EPiC Series in Education Science

Volume 1, 2017, Pages 316–323

AUBEA 2017: Australasian Universities Building Education Association Conference 2017

Towards Managing Iterative Changes in BIM Collaboration Workflows

Dr Muhammad Tariq Shafiq[1] and Prof. Steve Lockley [2]

[1]Assistant Professor, UAE University, UAE

[2]Professor of Building Modelling, Northumbria University, UK

Muhammad.tariq@uaeu.ac.ae

Abstract

Collaboration on Building Information Models (BIMs) requires iterative and distributed processes that make maximum reuse of the information being exchanged directly between models in a platform independent model collaboration environment. 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. This paper highlights the overall problem of managing iterative changes in BIMs and discusses various issues and challenges involved in controlling the collaboration transactions on a BIM data repository in a multi-model collaboration environment. This positional paper describes that model matching and comparison strategies are the keys to solve the problem of iterative change management, which may have better solutions in other knowledge domain such as software engineering. The future research is exploring Software Source Control (SSC) strategies to devise a signature-based model comparison approach for IFC models that can lead to potential solutions for effective management of collaboration operations with BIMs.

1 Introduction

The technology to efficiently create Building Information Models is now mature and can support the development of discipline-specific building information models; however, the technology to collaborate on models is still developing and growing. Collaboration on BIM models involves iterative and distributed processes that make maximum reuse of the information being exchanged directly between models in a model collaboration environment. BIMs are subject to constant evolution, where collaboration requires information to be created, coordinated and exchanged concurrently and most often in real time, allowing multiple users to manipulate information whilst requiring the data to be

synchronised in a shared data repository. 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 as a result of the collaboration operations and workflows that involve various project participants and heterogeneous applications. Emerging practice indicates that the aspirational single project model in a collaborative BIM environment is not usually a single database structuring all the related information but a combination of tightly and loosely coupled databases (federation) linked with clear rules and allowing controlled access to the different parts of the information available in the federated model for collaboration operations. A central issue to the management of this federation is an understanding of the concurrent and iterative processes required to support and execute tasks involved in the operations of a collaborative BIM environment.

2 Limitations of Current BIM Collaboration Practices

Collaboration on Building Information Models (BIMs) is currently undertaken through file-based exchanges, using a variety of methods, such as physical file transfer, extranets, project websites and proprietary collaboration tools (Isikdag et al. , 2007; Shafiq et al., 2012). A file-based collaboration on BIM models have several limitations (Adachi, 2001; Hietanen, 2002; Kiviniemi et al., 2005; Beetz et al., 2010), and thus offer limited opportunities to manipulate the data (Shafiq et al., 2012).

- Differences in the internal storage structure of BIM authoring platforms (e.g. Autodesk Revit, ArchiCAD) make it difficult to maintain the integrity of information in models when shared between applications. Even if the exchanges are in a platform- neutral file format, such as Industry Foundation Classes (IFC), native BIM authoring applications leave application imprints on the data. Moreover, the addition or loss of data during an IFC export may result in models becoming less interpretable once imported into other applications.
- Redundant data may occur in different versions in file-based model exchanges leading to data duplication and rework.
- As the information content grows within a BIM, its size significantly increases, which results in it becoming difficult to transfer through a file exchange mechanism.
- In many cases, only a partial view of the model is required to be exchanged or viewed due to data ownership and liability issues, which is difficult to manage through file based exchange.
- Maintaining proper versioning of objects is impractical in ad-hoc file based model exchanges. Controlling user rights, ownership and responsibility of the model's contents become compromised in a file-based information exchange.

Therefore, use of file-based exchanges of data to collaborate on BIM models is not a long-term solution to facilitate acclaimed benefits of using Building Information Modelling. In addressing this issue, the concept of model servers was introduced as a potential solution to improve the workflow and stimulate collaboration on BIMs. However, research has been limited to date in this fertile and emergent area.

3 BIM COLLABORATION AND MODEL SERVERS

The technology that can support database level exchanges is generally referred to as 'model servers', which can exploit and reuse information directly from a shared model repository and facilitate

collaboration among any number of participants. Plume and Mitchell (2007) and Jørgensen et al. (2008) define model server as 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. Attempts to develop model servers for the construction industry started alongside the development of Industry Foundation Class (IFC) schema (Adachi, 2001, 2002). The IFC schema is independent of mechanisms or tools used to generate data; its focus is to represent the core data of building components as object models. Thus, for database level transactions, the complete 'IFC data model' can be used to underpin a 'model server', which has been referred to as an 'IFC model server' (Adachi, 2002; Hietanen, 2002).

A BIM hosting 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 (Singh, Gu, & Wang, 2011). The potential of 'model servers' coupled with web-based technologies for effectively enabling information to be shared in a collaborative environment has been demonstrated (e.g., Kiviniemi et al., 2005; Jørgensen et al., 2008; Beetz et al., 2010). A considerable amount of research has been undertaken to develop model server capabilities through the use of the Standard for the Exchange of Products (STEP), IFC, as well as proprietary data formats, resulting in products such as IMSvr, SABLE, Express Data Manager (EDM), Share a Space, Activefacility and BIMserver (Adachi, 2002; SABLE, 2003; Beetz et al., 2010; Shafiq et al., 2012).

Model servers are the backbone technology that can enable the realisation of effective collaboration on BIMs, though they are still not technically mature. A platform independent model server should allow different discipline BIM applications to exchange data using IFC, and provide collaboration functionalities to enable a model to be uploaded/downloaded, viewed, split, merged, and compared. However, the collaboration workflows during model server transactions result in the creation of complex data structures, the management of which has several unresolved challenges, therefore limiting the practical application for end users (Kiviniemi et al., 2005; Koch and Firmenich, 2011). Consequently, there has been a tendency for them to be used within experimental and academic environments, due to a number of latent 'pitfalls', such as IFC export/import quality, access control and change management etc (Hjelseth and Nisbet, 2010; Jørgensen et al., 2008; Kiviniemi et al., 2005; Kiviniemi, 2006b; Singh et al., 2011).These problems are not solely attributable to the limitations of the technology but the inherent structure (e.g., procurement methods and contractual arrangements) and complex works flows and procedures that have been designed to deliver projects in a linear workflow.

4 Iterative Change Management in a model server workflow

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. The changes in these versions can be; (1) technical changes, such as selection of a design alternative; (2) modifications and detailing of objects as the design process precedes; (3) changes caused by data round-tripping, such as IFC import/export. Management of these changes, (1) & (2), involves complex workflows supported by computing

operations which enable a model server to handle simple and complex transitions. Apart from the technical challenges, there are user issues that need to be considered while managing iterative changes, however, this is outside the scope of this research.

The analysis, interrogation and collaboration of BIMs is generally reliant on the use of Globally Unique Identifiers (GUIDs) or arbitrary geometry representations, however, there is a tendency for both to fail in the complex situations arising as a result of simple and complex transactions in a model server environment (Liebich et al., 2010). 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.

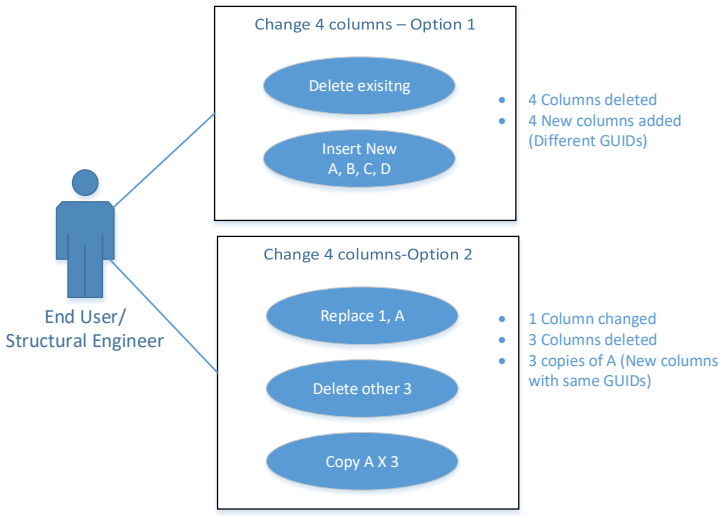

Figure 1: Use Case of a structural Engineer modifying columns

In similar design modifications, as represented in the use-case example in Figure 1, when exporting the latest version of an IFC model, the IFC versioning can be different as well as the associated GUIDS, leading to two different comparison results under the same editing option.

If the objects' GUIDs are identical, there is clearly a match for comparison or merging (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, such as Solibri or ArchiCAD, use GUIDs to establish candidates for comparison and therefore fail if they are different for two comparable objects. Tracking comparable objects across versions is the most basic operation in managing the workflows of model server enabled collaboration on BIMs. GUIDs are helpful when tracking such changes, but there

is a need to consider other characteristics in addition, such as location (same bounding box), containment (same address), name, specification, or function of the objects to perform a more meaningful comparison.

Also, when managing changes in model server environments, it is important to understand what types of changes are expected in two versions that are being subjected to a comparison process. In addition to obtaining an accurate comparison result, the process should minimise a number of unwanted change notifications to the end users. Model comparison is performed at an object level but the end users are typically concerned about the effectiveness of the outcome. For example, when a cost engineer runs a model comparison, if there isn't any change that affects cost information, then he or she may not be interested to know anything else that is different between the two versions of the model.

So, in terms of managing changes in IFC models, it is important to understand the scope of changes at an object level, but also to consider the end user requirements (e.g. comparing heating systems only), otherwise, the comparison process may produce accurate, but redundant results for the end users. A practical way of thinking about change management in the construction industry is linking a change to its subsequent actions, such as a design change leading to a new set of drawings or a specification change triggering a 'Request for Information'. These changes can affect the objects inside a model by changing their position, shape, and properties. If the position and shape of an object are changed, then it is a visible change and so can be communicated to an end-user through a graphical representation (i.e. a door has been moved from wall A to B).

5 Creating Signatures for IFC Objects: A Way Forward

5.1 Signature matching in Software Source Control

As argued in the previous sections, Model comparison strategies in IFC model comparison are heavily based on GUIDs, which have some limitations in providing accurate results. This comparison mechanism is called "static ID based matching approach" that suggests that each object in two comparing models has been assigned a unique identity upon its creation, which is a persistent and a non-volatile unique identifier for an object. This unique ID is referred as GUIDs or Universally Matching Unique Identifier (UUIDs) in software engineering. Software source control (SSC) is a similar process in software engineering that facilitates multiple-user database level collaboration in an environment like model servers. However, in SSC, comparison of two potentially matching objects is not limited to using a static ID based matching algorithm. In addition, a signature based, a similarity-based or a language specific matching algorithms can also be used to address inherent limitations of a static ID based matching and comparison algorithm (Kolovos et al, 2009). This study has analysed SSC comparison algorithms and has identified that a signature based matching approach can be applied to IFC model matching and comparison in model server collaboration transactions.

Reddy & France (2005) proposed a signature based matching approach in which 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. Oliveira & Breitman, (2009) 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" (Reddy & France, 2005). Furthermore, Oliveira & Breitman (2009) divided signature types intro 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 of an object does not rely on a 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 requires developer and end-user involvement to define a series of functions to calculate accurate signatures for model objects based on their signature-types. Once a user defines the set of values of signature calculation, 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 (Carroll & Krotoski, 2014). Signature matching is useful even if a complete solution is not feasible with signature matching, as signature matching can be used to eliminate a significant proportion of elements to downsize the comparison criteria for any further comparison.

5.2 Signature matching application in IFC models

The characteristics of objects can be used to create signatures for IFC objects, which then can be used in addition to GUIDs, to effectively compare corresponding objects in change management 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 any change in these components will require a significant change. Therefore, the characteristics of an object related to its position and shape provide an appropriate mechanism for creating recognition signatures. 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 'Prep' 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.

Creating a signature from element properties is complex, as properties have a degree of change and how that change is important in term of a comparison result and in the context of change management from an end user's perspective.
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.

In summary, the position and shape are a way of reflecting an object on a drawing and in a 3D model, which is easy to reflect an end user if there is any change. 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.

6 Limitations and Future Work

This study has proposed that the characteristics of an IFC object can be used to create unique signatures for IFC objects, can be used in addition to GUIDs, to effectively compare corresponding objects in change management process in model server collaboration transactions. 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.

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. A change or update in an object's properties can have multiple impacts, for example, it can affect its identity, position, shape, representation, subsequent drawings and specifications or both for the end user. As previously noted, the position and shape related properties of an element are strong candidates for a signature but the those remaining also need to be examined and cannot be ignored if effective change management and productivity improvements are to be attained, particularly during the model comparison process. Also, a critical issue is to consider is associated weighting of different signature-type attributes of IFC objects to constitute an effective signature for IFC objects. Future research will focus on creating object signatures using hash codes from IFC object characteristics, to formulate an object recognition and comparison strategy in managing model server enabled collaboration on BIMs.

References

Adachi, Y. (2001). Introduction of IFC model server. Finland: SECOM Co., Ltd./VTT Building and Transport. Retrieved May 3, 2013.

Adachi, Y. (2002). Brief of IFC Model Server Project. Transport, 2002–2002.

Beetz, J., de Laat, R., van Berlo, L., & van den Helm, P. (2010). Towards an Open Building Information Model Server. Bimserver.org, 1–8. Retrieved from http://bimserver.org/wp-content/uploads/2010/11/Beetz-Berlo_ddss2010.pdfz

Hietanen, J. (2002). BLIS Review : IMSvr Why a model server ? Which IFC releases are supported ? Why should I use IMSvr Can I still exchange IFC files ? Retrieved from http://www.blis-project.org/software/reviews/IMSvr_Review.pdf

Hjelseth, E., & Nisbet, N. (2010). Overview of concepts for model checking. In Life Sciences (pp. 16–18). Retrieved from http://itc.scix.net/data/works/att/w78-2010-53.pdf

Isikdag, U., Aouad, G., Underwood, J., & Wu, S. (2007). Building information models: a review on storage and exchange mechanisms. Bringing ITC Knowledge to Work. Salford, UK.

Jørgensen, K. A., Skauge, J., Christiansson, P., Svidt, K., Sørensen, K. B., & Mitchell, J. (2008). Use of IFC Model Servers-Modelling Collaboration Possibilities in Practice. differences. Retrieved from http://vbn.aau.dk/files/14804119/ReportIfcModelServer-Final.pdf

Kiviniemi, A. (2006). Ten Years of IFC Development Why are we not yet there ? International Alliance for Interoperability. In In Keynote lecture at the 2006 Joint International Conference on Computing and Decision Making in Civil and Building Engineering. Montreal, Canada.

Kiviniemi, A., Fischer, M., & Bazjanac, V. (2005). Integration of multiple product models: Ifc model servers as a potential solution. In Proceedings to the 22nd Conference on Information Technology in Construction CIB W78,. Dresden, Germany.

Kiviniemi, A., Fischer, M., & Bazjanac, V. (2005). 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.

Koch, C., & Firmenich, B. (2011). An approach to distributed building modeling on the basis of versions and changes. Advanced Engineering Informatics, 25(2), 297–310.

Kolovos, D. S., Pierantonio, A., Informatica, D., Ruscio, D. Di, & Paige, R. F. (2009). Different Models for Model Matching : An analysis of approaches to support model differencing. In Proceedings of the ICSE Workshop on Comparison and Versioning of Software Models Pages 1-6. Washington DC.

Liebich, T., Weise, M., Laine, T., & Jokela, M. (2010). InPro (2006-2010) "Open Information Environment for knowledge-based collaborative processes throughout the lifecycle of a building", various public deliverables, available at http://www.inpro-project.eu (visited

Oliveira, K. S., Breitman, K. K., & de Oliveira, T. C. (2009). A Flexible Strategy-Based Model Comparison Approach: Bridging the Syntactic and Semantic Gap. J. UCS, 15(11), 2225-2253.

Plume, J., & Mitchell, J. (2007). Collaborative design using a shared IFC building model—Learning from experience. Automation in Construction, 16(1), 28–36.

Reddy, R., & France, R. (2005). Model Composition - A Signature-Based Approach. In In AOM Workshop, 2005.

Shafiq, M. T., Matthews, J., & Lockley, S. R. (2012). Requirements for model server enabled collaborating on building information models. International Journal of 3-D Information Modeling (IJ3DIM), 1(4), 8-17.

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