EasyChair Preprint

№ 4085

Single Shared Model Approach for Building Information Modelling

Simo Ruokamo and Rauno Heikkilä

August 25, 2020

Single shared model approach for building information modelling

Simo Ruokamo[a] and Rauno Heikkilä[b]

[a]University of Oulu, Finland & Enterprixe Software Ltd, Finland
[b]University of Oulu, Finland
E-mail: simo.ruokamo@gmail.com, rauno.heikkila@oulu.fi

Abstract –

The current practice for information sharing with building information modelling (BIM) is a distributed data sharing based on conversions. Conversions are problematic due to data loss, redundancy, and conflicting information. A single data schema used by all applications is a requisite for a conversion-free data collaboration. In the study, a software development kit (SDK) was developed, which implements required features and guarantees compatibility between BIM programs. Three independent applications 3DTrussme, Leonardo, and Viewer were developed using SDK. A cloud service for handling the shared model was implemented. In the experiments, Leonardo was used for modelling walls, 3DTrussme for truss design, and Viewer for model viewing. All three applications were using the same shared model on the cloud.

In the experiments, the information exchange occurred without conversions and all data was saved only once on the cloud database. Without conversions and duplicates less conflicts and redundancies occurred, which lead to better data integrity and integration. Using SDK, there was no technical barrier for applications to join the single shared model ecosystem, but a drawback was that existing BIM programs are not compatible without remarkable changes. The performance was acceptable on the test run, but in real use, the size of the model and the number of applications and users, will be much larger. However, a conversion-free single shared model approach can be a possible trend to the development of the next generation BIM as well as a potential alternative for current data sharing methods using distributed files, conversions, and linked data.

Keywords –

Building Information Modelling, Data Conversion, Cloud Services

1 Introduction

The evolution of the building design has developed from handmade paper drawings to a fully digitalized process using computers. In the 1970s, the first 2D CAD (Computer Aided Design) software came to the market and in the 1980s, first pioneers developed 3D design applications for building design. Acronym BIM (Building Information Modelling) is nowadays commonly used for the digitalized information handling and it was probably Jerry Laiserin who first introduced the term BIM [1,2].

A successful collaboration between all stakeholders requires an efficient and functional sharing of the building information [3]. The amount of the BIM data grows significantly during the design and construction stages of the building project. After the construction stage, new information is still created but not at the same rate. Moreover, the flow of the data substantially breaks off when the construction is completed [4]. A continuous information flow is a necessity for improving the data utilizing within facility management [5].

The prevailing practice for data exchange is distributed data management (DDM) approach based on conversions. In general, an exchange format is used for data transfer. Using an exchange format requires two conversions between two applications, but reduces the total amount of import and export formats each application needs to implement [6]. Direct data exchange between native formats requires only one conversion and is less error prone but, on the other hand, each supported format must be programmed.

A conversion-free data exchange requires the use of only one data format. To fully avoid the problem of overlapping and conflicting information, a single data schema is not enough. Separate models, although in the same format, can still include inconsistent data. With the single shared model approach (SSM) no conversions are needed and all information is saved once, which reduces the data complexity and improves the data integrity and integration. The challenge with the single model is

concurrent changes made by different users. Rules must exist for defining which change is valid, when overlapping modifications happen. Figure 1 illustrates the different exchange methods from the perspective of conversions.

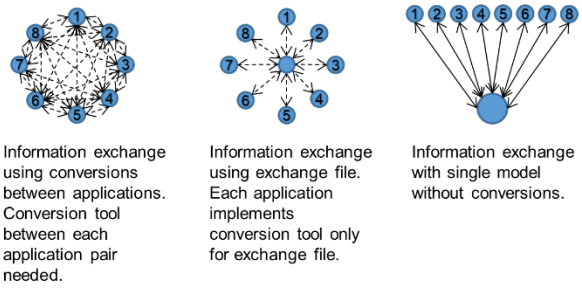

Figure 1.The basic differences for the needed conversions with the different data exchange methods. With application to application, the total number of conversions is the largest, whereas with a single shared model no conversions are made.

2 Development of the single shared building information modelling

For a regular user it is common to mix applications, data models, and storages. However, they have their own functions and objectives from a software technology point of view. When application is running, information is available at the main memory of the computer. When application is closed, the data must be stored to permanent storage which can retain its information even when powered off. The interconnection between the permanent storage and application is the data model. A file or database is read into program's run time memory as a data model, then modified during the application run and saved back to the permanent storage.

2.1 Data model and storage

In today's software technology, the common practice for a data model is an object oriented approach [7]. Each different object is encoded as a software class and a requisite set of classes constitutes the whole data model. The class instance is an object that is created from the definition of the class into the data model. Instances get a globally unique identifier when they are created into the data model and one software class can be instantiated as multiple objects, each having a different unique identifier.

The data on the permanent storage is read to object instances and saved back. For instantiating the correct class to the data model the definition of the class must also be saved on the storage. The class instantiating method can be either static or dynamic. With static binding (called also early binding), code of classes must be available when the program is compiled and built. The drawback of the static binding is that every change or addition of a class requires an updated version of the application. Modern programming languages, like Microsoft C# [8], support dynamic binding (called also late binding), which requires the availability of the class not until the application is started. That is a significant difference and makes changes into data model classes much more flexible. A class addition or change does not require a new version of the application.

In a building data model, classes and instances carry the information as a collection of value-name pairs. The IFC (Industry Foundation Classes) is a standardized object-based data format and model maintained by an international non-profit organization called buildingSMART. With IFC, value-name pairs are called property sets [9]. Term attribute is also commonly used with object-based data models. List of classes and properties are not constant which brings up the challenge of the data compatibility. Standards are common languages which realise the universal and admitted understanding for the content of the building information. But standards change slowly and are not adequate for commercial BIM applications since data content advancement is an endless and all the time running process. Therefore the flexibility and extensibility of the information content are key features for the single shared data model. Supporting standardised data is advisable, but by allowing applications to freely specify additional information content, technical barriers are eliminated from the use of a single data model schema. Both the data model and permanent storage must implement freely extendable data content.

The permanent storage can be a database or a file. A database has a more organised data schema and allows partial data access and sharing with several users. The following three alternatives are technically possible as a database schema for storing the data of classes:

1. Separate table for each class with a separate column for each attribute. This schema has traditionally been used.
2. The vertical database schema also called an entity-attribute-value model (EAV) [10].
3. XML schema, where whole data of the class is packed as XML data.

Juola [11] implemented all the three alternatives using a SQL Server database. With separate table for every class it is almost unfeasible to keep tables and columns up-to-date due class changes. Queries are very complicated with the entity-attribute-value schema. The result was, that the XML schema was best suitable for a building data model having always evolving content.

The number of object instances in a data model can grow huge. Some kind of organising and grouping is needed for maintaining the fluent manageability of the model. Objects can be organised based on their material, structural behaviour, type, name, position, feature, or some other data. As a result, the amount of active objects is decreased making the handling of the model more flexible. IFC supports Model View Definition (MVD and Information Delivery Manual (IDM) standards for defining a data subset [12]. Several studies have been made for extracting IFC partial model based on MVD [13].

Hierarchical organisation is a well-known arrangement method, but that method has very rarely been applied among AEC/FM applications. The hierarchical arrangement naturally enables a model division into partial models and hierarchy is simply constructed by defining a parent object for every object. A partial model is then made up of an object and all its descendants at all sub hierarchy levels. The IFC data model has a static and fixed hierarchy of *Project → Site → Building → BuildingStorey → Space* [9]. According to Singh, Gu and Wang [14] a static hierarchy is inadequate and the ordering should be flexible for fulfilling the requirements of users.

2.2 Data sharing

A data concurrency control is essential for the single shared data model. Simultaneous changes to the same data can be handled by an optimistic or pessimistic method [15]. With an optimistic control, it is assumed that conflicting data changes are rare and can be resolved. The pessimistic control is based on data locking, which fully prevents any concurrent data editing. The validity, reliability, and consistency of the information in the building model is best guaranteed by the pessimistic method. Locking the whole model is not realistic, but only a part of the model can be reserved for one user at one time. The hierarchical model arrangement implements partial models that can be reserved and released. The single data model system and hierarchical arrangement with reserving and releasing partial models together make up a pessimistic data concurrency control system for ensuring the validity, reliability, and consistency of the information in the building model.

With the single shared model, all information must be available for all stakeholders without delays and conversions continuously. In the current internet world that is best achieved with a cloud based system. Using the cloud database only is technically possible, but a synchronised local copy gives next advantages:

- Enables incremental updates reducing the amount and size of data transfers [9]. By keeping a change log, only changed data needs to be synchronized between the cloud and local storage.
- The reserved partial model can be first saved to the local storage before publishing it to the cloud. Unfinished work is then not available for other participants.
- Offline working without a connection to the internet is possible with the local synchronized storage.

The cloud storage cannot be accessed like the local storage. User rights on the server cannot be as extensive as they are on the local computer. It would be a clear risk for the security and data integrity to allow public and direct read-write access to the server storage for all. A cloud service implementing only needed functionality ensures that no data corruption occurs due to a false operation. For a safe and secure access to the cloud storage, next functions need to be implemented on the cloud service:

1. Registration of user.
2. Establishing a new model.
3. Getting a list of models available for user
4. Connecting to a model
5. Load for downloading the whole or partial model from the cloud to the local storage.
6. Reservation of the partial model for editing. Reserved part is locked permitting only reading for other users.
7. Releasing and publishing the reserved partial model to the cloud storage.
8. Get changes due to releases made by other users.
9. Adding a new node to the model hierarchy tree.
10. Removing a node form the model hierarchy tree.
11. Disconnect from the shared model and logout from the cloud service.

2.3 Programming principles of the single shared data model system

A derivation programming technique is a common practice with the coding of classes. With derivation, duplicate code for similar classes is avoided, since a derived class inherits everything as default from the parent. Derived classes can develop the inherited content further as much as needed. The amount of software data model classes that are needed during the whole life-cycle of the building is vast. Thus, the development and maintenance of data classes are not tasks for a single software house. However, for ensuring compatibility, base public classes used by all developers are needed. The derivation of new classes must start from public classes, which must implement the required functionality for forcing the compatibility between all developed applications. Especially reading the permanent storage as a data model into runtime memory of the application and saving it back are operations that must only exist on

public classes. Secondly, classes for geometry must be implemented as public classes so that applications can also view non-public class objects that are defined by other applications.

For uniformity, common understanding and ability to co-operate, the amount of public classes should be as large as possible. Many organisations in various countries are developing various BIM standards for diverse purposes [16] and the public classes can be seen as the standardised part. BuildingSMART International has published a few standards for building information content and data exchange [17]. However, also non-standard classes can be made publicly available.

2.4 Arrangement and execution of the experimentation

For a full scale testing of the single shared model system, a cloud service and applications were developed. Cloud service was running on a Windows Server operating system. The web service was implemented using Windows Communication Foundation (WCF) [8]. Storage system used was a relational database Microsoft SQL Server. For the development of applications a public software development kit (SDK) was created including the next four assemblies:

1. Base public classes (BPC) assembly includes base data model classes from which all application specific classes must be derived.
2. A local storage for client (LSC) assembly offers a synchronised local storage for applications.
3. A model toolkit for the client (MTC) implements functionality for synchronising the local and cloud storage.
4. A web service toolkit for the client (WSTC) is a helper assembly simplifying the use of the web service functions.

Three applications were developed using the public SDK. *3DTrussME* is a 3D modelling and structural analysis software for wooden trusses. It has been the first and main testing application for the single shared model system and has a large application specific class library. *3DTrussME* is owned by a Finnish company, Ristek Oy, and the programming is carried out by another Finnish company, Enterprixe Software Ltd [18]. *3DTrussME* is a commercial application currently used in Finland, Norway and Estonia. *Leonardo* is a 3D design application for concrete structures. The development of *Leonardo* is ongoing and it is not yet available for a practical use. The third application used in testing was *Viewer*, which was only used for viewing the model. Figure 2 shows the general arrangement of the testing environment.

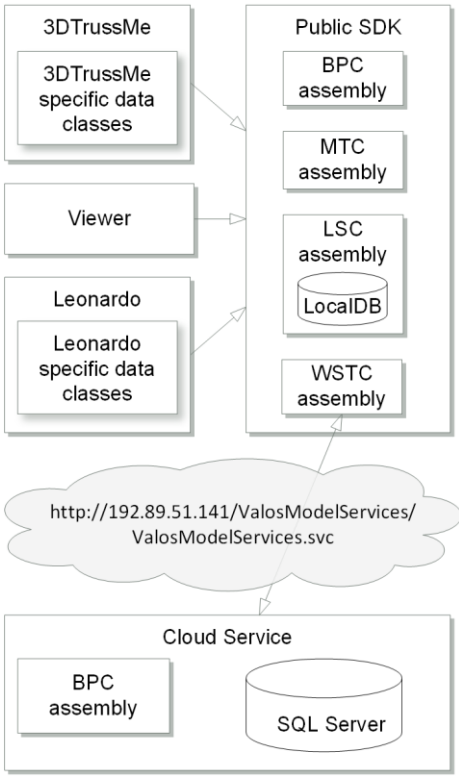

Figure 2. A diagram showing the general arrangement of the testing environment with three applications, public SDK package, and cloud service.

Three users were participating in the test event, one for each application. All three applications were connected to the same shared model on the cloud. *3DTrussME* was used for truss design, *Leonardo* for modelling walls and *Viewer* for viewing the model. The pessimistic concurrency control was in use and partial models were reserved and released. During the test, *Leonardo* data classes were further developed without any compatibility problems for other two applications. The test was executed with steps shown in the next list.

1. Registration of users.
2. A model was established on the cloud database and access to the model for users was granted.
3. User #1 using *3DTrussME* reserved the model, created the base hierarchy and released the model. Fig. 3 shows a screen snapshot after step #3.
4. User #2 using *Leonardo* reserved the *Walls* subtree and started modelling walls. User #1 saw the reservation when getting the latest from the cloud.
5. User #2 finished the modelling of walls, released the node *Walls* and local changes were updated to the cloud model. User #1 updated changes from the cloud and saw the modelled walls. *Leonardo* was using a private wall class, but by using a public wall class instead, walls were available at *3DTrussMe*.

6. User #1 reserved the node *Roof* for truss modelling. User #3 started *Viewer* and saw walls and reservation of the *Roof* node.
7. User #1 noticed that one wall was at wrong position, reserved it, made the correction, and released it.
8. The private data model of Leonardo was further developed by adding new attributes to the wall and by creating a new column class. User #2 started to use the new version of *Leonardo* and updated local model. No compatibility problems or data corruption occurred although *3DTrussMe* used older public wall class and *Leonardo* new changed wall class.
9. User #2 reserved *Walls* and *Columns* node, continued modelling and released nodes.
10. User #1 released Roof node.
11. All users updated their local model and can saw the whole model.
12. All users disconnected from the model and closed applications.

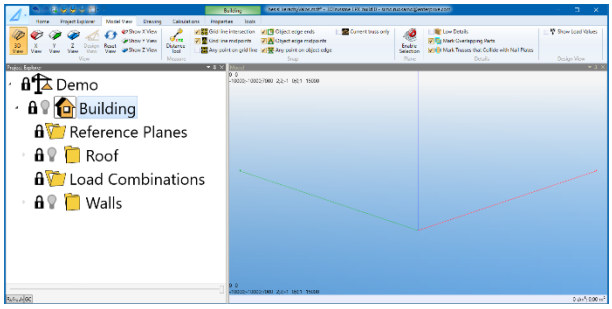

Figure 3. Basic hierarchy after step #3. All nodes are not reserved for changes which is marked as a closed black lock icon in the project explorer tree.

3 Results

As a result of the test execution, a shared model was built up on the cloud. Three applications were using the same shared model on the cloud and all data sharing occurred without conversions and data defects. Figure 4 shows the final model after the execution of the test. The permanent outcome of the executed test was the data stored in the cloud database. Totally five tables were used for data storing:

1. User table for registered users.
2. Model access table for defining the access of users to model.
3. Session table for connected users. With a connection to the model each user gets a session id that identifies the access to one model. Session ids are not permanent and are used instead of credentials after the connection to the model. Session ids are invalidated with disconnect or after defined unused timeout.
4. Event table for the model established, reservation and release events. Events enable the bookkeeping of reserved nodes and incremental updates.
5. Model table for the building model data.

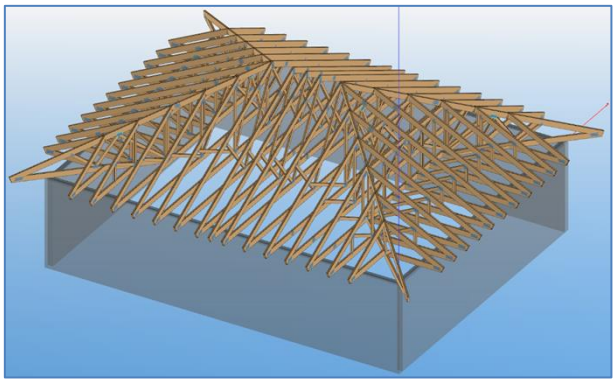

Figure 4. Final model after the execution of the test on *3DTrussME* application.

Just one data format is compulsory for the conversion-free data exchange used. The key points of the specification for a freely expandable data model schema keeping the compatibility backward and forward are as follows:

- The data model schema consists of public part and private application specific portions. All private classes must be derived from public classes.
- Data carriers and all data saving into and reading from the permanent storage are handled by public classes.
- Classes are instantiated using the dynamic binding method.
- Schemas of the data model and permanent storage must enable backward and forward compatibility allowing free changes to the content and number of data classes. This is achieved with a dynamic binding and XML storage format.
- A software development kit (SDK) implements all the key points of this list, and thus, applications developed using SDK automatically realize all key points and are compatible with each other.

Data duplicates are not prevented by using only one data model and storage structure. A single model approach accessed simultaneously by all participants is needed for removing the overlapping information. The following key elements are required for a workable single shared model system:

- The single shared database is placed on the cloud, enabling an equal access for all participants.

- Access control is implemented limiting the entry only for registered and authorized users. Furthermore, access to models per user is controlled.
- All connections to the shared cloud model are performed through a web service that has the required functionality. It is a security risk to allow direct access to the database, which may lead to illegal changes in the database.
- A pessimistic data concurrency control is used.
- An event log keeps track of reservations, releases and data change events enabling the monitoring of the reservation state and incremental updates of the local storage.
- The model arrangement is realized using a hierarchical approach which divides model into numerous and varying size partial models. Those subtrees can be used as units for reservation and for limiting the size of the model that is handled at a time. The hierarchical arrangement is free and the model can be divided as example by design discipline, storey, space or element type using one or combined dimension.
- SDK is used for the development of applications enforcing the compatibility and SDK also eases establishing connections to the single shared model.

4 Conclusions

The presented single data model system is available for all new and old BIM applications. An SDK is free and contains base classes as a start point for the development of application-specific data content. The schema of the data model is version-free allowing changes and additions without breaking the compatibility. In summary, the presented method makes up a conversion-free data exchange solution based on a single extendable data model schema. It is self-explanatory that without conversions all conversion defects will be eliminated. Data duplicates will vanish when a single model schema is extended as a single shared model approach. The data integrity and integration improve when data sharing occurs without conversions and when no overlapping information exists.

To obtain the greatest benefit from the single shared model, software from various disciplines should be available. There is no limit regarding what types of applications can join the ecosystem: the only requirement is to use an SDK. Anyway, many issues can be raised up for the wider industrial use. The incompatibility with the current convention, needed investments and lack of interest hinder the expansion. The reputation and reliability of a new technology is low in the beginning. Rogers [19] divides technology adopters into innovators (2.5%), early adopters (13.5%), early majority (34%), late majority (34%), and laggards (16%). Evidently, the conversion-free single shared model approach needs innovators for the start of the technology expansion.

No commercial single model system for multidisciplinary building information is available. Systems for sharing a model between the same applications have been developed, but none can cross the application boundary. The objective for model sharing with *Tekla*, *Archicad* and *Revit* is to enable multiple people to work simultaneously with the same model. There is no reason to limit the model sharing only between the same applications as long as user and access control prevents conflicting and illegal changes. Indeed, according to Lu, Wu, Chang and Li [21] , there is lack of BIM standards for model integration and management by multidisciplinary teams.

It is a common opinion among the AEC industry and BIM scientists that a single model BIM is an unfeasible solution. According to Day [22], a single building model is only a daydream. On the other hand, Howard and Björk [23] state that a single BIM is the holy grail, but there might not be willingness to achieve it. According to Turk [24], a centralized shared database is impossible but in the future, BIM will approach it. The reasoning for this is mostly not presented by these authors, but model differences between disciplines and the size of model are noted. Because of the rejection of the single BIM model, no research has been conducted of a true single shared model system. Under the umbrella term ‘single BIM’ scientific articles can be found, but they see single BIM as a common repository for distributed data sources. A cloud service or a single address to separate files is only one data delivery tool for distributed information. A true single model system is a shared database that can be accessed simultaneously by several users, and every piece of data is stored only once.

According to Johnson [25], the complexity of design tasks and software evolution raise questions about a single model solution. It is true that tasks performed by engineers and consultants are complex and various. Many kinds of applications are used for design tasks and all software is evolving continuously. However, is that complexity troublesome only for the single shared model approach? The distributed data sharing system uses many data formats and conversions. Is this more complex when compared with the single shared model operating without conversions? Johnson [25] list the next issues for alleviating the skepticism against “One BIM”:

- An open source vendor-neutral elastic data structure.
- Enabling the interoperation of applications from multiple vendors.
- Sharing data in the design ecosystem without explicit import or export.
- Supporting different kind users, tasks, workflows, and stages in the design process.

The presented single shared approach will implement all the above items, but a significant adjustment to current BIM practices and processes must come true.

According to Miettinen and Paavola [2], the benefits of using BIM is often reported by researchers and project participants, but the impact of BIM is difficult to isolate from the success of the entire project. Moreover, increases in the productivity in the construction business have been marginal when compared with other industry sectors [26]. Assuming, that information technology has a notable influence on the improvement of productivity, why has digitalization succeeded in other industry sectors but not in the construction business?

The testing of a software system is not a one-time process. In the test run only three applications and users were involved, but in the reality, many more applications are used with BIM. No technical limit for the number of applications exist, but a growing number of applications and users accessing the same model concurrently can slow down the performance

Microsoft SQL server was used as the cloud storage system and the maximum size of a SQL Server database is 524 272 terabytes [8], which is an incredible amount of information. Before that maximum size limit is reached, other performance issues will probably arise. Client applications do not need to make calls to the cloud service non-stop, but database queries can slow down the cloud service when the database is large. In the test run, there were only three users and one model on the cloud, which naturally cannot show much of the performance for real projects. Microsoft LocalDB was used as a local storage system on the client side. The maximum size of the LocalDB database is 10 GB [8] which is much less than the maximum size of a SQL Server database. Local storage capacity will most obviously be the first bottleneck before any performance problems on the cloud service appear.

An internet service provider (ISP) and independent software vendor (ISV) for web service are needed for offering the cloud service for the single shared model. Figure 5 shows all the major players of the building project. ISVs should not hold a monopoly position in their field of activity. Application development is freely available for all enabling multiple software on the same purpose. Developing the web service can also be done separately by multiple ISVs. There can only exist one public SDK and a private commercial enterprise is not the best ISV for SDK. A public non-profit corporation or alliance would be a better ISV operator for the public SDK.

Allowing all users to change all parts of the model after reservation might not be a desired course of action. As example architects do not usually allow designers from other disciplines to edit architectural plans. By organizing users to groups, more detailed user rights can be implemented. Each group can reserve partial models and control usage rights for other groups. When using both user and group access control the partial model availability can be restricted on many levels.

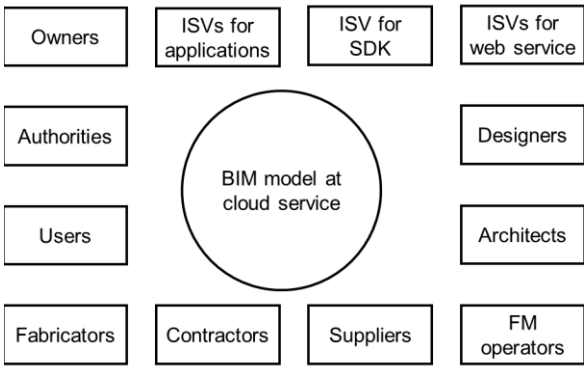

Figure 5. The players around BIM model during the whole life-cycle of the building project.

The main weakness of the single shared model approach is its incompatibility with the currently used data formats. All data classes in current applications must be redone for full compatibility which is likely a threshold for most software houses. Therefore, conversions from existing formats are needed for lowering the obstacle to the presented new single model method; otherwise, it will be isolated without any links to existing systems. Rewriting all IFC classes by starting the derivation from BPC classes could be one solution. After that, importing the IFC files could be done. However, exporting to IFC cannot properly support all the data on a single shared model using a version-free data schema since IFC is not a true open and extendable data format. IFC implements adding new property sets for the objects and the use of IfcProxy for entities that are not defined by IFC [27]. But, this will end up as an outstanding amount of IfcProxies having fully different content if all the native data by all applications is exported. Additionally, updating the schema of the IFC standard will break the compatibility backward and forward, requiring a new version of every application that is using IFC.

References

[1] Laiserin J. LaiserinLetter. 2002; Online: http://www.laiserin.com/features/issue15/feature01.php, Accessed 2013.

[2] Miettinen R, Paavola S. Beyond the BIM utopia: Approaches to the development and implementation of building information modeling. *Automation in Construction*. 2014 7;43:84-91.

[3] Shen W, Hao Q, Mak H, Neelamkavil J, Xie H, Dickinson J, et al. Systems integration and collaboration in architecture, engineering, construction, and facilities management: A review. *Advanced Engineering Informatics* 2010 4; 24(2):196-207.

[4] Xu X, Ma L, Ding L. A framework for BIM-enabled life-cycle information management of construction project. *International Journal of Advanced Robotics Systems* 2014; 11(1).

[5] Nicał AK, Wodyński W. Enhancing Facility Management through BIM 6D. *Procedia Engineering* 2016; 164:299-306.

[6] Isikdag U, Underwood J. Two design patterns for facilitating Building Information Model-based synchronous collaboration. *Automation in Construction* 2010; 19:544-53.

[7] Berdonosov V, Zhivotova A, Sycheva T. TRIZ Evolution of the Object-Oriented Programming Languages. *Procedia Engineering* 2015; 131:333-42.

[8] Microsoft Corporation. Microsoft developer network. 2019; Online: https://msdn.microsoft.com. Accessed 2019.

[9] Eastman C, Teicholz P, Sacks R, Liston K. *BIM Handbook*. Hoboken, New Jersey: John Wiley & Sons, Inc.; 2011.

[10] Nadkarni PM, Marenco L, Chen R, Skoufos E, Shepherd G, Miller P. Organization of heterogeneous scientific data using the EAV/CR representation. *Journal of the American Informatics Association*. 1999; 6(6):478-93.

[11] Rauno Juola. *Rakennuksen tietomallin palvelinpohjaisen tietokannan tallennusratkaisu*. Oulu: University of Oulu, Faculty of Information Technology and Electrical Engineering, Department of Computer Science and Engineering, Computer Science; 2014.

[12] Lee Y, Eastman CM, Solihin W. An ontology-based approach for developing data exchange requirements and model views of building information modeling. *Advanced Engineering Informatics* 2016 8; 30(3):354-67.

[13] Won J, Lee G, Cho C. No-Schema Algorithm for Extracting a Partial Model from an IFC Instance Model. *Journal of Computing in Civil Engineering*. 2013 NOV 1; 27(6):585-92.

[14] Singh V, Gu N, Wang X. A theoretical framework of a BIM-based multi-disciplinary collaboration platform. Automation in Construction 2011 3; 20(2):134-44.

[15] Wette C, Pierre S, Conan J. A comparative study of some concurrency control algorithms for cluster-based communication networks. *Computers & Electrical Engineering* 2004; 30:615-36.

[16] Barbosa MJ, Pauwels P, Ferreira V, Mateus L. Towards increased BIM usage for existing building interventions. *Structural Survey*. 2016;34(2):168-90.

[17] BuildingSMART. buildingSMART International.; Online: http://www.buildingsmart-tech.org/, Accessed 2019.

[18] Jarmo Kajava. *Development of the three-dimensional nail plate structure design software*. Oulu: University of Oulu, Faculty of Technology, Mechanical Engineering; 2017.

[19] Rogers EM. Diffusion of Innovations, 5th edition. New York, USA: Free Press; 2003.

[20] Wong J, Wang X, Li H, Chan G, Li H. A review of cloud-based bim technology in the construction sector. *Journal of Information Technology in Construction*. 2014;19:281-91.

[21] Lu Y, Wu Z, Chang R, Li Y. Building Information Modeling (BIM) for green buildings: A critical review and future directions. *Automation in Construction*. 2017; 83:134-48.

[22] Day M. The trouble with BIM. AECMagazine 2011;.

[23] Howard R, Björk B. Building information modelling – Experts' views on standardisation and industry deployment. *Advanced Engineering Informatics* 2008 4; 22(2):271-80.

[24] Turk Ž. Ten questions concerning building information modelling. *Building and Environment* 2016; 107:274-84.

[25] Johnson BR. One BIM to Rule Them All: Future Reality or Myth? In: Kensek KM, Noble DE, editors. *Building Infromation Modeling: BIM in Current and Future Practice*: John Wiley & Sons; 2014, p. 175-185.

[26] Koskenvesa A. Rakennustyön tuottavuus 1975-2010. *Rakentajan kalenteri* 2011: Rakennustieto Oy; 2011, p. 138-146.

[27] Borrmann A, Beetz J, Koch C, Liebich T, Muhic S. Industry Foundation Classes: A Standardization Data Model for the Vendor-Neutral Exchange of Digital Building Models. In: Borrmann A, König M, Koch C, Beetz J, editors. *Building Information Modeling: Technology Foundations and Industry Practice*: Springer International Publishing AG; 2018, p. 81-126.