Front. Eng. Manag. 2018, 5(3): 298–306
https://doi.org/10.15302/J-FEM-2018009

RESEARCH ARTICLE

Muhammad Tariq SHAFIQ, Jane MATTHEWS, Steve LOCKLEY, Peter E.D. LOVE

Model server enabled management of collaborative changes in building information models

Abstract The issues and challenges involved in controlling the collaborative changes in a Building Information Modeling (BIM) data repository, in a multi-model collaboration environment, are discussed. It is suggested that managing iterative changes in BIMs is a database problem, exacerbated by the long transaction times needed to support collaborative design progression. This is yet to be resolved in the construction industry and better solutions are needed to support the underlying workflows and computing operations for seamless collaboration on BIMs. With this in mind, this paper proposes the use of the structural and semantic characteristics of BIM objects as a mechanism for tracking changes across co-developed solutions. The creation of object signatures, using hash codes derived from their characteristics, provides a potential mechanism for object comparison and effective change recognition and management.

Keywords building information models, collaboration, IFC, change management, model-servers

Received January 18, 2018; accepted June 15, 2018

Muhammad Tariq SHAFIQ
Department of Architectural Engineering, University of the United Arab Emirates, UAE

Jane MATTHEWS
Department of Construction Management, Curtin University, Perth, Western Australia 6845, Australia

Steve LOCKLEY
Department of Architecture and Built Environment, Northumbria University, Northumbria, United Kingdom

Peter E.D. LOVE (✉)
Department of Civil Engineering, Curtin University, Perth, Western Australia 6845, Australia
E-mail: plove@iinet.net.au

1 Introduction

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). The requirements for BIM enabled collaboration which requires concurrent design, are currently hindered by this reliance on the existing file based tools (Shafiq et al., 2012), which have a number of limitations (Adachi, 2001; Hietanen, 2002; Kiviniemi et al., 2005a, b; Beetz et al., 2010), for example:

- differences in the internal storage structure of BIM authoring platforms 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. For example, it may be necessary for internal walls to live both in the Architects model as well as the structural engineers model;
- 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;
- versioning of individual objects within a BIM is not possible in ad hoc file based model exchanges; and
- controlling user rights, ownership and responsibility of the model's contents becomes compromised in file-based information exchange.

Given the above limitations, the use of file-based

exchanges of data are not viable as a long-term solution to collaboration on BIMs. Database level transactions have potential for achieving more sophisticated collaboration, but the practical realization and the technology to facilitate such transactions remain unsolved (Shafiq et al., 2012). In addressing this issue, the concept of model servers has been identified as having potential to improve the workflow and stimulate collaboration on BIMs. However, research has been limited to date in this fertile and emergent area. With this in mind, this paper provides a commentary on the research undertaken to date, specifically identifying problem areas, and proposing a way forward to a support increasingly sophisticated collaboration on BIMs.

2 Model server

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 servers as a type of database system that allow 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 from the 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). The architecture of an IFC model server is presented in Fig. 1.

A BIM hosting model server is expected to facilitate the exchange of information between the applications used throughout a project's lifecycle (e.g., design tools, analysis tools, document management systems, facility management tools) (Singh et al., 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; Beetz et al., 2010; Shafiq et al., 2012). 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).

Model servers are the backbone technology that can enable the realization of effective collaboration on BIMs, though they are still not technically mature. Consequently, there has been a tendency for them to be used within experimental and academic environments, due to a number of latent 'pitfalls' which include (Kiviniemi et al., 2005; Kiviniemi, 2006; Jørgensen et al., 2008; Hjelseth and Nisbet, 2010; Singh et al., 2011):

- transactions on model data repositories (e.g. upload or download models) are typically made via the Internet, and thus as the size of models issued increases, performance problems arise particularly in large scale projects;
- data locking for 'check-out' or 'check-in' procedures is a critical model server function required to control concurrent access and repository update operations. However, it imposes serious limitations on the modeling process, creates bottlenecks for different disciplines, and therefore adversely impacts productivity;
- restrictions on access rights and work permissions are necessary to maintain ownership and liability; while there is adequate provision to support this at the model level, at the object level of a model, however, there exists no standardized methods to facilitate and transfer such controls. Moreover, users' roles are not static, but switch with changing responsibilities in accordance with different stages of a project. In large scale projects, the roles and responsibilities are not only complex but often overlap. Thus, it becomes difficult to control user rights, access permissions, model data ownership and responsibility in large scale model collaboration environments;
- the quality of IFC export and import in BIM tools is inconsistent, which renders it difficult to interpret data consistently in a shared repository, particularly if it is generated from heterogeneous applications.
- the user interfaces of existing model servers are complex and confusing for the majority of users in a project. Invariably users are used to working with a number of Computer Aided Design (CAD) tools that vary depending on each discipline's preference and patterns of usage. The user interfaces of existing model server solutions are static and non-intuitive, which imposes yet another learning curve in order to use and adopt model server enabled collaboration systems.
- in model collaboration workflows, the change management of evolving content becomes very complex as a result of iterative changes and the creation of duplicate objects when concurrent modeling is undertaken. Fre-

quently, the same object is created or used in multiple discipline models that exist on the shared data repository and subsequently need to be reintegrated to avoid data redundancy and duplication. This involves accurately splitting, merging, comparing, and checking the model and its functions (Shafiq et al., 2013).

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. However, it is outside the scope of this paper to examine the non-technological issues that can hinder collaboration in construction projects. For a detailed review of the need for collaboration on construction projects, particularly from a structural, organizational, and cultural perspectives refer to Love and Gunasekaran (1997a, b), Holt et al., (2000), and Lloyd-Walker and Walker (2015).

Collaborative operations in a project requires data synchronization and management through long transactions with a model server (Weise and Katranuschkov, 2004; Nour, 2007; Counsell, 2012). A long transaction is an extended editing operation of data, independent from the shared data repository, which allows offline modifications and supports parallel working by involving versioning and merging of concurrently changed data. The shared data repository on the server must be updated with any changes as a result of modifications made in a long transaction, such as new data instances added, deleted or changed. This is the most important stage in managing changes in a model collaboration workflow as it involves a number of model server functions that enable comparisons, versioning, merging and checking to be undertaken.

Timing is critical when managing a collaboration workflow; if two events occur at different points in time, then meaningful change management cannot be achieved unless a model server is able to detect what has happened between these periods. Thus, the issue of model comparison becomes critical in managing changes for server enabled collaborative working with BIMs (Shafiq et al., 2013).

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 long transactions (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. A number of studies have addressed such technical issues, but the understanding of model server workflows and related technical solutions remains immature (e.g., Weise and Katranuschkov, 2004; Nour, 2007; Beetz et al., 2010; Hjelseth and Nisbet, 2010; Liebich et al., 2010).

3 Iterative change management workflow of a model server collaboration

The workflow in a model server enabled BIM collaboration relies on an iterative bi-directional exchange of data (i.e. 'round trip') in a long transaction (i.e. independent from the server, at a local workstation, over days not milliseconds). This includes (1) distribution/download of the data (i.e. data check out); (2) local data manipulation according to the user intent (i.e. model modification); and (3) uploading changes to the shared data repository on the model server (i.e. data check in), as reflected in the Fig. 2.

Figure 2 represents a typical workflow of a shared data repository (M), exchanging a neutral subset model (M_i) in a long transaction, which is imported into an end user BIM tool(S_i), modified (S_{i+1}) and then exported back into neutral data format (M_i+1) . The difference of M_i and M_{i+1} is the change set which is to be re-integrated with the shared repository to complete the workflow. This workflow assumes that the user is importing/exporting an IFC model in/out of a BIM authoring program, and not just editing the IFC via a text editor. The change management workflow in a model server transactions can be explained in three steps (Liebich et al., 2010): (1) Takeover of data from a model server; (2) Interpretation and manipulation of data and; (3) Update of changes to the model server.

3.1 Takeover of data from a model server

The workflow commences with the download/check out of the required data to execute a design/collaboration task (such as input for the HVAC design development). It has been established that a complete copy of the design data are often not required (Lockley et al., 1994; Sun and Lockley, 1997). Thus, this step often includes determining a subset or partial model from the data repository, which can satisfy the information requirement for the task. This is a separate research issue; it may be or may not be in the scope of a model server to determine and validate the information exchange requirements for a task. Once a piece of data are checked out from a model server it will be changed or modified on a local machine typically in a long transaction, (i.e. independent from the model server repository). Model server functions need to apply a change management or concurrency control (e.g. version management) policy to avoid data duplication and parallel work conflicts (such as data locking for an exclusive check out), which will help to manage data changes or updates to the shared data repository in the later steps.

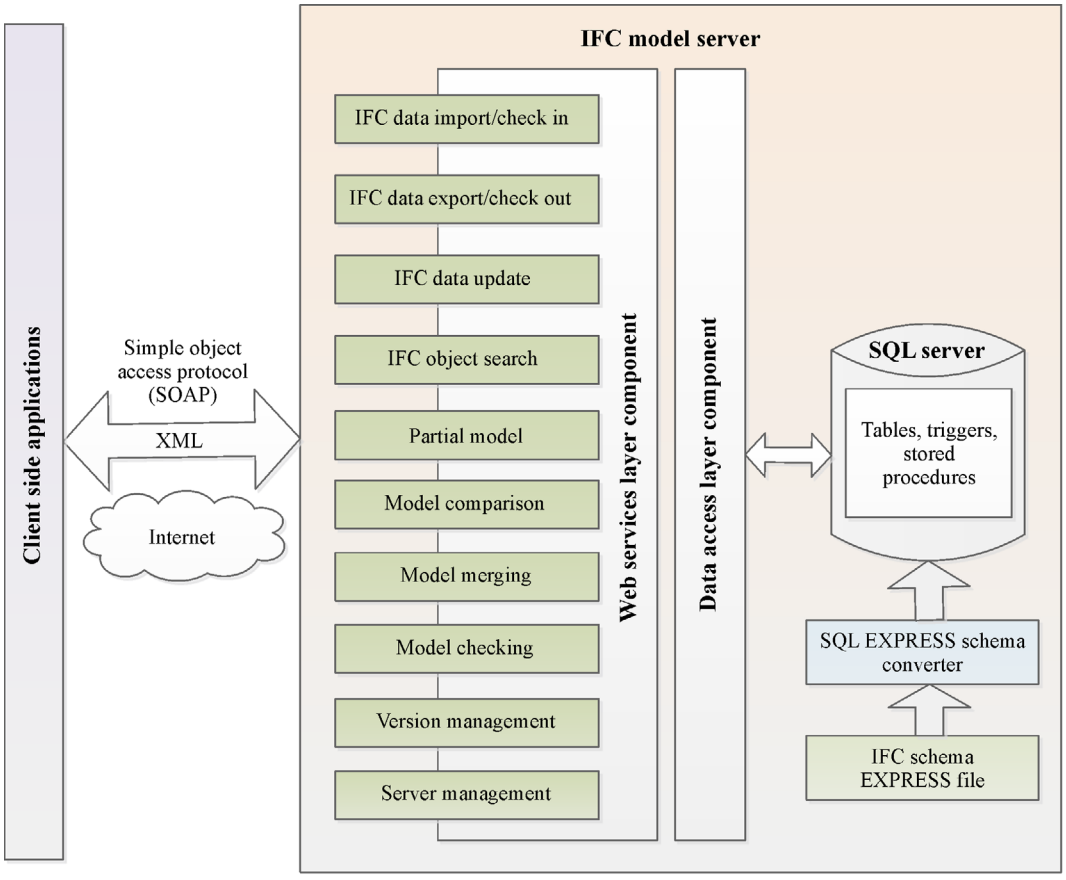

Fig. 1 Architecture of an 'IFC model server'

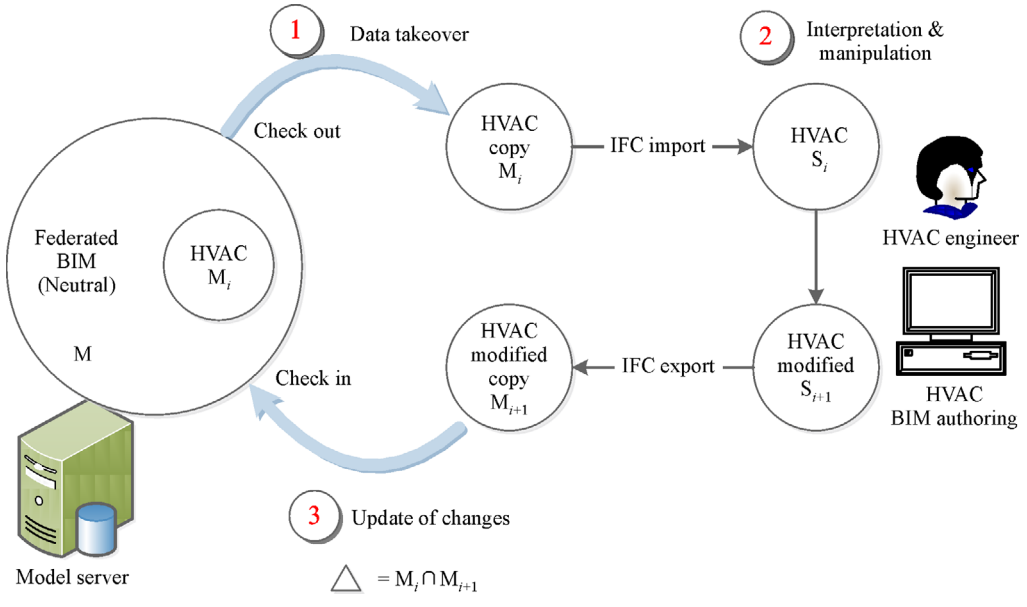

Fig. 2 Iterative cycle of data exchanges for model server enabled collaboration (Adapted from Liebich et al., 2010)

3.2 Interpretation and manipulation of data

Even with the use of an IFC model server (neutral shared data repository), it is not practical to consider all data, on the shared repository and on the client side, with the same level of sophistication and semantic understanding. The client-side applications need to transfer the IFC data into its native proprietary format for discipline specific tasks. The problems surrounding interoperability of neutral and proprietary data are well known, but have only been partially solved (e.g., Shafiq et al., 2012; Shafiq et al., 2013). Thus, data interpretation and manipulation on the client side often result in data loss, inaccuracies and unexpected changes, thus making the change management process even more complicated. Liebich et al. (2010) have suggested that when dealing with 'round trip' change management scenarios, a reliable basis for data management is the meaning of data and not the data itself. However, the gap in determining data meaning and data itself is the most challenging task in future BIM developments.

3.3 Update of changes to the shared data on a model server

The final step in the model server data 'round trip' is the synchronization of changes or updates with the shared data repository as a result of a long transaction (e.g. instances added, deleted, replaced, edited etc.). The success of change management depends on the quality of data produced by the client-side applications, irrespective of data loss or inconsistencies due to the transfer between IFC and native data formats. However, both the existing client side applications (BIM authoring tools), and current model collaboration systems typically provide poor or no support for advance data management features. Most BIM tools generate new model instances in a new version that contains old and new data, which cannot be used to replace original source data on the repository as these instances only represent a part of the whole repository (i.e., partial models). Therefore, a model server has to be capable of dealing with a number of changed partial models, and their versions and variants, as the result of an iterative design development process. In this instance, a model server must identify what has changed between two versions of the same data using a comparison strategy, and then apply a merging mechanism to update the changes back into the shared repository.

These two aspects, (1) model comparison to match two successive model versions to recognize the latest modified data (2) merging/update of concurrently made data changes, are critical for the success of the whole process of model server enabled collaboration. In the workflow represented in Fig. 2, if the whole repository can be exchanged (depending on the size and capacity of applications involved), step 1 can also be avoided as there will be no need for any subset data extraction. However, in all cases model comparison and merging will be required to manage changes for meaningful information exchange and for the consistency of the shared data repository.

4 Complexities in model server workflows

The most critical aspect of model-based collaboration is controlling changes in concurrent workflows. This entails all the complexities and technical challenges of effective data management detailed above, making the ability for meaningful model comparison essential to the success of model-based collaboration.

4.1 Validation of federated models

As stated a model server needs to deal with the concurrency of the collaboration operations as well as with the structural and semantic integrity of the data. Ideally, an IFC enabled model server will be hosting IFC data from multiple sources in a federated database. A federated BIM database consist of a number of linked but distinct models which retain their integrity so that a change in one component in one model will not affect the components in the other linked models (Lowe and Muncey, 2009). The contents of this federated database need to be validated for semantic and structural accuracy so that the changes or updates in the modeling environment do not violate the consistency of shared BIM data on the model server. The semantic correctness of modeling content is very difficult to validate due to the heterogeneous tools involved in the creation of the federated content. Most current tools focus on geometry integration of federated BIM content, but have generally failed to maintain the complete structural and semantic correctness in IFC round tripping (Van Berlo et al., 2012; Lockley et al., 2013). The result of this situation is a lack of assurance that the data in a federated BIM is free from faults and errors, leading to data interpretation that delivers incorrect results.

4.2 Post rationalization versus pre-defined collaboration

The core of the change management workflow in a model server environment is the process of finding the changes and updates (model comparison) and merging them back with the shared data repository. These are often post-rationalization activities in the collaboration workflow where data management systems (i.e. model servers) have to deal with what has already happened on a set of data with no or limited knowledge of any conditions applied to the incoming data. In post rationalization situations, establishing the right characteristics for determining change is a challenging process (for example, comparing corresponding objects in different model versions). For

this, GUIDs are typically used in a shallow or deep tree comparison of IFC models (Nour, 2007). Yet, this approach has been widely criticized due to the inconsistent persistence of GUIDS (Weise and Katranuschkov, 2004; Kiviniemi et al., 2005; Nour and Beucke, 2008; Hjelseth and Nisbet, 2010). These resulting inconsistent, missing, or replaced GUIDs can potentially lead to costly calculations, and quality compromises in a round trip data exchange.

An alternative approach is to use pre-defined protocols to manage changes and updates in model server collaborations (Liebich et al., 2010). In a pre-defined approach, model server workflows only use a pre-selected set of data, and its versions, which are assigned with predefined authority and responsibility for any changes. This approach is alleged to eliminate potential conflicts when integrating changes back into the shared repository by applying restrictions on unauthorised and unwanted changes. Liebich et al. (2010) advocated this approach and suggested splitting the possible changes into three categories, which are:

1) Invariant data: A type of data that is not allowed to change during a round trip exchange.

2) Read-only data: A type of data that is not expected to be updated, thus issued as read-only.

3) Changeable data: A type of data that can be changed and that is allowed to be updated in the shared repository.

Applying such restrictions to the data before starting a data exchange cycle significantly decreases the amount of costly calculations in model server workflows. However, predefined collaboration operations require more sophisticated control mechanisms, which are absent from current practice. The transfer from post rationalization to predefined collaboration operations has the potential to greatly improve the accuracy and quality of data collaboration in a model server enabled environment.

4.3 Inconsistent version control

A prerequisite of an effective model collaboration process is accurate versioning of IFC objects in the exchanged models. Existing collaboration systems consider BIMs as files and version them accordingly. Versioning of objects within a model is neglected, which restricts further processing of model content for collaboration operations. In the IFC2x3 or IFC4 schema, there is no definition for an actual Model (i.e., IfcModel); there is therefore no variable that can support versioning of an 'IfcModel' at the model level. Thus, there is an abstraction gap between current BIM tools and the structural nature of IFC models. As an alternative, object versioning is a developing approach that applies a control mechanism to an object in a BIM, at an attribute level, enabling conflict detection and change management on objects rather than on files or models (Nour and Beucke, 2008; 2010).

The key advantage of object level version control is the ability to track changes and updates through an audit trail during data exchanges, which can greatly improve the efficiency and effectiveness of model collaboration workflows. Existing BIM authoring tools maintain object versioning through the GUIDs of IFC objects and relationships. Nour and Beucke (2010) have suggested that in practice version control merely based on IFC GUIDs leads to inconstant results. An underlying problem with the use of GUIDs is that they are not managed correctly in existing BIM authoring tools and their resulting unreliability creates more problems than it solves for object versioning and model comparisons. Improved version control practices are needed to overcome these issues and provide more efficient results. This could include consideration of other metrics in addition to GUIDs (such as geometry, topology and semantics) for object recognition and versioning.

4.4 Granularity level of model comparison

When comparing models, a decision needs to be made as to what granularity level the comparison will be made at, for example $Model < objects < relationships < references < attributes < PropertySets < values$. Generally, models can be compared either using shallow or deep comparisons (Nour, 2007). A shallow comparison only considers mapping of primitive attributes, such as those defining geometry, and compares potential matches in two models using their GUIDs. Deep comparison also includes referenced attributes and compares the entire hierarchy of the model structure. Both of these approaches use GUIDs, which are difficult to maintain and inefficient (Nour and Beucke, 2010), leading to costly and inaccurate computations. In addition, these approaches rely on the structural properties of objects (i.e., properties that define geometry representation) that can be used to compare and identify differences and similarities. However, different BIM authoring tools may have different geometric representation for the same physical object. This leads to the following question: How can models constructed independently with different tools be compared when their internal representations vary from that exchanged through the IFC?

4.5 Data inconsistencies in IFC round tripping

IFC models are often composed of thousands of primitive and referenced objects and types which are represented differently within the internal structures of BIM tools. Therefore, the same IFC model exported from two different tools, can have added or deleted object types or associated attributes. These data inconsistencies occur due to the different internal structure of BIM authoring tools and application imprints on the IFC model during the round tripping process. Ma et al. (2006) identified that such data inconsistencies can include added attributes,

numerical precision loss, and differences in string length, attribute value calculations, and referenced objects. In addition, there can be schema inconsistencies caused during the export process, as some required attributes can be absent or lost during IFC export from a specific BIM tool. Such data inconstancies cause problems in data manipulation operations and yield inaccurate results.

5 Model server enabled collaboration on BIMs: A way forward

Collaboration on BIMs using a distributed shared repository is an iterative process involving long transactions. 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 a number of 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 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. 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. 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. In such design modifications, when exporting a new 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 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.

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 the amount 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 is 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).

5.1 Creating signatures for IFC objects: A way forward

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 looking into 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 an object's position, its shape and its 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 '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. Moreover, the position and shape are a way of reflecting an object on a drawing and in a 3D model, which is easy to reflect to 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.

If a method can determine the degree of importance of IFC properties, then it can be used to create a signature based on key properties. Hence, in terms of creating an element's signature, there are a number of issues that need to be considered in the broader perspective of change and relevance to the end user. 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 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. With this in mind, future research should focus on creating object signatures using hash codes from IFC object characteristics, as part of a signature matching strategy, previously successfully used in Software Engineering (Reddy and France, 2005), for object recognition and comparison in managing model server enabled collaboration on BIMs.

6 Conclusions

Collaboration on BIMs 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 while requiring the data to be synchronized 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 involves 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 the tasks involved in the operations of a collaborative BIM environment.

References

Adachi Y (2001). Introduction of IFC model server. http://cic.vtt.fi/projects/ifcsvr/memo/VTT-MEMO-ADA-05.pdf

Adachi Y (2002). Overview of IFC model server framework. ECPPM, 2002: 367–372

Beetz J, De Laat R, Van Berlo L, Van Den Helm P (2010). Towards an open building information model server. In: Proceedings of the 10th International Conference on Design & Decision Support Systems in Architecture and Urban Planning. The Netherlands

Counsell J (2012). Beyond level 2 BIM, web portals and collaboration tools. In: Proceedings of 16th International Conference on Information Visualisation. 510–515

Hietanen J (2002). BLIS review: IMSvr. http://www.blis-project.org/software/reviews/IMSvr_Review.pdf

Hjelseth E, Nisbet N (2010). Overview of concepts for model checking. http://itc.scix.net/data/works/att/w78-2010-53.pdf

Holt G D, Love P E, Jawahar Nesan L (2000). Employee empowerment in construction: An implementation model for process improvement. Team Performance Management: An International Journal, 6(3/4): 47–51

Isikdag U, Aouad G, Underwood J, Wu S (2007). Building information models: A review on storage and exchange mechanisms, bringing ITC Knowledge to Work. In: Proceedings of CIB 24th W78 Conference Maribor 2007

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. Aalborg: Aalborg University

Kiviniemi A, Fischer M, Bazjanac V (2005a). Integration of multiple product models: IFC model servers as a potential solution. In: Proceedings of the 22nd CIB-W78 Conference on Information Technology in Construction. 1–4

Kiviniemi A, Fischer M, Bazjanac V (2005b). Multi-model Environment: Links between Objects in Different Building Models. In: Proceedings of the 22nd Conference on Information Technology in Construction CIB W78. Dresden: 277–284

Kiviniemi A (2006). Ten years of IFC development why are we not yet there? International alliance for interoperability. In: Proceedings of 2006 Joint International Conference on Computing and Decision Making in Civil and Building Engineering. Montreal

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

Liebich T, Weise M, Laine T, Jokela M (2010). InPro building information model. https://www.yumpu.com/en/document/view/8106859/inpro-building-information-model

Lloyd-Walker B, Walker D (2015). Collaborative project procurement arrangements. Newtown Square: Project Management Institute

Lockley S, Greenwood D, Matthews J, Benghi C (2013). Constraints in authoring BIM components for optimal data reuse and interoperability: Results of some initial tests. International Journal of 3-D Information Modeling, 2(1): 29–44

Lockley S, Rombouts W, Plokker W (1994). The Combine data exchange system. In: First ECPPM Conference, Dresden

Love P E, Gunasekaran A (1997a). Process reengineering: A review of

enablers. International Journal of Production Economics, 50(2–3): 183–197

Love P E, Gunasekaran A (1997b). Concurrent engineering in the construction industry. Concurrent Engineering, 5(2): 155–162

Lowe R H, Muncey J M (2009). Consensus DOCS 301 BIM Addendum. Construction Lawyer, 29(1): 17

Ma H, Ha K, Chung C, Amor R (2006). Testing semantic interoperability. In: Proceedings of the Joint International Conference on Computing and Decision Making in Civil and Building Engineering. 1216–1225

Nour M (2007). Manipulating IFC sub-models in collaborative teamwork environments. In: Proceedings of the 24th CIB W-78 Conference on Information Technology in Construction. 1–12

Nour M, Beucke K (2008). An open platform for processing IFC model versions. Tsinghua Science and Technology, 13: 126–131

Nour M, Beucke K (2010). Object versioning as a basis for design change management within a BIM context. In: Proceedings of the 13th international conference on computing in civil and building engineering (ICCCBE-XIII). Nottingham.

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. AOM Workshop

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 Modelling, 1(4): 8–17

Shafiq M T, Matthews J, Lockley S R (2013). A study of BIM collaboration requirements and available features in existing model collaboration systems. Journal of Information Technology in Construction, 18: 148–161

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

Sun M, Lockley S R (1997). Data exchange system for an integrated building design system. Automation in Construction, 6(2): 147–155

Van Berlo L A H M, Beetz J, Bos P, Hendriks H, Van Tongeren R C J (2012). Collaborative engineering with IFC: New insights and technology. In: Proceedings of 9th European Conference on Product and Process Modelling. Iceland

Weise M, Katranuschkov P (2004). Generic services for the support of evolving building model data. In: Proceedings of the 10th international conference on computing in civil and building engineering (ICCCBE-XIII). 1–12