tailored in turn in different schemas (aka modules):

- Resource layer: it specifies classes that do not stand in taxonomical relationships with classes defined in the other layers, but they can be rather recalled by means of specific relationships. For example, it includes amongst its schemas the `IfcGeometryResource`, which contains entities needed to define geometric representations (e.g. `IfcCartesianPoint`, `IfcPlacement`, `IfcSurface`). In this way, a product (as a physical object) - defined in the Core Layer (see below) as a `IfcProduct` - can be characterised by a specific placement by the `ObjectPlacement` attribute that points to `IfcPlacement`;

- Core layer: it contains the most general concepts of the IFC data model. Its purpose is to provide the main backbone concepts and relationships of the IFC data model. It thus supports interoperability among the IFC layers and compatibility with the various IFC releases. The Core layer comprises two main schemas:

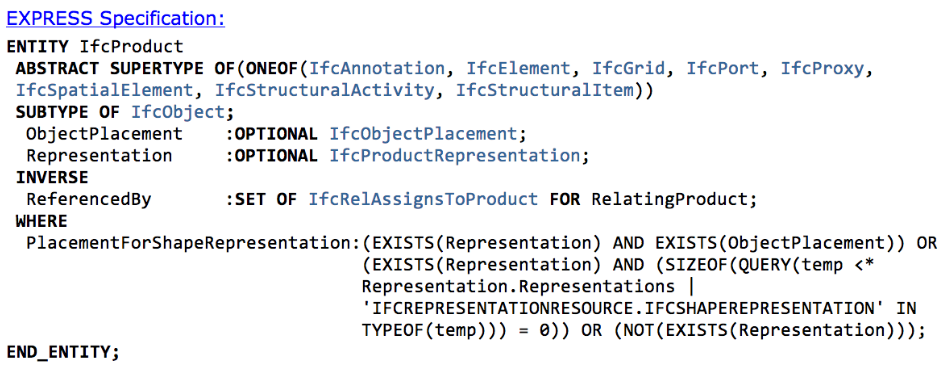
  1. *IfcKernel*, which collects the most general concepts of the standard like `IfcRoot`, `IfcObjectDefinition`, `IfcProcess`, `IfcContext`;
  2. *IfcCoreExtension*, which is further subdivided into three modules: *IfcControlExtension*, *IfcProcessExtension* and *IfcProductExtension*. These specialise the *IfcKernel* with AEC/FM concepts. In particular, the *IfcControlExtension* contains entities for control objects like `IfcPerformanceHistory` and `IfcControl`; the *IfcProcessExtension* specifies entities for the representation of process-like entities, e.g. `IfcEvent`, `IfcProcedure`, `IfcTask`; the *IfcProductExtension* contributes to the specialisation of entities related to product modelling like `IfcElement`, `IfcElementAssembly`, `IfcGrid`;

- Interoperability layer: it contains concepts, defined in the Domain layer (see below), shared by multiple domains and used for inter-domain exchange and sharing of construction information. Amongst its schemas, it includes the *IfcSharedBuildingElements* and the *IfcSharedFacilitiesElements*. The former specialises the *IfcProductExtension* (Core) schema by classes used for representing building structures. Amongst its entities, it includes `IfcChimney`, `IfcDoor`, `IfcRamp`. The latter provides a set of entities concerning facilities management. Some of them specialise the *IfcProductExtension* schema (e.g. `IfcFurniture`, `IfcFurnitureType`), while others are attached directly under the *IfcKernel* (e.g. `IfcInventory`, `IfcOccupant`);

- Domain layer: it contains the most specific concepts of IFC. The Domain layer organises concepts according to industry disciplines and amongst its schemas it includes the *IfcArchitectureDomain* and the *IfcBuildingControlsDomain*. The former defines concepts used in architecture, like `IfcDoorStyle`, `IfcWindowStyle`, `IfcWindowPanelProperties`, among others. The latter supports the modelling of building automation, control, instrumentation and alarm. Amongst its entities, it includes `IfcActuator`, `IfcAlarm` and `IfcSensor`.

The modular architecture operates on a so-called *gravity principle*: at any layer, an entity may refer only to an entity at the same or lower layer [21]. For instance, entities at the Core layer may refer to other Core classes, as well as to Resource layer classes, but cannot refer to entities within the Interoperability or Domain layers. The same principle applies also within the Core layer, in the sense that *IfcKernel* entities cannot refer to *IfcCoreExtensions*, while the reverse is allowed.

The concepts of IFC, modelled in EXPRESS by the **ENTITY** construct, are organised into taxonomies via the supertype/subtype partial ordering relation. For instance, `IfcProcess` is the supertype of `IfcEvent`, `IfcProcedure` and `IfcTask`. Some concepts (e.g. `IfcProcess`) are declared to be *abstract*, in the sense that they can only be instantiated through their subtypes. Inheritance is allowed along the hierarchy, so that subtypes inherit those attributes defined at the level of the supertypes. For example, Fig.1 shows the EXPRESS specification of `IfcProduct`. This is modelled by **ENTITY** and stands for the abstract super-type of different classes, which are mutually disjoint (the construct **ONEOF** specifies the disjointness). The relationship **SUBTYPE OF** states that `IfcProduct` is subsumed under `IfcObject`. *ObjectPlacement*, *Representation* and *ReferencedBy* are attributes, while **WHERE** is a rule specifying a certain condition.


EXPRESS Specification:

```
ENTITY IfcProduct
 ABSTRACT SUPERTYPE OF(ONEOF(IfcAnnotation, IfcElement, IfcGrid, IfcPort, IfcProxy,
 IfcSpatialElement, IfcStructuralActivity, IfcStructuralItem))
 SUBTYPE OF IfcObject;
  ObjectPlacement    :OPTIONAL IfcObjectPlacement;
  Representation     :OPTIONAL IfcProductRepresentation;
 INVERSE
  ReferencedBy       :SET OF IfcRelAssignsToProduct FOR RelatingProduct;
 WHERE
  PlacementForShapeRepresentation:(EXISTS(Representation) AND EXISTS(ObjectPlacement)) OR
                                  (EXISTS(Representation) AND (SIZEOF(QUERY(temp <*
                                  Representation.Representations |
                                  'IFCREPRESENTATIONRESOURCE.IFCSHAPEREPRESENTATION' IN
                                  TYPEOF(temp))) = 0)) OR (NOT(EXISTS(Representation)));
END_ENTITY;
```

Figure 1: `IfcProduct` in EXPRESS, from [6]

3 Owl-izing IFC

IFC supports data exchange among Building Information Modeling (BIM) applications [30]. Nevertheless, different communities have explored its conversion into the Web Ontology Language (OWL) [33] to overcome drawbacks due to some limitations of EXPRESS, and to benefit from the exploitation of Semantic Web technologies for the management of BIM data [18].

Beetz and colleagues [3, 4], as well as Krima et al. [1], draw attention to the EXPRESS lack of formal semantics, so that a logic-based language as OWL is preferable for the definition of axiomatic theories aimed at supporting knowledge representation and data sharing. Additionally, Beetz et al. [4] stress that the popularity of EXPRESS among the engineering community is very limited, apart from the STEP initiative, so that the reuse of existing ontologies or tools for interoperability is often inhibited, especially those related to the Semantic Web initiative. In the paper, the authors explore a semiautomatic method for lifting EXPRESS schemas onto OWL files.

Schevers et al. [24] as well as Zhang et al [35], argue that the conversion of IFC into OWL facilitates the linkage between different IFC models and databases, apart from enabling the exploitation of Semantic Web technologies for building information models. Katranuschkov and colleagues [13] develop an ontology framework aimed at supporting data modelling and data sharing in civil engineering by reusing the IFC data model. However, differently from other approaches, their library of ontologies is developed as an XML Schema. Pauwels and colleagues [18, 19] present a conversion service of the XML-based schema into OWL ontologies in order to represent building information through the enriched RDF graphs, which can be used with reasoning engines [19]. Following the Linked Data approach, the authors stress the advantages of a semantic rule checking environment for the AEC domain [18].

Terkaj et al. [27] propose a modular OWL ontology for virtual factory modelling and data sharing between heterogeneous and geographically distributed software tools. The main structure of the ontology, called Virtual Factory Data Model (VFDM), is based on IFC and the conversion of the standard from EXPRESS to OWL mainly follows the pattern proposed in [3, 4]. The VFDM models the main building blocks of manufacturing systems and their inter-relations, namely, products to be manufactured, manufacturing processes, manufacturing resources and the factory building. The ontology is used as a key enabler inside an integration framework [31] to interoperate different software applications by developing ontology-based plugins for both commercial (e.g. Arena by Rockwell Automation [28], Plant Simulation by Siemens PLM [12]) and non-commercial (e.g. GIOVE Virtual Factory [32], OntoGUI [29]) software tools, aiming at realising an integrated software platform that can support the design and management of a factory along its lifecycle phases [7]. The VFDM shows how an OWL version of IFC can be more easily extended and integrated with other ontologies to represent specific knowledge domains (e.g. factory sustainability [26, 8] and ambient assisted living [22]).

4 Types and occurrences in IFC

As it stands today, IFC is a data model that relies on human interpretation, in the sense that the meanings of its modelling elements are not constrained by means of formal semantics. As discussed in the previous section, semantic considerations lead to look for a formalization of IFC in some formal language, primarily OWL, which is frequently used for computational reasons (which we do not discuss in this paper). However, if the change of language helps to improve some aspects of the standard (e.g. making explicit semantics), it does not *per se* lead to the clarification of the ontological coherence laying behind IFC. For this reason, the use of ontological analysis can help to highlight possible drawbacks in the standard and in the conversion patterns used to build IFC-driven ontological systems.

There are different ways to analyse a standard from the ontological viewpoint. The aim of this section is to verify whether the main distinctions on which IFC relies are well defined and understood. In this initial work, we focus on the distinction between IFC modelling elements named 'type' and 'occurrence' aiming at clarifying what they amount to in ontological terms. The IFC conceptual schema applies this distinction to several modelling constructs, e.g. `IfcObject` and `IfcTypeObject`; `IfcProduct` and `IfcTypeProduct`. Types and occurrences are linked through the (objectified) relationship `IfcRelDefinesByType`, as showed in Fig.2. We want to understand what kind of entities are involved in the distinction, how they are classified, and what kind of relationships hold between them. The investigation aims at reducing the risk of an erroneous implementation of the standard in languages such as OWL by clarifying the notions and how they should be used for modelling purposes.

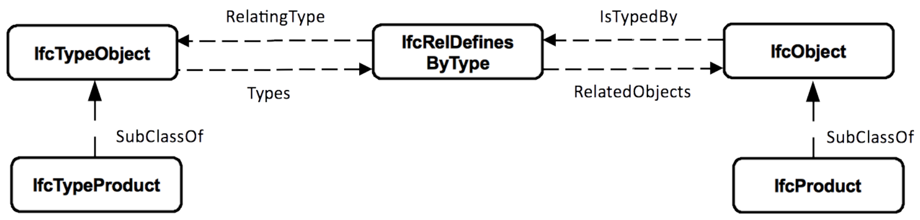


Figure 2: IfcObject and IfcTypeObject with the subclasses IfcTypeProduct and IfcProduct

Let us consider a paradigmatic case: `IfcTypeProduct` vs. `IfcProduct`. As anticipated in Section 2, `IfcProduct` is a subclass of `IfcObject`, while `IfcTypeProduct` is a subclass of `IfcTypeObject` (see Figures 1 and 2). According to the IFC documentation: "The `IfcProduct` is an abstract representation of any object that relates to a geometric or spatial context. Subtypes of `IfcProduct` usually hold a shape representation and an object placement within the project structure. [...] Any instance of `IfcProduct` defines a particular occurrence of a product[...]" ([6], section 5.1.3.10). Therefore, from the definition, `IfcProduct` is a class because it has (a) subtypes (thus structured in a hierarchy) and (b) instances.

The construct `IfcTypeProduct` is defined in the following way: "`IfcTypeProduct` defines a type definition of a product without being already inserted into a project structure (without having a placement), and not being included in the geometric representation context of the project. It is used to define a product specification, that is, the specific product information that is common to all occurrences of that product type" ([6], sect. 5.1.3.49). Although some terms are left unspecified, e.g. project structure, it seems clear that the `IfcTypeProduct` construct represents a class as well. In particular, an instance of `IfcTypeProduct` refers to a set of product specifications, i.e., properties that can be common a set of instances of `IfcProduct`.

The terms *instance* and *class* are commonly used with a variety of meanings in the literature, thus we need to make their interpretation precise. The main distinction between a class and an instance is that the former is a collection of entities (its members), whereas the latter is not[1]: a class is said to *have instances*, whereas an

[1]We do not distinguish classes from sets and call instance any member of a class which is not a class itself.

instance itself cannot instantiate. For example, a particular automobile produced by FIAT, call it `FIAT500#001`, is an instance of the class FIAT_CAR, and the distinction between the class and the instance relies on the way they are related to the expression *being* FIAT_CAR: we say that any instance of the class satisfies the expression and that the class is characterised by such an expression. More specifically, *being* FIAT_CAR is a property used to *talk about* instances: some instances are cars produced by FIAT, while others, e.g. a product manufactured by Microsoft as well as the Everest mountain, are not. Properties help to discriminate entities because they allow to state which entities are distinct and why. To be an instance of the class FIAT_CAR is thus equivalent to satisfy the property *being* FIAT_CAR, which is a shortcut for a conjunction of several more specific properties regarding the engine size, the chassis model, the chassis colour, among others. This complex property is known, in logical and ontological terms, as the *intension* of the class, while the collection of the entities satisfying the property is called the *extension* of the class. The difference between intensionality and extensionality plays a relevant role in ontological engineering, because it allows understanding whether a class is bound or not to the particulars it talks about [9].

Following the distinction between extensionality and intensionality, IFC occurrences can be understood as extensional classes, whose intensional properties can be also defined by the properties associated with the types. The type-ocurrence dichotomy can be further discussed by referring to Fig. 3. Recall that IFC, being formalized in EXPRESS, explicitly models the upper side of the schema only (`IfcTypeObject`, `IfcObject` and `RelDefByType`), while the boxes in the middle and lower side (`TypeTruck`, `IC_FIAT_500`, `id1`, `OccTruck`, `Fiat500#001`, among others) are here included to help in the interpretation.

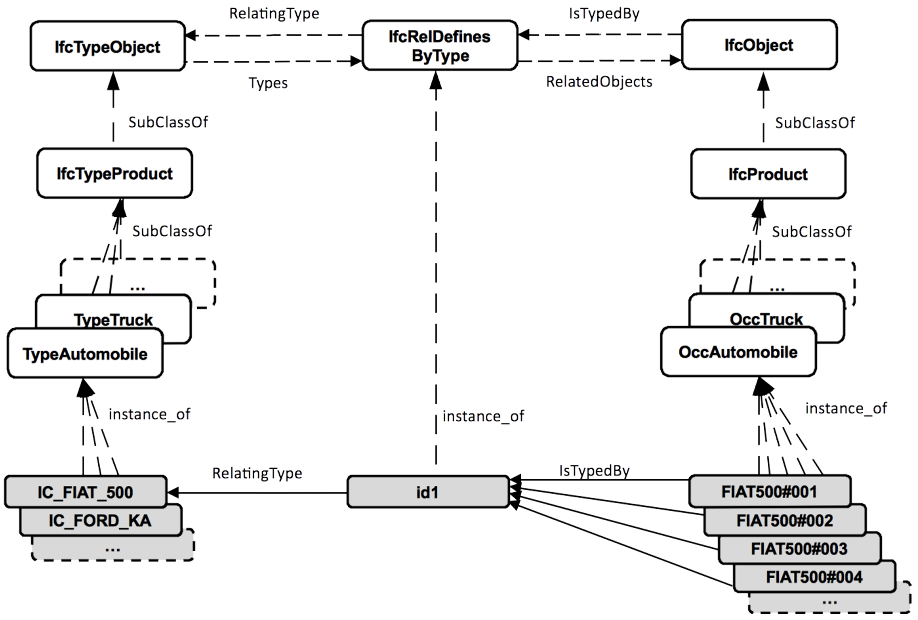


Figure 3: Example of IFC type/occurrence

The relationship of instantiation holding between classes and their corresponding members is labeled `instance_of` in Fig. 3. `RelatingType` and `IsTypedBy` are (non-objectified) relationships holding between the represented modelling constructs at both the class and the instance levels. The bottom-right elements (e.g. `Fiat500#001`) of the figure are instances representing particular objects, e.g. a particular car that someone owns and drives. This object is characterised by a conjunction of properties (like *having chassis model*, *having chassis color*, *having engine size*) that provide the identity criterion for belonging to the class *OccAutomobile*. The object `FIAT500#001` and the class *OccAutomobile* are thus connected by the instantiation relationship. Moreover, the qualifications of these properties (i.e. values and ranges like chassis model $\#F22$, chassis color $\#red3F$ and engine size 1200) are used as the criterion to link an instance (`FIAT500#001`) with its corresponding type (`IC_FIAT_500`), belonging to a type class (*TypeAutomobile*).

The class *OccAutomobile* is associated to a unique subclass of `IfcTypeObject`, namely *TypeAutomobile*. This relationship is at the intensional level, in the sense that it relies on the properties characterizing *OccAutomobile* and does not depend on its instances (whether they exist or else). We have seen some examples of these properties: *having chassis model*, *having chassis colour* and *having engine size*. *TypeAutomobile* thus refers to the collection of some common properties that characterize *OccAutomobile* but, generally speaking, not on their values. We call the relationship between the type and occurrence elements, i.e. the classes on the top of Fig. 3, *typization* since it allows to associate to each homogeneous collection of instances in the right hand-side of the diagram a single general description, i.e. a set of properties in terms of which it is meaningful to consider a comparison of instances[2].

Finally, an instance of *TypeAutomobile*, that is, an element in the bottom-left of Fig. 3, is associated with a collection of these very properties with specified values and ranges: in our example, *having chassis model* $\#F22$, *having chassis colour* $\#red3F$, *having engine size* 1200 and the like. We call this object `IC_FIAT_500`. In ontological terms, `IC_FIAT_500` is called an information object. Also, we could call the relationship between the collection of qualified properties `IC_FIAT_500` and the instance(s) `FIAT500#001` that satisfies them a *realization* since the object `FIAT500#001` manifests all the properties given by the information object `IC_FIAT_500` with the requested qualifications. This relationship differs from that between the class *TypeAutomobile* and the class *OccAutomobile*: the first (realization) is between instances and says that an entity satisfies a set of properties with given values; the latter (typization) is between classes and associates the class corresponding to a set of properties (a type) to the occurrence class whose members must satisfy those properties for some admissible value. Note that a realization does not need to be a physical entity; it may very well be a virtual element.

Currently, it is a matter of ambiguity whether an occurrence instance represents in IFC a physical entity (e.g. the car owned by someone), or a virtual representation in an information system. In our experience we note that IFC practitioners adopt both readings depending on their application tasks. Note however that physical and virtual entities have different ontological properties: on the one side, the former has e.g. a spatiotemporal location according to which it can be classified as a *physical object* in a formal ontology framework like the one proposed by the DOLCE foundational ontology [15]; on the other side, the latter lacks spatial location in the former sense and is classified in DOLCE as an *abstract object* when it also lacks temporal location (in a common sense perspective that is usually associated with physical objects); or as a *concept* when existing in time, for example in the form of a description. Since physical, abstract and conceptual elements are distinguished by distinct properties, an ontology artefact requires to clearly separate them. Note also that the identification of classes at the virtual (abstract) and the physical levels plays a relevant role in data sharing, communication and verification since it determines which data are affected by time and in which way. Consider a scenario in which a practitioner develops an ontology-based instance model (A-Box) with an occurrence instance o_1, which is a virtual entity. When a user accesses the ontology, s/he would not be able to properly interpret o_1 as a virtual entity unless this is somehow modelled in the system by specific formal constraints. An explicit specification of the ontological kind of entity that is represented (virtual or physical) could enhance the reliability of the information sources that use the standard.

5 Conclusion and further discussion

Current attempts towards the OWL formalization of IFC have not been systematically supported by the ontological analysis of its terminology and modelling constructs. While bringing the benefits of formal semantics into IFC, OWL-ized IFC ontologies do not clarify the meanings of its modelling elements. Indeed, choosing how to interpret IFC notions relies on the users' knowledge of the standard and their application needs, issues that threaten automated interoperability between IFC models developed by different communities. Our approach differs from the state of the art (Section 3), because it proposes a reading of the standard aimed at increasing its ontological soundness and conceptual clarity. The performed ontological analysis explains the type/occurrence modelling pattern in terms of the difference between descriptions (i.e., information entities) within the type hierarchy, and their realizations within the occurrence hierarchy, where realizations might be both physical and virtual elements in the system.

The distinction between IFC types and occurrence classes should not be confused with the distinction between meta-classes and classes [16]. Briefly said, meta-classes and classes differ on the abstraction level: the meta-class has the class as an instance. Yet, if a class instantiates a meta-class, then the meta-class should be associated with a collection of the properties that are *necessary* for a class to be classified as an instance. According to our analysis, *TypeAutomobile* is related to *OccAutomobile* via properties like *having chassis model*, *having chassis colour* and *having engine size*. However, these properties do not necessarily have to be linked to the properties of the class *OccAutomobile* for two reasons. First, IFC allows that some properties associated with types can be overridden [6], thus they are not treated as *necessary* properties. Second, analogous to the example

[2]Ontologically speaking, we could say that IFC is a contextual modelling framework since the notion of 'homogeneous instances' depends on the user or the application task. This observation is crucial for a suitable choice of the classes to consider in the diagram.

of `IfcBuilding` [6], which does not need to have specified `IfcBuildingType`, it is possible to have an instance of class *OccAutomobile* and directly characterize it with as many properties as possible (i.e. chassis et al.) without linking it to any instance of class *TypeAutomobile*. This example, together with the possibility to override some properties associated with the types, demonstrate that the standard currently does not treat the IFC types as meta-classes of the occurrence classes. Accordingly, the `Instance_of` relationship does not apply to types and classes of occurrences in IFC.

The difference between IFC type and occurrence classes may remind the materialization modelling pattern [20]. Materialization is a binary relationship holding between abstract and concrete entities, e.g. between the abstract `CarModel` and the concrete `Car`, where the former is a class of properties, and the latter is a class of particular automobiles defined by the class of properties. Concrete classes are said to materialise abstract ones; e.g. one could say that John's and Mary's Fiat500s materialize the same Fiat500 model, although they are two different particular cars. However, the abstract-concrete pairs, and thus the materialization relationships, are ambiguously defined: firstly, materialization holds that one and the same thing can be both concrete and abstract [20] while ontological theories force to distinguish them as different kinds [15]. Secondly, the semantics of the materialization relationship is defined through the instantiation and generalization/specialization relationships (together with a metaclass/class correspondence), but generalization and abstraction are quite different operators, similarly for specialization and concretization. It remains unclear how abstract and concrete entities are related by these operators. The study of materialization might enlighten the IFC type/occurrence dichotomy; nevertheless, this can be assessed only once the semantics of materialization has become conceptually transparent and (possibly) formally represented.

Acknowledgments

This research has been partially funded by MIUR under the Italian flagship project "Fabbrica del Futuro", Subproject 2, research project "Product and Process Co-Evolution Management via Modular Pallet configuration" (PRO2EVO).

References

[1] R. Barbau, S. Krima, S. Rachuri, A. Narayanan, X. Fiorentini, S. Foufou, and R.D. Sriram, "OntoSTEP: Enriching product model data using ontologies," in Computer-Aided Design, 44(6):575–590, 2012.

[2] V. Bazjanac and D.B. Crawley, "The implementation of Industry Foundation Classes in simulation tools for the building industry," tech. rep., Lawrence Berkeley National Laboratory, 1997.

[3] J. Beetz, J. van Leeuwen, and B. de Vries, "An Ontology Web Language notation of the Industry Foundation Classes," in *22nd CIB W78 Conference on Information Technology in Construction*, 2005.

[4] J. Beetz, J. Leeuwen, and B. Vries, "IfcOwl: A case of transforming express schemas into ontologies," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 23, pp. 89–101, 2009.

[5] S. Borgo, M. Carrara, P. Garbacz, and P. E. Vermaas, "A formalization of functions as operations on flows," *Journal of Computing and Information Science in Engineering*, 11(3):031007 1–14, 2011.

[6] buildingSMART International, "Industry Foundation Classes. IFC release candidate 4," 1999-2012.

[7] M. Colledani, G. Pedrelli, W. Terkaj, and M. Urgo, "Integrated virtual platform for manufacturing systems design," Procedia CIRP, vol. 7, pp. 425–430, 2013.

[8] S. Gagliardo, F. Giannini, M. Monti, G. Pedrielli, W. Terkaj, M. Sacco, M. Ghellere, F. Salamone, "An Ontology-based Framework for Sustainable Factories," Computer-Aided Design and Applications, vol. 12(2):198–207, 2015.

[9] N. Guarino and C. Welty, "An overview of OntoClean," in *Handbook on Ontologies* (S.Staab and R.Studer, eds.), pp. 201–220, Springer-Verlag Berlin Heidelberg, 2009.

[10] M. Grueninger, "Using the PSL ontology," in *Handbook on Ontologies* (S.Staab and R.Studer, eds.), pp. 423–443, Springer-Verlag Berlin Heidelberg, 2009.

[11] ISO, *Industrial Automation Systems and Integration - Product Data Representation and Exchange. Part 11: Description methods: The EXPRESS language reference manual*, iso 10303-11:2004(e) ed., 2004.

[12] B. Kádár, W. Terkaj, and M. Sacco, "Semantic Virtual Factory supporting interoperable modelling and evaluation of production systems," *CIRP Annals - Manufacturing Technology*, 62(1):443–446, 2013.

[13] P. Katranuschkov, A. Gehre, and R. Scherer, "An ontology framework to access IFC model data," *ITcon*, vol. 8, no. Special Issue, pp. 413–437, 2003.

[14] L. Khemlani, "The IFC building model: A look under the hood," 2004.

[15] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, "WonderWeb Deliverable D18, Ontology Library (final)". LOA-ISTC, CNR, Tech. Rep. 2003

[16] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis, "TELOS: Representing knowledge about information systems", in ACM Transactions on Information Systems (TOIS), 8(4), pp. 325-362.

[17] National Institute of Building Sciences, "National building information modeling standard, version 1 – part 1: Overview, principles and methodologies," tech. rep., 2007.

[18] P. Pauwels, D. Van Deursen, R. Verstraeten, J. De Roo, R. De Meyer, R. Van de Walle, and J. Van Campenhout, "A semantic rule checking environment for building performance checking" in *Automation in Construction*, vol. 20:5, pp. 506–518, 2011.

[19] P. Pauwels and D. Van Deursen ,"IFC/RDF: Adaptation, Aggregation and Enrichment", in *First International Workshop on Linked Data in Architecture and Construction (LDAC 2012)*, 2012

[20] A.Pirotte, E.Zimányi, D.Massart, T.Yakusheva, Materialization: A Powerful and Ubiquitous Abstraction Pattern, in Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), 1994

[21] L.C. Pouchard and A.F. Cutting-Decelle, "Ontologies and standard-based approaches to interoperability for concurrent engineering" in *Concurrent Engineering in Construction Projects*, Taylor & Francis 2007

[22] M. Sacco, E. Caldarola, G. Modoni, W. Terkaj, "Supporting the Design of AAL through a SW Integration Framework: The D4All Project," in *Universal Access in Human-Computer Interaction. Design and Development Methods for Universal Access, Lecture Notes in Computer Science*, vol. 8513, pp. 75-84, Springer International Publishing, 2014.

[23] E.M. Sanfilippo, S. Borgo, and C. Masolo. States, events, activities: Is there an ontology behind BPMN? In *International Conference on Formal Ontology in Information Systems (FOIS 2014)*, FAIA, IOS Press, 2014.

[24] H. Schevers and R. Drogenmuller, "Converting the industry foundation classes to the web ontology language," in *First International Conference on Semantics, Knowledge and Grid (SKG)*, 2006.

[25] S. Szykman, R.D. Sriram, and W.C. Regli, "The role of knowledge in next-generation product development systems," in *Journal of Computing and Information Science in Engineering*, vol. 1, pp. 3–11, 2001.

[26] W. Terkaj, L. Danza, A. Devitofrancesco, S. Gagliardo, M. Ghellere, F. Giannini, M. Monti, G. Pedrielli, M. Sacco, and F. Salamone, "A Semantic Framework for Sustainable Factories," in *Procedia CIRP*, vol. 17, pp. 547–552, 2014.

[27] W. Terkaj, G. Pedrelli, and M. Sacco, "Virtual Factory Data Model," in *Workshop on Ontology and Semantic Web for Manufacturing OSEMA 2012, CEUR Workshop Proceedings*, vol. 886, pp. 29–43, 2012.

[28] W. Terkaj and M. Urgo, "Virtual Dactory Data Model to support performance evaluation of production systems," in *Workshop on Ontology and Semantic Web for Manufacturing OSEMA 2012, CEUR Workshop Proceedings*, vol. 886, pp. 44–58, 2012.

[29] W. Terkaj and M. Urgo, "Ontology-based modeling of production systems for design and performance evaluation," in *Proceedings of 12th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 748–753, 2014.

[30] V. Thein, "Industry Foundation Classes (IFC).BIM interoperability through a vendor-independent file format. a bentley white paper," tech. rep., Bentley. Sustaining Infrastructure, 2011.

[31] T. Tolio, M. Sacco, W. Terkaj, and M. Urgo, " Virtual Factory: an Integrated Framework for Manufacturing Systems Design and Analysis," in *Procedia CIRP*, vol. 7, pp. 425–430, 2013.

[32] G.P. Viganò, L. Greci, S. Mottura, and M. Sacco, "Giove Virtual Factory: A new viewer for a more immersive role of the user during factory design," in *Digital Factory for Human-oriented Production Systems. The Integration of International Research Projects*, pp. 201–216, Springer London, 2011.

[33] W3C OWL Working Group, "OWL 2, Web Ontology Language document overview (second edition)."

[34] M. West, *Developing High Quality Data Models.* Morgan Kaufmann, 2011.

[35] L. Zhang and R. R. Issa, "Development of IFC-based construction industry ontology for information retrieval from ifc models" in *EG-ICE Workshop, University of Twente, The Netherlands, July. 2011.*