Automation in Construction 86 (2018) 53–68

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

IFCdiff: A content-based automatic comparison approach for IFC files

Xin Shi[a], Yu-Shen Liu[a,b,c,*], Ge Gao[a], Ming Gu[a], Haijiang Li[d]

[a] School of Software, Tsinghua University, Beijing 100084, China
[b] Key Laboratory for Information System Security, Ministry of Education of China,China
[c] Tsinghua National Laboratory for Information Science and Technology (TNList), China
[d] BRE Institute of Sustainable Engineering, Engineering School, Cardiff University, UK

ARTICLE INFO

Keywords:
Building Information Modeling (BIM)
Industry Foundation Classes (IFC)
IFC comparison
Change detection
Similarity and difference

ABSTRACT

With the growth in popularity of the IFC (Industry Foundation Classes) format used in construction industry, it often requires effective methods of IFC comparison to keep track of important changes during the lifecycle of construction projects. However, most IFC comparisons are based on a visual inspection, a manual count and a check of selective attributes. Although a few techniques about automatic IFC comparisons have been developed recently, they are usually time-consuming, and are sensitive to the GUID change or redundant instances in IFC files. To address these issues, we propose a content-based automatic comparison approach, named *IFCdiff*, for detecting differences between two IFC files. The proposed approach starts with a comprehensive analysis of the structure and content of each IFC file, and then constructs its hierarchical structure along with eliminating redundant instances. Next, the two hierarchical structures are compared with each other for detecting changes in an iterative bottom-up procedure. Our approach fully considers the content of IFC files without the need of flattening instances in IFC files. In contrast with previous methods, our approach can greatly reduce the computational time and space, and the comparison result is not sensitive to redundant instances in IFC files. Finally, we demonstrate a potential application to incremental backup of IFC files. The software can be found at: http://cgcad.thss.tsinghua.edu.cn/liuyushen/ifcdiff/.

1. Introduction

During the last decade, Building Information Modeling (BIM) has received a considerable amount of attention in the domain of Architecture, Engineering and Construction (AEC) to support lifecycle data sharing [1]. As an open and neutral data format specification for BIM, Industry Foundation Classes (IFC) [2] plays a crucial role to facilitate interoperability between various software platforms. The IFC data format has been widely supported by the market-leading BIM software vendors. Many recent studies also demonstrate the IFC viability in various applications, such as evaluation of design solutions [3], virtual construction [4], construction management [5], model checking [6,7], path planning [8], file optimization [9], semantic annotation [10] and information retrieval [11,12].

With the growth in popularity of the IFC format used in construction industry, it often requires an effective IFC comparison method to keep track of important changes during the lifecycle of construction projects. The IFC comparison aims to analyze and identify the differences and similarities between two IFC files. It is a fundamental problem which may arise in many BIM-based applications, such as collaborative building design [13], incremental backup of files, construction project management [5], product data exchange [14–16], conformance checking [15], handover for operation and maintenance [15]. Previous IFC comparisons are usually based on a visual inspection, a manual count and a check of selective attributes [15,17-19]. However, due to the large file sizes, the complex inheritance and referencing relationships of IFC files, such a way of manual inspection is often time-consuming and error-prone; furthermore, it can only report a partial and illustrative view of the compared files [14]. Although a few recent studies have been developed for automatic IFC comparison [14,18,20], their methods are usually very time-consuming, and are sensitive to the globally unique identifiers (GUID) change [18,20] or redundant instances [14] within IFC files.

To address these issues, we propose a content-based automatic IFC comparison approach, named *IFCdiff*, for tracking differences or detecting changes between two IFC files. Our approach starts with a comprehensive analysis of structure and content of each IFC file, and then constructs its hierarchical structure along with eliminating

* Corresponding author at: School of Software, Tsinghua University, Beijing 100084, China.
E-mail addresses: coolstone712@126.com (X. Shi), liuyushen@tsinghua.edu.cn (Y.-S. Liu), gg07@mails.tsinghua.edu.cn (G. Gao), guming@tsinghua.edu.cn (M. Gu), lih@Cardiff.ac.uk (H. Li).
URL: http://cgcad.thss.tsinghua.edu.cn/liuyushen/ (Y.-S. Liu).

https://doi.org/10.1016/j.autcon.2017.10.013
Received 29 June 2016; Received in revised form 17 August 2017; Accepted 17 October 2017
Available online 10 November 2017

redundant instances at each level. Next, the two hierarchical structures are compared with each other for detecting changes in an iterative bottom-up procedure. Our approach fully takes into account the content of IFC files and makes good use of the hierarchical structure of IFC files. First, our approach can significantly reduce the computational time and space. Furthermore, the comparison result using our method is not sensitive to redundant instances within IFC files. In addition, we also demonstrate a potential application of our approach to incremental backup of IFC files.

The paper is organized as follows. Section 2 reviews the related work and summarizes the existing problems. Section 3 introduces some basic concepts and terms of IFC files. Section 4 gives a detailed description of our approach. Section 5 demonstrates the experimental results and a potential application to incremental backup of IFC files. Finally, Section 6 concludes this paper, summarizes our contributions and discusses some future work.

2. Related work

Early studies of IFC comparison mainly conducted a visual inspection of models and a check of selective attributes in the original and exchange models [15,17-19]. The visual inspection can be done with various IFC viewers that are available, while the attribute analysis is usually a manual check for building elements. However, only using a visual and manual way for comparing IFC files is inaccurate and incomplete due to the complex referencing and inheritance structure of IFC files [14]. The manual way is useful for only small and simple IFC models, whereas it is not practical for large and complex models in the actual construction projects. Consequently, there is an urgent need for developing automatic IFC comparison tools in the scenario of IFC-based data management.

2.1. Plain text comparison

There are various approaches in use for performing automatic comparison of IFC files. An IFC file is a plain text (ASCII) format with the extension "*.ifc", which is specified by IFC and ISO 10303-21 [21] (also known as "STEP physical file"). Therefore, a direct approach is to use plain text comparison tools for directly comparing two IFC files, regardless of information content of models. Some widely used text comparison tools [22], such as *diff*, *DiffMerge*, *cmp*, *FileMerge*, *SVN*, *CVS* and *BCompare*, can be conducted for this purpose. These tools usually compute the longest common subsequence and highlight the differences between textual files. However, pure text comparison does not consider the specific data organization and representation of an IFC file. Therefore, the traditional text comparison tools are not suitable for IFC file comparison.

2.2. GUID-based IFC comparison

Another kind of approaches is based on the globally unique identifier (GUID) [18,20] which is an unique identifier for object instances across applications and systems. The general strategy of GUID-based comparison is as follows. If there is an instance in one IFC file which has the same GUID as an instance in another IFC file, they can be considered as the same instance; otherwise, they are considered to be different instances even with the same attributes of the entity or of its reference entities. The GUID-based comparison is widely adopted by many commercial BIM platforms such as *Autodesk Revit*, *Navisworks* and *Graphisoft ArchiCAD*. Some research articles [14,20] also discussed how to use the GUIDs for measuring the differences between IFC files.

However, in the IFC specification, only the entities inherited from *IfcRoot* have GUIDs in their attributes, while many other entities (e.g. *IfcPropertySingleValue*) not inherited from *IfcRoot* have no GUID [14,20]. In addition, the GUIDs of instances are often changed during the data exchange between different systems even without any modification to the model itself. Therefore, the GUID-based comparison is not a reliable approach to distinguish two IFC files even if it is quite simple and fast for comparison.

2.3. Graph-based IFC comparison

A third type of approaches was proposed by Arthaud and Lombardo [13] in the co-design scenario, which compares two oriented graphs generated by two IFC files. From this, it is possible to track the differences between two IFC models. However, the matching process of nodes between two oriented graphs still complies with the GUID comparison, where the instances without GUIDs are ignored in the comparison process. More recently, Oraskari et al. [23] presented RDF-based signature algorithms for computing differences of IFC models. They convert each IFC model into an RDF graph, in which anonymous nodes (i.e. those instances that do not have any GUID) are assigned GUIDs using a novel signature-based algorithm. As a result, comparisons of IFC models are reduced to graph matching. However, node comparison in the last step is still based on GUID comparison.

It is noteworthy that such graph-based IFC comparisons are non-trivial and time-consuming for large models. Furthermore, they do not handle duplicate data instances in IFC files. In practice, the IFC files generated by various software platforms often include a large number of duplicate data instances [9,14], which should be processed in the procedure of IFC comparison. We will discuss this issue in Section 2.5.2 in detail.

2.4. Flattening-based IFC comparison

The fourth type of approaches, presented by Lee et al. [14], utilizes a recursive strategy to flatten the instances in two IFC files, and then compares the flattened data instances instead of the original ones. The "flattening" process is to replace all the reference numbers with their actual values in each IFC file, which makes an IFC file into a structure that does not include any referencing or inheritance structure [14]. This overcomes the difference of reference numbers included in attribute values when comparing pairs of data instances. As a result, IFC comparison is simplified to pure string comparison after flattening.

This flattening-based method firstly reads two IFC files and parses data into instance name, entity name, and attribute values before comparison. In the following example of one data instance, #90 is the instance name, IFCSLAB is the entity name, and the remaining information within parentheses is the attribute values.

```
#90=IFCSLAB(2VLPPLMIR7fBUKZN0XN2MZ,                    #13,
SLAB_006, , $, #335, #320, $, .FLOOR.)
```

Since different BIM modeling systems might export IFC files in various ways, the instance names and reference numbers might be different. To overcome this difference in referencing mechanisms, the files should be "flattened" first, i.e., making files in a structure that does not include any referencing or inheritance structure by replacing the reference identifier numbers with their actual attribute values. The following shows the flattened data instance of #90.

```
#90=IFCSLAB (2VLPPLMIR7VLPPLMIR7fBUKZN0XN2MZ,
$, UNDEFINED, $, $, $, $, $, $, ORGANIZATIONNAME, $,
$, $, $, GS, GRAPHISOFT, GRAPHISOFT, $, $, 9.0, ACAD9.0,
ARCHICAD, $, .NOCHANGE., $, $, $, 1149148841, SLAB_006, ,
$, $, (0.,0.,0.), (0.,0.,1.), (1.,0.,0.), (0.,0.,0.), (0.,0.,1.), (1.,0.,0.),
(-43500.,14500.,-200.), (0.,0.,1.), IFCPARAMETERVALUE(0.)),
((0.,0.), IFCPARAMETERVALUE(90.)), .T., .CARTESIAN.), .F.,
(0.,0.,0.), (0.,0.,1.), (1.,0.,0.), (0.,0.,1.), 200.)), $, .FLOOR.)
```

Such a flattening process overcomes the difference of reference numbers included in attribute values when comparing pairs of data instances. As a result, IFC comparison is simplified to pure string comparison after flattening.