Enhanced Differencing and Merging of IFC Data by Processing Spatial, Semantic and Relational Model Aspects

Simon Daum, André Borrmann
Chair of Computational Modeling and Simulation
Technische Universität München, Germany
simon.daum@tum.de

Abstract. Building Information Modeling promotes collaborative work in which models are edited and enhanced in parallel by domain experts. In this environment, revised and extended datasets have to be combined to an overall database. To perform this merging, duplicates have to be identified. This paper describes a novel, comprehensive approach for detecting equivalences in datasets of the Industry Foundation Classes (IFC), an open and full-fledged schema for building models. In contrast to available methods, the presented approach is independent of object identifiers. Instead, the detection of duplicates is performed on entity level considering spatial, semantic and relational aspects of the IFC data.

1. Introduction

The paradigms of Building Information Modeling (BIM) promote the use of comprehensive digital building representations. Instead of previously adopted approaches, which focus on pure geometric data, a BIM comprises not only the 3D shapes of elements and enclosed spaces, but also type information and the relationships between elements. Therefore, a BIM has the potential to serve as a data pool used by all participants involved in the design, construction and operation of a building. Despite the benefits of a BIM-based planning, there are still open questions in the field. In this contribution, we delve into methods of combining several submodels to an overall BIM model.

The terminology we use is borrowed from textual comparison tools. For instance, these tools are employed for the management of source code which is concurrently edited by several developers. In this context, the process of creating one integrated dataset from changed files is referred to as *merging*. Here, textual data has to be classified as original, newly added or modified. Hence, the textual comparison tools can show all differences (also known as *diffs*) between two files.

In a BIM environment, merging takes place as part of a distributed planning in which various project participants separately enhance their sections of the building model (Weise et al., 2004). The workflow of one participant can be summarized as submodel extraction, model enhancement und submodel integration (Figure 1).

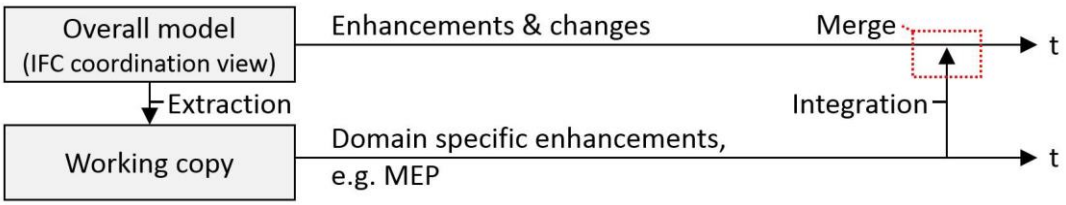

Figure 1: Merging in a multi-disciplinary BIM-based planning process.

The last step of the workflow comprises the necessary merging of the current overall model with the modified submodel. As the working copy contains original elements, some of them will be represented twice, once in the overall model and once in the working copy. This is critical as these duplicates lead to erroneous computations in downstream application, e.g. in quantity take-offs or simulations.

In the currently applied tools, the support for merging BIM data is limited. A manual solution is to tag the original entities at the time when domain specific enhancement starts. But this annotating demands a great amount of discipline of all involved editors, leads to recurrent manually work, is error-prone and hampers the editors in their actual tasks.

On the other hand, the support of a high quality merging process for BIM data is important because of the specific conditions of collaboration in the construction industry. Despite the former aspiration of one central BIM model, the reality shows that even in midsize projects several interrelated models are produced. Additionally, projects are developed during complex design and planning phases, in which models are often significantly changed. Finally, there are a large number of contributors involved in the ongoing detailing of a model. Here, each domain focuses on different views of the model whereas the used subsets of entities intersect.

The paper is structured as following. In the next section, related work is discussed. In Section 3 it is explained why a textual merging is not feasible. Instead, an entity-based approach is presented in Section 4, showing how doublets can be found by spatial, attributive and relational aspects of a model. The following section describes the prototypical implementation of the developed approach and presents a case study. Section 6 concludes this contribution and gives an outlook for further research.

2. Related Work and State of the Art

There are various applications for differencing and merging textual data. One of the best known tools is *diff3* which is used e.g. in revision control systems (Smith, 1988; Khanna et al., 2007). For textual merging, algorithms are facilitated which compute the *longest common subsequence* to identify unchanged parts between documents (Hunt and MacIlroy, 1976).

For differencing and merging of more structured data, various proposals are available in literature. In the domain of model-driven engineering, methods for merging are developed for the general Meta-Object Facility (MOF) and demonstrated for data of the Unified Modeling Language (UML) (Alanen and Porres, 2003). For documents written in the Extensible Markup Language (XML), various merging algorithms are available which process the tree structure of this encoding (Lindholm, 2004; La Fontaine, 2002). An algorithm for merging graph-like structures is discussed in Zündorf et al., (2001).

In the domain of 3D modelling, a system is presented that operates on the identifiers of meshes (Doboš and Steed, 2012). Also the BIM Server has the ability to check for duplicated entities by examine their object identifiers (Beetz et al., 2010). Later in this article, it is explained why this is not reliable if BIM-data is processed. More BIM-specific approaches are presented in Koch and Firmenich (2006, 2011). Here it is proposed to store all steps of operations in the model. This is beyond the scope of the current schemas and would require an extensive redesign of the transfer formats and all applied applications. In contrast, the approach which is described in this contribution can deal with the currently available Industry Foundation Classes (IFC).

To verify the spatial quality of BIM data, clash detection is employed (Tommelein and Gholami, 2012). Here, intersecting geometric representations of entities are reported. The implementation of clash detection is based on the identification of triangle intersections (Möller, 1997). Indeed, finding intersections can be used to identify doublings. But clashes do not indicate spatial duplicates by necessity as intersections can also occur in touching constellations. Thus, another geometrical examination is needed.

Appropriate strategies for a reliable geometrical matching can be found in the field of mechanical engineering. Here, the identification of similar 3D shapes in a database of components is an ongoing research topic. To efficiently select parts in this repository, the user introduces a lookup mesh. In the query, similarities between the lookup mesh and the stored shapes are computed and the most similar entities are returned (Rea et al., 2005; Tangelder and Veltkamp, 2008).

3. Textual Versus Entity-based Differencing

Whereas textual comparison tools match paragraphs and search for added and removed words and tokens, the merge process in a BIM environment requires a different strategy. As BIM data is based on identifiable entities of the considered Universe of Discourse (UoD), the differencing and merging has to be performed on entity level. In this contribution, entities are represented as instances of the IFC4 schema (Liebich, 2013).

To perform the merging, the changed entities of the current submodel have to be integrated in the overall model whereas certain entities supersede other entities. Three sets of IFC entities can be classified in the emerging overall model:

1. Some entities are represented twice and match each other exactly. This comprises the entity's attribute values, its use in relations as well as its geometric representation.
2. Some entities are represented twice but do not match each other exactly as some or all of the following aspects have changed.
    a. The geometric representations of the entity pair differ.
    b. The attribute values of an examined entity pair differ.
    c. The use of the entities in relations differs.
3. Some entities are added to the datasets independently and therefore have no matches.

The recognition of spatial, attributive and relational differences between the datasets enables for a fine-grained merging during model integration. In the first case, which represents a complete match, duplicates can be eliminated automatically. This is also the group of entities we focus on in this contribution. The second and third cases may call for intervention by the end user, e.g. if conflicts emerge because of simultaneous editing.

To extract the three types of differences, the identification of entities is required. In the IFC, entity identification is based on the *GlobalId* attribute with holds a unique string. However, there is a high risk of losing the GlobalId of an entity during processes of import, export and conversion.

As an example for the weak support of consistent GlobalId handling, a minimalistic IFC file is exported from Autodesk Revit 2015 by use of the extended IFC exporter plug-in (Autodesk, 2016). In the first case, a single wall with an opening forms the dataset. For the second export, the opening is deleted and the plain wall is exported. A textual differencing of

the two files shows that the GlobalIds of the associated relationships get lost, although the entities reflect the identical, unchanged information (

Figure 2). Note, that the textual comparison can only help to identify differences in these minimalistic IFC files which approximately comprise 250 lines of code. Files in realistic productive scenarios contain about 10.000 – 1.000.000 lines of code and are thus by far too large for textual comparison.

```
#209= IFCRELDEFINESBYPROPERTIES('3uhd6lbe190x2omI6GDmTm',#41,$,$,(#110),#207);
#217= IFCRELAGGREGATES('1F0zP3_s91Sgv41B5mmR_6',#41,$,$,#91,(#201));
---
#209= IFCRELDEFINESBYPROPERTIES('1qNfZoYpf52vnboENcp0uY',#41,$,$,(#110),#207);
#217= IFCRELAGGREGATES('1OeBRZBMj68B3HEWkiN2IR',#41,$,$,#91,(#201));
```

Figure 2: Varying GlobalIds of unchanged entities in two IFC exports.

To demonstrate the inadequacy of a textual comparison, a common IFC sample file is imported in Allplan 2016 and exported again (KIT, 2008). As the original file contains about 50.000 IFC entities, the exported one is made of about 150.000 entities. The multiplication of entities can be explained because of the duplication of geometrical elements, e.g. IfcCartesianPoint instances. A textual comparison of the two files does not detect any identical part of the two datasets. Nevertheless, from the view of a domain expert, both sets represent the identical model.

Thus, the matching of IFC files on a textual basis is not feasible. Furthermore, identifying entities by their GlobalIds is considered as unreliable. Other approaches for finding duplicates are necessary. To close this technology gap, a novel approach for matching IFC entities is presented next.

4. An Entity-Centric Matching Approach

All versions of the IFC schema are defined by use of the EXPRESS modelling language. The most central modelling elements of the language are the *Entity* and the *Type*. The first can be seen as a full-fledged object description, the latter depicts more simple types. Furthermore, there are elements for representing containers such as list and sets, enumerations and selection types. A detailed discussion of EXPRESS-based modelling is available in Schenck and Wilson (1994).

Although the IFC contains about 650 different entity declarations, only subclasses of IfcRoot have an unambitious identity. Other entities must be referenced by rooted instances and cannot be stored independently. This modelling approach leads to an elementary consequence for the matching of IFC datasets. Searching duplicates of non-rooted objects is not feasible as these instances attain their semantics solely by references from rooted objects. Therefore, the matching between entities must start at rooted objects, which are traversed for all referenced entities.

IfcRoot splits up in three subclasses. These are IfcObjectDefinition, IfcRelationship and IfcPropertyDefinition. The first one represents semantically described objects or processes on instance and type level. IfcRelationship is used to express entity-to-entity connections by explicitly instantiated objectified relationships. For example, this can indicate the spatial assignment of a wall to a storey or that a floor is covered by a specific coating. The

IfcPropertyDefinition stands in for pre- and user-defined semantically related sets of properties. Again, the concept of objectified relationships is used to establish the connection between an entity and a property set. A set can dynamically be attached to an entity by an IfcRelDefinesByProperties instance, subclass of IfcRelationship. Finally, a common type of some building element can be represented in the IFC schema by an IfcTypeObject object. The type comprises property sets and geometric representations which are reused by the linked entities. To establish the connection between elements and their IfcTypeObject, the IfcRelDefinesByType relationship is used.

For the user, the concepts of types, property sets and objectified relationships are quite abstract. The most tangible objects are IfcObjects, subclass of IfcObjectDefinition which stand in e.g. for building elements, processes and actors. For a domain expert, these objects and their mutual relations represent the primary informative content of the model.

Thus, from a user's point of view, a differencing algorithm for IFC data should be based on comparing IfcObjects. Other entities e.g. instances of IfcPropertyDefinition, IfcRelationship and IfcTypeObject are examined indirectly by their interactions with IfcObjects. This is demonstrated in Figure 3. Property sets which are connected to an entity or its type object are considered as original entity attributes (a, b). Objectified relationships, subclasses of IfcRelationship are processed as attributes which holds between a pair of IfcObjects (c).

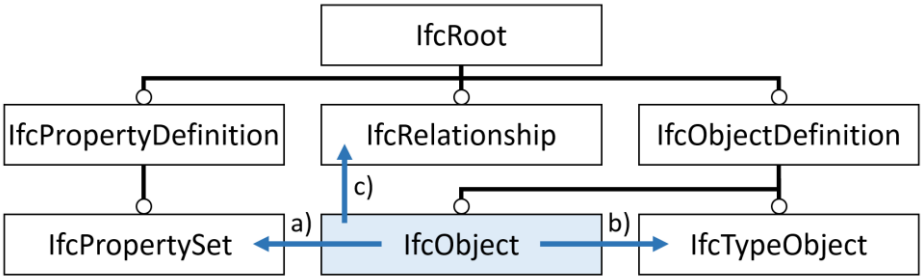

Figure 3: IfcObject instances interact with property sets, type objects and inter-entity relations which makes the class the appropriate starting point in the developed matching algorithm.

In the following the developed matching approach is presented. It comprises two steps. In the first one, the comparison of geometric representations is used to identify duplicates. In the next step, a semantical matching can find attributive modifications of already matched geometrical entities. The semantical matching is also necessary to recognise doublets of non-geometrical entities such as tasks or processes.

4.1 Geometrical Matching

In a BIM dataset, the semantics of an entity are strongly influenced by the entity's geometric representation and the resulting interactions with other spatial objects. Therefore, processing geometric shapes is an essential part in the proposed IFC differencing and merging.

The geometry of IFC entities can be expressed through several modelling approaches. There are data structures for procedural geometry such as lofts and sweeps. In addition, constructive solid geometry (CSG) can be facilitated. Finally, for static and unmodifiable spatial entities, an explicit boundary representation (B-Rep) can be used. A B-Rep structure can also be produced from procedural geometry and CSG. Thus, a matching for all variants of geometric modelling in the IFC can be based on processing the B-Reps of entities.

However, the creation of B-Rep structures from procedural geometry and CSG is ambiguous in concern of their tessellations. This means that various triangle meshes can be produced

from one procedural geometry or CSG. Thus, a geometry-based identification must be independent from the actual triangulation which can differ in each BIM tool. Only the permanent information about the entity's boundary should be considered.

Therefore, approaches for 3D database lookup are adjusted for a spatial BIM matching. In contrast to geometric database lookups, the comparison of shapes should not be invariant to linear transformations such as transition, rotation and scaling. These operations change the semantics of entities in a building model. Instead, the shape matching is performed on globally oriented representations.

In the suggested processing, sample points are distributed on the surface of an entity. For each sample point, the minimal distance to another meshes is computed (Figure 4). To speed up this computation, an upper bound of distance d_{max} is introduced and the triangles of entities are indexed in R*-Trees (Beckmann et al., 1990). By extending the bounding box of an entity with d_{max}, corresponding candidate meshes can be found (a). In the next step, an extended triangle box is used in the candidate's tree to identify triangle candidates (b). These triangles are then examined for distances to the sample points on the original triangle (c). Finally, the minimal distance found for each sample point is stored.

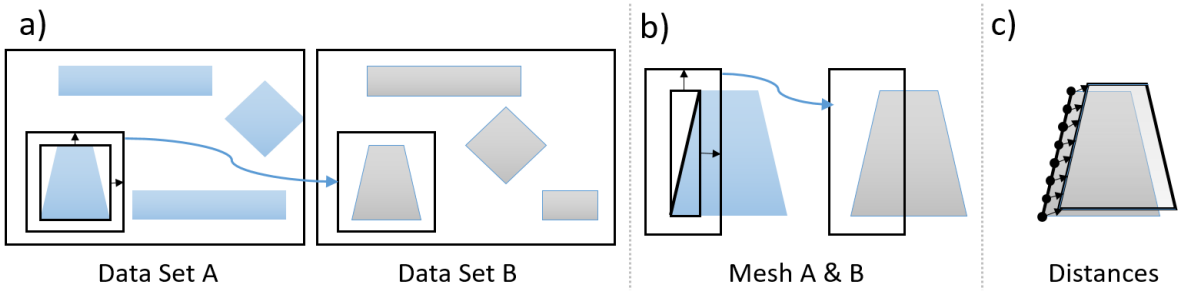

Figure 4: Matching geometric representations by using sample points and computations of minimal point/triangle distance (2D view for simplicity).

If all sample points are verified for lying on triangles of the second mesh, a geometric duplicate is identified. In contrast to clash detection, the presented method matches all surface areas of two entities. In addition, the method can identify the most similar shape in a group of candidates. The number of used sample points can be adjusted which enables for a trade-off between the robustness of the algorithm against its runtime.

The presented matching on geometry level is performed for all IfcObject instances which offer a geometric representation. During a pre-processing step, the necessary B-Rep structures are computed. After the geometric matching has been executed, the semantic matching is invoked. Accordingly, in the semantic matching, links to geometric representations are not considered.

4.2 Semantic Matching

In computer science, two equality comparisons are common. The first one is called *value equality*, also known as *equivalence*. This means that the actual values of two variables are equal. The second equality determines if two variables point to the same address in memory. This type of equality check is called *reference equality*, or *identity* (Albahari and Albahari, 2012).

In IFC modelling, the worldwide unique GlobalId value forms the identity of an entity and thus can be used for checking a specific kind of reference equality between two instances. If

this kind of equality would be robust, it would be an appropriate basis for matching and differencing between IFC datasets. Actually, it had the potential for extracting a change history for entities. But, as explained in Section 3, GlobalIds are often handled inconsistently during import/export steps. A hypothesis of this contribution is that differencing and matching on reference level is not necessary for an IFC merge. Instead, an approach based on the former described spatial comparison in combination with equivalence checks is proposed.

In the following subsection, the algorithmic approach for semantic matching is depicted. The algorithm for finding duplicates on the semantical level iterates through all IfcObjects of the two IFC datasets under examination. For each IfcObject pair the ValueEquality function is called. In this process, attributes with simple types and references to other entities are recursively compared. The two overloaded ValueEquality functions are stated in Figure 5.

```
Algorithm bool ValueEquality (Entity entityA, Entity entityB)
        if TypeOf(entityA) != TypeOf(entityB)
                return false
        foreach attributeA of entityA.AttributesWithoutGlobalId
                attributeB = entityB.FindAttribute (attributeA)
                if ! ValueEquality(attributeA, attributeB)
                        return false
        foreach referenceA of entityA.References
                referenceB = entityB.FindReference (referenceA)
                if ! ValueEquality(referenceA, referenceB)
                        return false
Algorithm bool ValueEquality (Attribute attributeA, Attribute attributeB)
        if TypeOf(attributeA) != TypeOf(attributeB)
                return false
        return attributeA == attributeB
```

Figure 5: The two overloads of the ValueEqual function.

For example, an IfcBuilding is a subclass of IfcObject and accordingly an evaluated entity. Figure 6 shows two IfcBuilding instances. The recursive ValueEquality function traverses both object structures and yields a difference in the *Description* attribute of the IfcOrganization entity.

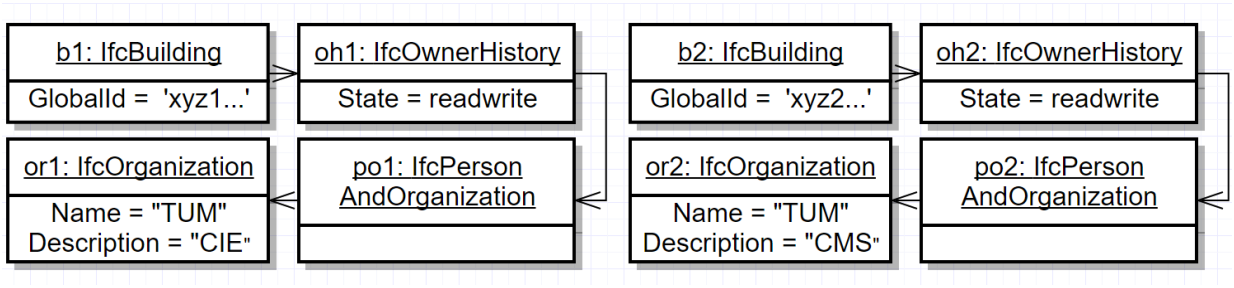

Figure 6: Two IfcBuilding instances which are not value equal because of a difference in the referenced IfcOrganization entity.

5. Prototypical Implementation and Case Study

The presented approach of an IFC merging is evaluated in a C# implementation. As a case study, two data sets of an office building are compared (Figure 7). The original data is taken from KIT (2008). The first set comprises 458 spatial building elements and 17732 IFC entities in total (a). In the second set, 283 instances of IfcDoor and IfcWindow are added which extends the number of IFC entities to 22982 (b). In the initial geometrical matching, all duplicates of the doubled building elements are identified. Figure 7c shows the resulting diff between data set a) and b).

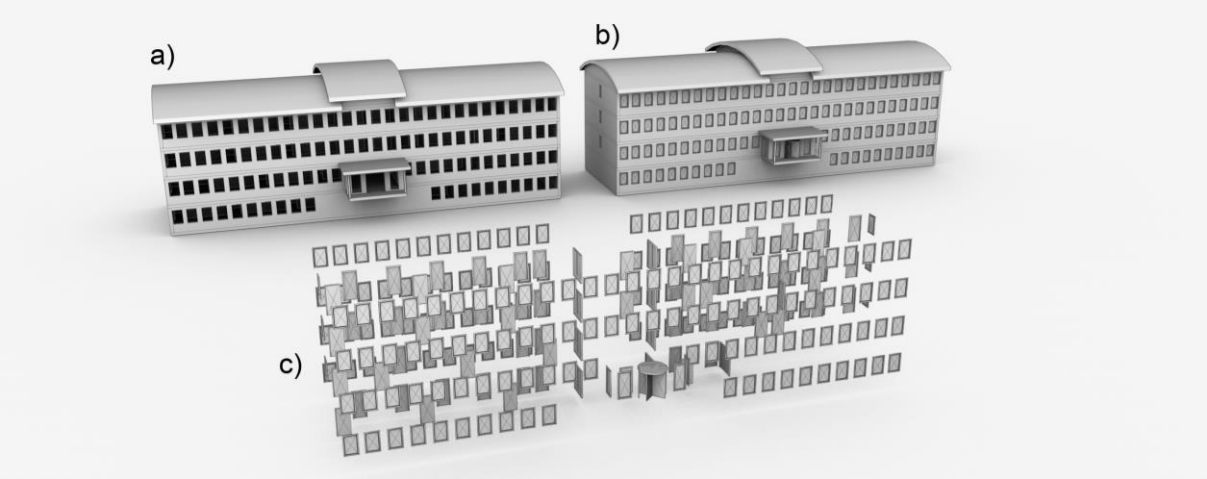

Figure 7: Two IFC data sets and their computed geometrical differences.

The following semantic processing matches entities without geometry representations. In the sample data, these are instances of spatial structures such as IfcSite and IfcBuildingStorey. In addition, correlating type objects such as IfcWallType and IfcRailingType instances are recognised.

The prototype showed that the computational accuracy of the geometrical matching is high. The divergences between actual matching entities were in the range of 1E-4 and 1E-6 millimetres. The computation of the example takes about 25 seconds in the single threaded prototype on a standard workstation.

6. Summary and Outlook

This contribution demonstrates a novel approach for file based collaborative work in a BIM environment. Here, submodels with overlapping entity sets have to be merged to an overall model. Instead of relying on the matching of object identifiers, the presented method is based on a two-level processing. In the first step, geometrically matching entities are identified. Then, a sematic matching is used to link entities which are equivalent. Instead of a time consuming and error-prone manual investigation of model differences, the method helps to produce high quality overall models more efficiently.

There are various research questions emerging. The most centric one is dealing with a matching algorithm based on a similarity metric. This is necessary when entities without geometrical representation have changed slightly. Here, the presented approach will not find duplicates. With a metric, the semantically most similar entity can be identified. Finally, more

research is necessary to extend the method and handle more complex scenarios such as a three way diff with simultaneous modifications, additions and deletions in the entity sets.

References

Alanen, M. and Porres, I. (2003) *Difference and union of models*.

Albahari, J. and Albahari, B. (2012) *C# 5.0 in a Nutshell: The Definitive Reference*, O'Reilly Media, Inc.

Autodesk (2016) *IFC for Revit: .NET code for the Revit 2012-2016 IFC open source* [Online]. Available at https://sourceforge.net/projects/ifcexporter/.

Beckmann, N., Kriegel, H.-P., Schneider, R. and Seeger, B. (1990) *The R*-tree: an efficient and robust access method for points and rectangles*, ACM.

Beetz, J., van Berlo, L., Laat, R. de and van den Helm, Pim (2010) 'Bimserver. org-an Open Source IFC model server', *Proceedings of the CIP W78 conference*.

Doboš, J. and Steed, A. (2012) '3D Diff: an interactive approach to mesh differencing and conflict resolution', *SIGGRAPH Asia 2012 Technical Briefs*, p. 20.

Hunt, J. W. and MacIlroy, M. D. (1976) *An algorithm for differential file comparison*.

Khanna, S., Kunal, K. and Pierce, B. C. (2007) 'A formal investigation of diff3', in *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science*, Springer, pp. 485–496.

KIT (2008) *Office Building: Office Building ArchiCAD 11 IFC 2x3* [Online]. Available at http://iai-typo3.iai.fzk.de/www-extern/index.php?id=1184&L=1.

Koch, C. and Firmenich, B. (2006) 'A novel diff and merge approach on the basis of operative models', *Proc. of the 11th Int. Conf. on Civil and Building Engineering (ICCCBE-XI)*.

Koch, C. and Firmenich, B. (2011) 'An approach to distributed building modeling on the basis of versions and changes', *Advanced Engineering Informatics*, vol. 25, no. 2, pp. 297–310.

La Fontaine, R. (2002) 'Merging XML files: a new approach providing intelligent merge of XML data sets', *In Proceedings of XML Europe 2002*.

Liebich, T. (2013) *IFC4 specification* [Online], no. 4. Available at http://www.buildingsmart-tech.org/ifc/IFC2x4/rc4/html/index.htm.

Lindholm, T. (2004) 'A three-way merge for XML documents', *the 2004 ACM symposium*. Milwaukee, Wisconsin, USA, p. 1.

Möller, T. (1997) 'A fast triangle-triangle intersection test', *Journal of graphics tools*, vol. 2, no. 2, pp. 25–30.

Rea, H. J., Sung, R., Corney, J. R., Clark, D. E. R. and Taylor, N. K. (2005) ‘Interpreting Three-Dimensional Shape Distributions’, *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 219, no. 6, pp. 553–566.

Schenck, D. and Wilson, P. R. (1994) *Information modeling: the EXPRESS way*, Oxford University Press New York.

Smith, R. (1988) *GNU diff3: Version 2.8. 1* [Online], distributed with GNU diffutils package.

Tangelder, J. W. H. and Veltkamp, R. C. (2008) ‘A survey of content based 3D shape retrieval methods’, *Multimedia Tools and Applications*, vol. 39, no. 3, pp. 441–471.

Tommelein, I. D. and Gholami, S. (2012) ‘Root causes of clashes in building information models’, *Proceedings of IGLC20: 20th Annual Conference of the International Group on Lean Construction*.

Weise, M., Katranuschkov, P. and Scherer, R. J. (2004) ‘Managing long transactions in model server based collaboration’, *eWork and eBusiness in Architecture, Engineering and Construction: Proceedings of the 5th European Conference on Product and Process Modelling in the Building and Construction Industry-ECPPM 2004, 8-10 September 2004, Istanbul, Turkey*, p. 311.

Zündorf, A., Wadsack, J. P. and Rockel, I. (2001) ‘Merging graph-like object structures’, *Proceedings of the Tenth International Workshop on Software Configuration Management, May*, pp. 12–19.