Requirements for event-driven architectures in open BIM collaboration

Esser, S., Abualdenien, J., Vilgertshofer, S., Borrmann, A.
Technical University of Munich, Germany
sebastian.esser@tum.de

Abstract. Current practice of BIM-based collaboration relies on the exchange of entire domain models, which are managed and shared using Common Data Environments. Such platforms are well standardized in terms of states, roles, and participants. However, the coordination of changes by exchanging entire models requires manual identification of any added, modified, or deleted information from one shared version to another. This paper proposes the application of an event-driven system architecture, which is in widespread use in modern communication systems. Combining the publish-subscribe design pattern, patch-based update mechanisms for BIM models, and the established concept of asynchronous, decentralized collaboration in BIM projects can significantly ease the process of understanding design changes and reduce the overall overhead of information already shared in previous versions among project members.

1. Introduction

A major advantage of the Building Information Modeling (BIM) methodology is to facilitate collaboration and coordination among the various domain experts in designing and constructing built assets. In today's practice, reflected in ISO 19650, data management in BIM projects is realized through Common Data Environments (CDE). While the standard does not define the technical implementation, a typical CDE is a web application with centralized file storage for sharing model files and associated documents. Domain experts involved in a project can upload and download files from and to the CDE. It is important to note that current BIM practice relies on the notion of federated domain models (such as architectural, structural, and electrical models) as specified in ISO 19650 (CEN, 2018). Accordingly, no overall model is collaboratively edited, but each domain creates individual models and coordinates them with other parties in discrete time intervals. Currently, this federation concept is realized by uploading information containers comprising entire domain models (typically in an open BIM format) to a CDE. However, today's practice entails significant limitations, as modifications cannot be tracked for individual objects, forcing all collaborators to perform global checks for potential coordination with their domain models.

Numerous studies over the past decades have highlighted the advantages of vendor-neutral data exchange over closed-BIM solutions (Shafiq et al., 2012; Solihin et al., 2016; Zhang et al., 2015). Design processes are often multi-disciplinary and iterative, demanding the use and integration of various design and simulation tools. Thus, novel methods for distributing design changes are highly required to support decisions and project management. Currently available open-BIM solutions can only facilitate exchanging complete BIM files rather than specific changes on object level, which impedes a seamless integration and evaluation of the exchanged updates.

To overcome this deficiency, a patch-based update mechanism can be used to track model modifications on object level representing them as graph-based transformations (Esser et al., 2021). A core requirement of realizing such a decentralized object synchronization system lies in a suitable network architecture and adequate communication protocols. Advanced techniques of distributed computing systems appear promising as various protocols support bidirectional

https://doi.org/10.7146/aul.455.c195

and asynchronous communication, particularly relevant for the fragmented construction industry with highly specialized domains.

This paper proposes an event-based mechanism to transfer update patches among project stakeholders and highlights different protocols that support bidirectional communication in a distributed, asynchronous environment. The approach enhances the information value of BIM-based exchange workflows because it provides direct access to the modifications, which helps to assess the impact on other domain models. Nevertheless, it is essential to state that each domain will continue authoring individual domain models, which other parties should use for reference purposes. The benefits of the proposed approach are twofold: First, clear authorship for each dataset shared within a project remains existing. Second, the proposed system reduces the overall data exchange and simplifies the evaluation of shared modifications as each party can register itself for events relevant to its design and simulation tasks. Furthermore, the subscription to specific change events assists in raising alerts to the user in case of critical updates that affect interdisciplinary dependencies.

2. Existing approaches in the context of BIM-based collaboration systems

The design and planning of civil and building assets are typically iterative processes as the achieved results must fit various boundary conditions, such as environmental and economic aspects. Finding consensus across the involved parties typically involves consistency and clash detection checks, performed on coordinated models that assemble all partial domain models (Preidel & Borrmann, 2015). Studies have shown that decentralized federation-based collaboration using independent domain models and merging them regularly into coordination models shows higher performance than jointly working on a single centralized BIM model (Counsell, 2012). Shafiq et al. (2013) have researched user requirements against CDEs and how existing platforms supply suitable functions accordingly. Even though the results achieved back then are not directly comparable to today's market situation, some key findings are still valid. Mainly deviating naming conventions and considering proprietary formats produced by desktop products hinder an unbiased comparison of various processes. From a technical perspective, current CDE systems require sharing BIM models as monolithic files and uploading them manually to the project platform.

Contrary to this practice, Sattler et al. (2020) proposed a query-based approach to consume information from federated models. The extracted information assists in interdisciplinary problems as their framework can handle heterogeneous data sets. However, project-wide coordination of all delivered models remains not possible with their approach.

Chen & Hou (2014) have presented a hybrid approach consisting of peer2peer and Client-Server connections for model-based collaboration workflows. Each domain connects to the centralized project hub, enabling project-wide management and coordination of interdisciplinary tasks. Additionally, all members of a specific domain can jointly collaborate within their domain system to synchronize their achievements in real-time. However, a downside of their approach is the strong focus on aspects related to the architectural and MEP domain. Following the vision of extending the decentralized nature of design processes with the ability to share updated information in an event-driven style, a flexible system architecture is required to respect existing paradigms and enhance the information flow among experts and disciplines participating in a project.

Recently, multiple researchers have investigated leveraging cloud capabilities and linked data techniques to foster the delivery and exchange of project files, including BIM models, drawings, images, and others. Senthilvel et al. developed a micro-service approach for delivering project files following the information containers ISO standard 21597 (Senthilvel et al., 2021). Additionally, Karlapudi et al. (2021) have analyzed the capabilities of ISO°21597 and presented a case study evaluating SPARQL queries in information containers. Schulz & Beetz (2021) proposed the use of the BIM collaboration format (BCF) on a centralized web server to document existing buildings.

3. Distributed architectures and technologies

The concept of distributed computing has been researched and applied in various research and industry applications. This section highlights best-practice approaches to event-oriented programming, network architectures, and related communication protocols.

3.1 Design patterns for event-driven programming

Design patterns assist in defining standard implementation recipes on how a specific feature should handle data in a best-practice manner. Two prominent patterns must be considered when discussing the application of event-driven architectures for BIM-based collaboration workflows.

Observer pattern: Consider a set of objects that must be notified about the change applied to object A. Then, object A maintains a list of observers. Every time the state of this object changes (e.g., a new value is set to a particular object property), the object instance reports to all observers about the change. Each observer, in turn, can then perform actions to react to this event. Even though the object knows its observers, there is no vice-versa interaction between an observer and the object itself, like notifying the entity about the observers' state. Nevertheless, one can implement a second observer to enable a bidirectional communication structure between both objects.

Publish-Subscribe pattern: Contrary to the principle of having a single consumer for each message, the publish-subscribe pattern features the distribution of an incoming message to many receivers. Clients can specify a topic or channel within their message, which is then used in the middleware to distinguish to whom the message should be delivered. Clients can subscribe to topics relevant to their business and will automatically receive incoming messages from the middleware if they contain the subscribed topic.

In addition to the observer and pub-sub approach, the Point-To-Point pattern distributes occurring events to other parts of the system. Each connected object can formulate a message and send it to a centralized middleware. The middleware uses a message queue following the First-In-First-Out approach to store all incoming messages. Then, it forwards the message to (exactly) one receiver with free capacities. Usually, each message is delivered to only one receiver and exactly once. As an advantage of this restriction, the receivers do not need to coordinate after receiving a message but can perform the necessary tasks. Such an implementation is often applied for load-balancing use cases (Curry, 2004).

3.2 Event-driven architectures in distributed network systems

The outlined design patterns typically act on a single computer or even within a single application. However, these concepts are also extensively used for distributed systems consisting of multiple devices connected to each other or through a central server. In the latter case, the server exposes interfaces and enables several clients to connect simultaneously and to exchange information using standardized communication protocols. A client can emit a message and receive data other clients (or the server itself) has published to the system. In addition to client-server architectures, peer2peer systems support non-hierarchical network layouts, in which each computer can communicate with others in a direct manner.

In all types of network architectures, standard web-based communication protocols, such as Hypertext Transfer Protocol (HTTP) and Transmission Control Protocol (TCP), can realize the information transfer. Accordingly, REpresentational State Transfer (REST) is a popular means for creating Web APIs (Fielding, 2000). RESTful web APIs are typically based on HTTP methods to access resources (API functions) via URL-encoded parameters and the use of JSON or XML to transmit data. Each HTTP method is executed as a request to a server that responds with content and a status code (indicating whether the request was successful or not). REST is well suited for various applications that involve manipulating and maintaining the status of data at the client-side while requiring efficient communication with a server. However, the server is typically stateless when using REST, i.e., no session information is retained by the server. As a result, information about the requesting clients (e.g., their IP addresses) is not stored. Hence, achieving an event-driven communication requires the client to periodically send requests to the server, check whether an event has occurred, and then retrieve the data. Hence, the stateful WebSocket communication protocol is more suitable for the envisioned application in the field of BIM-based collaboration as it can provide (using its built-in capabilities) a full-duplex communication between clients and servers. Communication using WebSockets ensures a real-time notification of clients by the server when a particular event occurs as the clients' connection address and state is maintained at the server. From a systematic perspective on the network architecture, servers implementing event-driven mechanisms are often named *Message-oriented middleware* (MOM). The server itself manages incoming messages and distributes them to all connected clients. Curry (2004) compares this architecture to the idea of a classical postal service. Messages are delivered to a post office; then, the postal service is responsible for providing the item to the correct receiver.

Looking into systems featuring the Internet-of-Things (IoT), prominent MoM examples are the MQTT and the CoAP protocols, which are often used in machine-to-machine (M2M) communication. The MQTT protocol facilitates topic-based communication where each client can publish payloads under hierarchically structured topic lists. In turn, clients can listen to any level of the topic hierarchy and receive a notification in case of a new message. The MQTT protocol uses a TCP-based communication based on IP routing (Light, 2017). In contrast to MQTT, the CoAP protocol specifies events by *Universal Resource Identifiers* (URIs) instead of topics. Thus, receivers subscribe to a particular resource rather than a topic (Bormann et al., 2012). Both protocols focus on situations where collecting, storing, sending, and receiving data with a minimum of overhead to protect the available power and network resources. Therefore, a key objective of MQTT and CoAP is a data transfer with almost no data overhead.

3.3 Serverless approaches

The latest advancements in cloud computing have reduced the burden of configuring, securing, and deploying servers. A new paradigm that is recently gaining popularity is the serverless

architecture, supported by leading cloud providers like Amazon Web Services, Google Cloud Platform, and Microsoft Azure. "Serverless" means that a developer does not need to manage an entire server infrastructure but uses the existing infrastructure (managed by the cloud provider) and mounts single scripts and partial implementations into the provided system. These features can be triggered by a series of events like uploading a new file into a storage system or by any other system interaction. A significant benefit of serverless approaches is the extreme flexibility in scaling system resources to demand.

3.4 Summary

The concepts and protocols presented in the previous paragraphs provide a sound technical base to realize an open system that is applicable to any kind of information exchanged within a project. Architecting an open BIM cloud-based collaboration platform requires deep knowledge about both, capabilities of communication techniques as well as domain use cases and needs. Some methods and techniques are better suited for exchanging large files (when comparing REST and Sockets vs. MQTT) or establishing a bidirectional communication channel (Sockets vs. REST). The realization of these approaches would combine diverse design patterns, fulfilling the underlaying use case and purpose. Given the architectures and patterns presented in the previous paragraphs, it appears promising to combine the advantages of event-driven approaches, transferring only updates applied to models instead of entire monolithic data items.

4. Proposed concept

We describe an exemplary situation of two interacting domains to motivate the proposed concept, as depicted in Figure 1. The architect creates a design model and hands it over to the structural engineer. The structural engineer subsequently develops a structural model using the information delivered by the architect. Later, the architect modifies the architectural model by inserting a new balcony. This operation leads to a change of the slab geometry and a new door and railing insertion into the architectural model. As the update affects load-bearing elements, the structural engineer should get notified about the applied change because the structural application referenced it. To this end, the structural engineer needs to consider the impact of the applied architectural change on the structural model (e.g., the extended area for vertical loads on the balcony and horizontal loads on the railing). After modifying the structural model and re-running subsequent simulations, the structural engineer may send the results back to the project platform reporting the corresponding changes in the structural model.

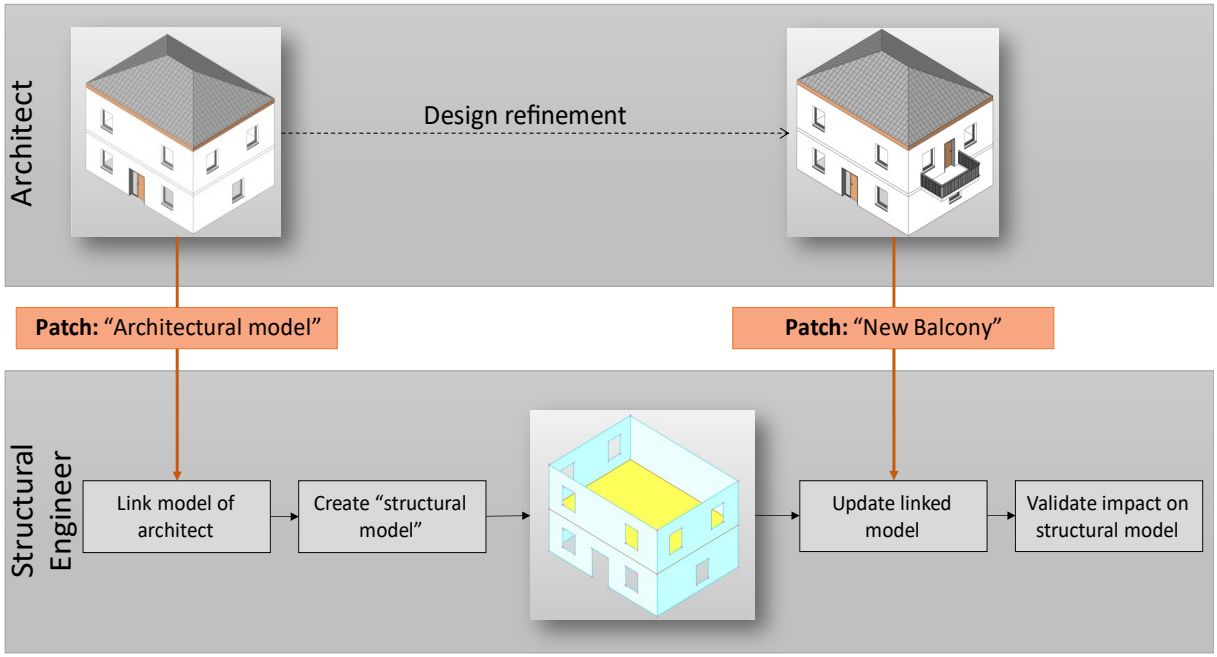

Figure 1: Exemplary situation of a patch-based model update between two domains.

4.1 Requirements resulting from cross-domain collaboration

Inspired by the current practice of BIM-based collaboration, its supplementary standards, and the outlined scenario, an event-driven CDE should meet the following set of criteria:

- Each domain continues to work in specialized authoring and simulation tools suited to fulfill assigned tasks. Accordingly, each domain publishes resulting model updates using update patches and remains full ownership over their respective domain models.
- New parties can join the project at any time and may subscribe to events that are particularly interesting for their tasks. Depending on the role and project setup, the project platform should provide appropriate access control and other business-related mechanisms.
- A copy of all discipline models is maintained on a centralized server, enabling the system to provide access to the latest state of each discipline model on-demand without the need to re-interpret the entire chain of change events. Additionally, the whole history of all changes communicated within the project is stored on the server to enable switching between several versions of BIM models.

4.2 Proposed solution

Combining the shortcomings of current CDE platforms and already existing techniques for distributed computing, the following architecture is proposed:
The publish-subscribe approach is applied to consider the distributed, asynchronous nature of design processes in the AEC industry while expressing necessary dependencies among various domains. Additionally, each involved party in the project can emit updates and subscribe to events relevant to its specific design and simulation tasks. A central server acts as a message broker, managing access rights, storing clients' subscriptions, and forwarding events according to their specified subscriptions. The server maintains a history of all raised modification events and supplies the latest model versions using HTTP-based interfaces. The data transfer between

each client and the server is realized through WebSocket connections offering bidirectional communication.

Each event consists of a topic and the patch content transferring the actual modification applied to a particular model. As design information produced and distributed during AEC projects varies in complexity and representations, the topic hierarchy must be agreed upon individually. Like other project-specific agreements, the concept of Employer's information requirements (EIR) and BIM execution plans (BEPs) are reasonable mechanisms to state these project-wide definitions. The topic of each event is then chosen according to this topic hierarchy and helps the server forward each event to the correct clients without analyzing the individual patch content. Figure 2 depicts an exemplary situation for a design project of a building.

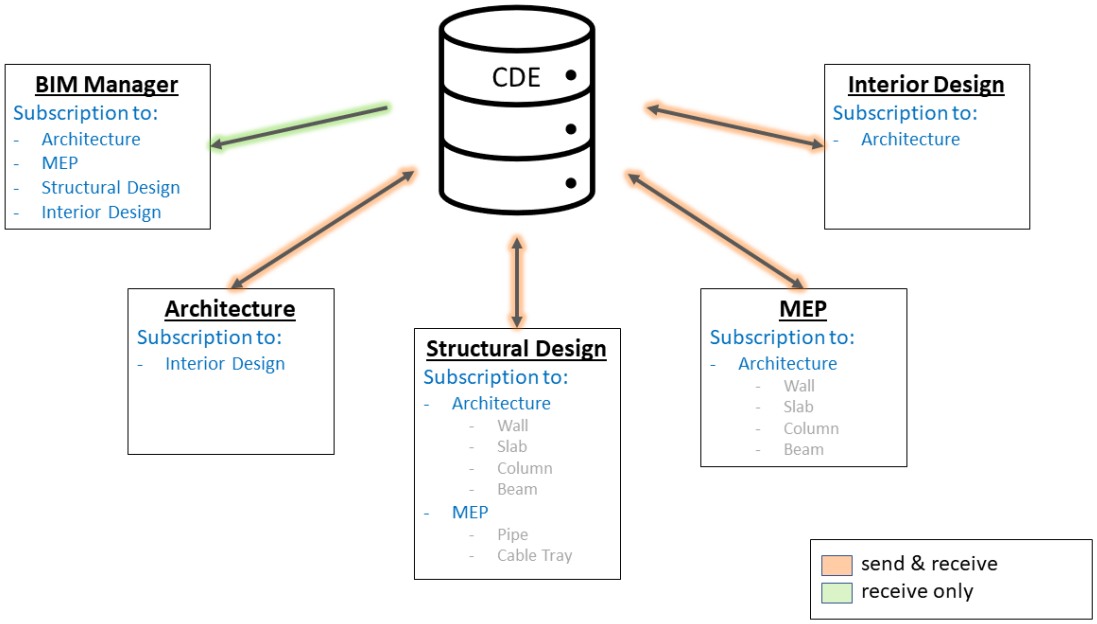

Figure 2: Topic subscription model for a building project consisting of various parties and diverse topic subscriptions

The subscription of a client happens either *implicitly* or *explicitly*. The implicit subscription method is envisioned as an automatic process if a domain requests a copy of a foreign discipline model for reference purposes. If an update affects these models, the collaboration system forwards the event to the subscribers. The update is applied to their local machines without further notice ("silent" integration). This way, it is ensured to keep all copies of a shared discipline model up to date at any storage location.

Contrary to implicit subscriptions, clients can express their explicit interest in specific object types or all objects within a logical container (e.g., building, building storey, system, etc.). Spatial and geometrical relationships within a BIM model could facilitate advanced subscription topics (e.g., all objects of any domain within a volume/box). For such cases, approaches like BIMQL and GraphQL may be employed to formally describe interests in a computer-readable manner (Daum & Borrmann, 2014). If a client receives an update of explicit interest, an alert is triggered, pointing the user to evaluate the event's impact on his domain tasks and models.

Additionally, explicit subscriptions help express interdisciplinary dependencies even though the essential principle of asynchronous project management is retained. Given the example in Figure 1, the structural engineer may subscribe to model changes affecting specific building elements such as walls, beams, columns, and slabs because of their load-bearing capacities. If

the architect adds the balcony and thus edits the geometric shape of the slab, the structural engineer receives a notification about the corresponding changes in the referenced architectural model. Subsequently, the engineer evaluates this modification in the architectural model and updates the structural model accordingly. After emitting another change event by the structural engineer transferring the changes applied to the structural model, all copies of any discipline model are synchronized again and reflect the latest state of the design.

5. Analysis of advantages and disadvantages of the proposed system

The proposed system features event-driven communication between clients and a centralized server. Synchronization on any desired time interval can be achieved. Intervals range from weekly synchronization of all models to an almost real-time collaboration if modification events are emitted immediately after applying changes in the authoring system. Letting the user choose the synchronization frequency according to specific needs enables a flexible yet controlled collaboration environment that supports agile and proactive project management.

Similar to the definition of suitable topic classifications, the choice of an appropriate synchronization interval depends on the individual project characteristics and is therefore difficult to define in a standardized manner. Furthermore, the transfer of changes on object-level instead of federating entire models demands dedicated checks to ensure the correct receipt of patches and their proper integration on all receivers. Besides integration verification, accompanying mechanisms are essential to indicate a specific version of foreign models as an engineer bases domain tasks on one state. Indeed, the proposed architecture drastically reduces the risk of making decisions on superseded models if a frequent information exchange is implemented in the project. However, it is of considerable benefit for conflict and clash resolution if tracing back design decisions and associated versions is possible. Therefore, the benefits of the proposed system outweigh the disadvantages, especially if all model replicas are synchronized frequently and users take design changes of foreign disciplines into account in their planning.

6. Summary and Outlook

Implementing event-driven exchange mechanisms can foster an intelligent and smooth BIM-based collaboration. It enables a quicker exchange of modifications applied to shared discipline models. Furthermore, it supports the understanding of whether and to what extent an update in foreign models impacts design tasks of a specific domain. Clients can register their interest in particular update scenarios by subscribing to events from similar or different domains. Each domain keeps the author's rights on its models and publishes patches that comprise the modification logic to a specific model. Thus, the established and well-adopted principle of asynchronous design environments in federated systems and the coordination of all domain models in regular time intervals remains unchanged. Nevertheless, the proposed approach reduces the amount of information a client must evaluate and provides notification mechanisms to highlight updates relevant to an individual client.

The authors will report on experiences and results after validating the proposed system in real-world projects.

References

Bormann, C., Castellani, A. P., & Shelby, Z. (2012). CoAP: An application protocol for billions of tiny internet nodes. IEEE Internet Computing, 16(2), 62–67. https://doi.org/10.1109/MIC.2012.29

CEN. (2018). DIN EN ISO 19650-2:2018 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling — Part 2: Delivery phase of the assets.

Chen, H. M., & Hou, C. C. (2014). Asynchronous online collaboration in BIM generation using hybrid client-server and P2P network. Automation in Construction, 45, 72–85. https://doi.org/10.1016/j.autcon.2014.05.007

Counsell, J. (2012). Beyond level 2 BIM, web portals and collaboration tools. Proceedings of the International Conference on Information Visualisation, 510–515. https://doi.org/10.1109/IV.2012.88

Curry, E. (2004). Message-Oriented Middleware. Middleware for Communications, 1–28.

Daum, S., & Borrmann, A. (2014). Processing of topological BIM queries using boundary representation based methods. Advanced Engineering Informatics, 28(4), 272–286. https://doi.org/10.1016/j.aei.2014.06.001

Esser, S., Vilgertshofer, S., & Borrmann, A. (2021). Graph-based version control for asynchronous BIM level 3 collaboration. EG-ICE 2021 Workshop on Intelligent Computing in Engineering, 98–107. https://doi.org/10.14279/depositonce-12021

Fielding R. (2000). Architectural styles and the design of network-based software architectures. https://www.ics.uci.edu/~fielding/pubs/dissertation

Karlapudi, J., Valluru, P., & Menzel, K. (2021). An explanatory use case for the implementation of Information Container for linked Document Delivery in Common Data Environments. EG-ICE 2021 Workshop on Intelligent Computing in Engineering, 76–86.

Light, R. A. (2017). Mosquitto: server and client implementation of the MQTT protocol. The Journal of Open Source Software, 2(13), 265. https://doi.org/10.21105/joss.00265

Preidel, C., & Borrmann, A. (2015). Automated Code Compliance Checking Based on a Visual Language and Building Information Modeling. ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction, 1–8.

Sattler, L., Lamouri, S., Pellerin, R., Fortineau, V., Larabi, M., & Maigne, T. (2020). A query-based framework to improve BIM multi-domain collaboration. Enterprise Information Systems, 00(00), 1–23. https://doi.org/10.1080/17517575.2020.1845810

Schulz, O., & Beetz, J. (2021). Image-documentation of existing buildings using a serverbased bim collaboration format workflow. EG-ICE 2021 Workshop on Intelligent Computing in Engineering, 108–117.

Senthilvel, M., Oraskari, J., & Beetz, J. (2021). Implementing Information Container for linked Document Delivery (ICDD) as a micro-service. EG-ICE 2021 Workshop on Intelligent Computing in Engineering, 66–73.

Shafiq, M. T., Matthews, J., & Lockley, S. (2012). Requirements for model server enabled collaborating on building information models. First UK Academic Conference on BIM, 5 -7 September 2012, Northumbria University, Newcastle upon Tyne, UK, 23–35. https://doi.org/10.1108/17410391111097438

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 (ITcon), 18(18), 148–161. http://www.itcon.org/2013/8

Solihin, W., Eastman, C., & Lee, Y. C. (2016). A framework for fully integrated building information models in a federated environment. Advanced Engineering Informatics, 30(2), 168–189. https://doi.org/10.1016/j.aei.2016.02.007

Zhang, C., Beetz, J., & Weise, M. (2015). Interoperable validation for IFC building models using open standards. Journal of Information Technology in Construction (ITcon), 20 (Special issue ECPPM 2014-10th European Conference on Product and Process Modelling), 24–39. https://doi.org/10.18154/RWTH-CONV-213541