# Signature-based matching of IFC Models

**Muhammad Tariq Shafiq[a] and Stephen R Lockley[b]**

[a]Department Architectural Engineering, UAE University, UAE
[b]School of Architecture and Built Environment, Northumbria University, UK
E-mail: muhammad.tariq@uaeu.ac.ae

**Abstract –**
**This paper presents the problem of model comparison in platform-neutral collaboration of Building Information Models. The paper describes that model matching and comparison strategies for platform-neutral models (i.e. IFC models) are the keys to solve the problem of iterative change management during BIM collaboration workflows. It is highlighted that the current model comparison strategies are centric towards using GUIDS which may not lead to accurate results due to the complexity of modelling operations and internal data structures of modelling tools. The paper proposes a signature-based model matching approach for IFC models that use model object characteristics to define object signatures for object recognition and comparison. Examples of creating signatures from IFC object characteristics are presented. The proposed methodologies for creating IFC object signatures are implemented in a custom-built tool, XBIM Signatures exporter, which then demonstrates a successful export of object signatures on a test case. The paper concludes that the proposed object signatures can be useful to establish accurate candidates for comparison and can reduce the workload of the overall model comparison process. However, a robust solution for IFC model comparison would require a weighting formula for various object characteristics to formulate an object recognition and comparison strategy.**

**Keywords –**
**BIM, IFC, Model Server, Model Comparison**

## 1 Introduction

Building Information Modelling (BIM) has presented effective solutions to resolved information collaboration problems in the AEC industry. The technology to create efficient Building Information Models is now mature and can support the development of discipline-specific (i.e. Architecture, MEP, Structural) Building Information Models. As the information in a BIM grows during an iterative design and production process and even beyond into maintenance, a critical issue is how to manage the iterative changes because of the collaboration operations and workflows that involve various project participants and heterogeneous applications. The problem of interoperable information exchanges using heterogeneous BIM application has been well researched and has led the development and implementation of Industry Foundation Classes (IFC) over the last 20 years. IFC has now become a widely accepted open and neutral data format specification to facilitate interoperable information exchanges using Building Information Models in the AEC industry.

Today, a platform-neutral BIM enabled collaboration relies on effective management of IFC models using a Model Collaboration System or a Model Server. A model server is a type of database system that allows upload, download, sharing and coordination (e.g., model comparison, and model checking) of models or components by multiple users [1].

An IFC enabled model server is expected to facilitate the exchange of information between the applications used throughout a building project lifecycle (e.g., design tools, analysis tools, document management systems, facility management tools) used throughout a project's lifecycle [2]. The workflow of such a collaboration is built around the concept of finding changes in two versions or variants of IFC model instances and merging them together with the shared data repository. Thus, the issue of IFC model comparison becomes critical to keep track of the changes during the information exchange workflows. The purpose of IFC comparison process is to identify similarities and differences between two IFC models. A fundamental requirement of an effective IFC comparison process is identifying comparable objects or identifying candidates for comparison in the two matching IFC files. Existing IFC Comparisons mechanisms are usually based on using Globally Unique Identifiers (GUIDS) to establish candidates for comparison, in a shallow or a deep tree comparison procedure. GUIDS are a reliable base if these are properly maintained during the IFC roundtripping and the involving editing operations using BIM software applications. However, GUIDS exports are not always consistent due to the complexity of modelling operations, application imprints and different internal data structures of BIM tools, which leads to costly calculations, quality compromises and inaccuracies due to inconsistent GUIDs in IFC model comparison processes.

To address these issues, this paper presents a signature-matching approach that advocates using dynamic object identities, calculated from the value of its properties and attributes using hash keys that can create a unique signature for an object. In BIM collaboration operations, a change can be in (1) an object's position, (2) its shape and its (3) properties. Therefore, an object's position, shape and properties can be used to formulate partial, default or complete signature for an object, which would be useful in establishing candidates for IFC comparison process, independently from GUIDs, and would reduce the overall workload of the IFC compression process. Signature matching is useful even if a complete solution is not feasible, as signature matching can be used to eliminate a significant proportion of elements to downsize the comparison criteria for any further comparison.

## 2 The Problem of IFC Model Comparison

A critical issue in managing collaboration operations on a model server is the management of iterative changes during BIM collaboration operations. The shared data repository on the server must be updated with any changes because of modifications made in a check in/check out the operation, such as new data instances added, deleted or changed. The information in the shared repository (i.e. Model server) is defined in terms of a moment in time and associated versions in other moments in time, resulting in several versions and variants of a shared repository instance and discipline-specific information models. Thus, the issue of IFC model comparison becomes critical to keep track of the changes during the information exchange workflows.

Typically, the IFC model comparison is a post-rationalisation activity in a collaboration operation where the data management system (i.e. model server or another comparison tool) must deal with what has already happened on a set of data with no or limited knowledge of prerequisites about the incoming data. The comprising IFC files may include changes, application imprints and other data instances due to IFC round-tripping. The process of IFC comparison will require establishing right candidates for comparison before a comparison algorithm can be applied to determine similarities and differences. In post rationalization situations, establishing the right candidates (corresponding objects in comparing versions) for comparisons becomes difficult, for which Globally Unique Identifiers (GUIDs) are typically used in a shallow or deep tree comparison of IFC models [3]. The use of GUIDs in IFC model comparison has been heavily criticised in the literature [[4][5][5][7] as it leads to costly calculations, quality compromises and inaccuracies due to inconsistent GUIDs in IFC data round-tripping. Moreover, different BIM authoring tools may have different semantic representation for a same physical object which triggers another concern that is how to compare models which are constructed independently using different tools because the tools internal representation differs from the IFC exchanged representation

Furthermore, support is required from the client-side application when submitting changes to the model server, for example, to maintain object owner history and the consistent preservation of GUIDs. However, the internal data storage structures of client-side BIM applications tend to be different from each other, with limited support for database level change management within proprietary BIM applications for server enabled collaboration. For example, if a structural engineer decides to change the position of 4 columns, there is no predefined standard workflow for executing this change. The engineer may change each column individually, leading to four changes in the model; or may decide to change one, delete the other three and then copy and paste the first one three times (Figure 1). This will result in one change, three deletions and three new columns in a model, but ultimately the design change would be the same in both cases.

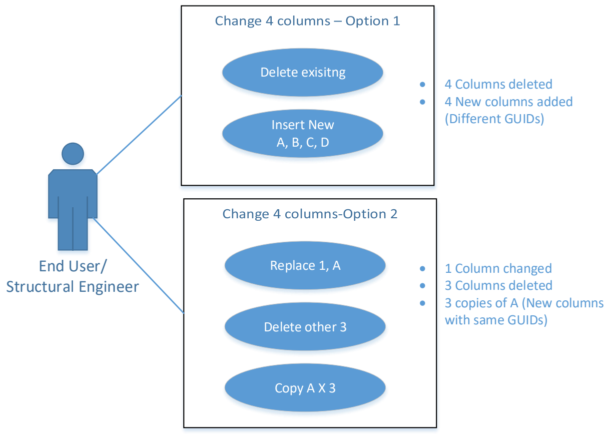
Change 4 columns – Option 1
Delete exisitng
Insert New A, B, C, D
4 Columns deleted
4 New columns added (Different GUIDs)
End User/ Structural Engineer
Change 4 columns-Option 2
Replace 1, A
Delete other 3
Copy A X 3
1 Column changed
3 Columns deleted
3 copies of A (New columns with same GUIDs)

Figure1: Use case of a structural engineer modifying columns in a shared model

In such design modifications, when exporting the latest version of an IFC model, the IFC versioning can be different, as the two editing operations could result in different GUIDs for the same model elements. If the objects' GUIDs are identical, there is clearly a match for comparison (as GUIDs are designed to be unique).

However, two objects can still constitute a comparable pair even if their GUIDs are different. Most model comparison applications (e.g. Solibri, BIMServices, or proprietary applications etc) use GUIDs to establish candidates for comparison and therefore fail if these are different for two comparable objects. Tracking comparable objects across versions is a fundamental task in managing the workflows of model server enabled collaboration on BIMs. In summary, GUIDs are helpful when tracking such changes, but there is a need to consider other characteristics, such as location (same bounding box), containment (same address), name, specification, in addition to using GUIDs for object recognition and identification in a comparison process.

## 3 Previous Work

Several authors and research projects have addressed the general issue of change management in IFC models [7]**Error! Reference source not found.Error! Reference source not found.** and only a few studies have focused on IFC model comparison strategies [11]**Error! Reference source not found.**[13][14].

Traditional plain text-based comparison tools do not consider specific data organization and representation of comparable files and therefore are not suitable for IFC Comparison. Alternatively, GUID-based comparison strategies were presented to compare IFC models [15] [13]. GUID-based strategies consider two instances in two comparing files as same if their GUIDS are matching and vice versa. [13] developed EVASYS (EXPRESS Evaluation System), one of the first tools that can calculate a number of identical instances in two comparing IFC models under EXPRESS schema. The comparison algorithm proposed in EVASYS was based on using GUIDs (Globally Unique Identifiers) to identify matching object pairs and then used a sequence of steps to filter out matching and different objects by comparing object types<instances<attribute value. GUID (globally unique identifier) is a unique identifier for object instances that follow the universally unique identifier standard (UUID) that provides a way of uniquely identifying an object. **Error! Reference source not found.** reported four weaknesses of EVASYS: (1) it ignores redundant instances; (2) it does not provide any mechanism to analyses IFC owner history; (3) it does not consider if GUIDs are changed but that the property values are maintained; and (4) it is limited to counting structural instances in comparison, ignoring semantics. In IFC models, only entities inherited from IFCRoot will have GUIDs, while many other entities (e.g. IfcPropertySingleValue) not inherited from IfcRoot have GUID **Error! Reference source not found.**. In addition, the GUIDs of instances are often changed during the data exchange between different systems even without a modification to the model itself [14]. Although GUID-based comparison approach is simple and can provide fast results but is error-prone and therefore cannot be relied on.

A graph-based approach is presented by [16] which compares two oriented graphs generated by two IFC files. Similarly, [13] proposed converting IFC file into RFD-RDF graph-based signature algorithms for computing differences of IFC models. However, the matching process in graph-based comparisons still complies with GUIDs comparison.

**Error! Reference source not found.** proposed a flattening-based comparison approach, which suggests replacing all referencing with actual values in two IFC files and then compares the flattened data instances instead of original files. The comparison process becomes simplified to a string comparison as it overcomes the differences of reference numbers included in attribute values. However, the proposed flattening process can result in quite long strings of data due to complex inheritance and reference of IFC instances. Therefore, the flattening-based approach is time-consuming and not suitable to compare large IFC files [14].

Another IFC comparison approach uses a tree-based comparison of two IFC models using the hierarchal structure of IFC to map instances of two comparing files. This approach was presented by [11] and implemented in an IFC comparison tool “BIMServices”. This approach uses IFC spatial and containment hierarchies to base model comparison as the path along model hierarchy tree will typically have incremental changes, starting from IfcProject, across different versions of the same IFC model. Also, the IFC spatial hierarchy (geometry) is a relatively reliable base across BIM authoring tools and end users, which can help better understanding of the comparison results. The bimServices tool provided a better way of comparing IFC models through a sequential pass to calculate differences and similarities by using a tree comparison. However, the identification of comparable pairs still relies on the assumption that GUIDs of two instances will match in the comparing models. More recently, [16] presented a similar tree comparison approach which uses data instances, instead of GUIDS, to establish candidates for comparison to initiate the comparison process. Their approach extracts three basic terms (i.e. instance name, entity name and attribute values from each data instances) and their referencing relationships to construct IFC file hierarchy to perform the further comparison. This approach can reduce the number of redundant instances and can provide fast comparison results without relying on the matching GUIDS. However, various BIM tools use the different naming structure for instances and can export two different names for the same object in an IFC Import/Export operation. Therefore, constructing the IFC tree comparison using instance or entity names may not lead to accurate candidates for comparison to support the rest of the comparison process.

Previous authors have recognised that in order to get a more meaningful result for IFC model comparisons an extended set of metrics need to be considered that is not solely based on GUIDs but also consider other physical, structural and semantic properties of the IFC models [6], [8]. Preservation of GUIDs is unanimously stressed by previous researchers, as accurate GUIDs can provide quick and useful comparison results. If BIM authoring tools can preserve GUID in the round trip IFC exchange, then it will simplify the IFC comparison problem to a significant degree, however, GUIDs are often lost or changed during a data exchange due to BIM authoring tool imprints or end-user modelling preferences. A fundamental problem with inaccurate GUIDs is an inaccurate selection of candidates for comparison that can lead to completely wrong results. Therefore, a more sophisticated approach is needed to establish candidates for model comparison using different model matching strategies in addition to GUIDs.

## 4 A Signature Matching Approach for IFC Models

This paper proposes a signature-matching approach to establish object identities in model matching and comparison processes. In a signature-matching approach, the identity of an object is not static but calculated dynamically from the value of its properties and attributes creating a unique signature for an object[17].

[18] defined that "the signature is the collection of values assigned to a subset of syntactic properties in the model elements". The set of values that can be used to determine a signature for an object is called "signature type" [17]. Furthermore, [18] divided signature types into three categories, which are (1) a complete signature that covers all the syntactic properties associated with a model element; (2) a partial signature that covers a certain range of element properties; and (3) a default signature that is only composed of name properties. The selection of a signature type for an object requires user configuration which can be defined using a query language. A signature can be calculated using any number of properties associated with an object that can provide a distinguisher for the object. For example, a signature for an object can be its name, type and location or a combination of these characteristics (e.g. wall, wall type, position coordinates).

A signature does not rely on precedent identity (e.g. GUIDs); therefore, it can also be applied to the models which are created independently of each other using different tools (i.e. as in the case of IFC models populated from Revit or AECOsim). However, the creation of unique object signatures needs to define a series of functions to calculate accurate signatures for model objects based on their signature types. Using the pre-defined signature types, a hash value can be computed for the relevant properties sets which are used to establish a match during the comparison process. A hash function is used to map digital data of arbitrary size to digital data of fixed size, the values returned by hash functions are called hash values or hash codes or hash keys or simply hashes. Signature matching is useful even if a complete solution is not feasible, as signature matching can be used to eliminate a significant proportion of elements to downsize the comparison criteria for any further comparison.

### 4.1 Signature Matching Application in IFC Models

This paper proposes that characteristics of IFC objects can be used to create signatures for IFC objects, which then can be used in addition to GUIDs, to effectively compare corresponding objects an IFC model comparison process. This leads to a new research question: "What should constitute an effective signature for IFC objects?". It is suggested that this can be answered by considering the structural and semantic characteristics of an IFC model and the type of changes an IFC object can undergo. Fundamentally, a change can be in (1) *an object's position,* (2) *its shape* and its (3) *properties.* Position and shape are absolute as if two objects, in a comparing process, are at the same position and contains the same geometrical shape, then these are likely to be a similar object and thus, are candidates for comparison. Therefore, the characteristics of an object related to its position and shape can be used to create a signature for object recognition. For example, in an 'IfcDoor', the 'IfcLocalPlacement' defines the local coordinate system that is referenced by all geometric representations. The three-dimensional (3D) shape of 'IfcDoor' is represented using 'SweptSolid', 'SurfaceModel', or 'Brep' to define the door geometry. Most BIM authoring tools exchange arbitrary shape extrusions in IFC, which can be used to create a unique object signature.

On the other hand, creating a signature from element properties is complex as it involves a degree of change in object properties. For example, in versions A & B of the same IFC model, change in properties can have following scenarios

Properties of object A (version 1 of a dataset) = P [A]
Properties of object B (version 2 of a dataset) = P [B]
P [A] = P [B] = No change in properties
P [A] > P [B] = some properties have been deleted
P [A] < P [B] = some properties have been added
P [A] ~ P [B] = Properties have been updated/edited

Therefore, it is important to determine what properties are important and if we can determine the degree of importance in IFC properties, then it can be used to create a signature based on the key properties. Several signatures can be created from object properties, such as (1) The total number of property count can be used as a signature; (2) A name key for all the property sets names attached to an object can be used as a signature; (3) A name key of property names can be used as a signature and (4) A property value key can be used as a signature etc. These signatures can be used as passes to determine the degree of change in a potentially matching pair in a model comparison process.

Moreover, the position and shape are a way of reflecting an object on a drawing and in a 3D presentation. Therefore, these two components are compelling candidates to create a unique object signature. However, the case with object properties is different as it involves the degree of change that needs to be incorporated into creating the signature. The following examples demonstrate the development of IFC object signatures from shape and position characteristics.

**Example 1: creating a signature from geometric representations (Shape)**

The geometry of an IFC element is defined by a shape definition (i.e. 2D, 3D body and 3D path), the IfcProductDefinitionShape & IfcLocalPlacement allowing multiple geometric representations. For example, an IfcDoor may have a profile, footprint, bounding box and several 3D body definitions at different levels of detail. There is an unlimited number of shapes within the IFC schema, but it is schema requirement that it must have at least one 3D body. Therefore, the 3D shape of an element can be a signature type as it is populated every time for an IFC element. Another aspect is bounding box representation, as an IfcBuildingElement may be represented as a bounding box, which shows the maximum extent of the body within the coordinate system established by the IfcLocalPlacement. The bounding box representation is the simplest geometric representation available. The same element shape should always be contained in a same bounding box representation. Therefore, bounding box representation & 3D shape definitions can be used as strong signature types. In simple words, if two objects occupy a similar 3D space then they are candidates for similarity checking. For example, if in two comparing datasets, DS1 = IFC model version A; and DS2 = IFC model version B

RadiusBoundingSphere signature matching

Signature Type: Partial signature
Description: A hash code signature can be calculated by the radius from the middle point that contains all the points in the *shepheId* that contains the shape of an element. MatchElement RadiusBoundingSphere. (DS1, DS2)

*>Element. RadiusBoundingSphere.hashcode.DS1 = Element. RadiusBoundingSphere.hashcode.DS2 = Filter for further analysis as potentially matching pairs*

*>>Element. RadiusBoundingSphere.hashcode.DS1 ≠ Element. RadiusBoundingSphere.hashcode.DS2 = Filter as changed object or new object*

See Figure 2 for graphical representation

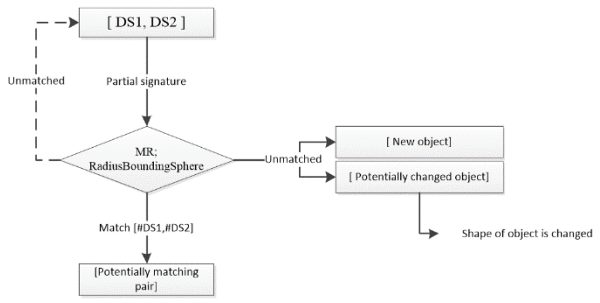

Figure 2. Matching rule for RadiusBoundingSphere signatures

A key aspect in defining shape signature type is to determine how 3D shapes are being generated from BIM authoring tools, as different BIM authoring tool may populate same 3D shape using different profile definitions, due to their internal structural differences and end-user domain applications. For example, Autodesk Revit uses arbitrary profile definitions to generate 3D shapes, whereas Tekla Structures uses parameterized profile definitions (IfcIShapeProfileDef, IfcEllipseProfileDef etc) as these profiles can create more efficient shapes related to structural elements (e.g. beam, columns etc). Therefore, shape signature is not a compound or complete signature, but it can have its contribution to reducing the size of the datasets for comparison. A shape-based partial signature is particularly important in determining the changes, rather identifying the candidates for comparison.

**Example 2: Creating signature from Local placement**

The geometry of an IFC element is defined by a shape definition and a local placement (i.e. IfcLocalPlacement). The IfcLocalPlacement defines the relative placement of a product in relation to the placement of another product or the absolute placement of a product within the geometric representation context of the project (BuildingSMART, 2014). A signature for local placement can be calculated by the X, Y & Z positions from the middle of an element (Centroid x, Centroid y & Centroid z). A matching centroid value will indicate an equivalent element with no change, whereas an unlatching centroid value will identify a change in the position of the object. For example, if in two comparing datasets, DS1 = IFC model version A & DS2 = IFC model version B

Position (Centroid) signature matching
Signature type: Partial signature
Description: A hash code that represents position signature of an element calculated from its local placement; Centroid for x, y & z

>ElementIfcLocalPlacement.Centroid. hashcode.DS1 = ElementIfcLocalPlacement.Centroid. hashcode.DS2 = Potentially matching pair, filter for further analysis, no change in position

>>ElementIfcLocalPlacement.Centroid. hashcode.DS1 ≠ ElementIfcLocalPlacement.Centroid. hashcode.DS2 = Filter as changed object; Position of the object changed

See Figure 3 for graphical representation

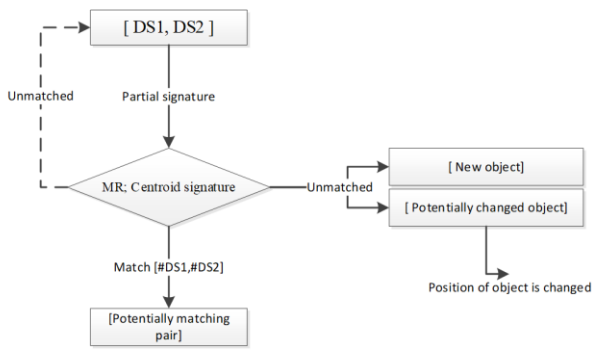

Figure 3. Matching rule for position (centroid) signatures

Likewise, the shape signature, the local placement signature cannot decide on its own for identification of corresponding element pairs in the comparison process, but it can be used in combination with other signature types, to detect potentially matching pairs or also to reduce the amount of work in calculating and identifying potential changes. If two elements have same local placement and shape, it is quite strong to say that it is a potentially equivalent pair, and can be a candidate for comparison. If the local placement coordinates are different for an element, in relation to its corresponding element in a comparable dataset, then it indicates that the element is “moved” in the updated version of the dataset, thus helps to identify the changes undergone. Similarly, signatures can be created from other IFC object characteristics (e.g. Schema type, Object Type, Name, Property Count- see Table 1) and can be used to support establish similar candidates for comparison. It is to be noted that GUID itself can be used as a signatures type as matching GUIDs would constitute a match but different GUIDs can be ignored (i.e. same objects can have different GUIDS).

## 4.2 Implementation: XBIM Signature Exporter

The proposed signature matching approach is implemented in a custom-built software tool, xBIM signature exporter, a tool that can create a unique signature using various characteristics of IFC building elements according to the proposed methodologies in this research. The xBIM signature exporter is developed on the platform of the xBIM toolkit, the eXtensible Building Information Modelling (xBIM) Toolkit, which is an open source, software development tool that supports IFC schema. The xBIM toolkit supports IFC models for geometric, topological operations and visualisation that can be used to create bespoke BIM middleware for IFC-based applications. The XBIM signature exporter was developed in a commercially funded research project; Therefore, the source code of the tool is not discussed.

The XBIM signature exporter has extracted serval signatures from the test model; however, only a few signatures are useful in IFC model comparison process. The xBIM signature exporter tool is implemented to calculate signatures for “Standard classroom” model using IGC 2.0 coordination view specifications. The test model contained an IfcDoor (1), IfcSlab (2), IfcWallStandardCase (5), IfcWindow (6) and an IfcOpeningElement . The XBIM signature exporter has extracted serval signatures from the test model using the signature types defined in Table 1. Examples of extracted signatures for “Door” and “Slab” are also presented in Table 1.

The limitations of this paper do not allow to present the test case and demonstrate the comparison results obtained using the extracted signatures. Therefore, only examples of extracted signatures are presented in Table 1 to follow the discussion.

Table 1: IFC object signatures definitions in XBIM signature exporter

| Signature Type | Description | Example of a "Door" Signature | Example of "Slab" |
|---|---|---|---|
| Model ID | A number that represents the internal ID of an element inside the model. E.g. 2552 for an IfcDoor. | 2552 | 249 |
| SchemaType | A name for a type of building element. E.g. IfcDoor for a door. | IfcDoor | IfcSlab |
| DefinedTypeId | A total number of defined types attached to an element. E.g. the value for the defined type of IfcDoor is calculated as 2539, in the test model. | 2539 | 0 |
| GUID | A fixed 22-character length string, which is associated with each element and is exchanged with the IFC exchange file structure. | 3A3Tihl61FQRUYE_9VB72L | 3A3Tihl61FQRUYE_9VB75i |
| Name | The name of an IFC element is a label in the term by which something may be referred to, i.e. IfcLabel = String (Max.255 characteristics). | IntSgl (1):1010 x 2110mm:1010 x 2110mm:195846 | Floor:Ground Bearing Concrete:195839 |
| ObjectType | The type of object of an element. | 1010 x 2110mm | Floor: Ground Bearing Concrete |
| PropertyCount | A number representing the count of total properties given to an element by the author in addition to what it would have by default (i.e. Extension properties in Revit) | 56 | 29 |
| PropertySetNamesKey | These are single value property sets that can be used to generate a signature, considering that the complex property sets are hardly ever used in real examples of IFC models | -863509959 | 1452519724 |
| PropertyNamesKey | Similarly, a hash key is created from "Names" of the properties (e.g. level, the height of offset from level etc) by sorting out names of the properties in an alphabetical order and creating a hash code signature. | 1703689443 | 1976711505 |
| PropertyValuesKey | Similarly, a signature is calculated for the values of the properties. | 570000713 | 1589767974 |
| MaterialId | This is the name of the material attached to an IFC building element. | (empty, no material assigned) | Floor: Ground Bearing Concrete |
| Centroid X | This is the position signature calculated by a x,y,z positions of the middle of an object. | -3908.871582 | -1556.871582 |
| Centroid Y | Thus, centroid determines the local placement of the object that can be used as a unique | 4958.726563 | 828.9765625 |
| Centroid Z | identification signature for the object recognition and also detect if there is any change in the object's position. | 1072.5 | -250 |
| RadiusBoundingSphere | This is a number that represents radius from a middle point that will contain all the point in the shepherd that contains the shape of an element. In simple words, it is the radius of the bounding box of the shape of an element. If the shape of an element is changed, it will change the bounding box and vice versa. Thus, it can be used as a unique signature for the shape of an object. | 1205.268555 | 6259.641113 |
| ShapeId | This is the shape ID that defines the shape of an object | -914375451 | -444593825 |

The results suggest that the schema type, GUID, position and bounding box signatures are strong signature types. Considering the volatile nature of GUIDs, schema type, position and centroid signatures provide enough evidence to establish candidates for comparison in the first pass, even if the GUIDs are not matched. After establishing matching pairs for a comparison analysis, signature matching can also help to reduce the size data for detecting fine grain changes if there are any. For example, if a matching pair has a different PropertyCount or PropertySetNamesKey, it means some properties have been added or deleted, and it should be checked for more detailed analysis. Thus, the signature types, such as NAME, Object Type, properties, can reduce the amount of workload to precisely detect changes in further analysis. It is not a brilliant way of calculating differences, but it reduces the amount of work required for further analysis.

## 5 Conclusions and Future Work

This paper describes that model matching and comparison strategies for platform-neutral models (i.e. IFC models) are the keys to solve the problem of iterative change management. A critical issue IFC model comparison is establishing right candidates for comparison, regardless of what comparison mechanical is being used. Typically, GUIDs are being used to identify corresponding objects in IFC model comparison methodologies, which is criticised for their accuracy and efficiency. This paper proposes that the characteristics of an IFC object can be used to create unique signatures for IFC elements, in addition to GUIDs, to effectively compare corresponding objects in a model comparison process. An object's position, shape and properties can be used to formulate partial, default or complete signature for an object, which can be used as passes to establish candidates for comparison and to highlight changes in a comparable object pair. This approach is useful in establishing accurate candidates for comparison without GUID dependence and can reduce the workload of the overall model comparison process.

In terms of creating an element's signature, there are several issues that need to be considered in the broader perspective of change management in a model server environment. This research has only developed signatures for IfcBuildingElements as it has the highest importance for an end-user from a change management prospectus. Even for IfcBuildingElement, the research has defined signatures for IFC object characteristics which are largely optional and may have different population if models are created using heterogeneous applications. In the context of creating unique signatures from these optional attributes, such questions need to be answered that what level of utilisation these optional properties are in the IFC models? And how likely these are to be found in real-world examples? Potentially, a subset of data can be derived, with the statistical reasoning that some characteristics, relationships and properties of IFC objects can be neglected or be focused on in creating object signatures, depending upon the usability proof from a wider and real-world dataset. This abstraction would require further empirical evidence from the analysis of real-world IFC models, reflecting a cross-sectional view of different BIM authoring tools that are used in creating IFC models. A critical issue is to consider associated weighting of different signature-type attributes of IFC objects to constitute an effective signature for IFC objects.

This study is limited to using optional object properties to create a property-based signature (such as PropertyCount, PropertyValuesKey), however, these optional properties can be referring to the same object but on a different Level of Detail (LOD). Therefore, Future research is needed to explore property signatures in more details, considering associated Level of Details and degree of change in an object's properties. Future research will explore the application of the proposed signature-based matching strategy on IFC models with different level of details and complexity within a model server enabled collaboration environment.

## 6 References

[1] Plume, J., & Mitchell, J. Collaborative design using a shared IFC building model—Learning from experience. Automation in Construction. 6 (1): 28-36, 2007.

[2] Singh, V., Gu, N., & Wang, X. A theoretical framework of a BIM-based multi-disciplinary collaboration platform. Automation in Construction, 20(2):134–144, 2011.

[3] Nour, M. Manipulating IFC sub-models in collaborative teamwork environments. In Proc. of the 24th CIB W-78 Conference on Information Technology in Construction, Pages 26–29, Maribor, Slovenia. 2007

[4] Hjelseth, E., & Nisbet, N. Overview of concepts for model checking. Proceedings of the CIB W78: 27th International Conference –Cairo, Egypt. 2010

[5] Kiviniemi, A., Fischer, M., & Bazjanac, V. Multi-model Environment: Links between Objects in Different Building Models. In Proceedings to the 22nd Conference on Information Technology in Construction CIB W78, Dresden, Germany. 2005.

[6] Nour, M., & Beucke, K. An Open Platform for Processing IFC Model Versions. In Tsinghua

Science & Technology, 13; 126-131. 2008.

[7] Weise, M., Katranuschkov, P., & Scherer, R.J. Generalised model subset definition schema. Proc. CIB-W78 Conf. 2003—Information Technology for Construction. Auckland, New Zeeland, 2003.

[8] Liebich, T., Weise, M., Laine, T., & Jokela, M. InPro Building Information Model, European Commission 6th Framework Programme, Public Report of Deliverable 19, 2010.

[9] Nour, M., and Beucke, K. Object versioning as a basis for design change management within a BIM context. International Conference on Computing in Civil and Building Engineering, W. Tizani (ed.), (ICCCBE-XIII), Nottingham, UK.2010

[10] Weise, M., Katranuschkov, P., Scherer, R.. Generic Services for the Support of Evolving Building Model Data. in Proceedings of the Xth International Conference on Computing in Civil and Building Engineering, Weimar, Germany, 2004.

[11] East, E., Nisbet, N., & Wix, J. . Lightweight Capture of As-Built Construction Information. In: 26th International Conference on IT in Construction, 2009, Istanbul, Turkey.2009

[12] G. Lee, J. Won, S. Ham, Y. Shin, Metrics for quantifying the similarities and differences between IFC files, Journal of Computing in Civil Engineering 25 (2); 172–181. 2011

[13] Ma, H., Ha, K., Chung, C., & Amor, R. Testing semantic interoperability. Interoperability, Proceedings of the Joint International Conference on Computing and Decision Making in Civil and Building Engineering. Montreal, Canada. 2006

[14] Shi, X., Liu, Y. S., Gao, G., Gu, M., & Li, H. IFCdiff: A content-based automatic comparison approach for IFC files. Automation in Construction, 86; 53–68. 2018

[15] Jeong, Y. S., Eastman, C. M., Sacks, R., & Kaner, I. Benchmark tests for BIM data exchanges of precast concrete. Automation in Construction, 18(4); 469–484. 2009

[16] Arthaud, G., & Lombardo, J. C. Automatic Semantic Comparison of STEP Product Models. Jos P. van Leeuwen and Harry J.P. Timmermans (eds.) Innovations in Design & Decision Support Systems in Architecture and Urban Planning, 447-463. Springer. 2006

[17] Reddy, R., & France, R. (2005). Model Composition; A Signature-Based Approach. In AOM Workshop, 2005. Online: https://pdfs.semanticscholar.org/1cd6/de7807b59d5d7d82f323adc88d26753c74a0.pdf; 08/03/2018

[18] Oliveira, K., & Breitman, K. A Flexible Strategy-Based Model Comparison Approach: Bridging the Syntactic and Semantic Gap, 15(11); 2225–2253. 2009.