The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLIV-4/W1-2020, 2020
3rd BIM/GIS Integration Workshop and 15th 3D GeoInfo Conference, 7–11 September 2020, London, UK

TOWARDS A GENERIC MAPPING FOR IFC-CITYGML DATA INTEGRATION

Helga Tauscher [1,*]

[1] HTW Dresden, University of Applied Sciences, Dresden, Germany - helga.tauscher@htw-dresden.de

KEY WORDS: BIM-GIS, IFC, CityGML, Graph transformation, Synchronization, Link generation

ABSTRACT:

Much work has been carried out on the topic of BIM-GIS integration. As a technical challenge in particular, research and development tackle the standard data formats of the two areas and aim for the conversion between, linking of or overarching querying over data sources of these formats. Usually, these operational cases (conversion, linking, querying) are examined in isolation or even treated as mutually exclusive and competing approaches. With Triple Graph Grammars, we propose to apply a method that allows to derive solutions for these operational cases from a common generic ruleset. We demonstrate this approach in a proof-of-concept implementation using eMoflon. Our work focusses on IFC and CityGML and we present and discuss a first end-to-end demonstration of integrating these standards with the proposed method. Going forward such representation of the correlation between IFC and CityGML, declarative, independent of particular operational implementations, can serve as a vehicle to capture and document acknowledged integration schemes for IFC and CityGML, complementing these two standards with a specification of their correlation.

1. INTRODUCTION

The integration of BIM and GIS is a precondition to holistic analysis and planning of the built environment on the building and urban scale. In particular, we find growing interest in establishing a connection between the two prevalent standard formats IFC in the construction domain and CityGML on the urban scale, with an abundant body of research and development. Several researchers have studied unidirectional transformation from IFC to CityGML (e.g. Donkers et al., 2016). Other efforts have tackled transformation in the opposite direction (e.g. Salheb, 2019) or integrated queries over both IFC and CityGML instances for holistic analyses (e.g. Daum et al., 2017). Other use case scenarios require the generation of explicit links between two respective instances or incremental updates. In this paper we report on work in progress to implement IFC-CityGML integration in a generalized manner.

The different operational scenarios — forward, backward transformation, synchronization, consistency checking, consistent model and correspondence generation — are effectively based on the same correlations between the IFC and the CityGML model. Thus if we had a universal representation of this correlation, the other operational transformation systems could be deduced from there. Triple Graph Grammars (TGGs) are built on that idea and represent the correlation between the two related domain-specific metamodels (schemas) in the form of a graph grammar (Schürr, 1995). Such a grammar consists of a set of graph transformation rules, that collectively describe all possible triples of two model instance graphs and one correlation graph. In other words: a TGG is a graph transformation system for the generation of consistently correlated pairs of model instances.

* Corresponding author

Building on a previous implementation for IFC-to-CityGML (forward) transformation (Stouffs et al., 2018; Tauscher, 2019; Lim et al., 2019; Tauscher and Stouffs, 2019) based on graph-transformation, we are now reimplementing the existing ruleset as a proper TGG ruleset. For authoring the ruleset as well as derivation and execution of operational transformations we are now using eMoflon (e.g. Leblebici et al., 2014). In Section 2 we describe the workflows to create the rulesets and show how eMoflon supports and simplifies rule development compared to our old system with a simple example ruleset.

EMoflon is based on the Eclipse Modelling Framework (EMF) and uses ECore representations of the two type graphs to be integrated. We have set up workflows to generate ECore files for both IFC and CityGML from the respective published standards documents. Our current ruleset uses IFC4 Add2 TC1 (ISO 16739, 2013) and the current CityGML 3 draft (Kutzner et al., 2020). We discuss how to derive the eCore models and parse/serialize instance models in Section 3.

In Section 4 we demonstrate the operational cases implemented and tested using a simple example data set.

2. FROM FORWARD TRANSFORMATION TO A GENERIC TGG

For the conversion from IFC to CityGML in the IFC2CityGML project, we had used unidirectional graph transformation due to the scope of the project and considerations of efficient implementation. We had already discussed the potential of triple graph grammars and shown how forward transformation rules can be derived from generic ones, that generate triples (Stouffs et al., 2018). The current reimplementation does specify the generic rules in the first place. Forward rules are then only one of multiple operational transformation systems

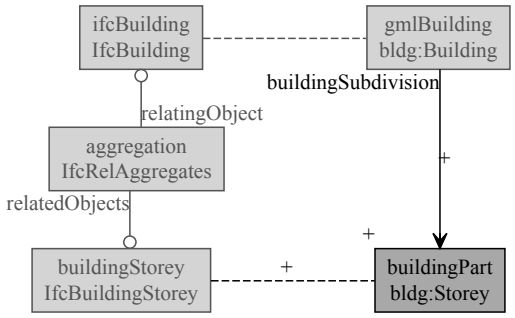

Figure 1. Forward transformation rule in old system

to be derived. As an example of the reimplementation, Figure 1 shows a forward transformation rule from our old rule authoring and management system, Figure 6 shows the corresponding generic TGG rule as reimplemented.

The visual representation of the rules is similar in both systems and like or old system and eMoflon also uses a domain specific language (DSL) to specify the rules. The diagrams are then generated from the (textual) DSL with PlantUML[1] which is based on GraphViz, same as our earlier rule management platform (Tauscher, 2019). The visualization support in eMoflon goes beyond rule diagrams and does also help with visualization of model instances or selected instance elements (see Figures 8 and 9 for examples). As a downside, we would like to see an improved layout algorithm in eMoflon.

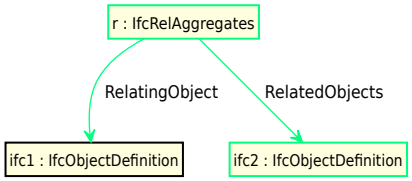

Figure 2. Abstract rule for aggregation

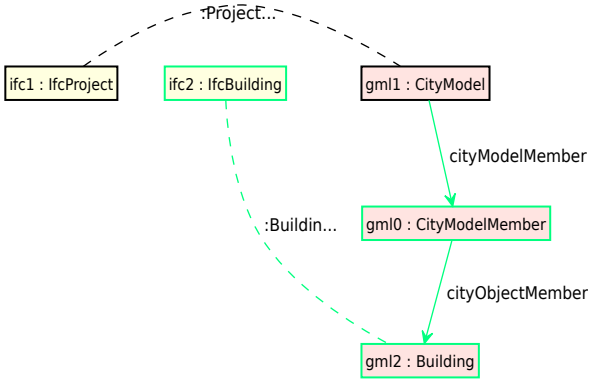

Figure 3. Abstract rule for buildings

Also similar to our old application, eMoflon checks the syntax of the rule DSL as well as the metamodels involved. Here, the functionality in eMoflon supersedes ours, because it allows metamodel checking for arbitrary models, while we only implemented checking on the IFC side. Beyond checking, eMoflon simplifies rule authoring with autocompletion both for the rule DSL and

[1] PlantUML: http://plantuml.com

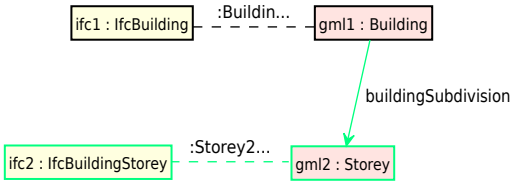

Figure 4. Abstract rule for storeys

metamodels elements (e.g. type and attribute names). The eMoflon implementation is based on Eclipse editors.

In our old rule system implementation, we ended up with a large number of similar rules, which do only differ in details, e.g. particular type or attribute names and are identical otherwise. While we had implemented ways to reuse rules in multiple rulesets, we could not reuse smaller parts of rules. This is now possible with eMoflon through a mechanism called “rule refinement”. Rules can be marked as *abstract* and hence not to be applied in a transformation directly, but only be used as blueprints for derived (refined) rules. Figures 2, 3 and 4 show such abstract rules. The rule in Figure 2 defines the IFC part of an aggregation relation which is used identically for the aggregation of buildings in projects (Figure 5) or storeys in buildings (Figure 6), among others. Instead of fully typing out these two rules, the rule in Figure 5 is created as a refinement of the rules in Figures 2 and 3, whereas the rule in Figure 6 is created as a refinement of the rules in Figures 2 and 4. This allows for more fine-grained modularity of rules.

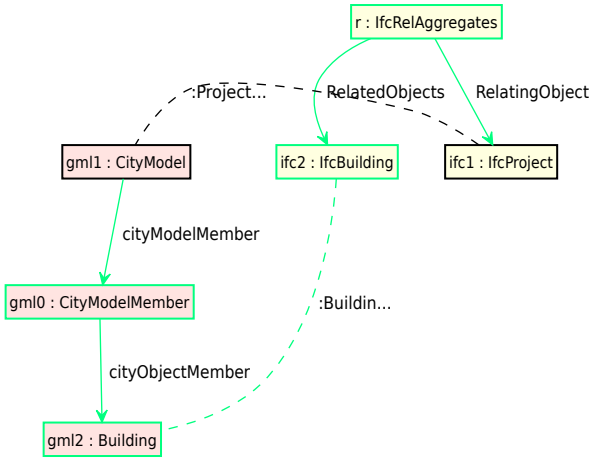

Figure 5. Concrete rule for buildings (extends aggregation, building rule, see Figures 2 and 3)

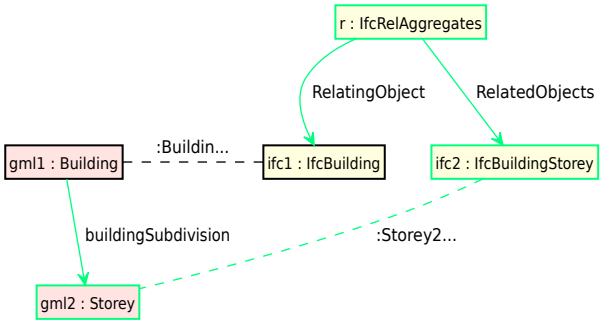

Figure 6. Concrete rule for storeys (extends aggregation, storey rule, see Figures 2 and 4)

Apart from these functionalities that support editing and maintenance of rulesets better than our old implementation, eMoflon has also some further advantages with regard to rule application, formal rigidity, and completeness of implementation. EMoflon allows to substitute the pattern matching engine and is currently distributed with two different engines, Democles and HiPE. The latter is an incremental pattern matcher with can employ multiple cores with parallelized rule application. Ecore metamodels are not only to be defined on the source and target side, but also for the correspondence graph. Further, attribute conditions can be employed to convert values (e.g. coordinate transformation) or establish criteria for the equality of measures to be used when searching for matches — something that our previous conversion engine did only implement partially. In our first implementation, the conversion engine could only handle graph patterns of a specific structure and with a fixed direction of edges (from entry to exit pair). For this reason, inverse IFC attributes where essential and some rules could not be interpreted by the conversion engine due to the lack of inverses. EMoflon's pattern matching does not suffer from this restriction.

The most relevant advantages of eMoflon, however, are the capabilities to (a) employ the Eclipse Modelling Framework for arbitrary domain models, schemas, serializations as described in Section 3 and (b) derive various operational transformations as described in Section 4.

3. WORK AND DATA FLOW TO DERIVE ECORE AND INSTANCE MODELS

Figure 7 shows the larger context of the IFC2CityGML transformation. The parts that are not directly in the focus of this work are greyed out, while the core parts are highlighted in deep black. The core transformation part, the square labeled "IFC2CityGML", consists of four models: the two metamodels on the left (IFC4.ECore.emf and CityGML.ECore) represent the IFC4 and CityGML schemas as EMF ECore. The TGG rules refer to these models for the types of their elements. Similarly, the two instance models on the right (BldgModel.ifc.emf and BldgModel.citygml.emf) refer to these models as their metamodels. Transformation, synchronization, and generation is carried out over these instance models.

The surrounding, greyed out part of the diagram in Figure 7 represents where the various input for the transformation is originating from. This applies to both the metamodels and the instance models as explicated in the following sections.

3.1 Meta models

eMoflon is based on the Eclipse Modelling Framework (EMF, Steinberg, 2009) and needs metamodels specified as ECore. ECore is an implementation of EMOF (Essential MOF) which in turn is a reduced variant of MOF (Meta-Object Facility), a metamodelling standard architecture defined by the Object Management Group (OMG) in order to describe the Unified Modelling Language (UML, ISO 19505, 2012; Fowler, 2004). The EMOF is a subset of the Complete Meta-Object Facility (CMOF). As such the EMF with ECore constitutes a suitable framework for the implementation of metamodelling solutions aiming to be interoperable with UML.

There are various ways to define metamodels and schemas which can serve as the origin for the ECore models, besides direct specification of the ECore in an Eclipse editor using some graphical user interface. These options include tools supporting model driven engineering (MDE), for example Sparx System's software Enterprise Architect (EA), or their predecessors, CASE tools, as well as visual and textual modelling and schema languages such as UML or EXPRESS, the XML schema definition (XSD), JSON schema, and in a wider sense the Web Ontology Language (OWL). There is also a lightweight domain-specific language called Emfatic[2] to describe ECore models. In the following, we focus on those two options that are used to specify IFC4 and CityGML 3 and to publish the respective standards documents.

Up to the current version 4, the IFC schemas have traditionally been defined in EXPRESS (ISO 10303-11, 2004) with the *.exp files being published as machine-readable and formal specification of the schema alongside informal documentation and description of the information model[3]. To derive ECore models from EXPRESS, Jakob Beetz has developed an early conversion library called buildingSMART library[4] which is used in BIMServer (Beetz et al., 2010). For this project, we use a new development, which is work in progress and based on the same graph transformation approach as the conversion from IFC to CityGML.

To understand the similarity, look again at Figure 7. In the upper left we can identify a square structure labeled "EXP2EMF" (mainly greyed out), which consists of four models, with two metamodels (EXPRESS.ECore and ECORE.ECore) on the left and two instance models (IFC4.exp.emf and IFC4.ECore.emf) on the right. This resembles the same structure as the square "IFC2CityGML" in the bottom right (deep black). In fact, we are using the same principles, methods, and tools for the integration of the IFC schema in EXPRESS and ECore as we do for the integration of the building data in IFC and CityGML. This is possible, because ECore models themselves are defined with ECore. The details and application cases for this integration go well beyond the scope of this paper and implementation is in an early stage. Here, we use Xtext (Bettini, 2016) to generate the ECore model for EXPRESS and a parser for *.exp schema definition files. Parsing of IFC4.exp will result in an instance of the EXPRESS ECore model (IFC4.exp.emf). From the TGG, we currently only use the forward transformation from EXPRESS to ECore. The instance model (IFC4.ECore.emf) generated from the EXP2EMF transformation is henceforth used as a metamodel in the IFC2CityGML transformation.

The new version of CityGML version 3, upcoming for adoption, is for the first time modelled with Enterprise

[2] Emfatic. A textual syntax for EMF ECore (meta-)models: https://www.eclipse.org/emfatic/

[3] buildingSMART IFC Specifications Database with official IFC releases: https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/

[4] OpenSourceBIM buildingSMART library: https://github.com/opensourceBIM/BuildingSMARTLibrary

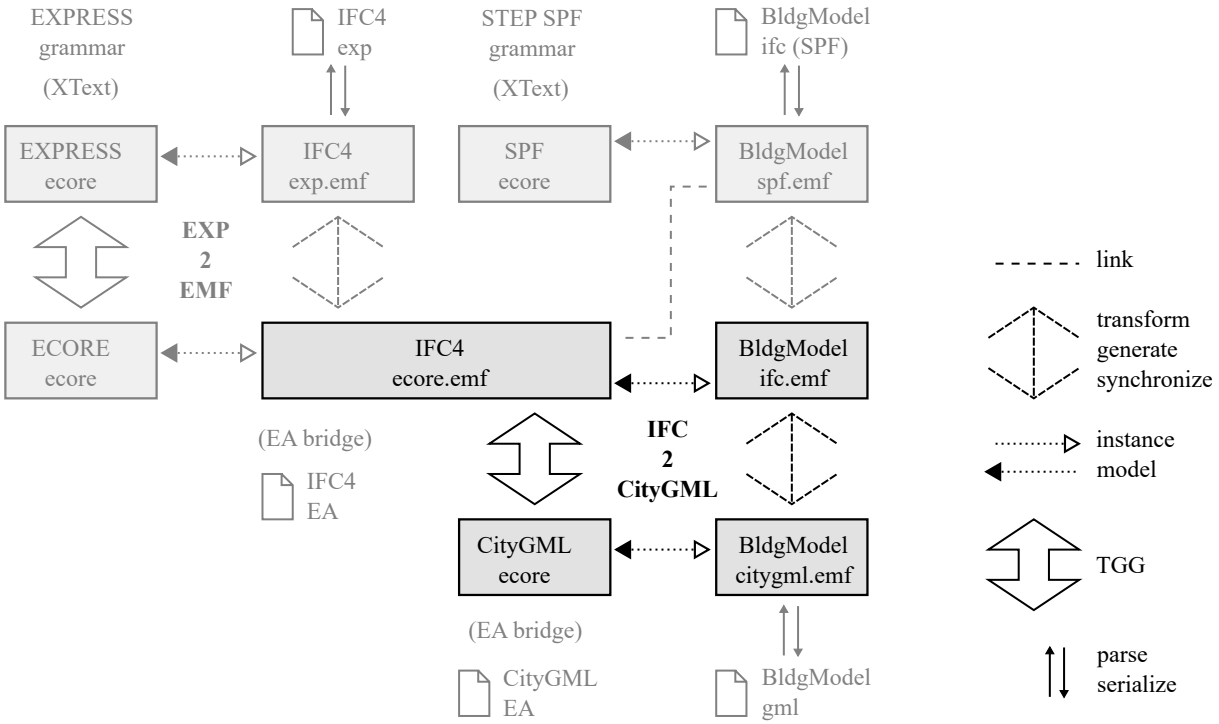

Figure 7. Work and data flow to integrate meta and instance models for IFC and CityGML integration

Architect and the respective EA project files (*.eap) are published in the OGC Repository[5]. These files can be converted to UML and further to ECore. For the first step, we are using itemis' YAKINDU EA bridge[6] and for the second step the Eclipse UML2 plugin[7]. This process is potentially (and to a certain extend intentionally) lossy. The UML is a visual modelling language, providing diagrams to describe various aspects of software systems. First, we are only interested in particular diagrams: class and package diagrams. Second, we only need the abstract syntax, not the visual diagram definitions. Third, with ECore, we can directly handle only a subset of UML (corresponding to the EMOF) and no extensions or profiles. The use of EA files through the EA bridge (or the direct use of UML) can also serve as an alternative for IFC metamodel input in upcoming IFC versions.

Both in the case of EXPRESS metamodel input and EA or UML input we will encounter concepts that are not directly covered by ECore. There is limited extensibility of ECore through so-called annotations. In some cases the EXPRESS construction can be replaced by an equivalent EMOF construction without loss. But since the focus of this paper is on the integration and transformation of building models, we must refer the reader to future publications.

3.2 Instance models

The ECore instance models (BldgModel.ifc.emf and BldgModel.citygml.emf in Figure 7) are expected as input or to be generated depending on the operational transformation scenario. Accordingly they either need to be parsed (if input) or serialized (if generated) to the respective formats. In the diagram, the direction of the parse/serialize arrow depends on the scenarios as follows:

- Forward transformation: parse IFC, serialize CityGML (optionally link model)
- Backward transformation: parse CityGML, serialize IFC (optionally link model)
- Linking: parse IFC and CityGML, serialize link model
- Synchronization: parse everything, serialize everything (as required)

EMF provides methods for persistence with the XMI format (ISO 19509, 2014) and EMoflon supports these out of the box consuming or producing XMI for the source, target and correspondence model instance as well as a protocol of the transformation. Listings 3 and 4 show the IFC and CityGML building model instances as XMI: BldgModel.ifc.emf and BldgModel.citygml.emf refering to the names of instance models in Figure 7. However, in an operational engineering scenario we want to use instance representations that are more commonly used for data exchange in practice.

As serialization format on the IFC side we currently consider only the STEP physical file (SPF, ISO 10303-21 (2016)) format, because it is the most compact so-called implementation method for data following the Standard for the Exchange of Product Model Data (STEP, ISO10303 series). An example is shown in Listing 1. To populate the IFC building model, we develop an Xtext-based parser. Xtext creates again an ECore model from the SPF grammar (SPF.ecore) which is used as the metamodel to be instantiated during parsing (Listing 2, BldgModel.spf.emf in Figure 7). In addition,

[5] OGC CityGML 3 conceptual model repository: https://github.com/opengeospatial/CityGML-3.0CMl
[6] itemis YAKINDU EA-Bridge: https://www.itemis.com/en/yakindu/ea-bridge/
[7] Eclipse Model Development Tools (MDT) UML2: https://www.eclipse.org/modeling/mdt/?project=uml2

the grammar imports and links against the IFC4 Ecore (IFC4.ecore.emf). The schema model should be dynamically imported in the future (not yet implemented). An additional transformation step is done to actually instantiate/bind the IFC model (bound instance model: Listing 3, BldgModel.ifc.emf). This is currently implemented in one direction only, using the Xtend language.

On the CityGML side we consider the usual XML-based serialization, but have not implemented anything yet.

Likewise, serialization of the correspondence instance model is not yet covered apart from the default XMI serialization. Viable formats would be, for example, the multi model container according to Fuchs (2015) as developed in the German mefisto project[8] (see also Fuchs and Scherer, 2017; Fuchs et al., 2011) or the Information Container for Data Drop (ICDD), ISO 215970 (2018).

The following listings illustrate the transformations from IFC-SPF as input through a forward transformation with excerpts from the serializations, only excluding the CityGML serialization, which would be the final derivative. Figures 8 and 9 show the PlantUML generated diagrams of these instances.

Listing 1. IFC-SPF of the spatial structure sample (BldgModel.ifc), input to the forward transformation

```
#1= IFCSITE('0KMpiAlnb52RgQuM1CwVfd',$,'Gelaende',
    'Ebenes Gelaende','LandUse',$,$,$,.ELEMENT.,
    (49,6,1,566000),(8,26,11,540400),110.,$,$);
#2= IFCRELAGGREGATES('1GO86...G9dnQ',$,$,$,#3,(#1));
#3= IFCPROJECT('0lY6P5Ur90TAQnnnI6wtnb',$,
    'Projekt-FZK-Haus',$,$,$,$,$,$);
#4= IFCRELAGGREGATES('0FWMH...icROM',$,$,$,#1,(#5));
#5= IFCBUILDING('2hQBAVPOr5VxhS3J1O047h',$,
    'FZK-Haus',$,$,$,$,.ELEMENT.,$,$,$);
#6= IFCRELAGGREGATES('1Y0uy...blObD',$,$,$,#5,(#7,#8));
#7= IFCBUILDINGSTOREY('2eyxpyOx95m90jmsXLOuR0',$,
    'Erdgeschoss',$,$,$,$,$,.ELEMENT.,0.);
#8= IFCBUILDINGSTOREY('273g3wqLzDtfYIl7qqkgcO',$,
    'Dachgeschoss',$,$,$,$,$,.ELEMENT.,2.7);
```

Listing 2. SPF-XMI of the spatial structure sample (BldgModel.spf.emf), parsed SPF linked to the schema, but not bound

```
<entities name="#6">
  <type href="IFC4.ecore#//IfcRelAggregates"/>
  <parameters>
    <values v="1Y0uyqfGvXQyvJl5QblObD"/>
    <values v="$"/>
    <values v="$"/>
    <values v="$"/>
    <values e="//@data/@entities.6"/>
    <values xsi:type="step:ParameterList">
      <values e="//@data/@entities.15"/>
      <values e="//@data/@entities.16"/>
    </values>
  </parameters>
</entities>
<entities name="#7">
  <type href="IFC4.ecore#//IfcBuildingStorey"/>
  <parameters>
    <values v="2eyxpyOx95m90jmsXLOuR0"/>
    <values v="$"/>
    <values v="Erdgeschoss"/>
    <!-- more values -->
    <values v=".ELEMENT."/>
    <values v="0.0"/>
  </parameters>
</entities>
<entities name="#8">
  <type href="IFC4.ecore#//IfcBuildingStorey"/>
  <parameters>
    <values v="273g3wqLzDtfYIl7qqkgcO"/>
    <values v="\$"/>
    <values v="Dachgeschoss"/>
    <!-- more values -->
    <values v=".ELEMENT."/>
    <values v="2.7"/>
  </parameters>
</entities>
```

Listing 3. IFC-XMI of the spatial structure sample (BldgModel.ifc.emf), bound to the schema (after instantiation)

```
<xmi:XMI ... xmlns="http://IFC4.ecore">
  <IfcSite Name="Gelaende"/>
  <IfcRelAggregates RelatingObject="/7"
                    RelatedObjects="/0"/>
  <IfcRelAggregates RelatingObject="/0"
                    RelatedObjects="/6"/>
  <IfcRelAggregates RelatingObject="/6"
                    RelatedObjects="/15␣/16"/>
  <!-- skipping 2 objects -->
  <IfcBuilding Name="FZK-Haus"/>
  <IfcProject Name="Projekt-FZK-Haus"/>
  <!-- skipping 7 objects -->
  <IfcBuildingStorey Name="Erdgeschoss"
                     Elevation="0.0"/>
  <IfcBuildingStorey Name="Dachgeschoss"
                     Elevation="2.7"/>
</xmi:XMI>
```

Listing 4. CityGML-XMI of the spatial structure sample (BldgModel.citygml.emf), after transformation

```
<Model.CityGML.Core:CityModel>
  <cityModelMember cityObjectMember="/1"/>
</Model.CityGML.Core:CityModel>
<Model.CityGML.Building:Building buildingSubdivision="/2␣/3"/>
<Model.CityGML.Building:Storey name="Erdgeschoss">
  <elevation elevation="/4"/>
</Model.CityGML.Building:Storey>
<Model.CityGML.Building:Storey name="Dachgeschoss">
  <elevation elevation="/5"/>
</Model.CityGML.Building:Storey>
<Model.CityGML.Construction:Elevation />
<Model.CityGML.Construction:Elevation />
```

4. OPERATIONAL CASES DERIVED

To test the operational transformation systems derived by eMoflon, we use very simple use cases restricted to the spatial structure of project, buildings, storeys and spaces. We limit our explanations here to buildings and storeys. We have developed the respective provisional generic TGG ruleset in Section 2 and use the IFC-STEP import from Section 3.2.

As IFC input, where needed, we use a stripped-down versions of the KIT sample data[9], mainly the FZK house.

[8] Mefisto – Management, Leadership, Information and Simulation in Construction: http://mefisto-bau.de

[9] KIT IFC samples: http://www.ifcwiki.org/index.php?title=KIT_IFC_Examples

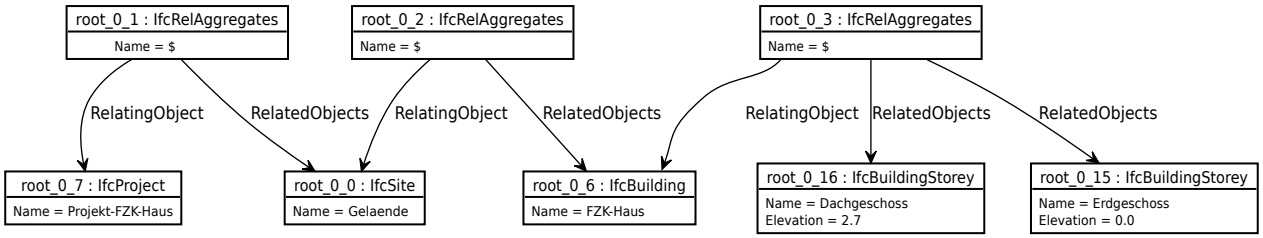

Figure 8. IFC instance of the spatial building structure

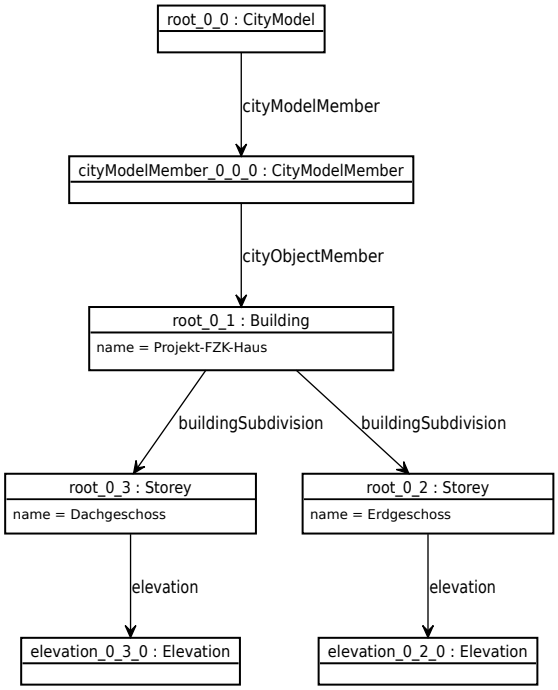

Figure 9. CityGML instance of the spatial building structure

The original IFC files are filtered for the required entities of types `IfcProject`, `IfcSpatialStructureElement`, `IfcRelAggregates` using the Opensource BIMserver with respective queries.

We derive and test the following operational transformation systems, all of which are covered by the incremental model transformation in eMoflon.

4.1 Sample data generation

This operational system generates triples of IFC and CityGML with consistent correspondence according to the given grammar. This can be useful to generate synthetic sample data for the evaluation of new methods in construction and urban planning as well as management and communication in these processes (e.g. simulation, data exchange, code checking).

The triple graph structure to be generated can be limited through the maximum number of applications for a particular rule in eMoflon. The generated attributes are controlled through operational attribute constraints. EMoflon supports random value generation for common data types out of the box. For enumerations, we have implemented a new custom operational attribute constraint. One problem with sample data generation is that the generated instance models might not have a reasonable interpretation, for example if storey elevations and heights are randomly assigned.

4.2 Bi-directional synchronization

This includes forward and backward transformation (where only one of the two instance graphs, IFC or CityGML, exists a priori) as well as updates of corresponding models where either side has changed. The latter case is useful for example for updating an existing city model with new building models derived from the planning process or building application submissions.

We test forward (IFC2CityGLM) transformation with the IFC-SPF input. For backward transformation we use the forward transformation results and manually edited ECore instance model. To test the update case, change storey elevation, add another storey and additional properties (not handled by the rule set) on either side.

4.3 Consistency checking

This includes the case of an existing correspondence model, where the whole triple is validated against the TGG ruleset as well as the case of establishing a correspondence model to complete a consistent triple according to the grammar. The latter of these two is more interesting for current use cases in BIM-GIS integration, where data needs to be connected instead of converted.

To test this case, we use the variations generated during bi-directional synchronization.

5. DISCUSSION

5.1 Summary

We have shown how eMoflon supports and simplifies rule authoring as compared to our old implementation (Section 2), how arbitrary domain models, schemas and serializations can be integrated, where our previous implementation was limited to IFC and CityGML (Section 3) and how various operational transformations can be derived from one general ruleset, where the previous implementation only supported forward transformation (Section 4).

We have achieved the goal of demonstrating a first minimal End-to-end (except CityGML serialization) integration. In this process, manual adjustments where still needed (e.g. ECore, XMI editing) to shortcut the development process, but we are working towards a fully automatic solution. Also, we have made deferments with regard of complete implementation which are listed in the following section.

5.2 Limitations of our work

5.2.1 EXPRESS parser. The EXPRESS parser implements most of the grammar, with the exception of expressions, functions, procedures, rules. It would be interesting, but not first priority, to evaluate whether, to which extent and how constraint integration is useful and feasible for the IFC-CityGML case.

5.2.2 EXPRESS-ECore ruleset. The ruleset for metamodel integration (EXPRESS2EMF) is restricted to the minimal useful ruleset: All named types are handled, but no internals for declared, enumeration and select types. Typed attributes and simple attributes are not yet implemented. Further we did not cover inverse attributes yet, since contrary to our earlier implementation they are not necessarily needed.

5.2.3 STEP parser. The STEP parser does currently not handle escaped quotation marks, high codepoints and empty aggregations. Input files have to be preprocessed to get rid of such constructs if existing. Enumerations are ignored. Import of IFC Ecore model on SPF parsing is still static, but dynamic import would be preferable.

5.2.4 CityGML EMF ECore does not allow overriding of attributes ("redefining" in UML terminology). For example, the Core.AbstractSpace.boundary is redefined for the AbstractSpace subtype Building.Room.boundary. As a workaround we have renamed the base feature, but a better solution is needed. Some manual work is needed to handle names which are allowed in UML, but not in Ecore, such as arithmetic operations (*,+,-,/,...). The EMF does a great job with validation of ECore and display of errors. Apart from the ones mentioned above, there are issues in the ISOTC211 subpackage, which we have not yet resolved. Subpackages in general seem not to be handled well in emoflon, which needs further investigation. Also data types did not come through the YAKINDU EA bridge conversion from EA. Further investigation of these issues is a larger amount of work, since CityGML 3 does pull in a lot of dependencies resulting in a huge model. For now, we rebuild a small portion as needed for conversion with emfatic.

5.2.5 IFC2CityGML ruleset The IFC2CityGML ruleset is far from completeness due to the goal of an end-to-end demonstration of the integration. On a more general level, the handling of optional attributes is not implemented and needs to be investigated for the TGG rules, but also for the metamodels.

5.3 Future work

Going forward from this first end-to-end proof of concept implementation, completion of the various parts with their limitations outlined in the previous section forms the immediate agenda.

On the level of metamodels, EXPRESS-EMF integration and EA/UML-EMF integration are larger interesting subjects of research and development also for other use cases rather than IFC2CityGML integration.

We aim to extend the first proof-of-concept ruleset to substantial coverage of the IFC-CityGML correspondence, such that we can than test operational transformations with realistic IFC and CityGML data sets. This can be achieved by reimplementation of the full existing IFC-to-CityGML forward transformation. Even though this ruleset provides a good base, the conversion to generic TGG rules for eMoflon will require substantial work to factor out some of the shortcuts in the forward transformation where correspondence is currently not expressed as declarative graph transformation rule but in an imperative way with so-called “converters”.

A more extensive implementation will allow us to compare our previous implementation and the eMoflon implementation with regard to criteria such as performance and expressiveness.

EMoflon supports various pattern matching engines to be used during rule interpretation. Currently, we are using the Democles pattern matcher. We will also try the newer parallel matcher HiPE and compare the two pattern matchers in the context of the IFC2CityGML use case. This will be interesting once we work with larger datasets and rulesets.

6. CONCLUSION

This paper shows for the first time a proof-of-concept for the application of a generic triple graph grammar to the integration of BIM and GIS in general and IFC and CityGML in particular. We describe workflows for the generation of IFC and CityGML ECore and the workflow for the creation of integration rules, such that the approach can be applied to future standards versions or schema extensions. A basic (but self-contained) IFC-CityGML ruleset is being developed and operational transformation systems demonstrated and tested. We finally assess aspects of usability, expressiveness and performance of the approach as compared to our earlier forward-only implementation.

This work is in progress, under active development and in an early stage. We document the development state, progress, source code and usage notes in the Github repository at http://github.com/hlg/EMF2EXPRESS.
We hope the development of the generic TGG ruleset can foster a community discussion about BIM-GIS integration, in particular the details of the mapping between IFC and CityGML. With the generic nature of the rules and their independence of specific operational dataflow requirements, they are suited to provide a technical base for discussions about a canonical mapping between IFC and CityGML and to facilitate the involvement of domain experts into the process. In our opinion, this fits well with ongoing efforts to standardize BIM-GIS integration such as the efforts of the IDBE working group to identify use cases, semantic overlaps and incompatibilities across the BIM and GIS domains or the joint ISO/TC 59/211 working group on GIS–BIM interoperability (ISO/CD TR 23262). The approach can also be transferred to other data formats, schemas and metamodels such as GAEB or gbXML on the building model and CityJSON or IndoorGML on the city model side.

ACKNOWLEDGEMENTS

The author would like to thank Ordnance Survey GB (www.ordnancesurvey.co.uk) and 1Spatial (www.1spatial.com/) for sponsoring the publication of this paper.

REFERENCES

Beetz, J., van Berlo, L., de Laat, R., van den Helm, P., 2010. BIMserver.org: An open source IFC model server. *Proceedings of the CIB W78 2010: 27th International Conference*. Cairo, Egypt.

Bettini, L., 2016. *Implementing domain-specific languages with Xtext and Xtend: Learn how to implement a DSL with Xtext and Xtend using easy-to-understand examples and best practices*. Community experience distilled. Packt Publishing, Birmingham, UK, 2nd edition edition.

Daum, S., Borrmann, A., Kolbe, T., 2017. *A Spatio-Semantic Query Language for the Integrated Analysis of City Models and Building Information Models*, Springer International Publishing, Cham, 79–93. doi:10.1007/978-3-319-25691-7_5.

Donkers, S., Ledoux, H., Zhao, J., Stoter, J., 2016. Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings. *Transactions in GIS*, 20(4), 547–569. doi:10.1111/tgis.12162.

Fowler, M., 2004. *UML distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley, Safari, Boston, MA, 3rd ed. edition.

Fuchs, S., 2015. *Erschließung domäneübergreifender Informationsräume mit Multimodellen*. Ph.D. thesis, TU Dresden.

Fuchs, S., Kadolsky, M., Scherer, R.J., 2011. Formal description of a generic multi-model. *WETICE - 20th International Conference on Collaboration Technologies and Infrastructures*. Paris, France.

Fuchs, S., Scherer, R.J., 2017. Multimodels: Instant nD-modeling using original data. *Automation in Construction*, 75, 22–32.

ISO 10303-11, 2004. Industrial automation systems and integration. product data representation and exchange. part 11: Description methods: The express language reference manual. Technical Report 10303-11, International Organization for Standardization, Geneva, Switzerland.

ISO 10303-21, 2016. Industrial automation systems and integration. product data representation and exchange. part 21: Implementation methods: Clear text encoding of the exchange structure. Technical Report 10303-21, International Organization for Standardization, Geneva, Switzerland.

ISO 16739, 2013. Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries. Technical Report 16739, International Organization for Standardization, Geneva, Switzerland.

ISO 19505, 2012. Information technology. object management group unified modeling lanuage (OMG UML). part 2: Superstructure. Technical Report 19505-2, International Organization for Standardization, Geneva, Switzerland.

ISO 19509, 2014. Information technology. object management group XML metadata interchange (XMI). Technical Report 19509, International Organization for Standardization, Geneva, Switzerland.

ISO 215970, 2018. Information container for data drop. exchange specification. part 1: Container. Technical Report 21597-1, International Organization for Standardization, Geneva, Switzerland.

Kutzner, T., Chaturvedi, K., Kolbe, T.H., 2020. CityGML 3.0: New functions open up new applications. *Journal of Photogrammetry, Remote Sensing and Geoinformation Science (PGF)*, 88(1), 43–61. doi:10.1007/s41064-020-00095-z.

Leblebici, E., Anjorin, A., Schürr, A., 2014. Developing eMoflon with eMoflon. D.D. Ruscio, D. Varró (eds.), *International Conference on Theory and Practice of Model Transformations (ICMT)*. Springer International Publishing, Cham, 138–145. doi:10.1007/978-3-319-08789-4_10.

Lim, J., Tauscher, H., Biljecki, F., 2019. Graph transformation rules for IFC-to-CityGML attribute conversion. *Proceedings of the 14th 3D GeoInfo Conference*. Singapore, 83–90. doi:10.5194/isprs-annals-IV-4-W8-83-2019.

Salheb, N., 2019. *Automatic Conversion of CityGML to IFC*. Master's thesis, TU Delft.

Schürr, A., 1995. Specification of graph translators with triple graph grammars. E.W. Mayr, G. Schmidt, G. Tinhofer (eds.), *Proc. of the 20th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '94)*, Springer, Berlin, Heidelberg, 151–163.

Steinberg, D., 2009. *EMF: Eclipse modeling framework*. The eclipse series. Addison-Wesley, Boston, 2nd edition, revised and updated edition.

Stouffs, R., Tauscher, H., Biljecki, F., 2018. Achieving complete and near-lossless conversion from IFC to CityGML. *International Journal of Geo-Information (IJGI)*, 7(9), 355. doi:10.3390/ijgi7090355.

Tauscher, H., 2019. Creating and maintaining IFC–CityGML conversion rules. *Proceedings of the 14th 3D GeoInfo Conference*. Singapore, 115–122. doi:10.5194/isprs-annals-IV-4-W8-115-2019.

Tauscher, H., Stouffs, R., 2019. Extracting different spatio-semantic structures from IFC using a triple graph grammar. *Intelligent and informed: 24th Annual Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA 2019)*. Wellington, New Zealand, 605–614.

Revised August 2020