2025 European Conference on Computing in Construction

CIB W78 Conference on IT in Construction

Porto, Portugal
July 14-17, 2025

UTILIZING INTERDISCIPLINARY OBJECT DEPENDENCIES FOR SEMI-AUTOMATED MODEL UPDATES

Sebastian Esser[1,2,3] and André Borrmann[2,3]
[1]Chair of Computational Modeling and Simulation
[2]Chair of Computing in Civil and Building Engineering
[3] TUM Georg Nemetschek Institute
School of Engineering and Design, Technical University of Munich, Germany

Abstract

Model-based collaboration in the AEC domain lacks seamless cross-disciplinary interaction when updates impact foreign models. This paper investigates how to integrate model updates across disciplines by employing graph transformations to capture, describe, and interpret changes, then suggest corresponding updates to facilitate changes authored in foreign domains. Our approach automates repetitive, predictable modifications and presents them as actionable proposals. Consequently, the workflow reduces model authors' workload by handling tedious tasks, allowing them to focus on complex design coordination and enhancing productivity. The proposed system architecture offers a scalable framework adaptable to various synchronization cycles, providing a foundation for future research.

Introduction

In recent years, remarkable advances have been witnessed in interdisciplinary model coordination within the Architecture, Engineering, and Construction (AEC) sector, mainly through integrating Building Information Modeling (BIM) techniques into collaborative design methods. These advancements have led to the widespread adoption of collaboration principles documented in regulations such as ISO 19650 and PAS 1192 in many collaboration platforms, known as Common Data Environments (CDEs).Such applications enable model-based data exchange between disciplines and offer additional management tools. Design information is predominantly recorded in discipline models, which users manage as monolithic files. Upon completion, the single-discipline models are federated using CDE platforms and can be utilized by other disciplines and stakeholders in the project. If changes must be applied to a model, a new model file is created and shared in the project space. This approach presents several drawbacks, including inadequate tracking mechanisms at the object level and a lack of effective methods for implementing fine-grained notifications when updates impact multiple disciplines.

Researchers have increasingly worked on overcoming file-based data exchange and recently presented new approaches to version management of BIM models based on their underlying object networks (Wang et al., 2023; Esser et al., 2022). These approaches have reduced the amount of data exchanged in case of model updates. In addition, they claim to provide a more streamlined integration of model updates in interdisciplinary setups. Sensitive notifications on objects affected by updates from foreign disciplines can accelerate conflict resolution. They can help point the model author in charge to investigate only those parts of the model that have been modified. While these approaches enhance progress and decrease the manual effort required to identify dependent model elements across disciplines, the assessment of model changes and their subsequent impact on other discipline models still depends on the individual responsibility of each model author. Additional considerations can be performed to extend version control principles by proposing sufficient model modifications based on foreign changes to further automate the consumption of incoming foreign discipline updates.

Contribution

This paper presents a design support concept to assist model authors in identifying and resolving conflicts based on design changes issued in foreign discipline models. The proposed framework primarily targets simple yet repetitive and time-consuming model updates, such as the interaction between duct systems of HVAC and MEP disciplines and the modification of adequate openings in architectural models. While the system's current capabilities are limited in handling complex scenarios, it provides a significant opportunity to automate mundane tasks, freeing up model authors to focus on complex design challenges.

At the framework's core, users can specify events relevant to their design or planning activities. By leveraging principles of event-driven database systems, event patterns, application conditions, and respective modification actions can be defined and applied to specific domain components. This enables users to filter events exchanged within a project and establish effective update routines based on incoming information from other disciplines, streamlining the evaluation of the impact of external model updates.

The framework utilizes available information about model changes in foreign disciplines to identify affected elements and define boundary conditions for semi-automated execution. Specifying conditions and actions provides a technical means to semi-automatically update a discipline

model based on change information retrieved from other domains. A case study outlines the feasibility of the proposed concept. Identified limitations and potential future research directions are also discussed at the end of this paper.

Related Works

The following paragraphs briefly summarize recent initiatives related to object-based version control of BIM models, which provides the basis for a direct interpretation of model changes. Furthermore, principles of active database management systems with capacities to handle events and triggers are introduced. They provide promising notions to be incorporated to directly consume incoming updates of foreign discipline BIM models.

Change propagation, model healing, and version control of BIM models

The design and planning of civil and building assets are inherently iterative, as the outcomes must satisfy various boundary conditions, such as environmental and economic constraints. Achieving an overall consistent design meeting all stakeholders' demands often relies on consistency and clash detection checks conducted on coordinated models that integrate all partial domain models. As mentioned before, version and revisions of discipline models are predominantly managed using monolithic model files, which are exchanged repeatedly with each new model iteration. This approach presents several drawbacks, including the redundant transfer of information already shared in earlier versions and the inability to directly analyze or reason on the specific changes made to a discipline model.

To address these challenges, Esser et al. (2022) introduced version control mechanisms based on graph representations of BIM models, enabling a more granular and efficient approach to manage model updates. In essence, they transform the complex, highly interconnected object structures of BIM models into Labeled Property Graphs (LPGs) to overcome artifacts induced by serialization processes. A Maximum Common Subgraph is then determined based on the graph representations, representing the model's unaltered parts. Any nodes and edges not part of this graph are either considered removed (if they appear in the initial model version) or newly inserted (if nodes and edges are only present in the updated version of the model). These identified changes are then assembled into a graph transformation rule, which is exchanged as an incremental description of the model change. Receiving units can apply the transformation rule on the outdated model version or its graph, resulting in the updated version. The underlying principles will be further introduced later.

The propagation of proposed design changes has also been explored in other contexts using BIM techniques. For instance, Wu et al. (2025) investigated a design healing method that adapts inconsistent models to predefined routines. This method employs a graph-based propagation mechanism, utilizing predefined design parameters and constraints to progressively identify essential building subparts for code compliance. Forth et al. (2023) examined the enrichment of BIM models with additional semantic information relevant to Life Cycle Assessments and associated decision-making. While the BIM Collaboration Format (BCF) was used in the latter case to capture design decisions and transfer the necessary modification information back into BIM authoring systems, neither approach employed a comprehensive model increment description that could be directly applied in receiving applications.

Event-based Database Systems

In numerous applications and domains, vast volumes of data are generated and exchanged as continuous data flows. To manage and utilize this data effectively, various data management systems serving different data types have been developed to store information and provide access through query interfaces. These data flows can vary significantly in size and timeliness. However, in domains such as production lines, traffic control, or stock exchanges, it is critical to enable immediate reactions to extraordinary events (Geisler, 2013). Classical Database Management Systems (DBMS) have been found inadequate for real-time processing, as these systems typically store and index incoming data before making it available for queries, introducing delays incompatible with real-time requirements. To address this limitation, extensive research has been conducted in the field of Information Flow Processing (IFP), which has its foundation in the works of Dittrich et al. (1995). They introduced *Event-Condition-Action* model as the core component for Active Database Management Systems (ADBMS). The key idea is summarized as follows: When an event is triggered, the associated condition is evaluated, and if the condition is satisfied, the corresponding action is executed.

Cugola and Margara (2012) identify two approaches to implementing Information Flow Processing (IFP) as a technical solution to handle events transiently. First, *Data Stream Management Systems* (DSMS) are designed to handle transient data streams by executing queries continuously and providing updated results as new data arrives. Second, Cugola and Margara (2012) highlight the *Complex Event Processing* (CEP) model, which filters and aggregates low-level events into higher-level abstractions that can be distributed to relevant stakeholders. These two approaches are not mutually exclusive, as combinations of DSMS and CEP can be employed to address diverse application needs. Today, several solutions implementing CEP principles are available on the market, with Apache Kafka being one of the most prominent (Garg, 2013; Shapira et al., 2022). These solutions are based on the general publish-subscribe pattern, where incoming messages are distributed to multiple receivers. Clients can specify a topic or channel within their message, which is then used to route the message to subscribers who have expressed interest in that specific topic.

Milosevic et al. (2016) underscore the advantages of event-based systems in real-time and near-real-time application scenarios. At the same time, they note that the fundamental principles of event detection and subsequent action execution can also support use cases with less stringent timeliness requirements. Consequently, these core concepts align well with the challenge of processing and consuming incoming model updates from foreign disciplines, offering a viable framework for addressing this problem.

Identified research gap

The literature review has identified a range of activities related to object-based version tracking in BIM models. While some studies discuss the consumption of identified changes, there appears to be significant potential to further streamline the process of incorporating model updates from foreign domains. This could be achieved by transforming these updates into actionable design proposals tailored to a specific discipline model. The event-condition-action paradigm, as introduced by Dittrich et al. (1995), emerges as a promising approach for implementing an immediate cross-domain update mechanism within a BIM application that receives updates from external disciplines.

Proposed Methodology

The proposed approach builds upon preliminary work in object-based version control designed to manage BIM models. It assumes that project coordination is carried out using vendor-neutral data models, such as the Industry Foundation Classes (IFC) or other related product models. Each discipline involved in a collaborative workflow creates dedicated discipline models, which are then shared with other disciplines for coordination. The following sections outline key aspects of the preliminary work that form the foundation for developing an event-driven update distribution system. Ultimately, a proposal is presented for how a client can process incoming updates from external domains, ensuring overall design consistency is maintained.

Preliminary Work

The work published in this paper builds upon a series of recent publications, most importantly (Esser et al., 2022, 2023; Esser and Borrmann, 2025). As introduced in the previous sections, various researchers have recently employed Labeled Property Graphs (LPGs) to represent interconnected information of a built environment domain. This type of semantic graph closely aligns with the fundamental principles of object-oriented data modeling, which underpin most data models currently used in the AEC sector. In the approaches presented by Esser et al. (2022) and Zhu et al. (2023), each class instance corresponds to a single node, with instance-specific attributes stored as properties attached to the node. Directed edges in the graph represent associations between instances. In addition to translating IFC models into graph representations, they introduced a comprehensive method for detecting changes in BIM model versions using LPGs, formalizing these changes into update patches, and applying them to outdated model versions. Figure 1 illustrates the approach to derive and apply version increments based on graph representations.

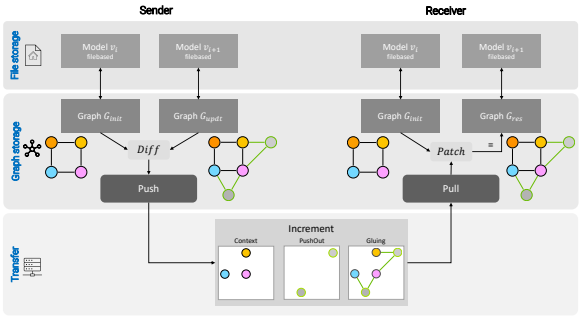

Figure 1: Overall Approach for the object-based version control system proposed in Esser et al. (2022)

Three graph patterns are used to define the version increments. The context pattern describes a subgraph in the host graph that must be matched to properly (un-)connect the removed or inserted graphlet with the unaltered parts of the model (i.e., the graph). The PushOut pattern represents the inserted and removed nodes and edges corresponding to the added and deleted objects in the BIM model. Finally, the gluing pattern describes the edges that link the push-out pattern to the host graph. If only the properties attached to a node are edited, these modifications do not impact the graph's topology. Instead, this information is handled as a semantic modification, which consists of a unique path to unambiguously identify the modified node and the attribute data to be updated. Figure 2 presents the data model for increments expressed in the Unified Modeling Language (UML).

Event-Condition-Action notion for Collaboration Hub

As previously mentioned, there is limited research in the literature on delivering sensitive notifications to disciplines that a model update may impact. Missing relevant updates can lead to an inconsistent overall design, as model updates might conflict with other discipline models developed based on earlier versions of the modified model. Illustrative examples of such design events include, among others, the insertion and modification of objects that must be allocated to other domain niches or spaces. High manual effort is required in the classical file-based collaboration approach to identify such demands. To address this gap, object-based version control combined with event-driven data distribution systems can be a potential solution, extending the Transfer layer illustrated in Figure 1. Figure 3 presents the system architecture for managing interactions between multiple domain clients via a centralized coordination hub. Each domain client is equipped with a set of version control commands. These actions comprise creating new version increments, sending them to the hub, and pulling them from the coordination hub.
The hub implements the principles known from Common

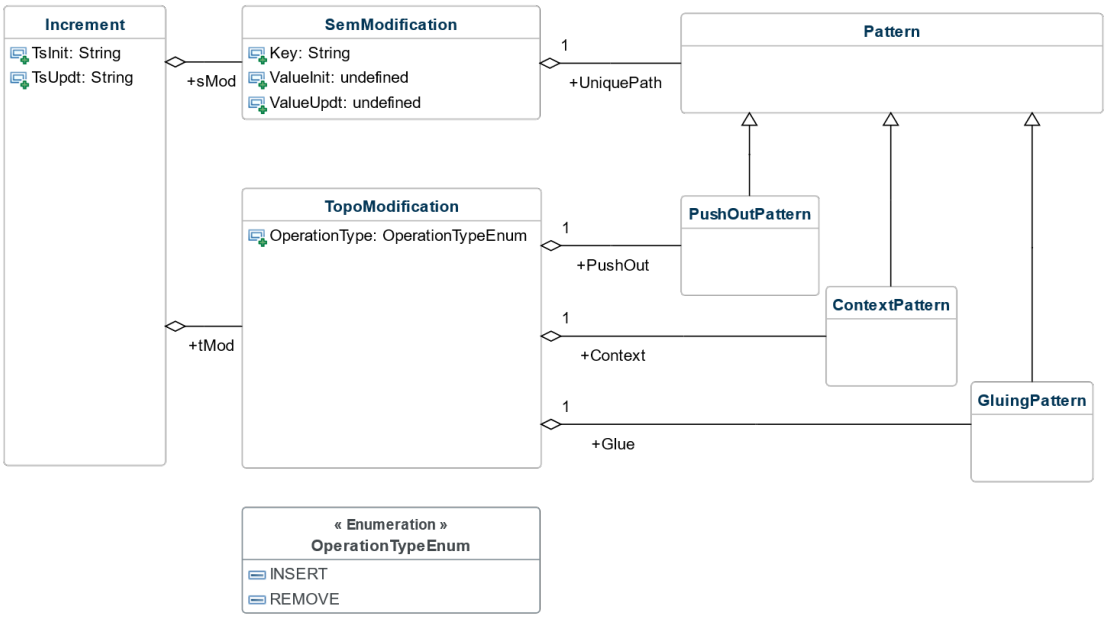

Figure 2: The data model used to describe model changes through graph transformations. Semantic modifications capture changes in node properties, while topological modifications describe the insertion or removal of subgraphs representing deleted or newly created model objects.

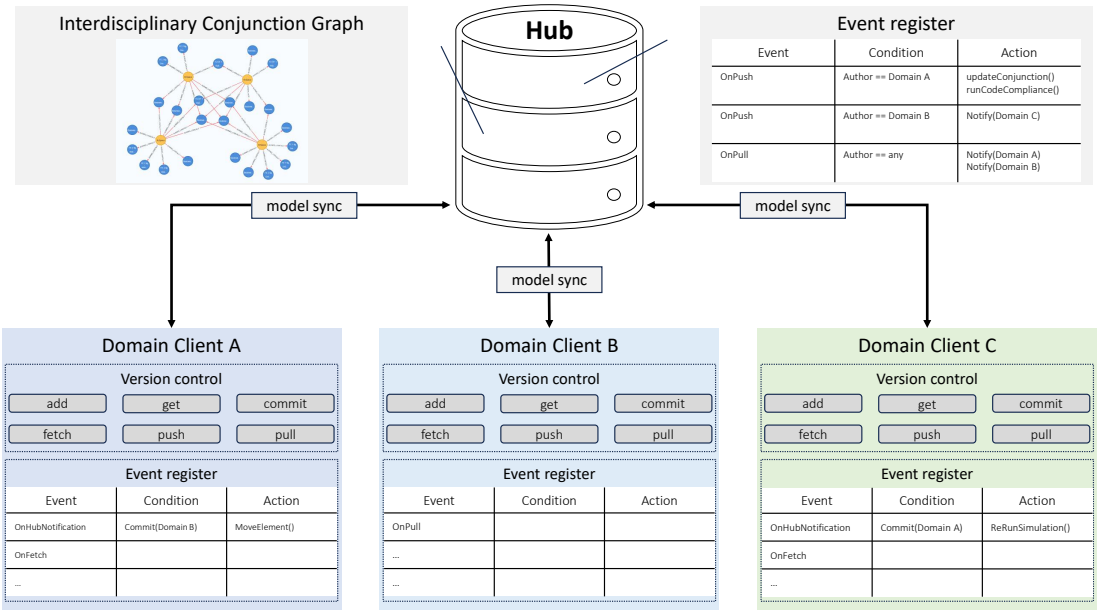

Figure 3: Proposed event registers for clients and the central collaboration Hub. Events capture certain actions relevant to a subsequent action, where conditions are employed to filter and streamline events based on respective features.

Data Environments (CDEs) and extends them by various additions required for managing model versions on object level. All models pushed to the hub are available in a graph store and can be handed out to the user as a graph representation or in a serialized form according to the underlying data model directives. The coordination hub employs two additional concepts to implement event-based principles: an event register and an Interdisciplinary Conjunction Graph (ICG). The event register stores and manages event-condition-action rules relevant to the overall project coordination. All events are defined according to the notion of event-condition-action. An event can essentially be any action performed on the coordination hub, including push, fetch, and pull events and any other user interaction on the hub itself (e.g., changes in user permissions, registration of new participating domains, acceptance workflows). Further conditions can be defined depending on the events in question to filter event types, authors, and event contents and trigger respective actions. Conditions can be concatenated using the boolean expressions AND,

OR, and NOT. Subsequently, actions can initiate processes on the hub itself or raise other events on the hub. One action can comprise one or multiple functions or processes to be performed. Hub actions are also employed to send sensitive notifications to the discipline clients, informing them about processes carried out on the hub. Users can submit event-condition-action statements to the hub's event register to realize the retrieval of sensitive notifications.

By assuming collaboration based on distinct federated discipline models, further emphasis is directed to realizing cross-domain updates or, more precisely, the notification of relevant updates issued in foreign domains. The coordination hub has an Interdisciplinary Conjunction Graph (Esser and Borrmann, 2025). This graph integrates all discipline models pushed to the coordination hub, forming unconnected partial graphs in the graph database. Interdisciplinary relationships are then added, connecting the different discipline graphs based on interdisciplinary relationships. These additional dependencies can arise from different representations or abstractions of an object that appear in different discipline models. Furthermore, objects with functional dependencies may also be connected in the ICG. Possible examples include, among others, the linkage of installed components such as pipes or other building services systems with corresponding openings and voidings. Cross-domain relationships are computed using coordination tools such as Solibri or DESITE BIM, which facilitate identifying and managing dependencies, conflicts, and shared elements across disciplines.

Increment utilization on Domain Client

Similarly to the coordination hub, clients are equipped with an event register as well. Here, the events to be processed are mainly triggered by notifications sent from the hub to a dedicated domain. Like before, conditions support further filtering and the setup of fine-grained actions related to the specific event content. Lastly, the subsequent actions are implemented according to the specific needs of the client, its role in the project, and the tools employed to interact with the exchanged data in the project. Each action can be implemented into any application interacting with the discipline-specific BIM models and act according to the respective use case performed in this application. Most obviously, actions in authoring tools can alter a discipline model. However, further actions can be defined as triggers for subsequent model data processing. Hence, advanced use cases like quantity take-off, clash detection workflows, or fabrication planning may benefit accordingly. The latter actions are not further reflected in this paper and may be discussed in future publications.

Human in the loop

Although fully automated model updates can be triggered by changes in external disciplines, human intervention remains essential for approving or rejecting any design modifications proposed by the predefined actions. Consequently, model authors retain complete control over the design information within their discipline models and any subsequent processes utilizing these models. Adequate visualization and presentation of proposed design changes are expected to be critical for successful adoption. When information takeoff workflows (e.g., quantity takeoff for tendering) are well-defined, proposed model changes should be presented alongside key indicators that illustrate the impact of these changes on derived documents. These indicators can reflect implications on costs and provide a quantification of change impacts, providing the user with valuable insights and enabling informed decision-making regarding the proposed updates.

Case Study

Figure 4 depicts a scenario in which interdisciplinary model updates play a crucial role in maintaining the consistency of the overall coordination model. The case study focuses on coordinating an architectural model and a duct system. Initially, both models are consistent with each other, with the architectural model providing adequately sized voids in the walls to accommodate the planned routing of the duct system. However, as the duct system undergoes modification, a shift occurs: one duct is repositioned, which in turn necessitates the adjustment of a duct fitting and a corresponding change in the geometry of the associated duct.

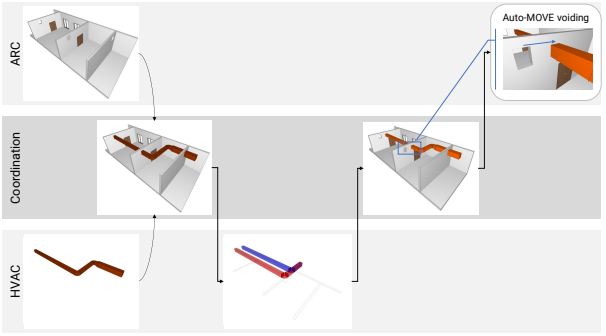

Figure 4: Initial consistency between two discipline models. Modifications to the HVAC model necessitate further changes in the architectural model.

The change is captured in an increment issued by the HVAC domain. Among some topological modifications, a set of semantic modifications is identified, of which the most relevant are denoted in Table 1. The corresponding unique paths required to unambiguously identify the altered nodes are illustrated in Figure 5.

The project coordination has established conjunction edges on the collaboration hub between the pipes and the wall voids (as schematically illustrated in Figure 6).

Upon receiving the HVAC update, the hub identifies modified HVAC elements described in the pushed increment, which affect dependent elements in the architectural model. The affected elements are identified using the *PassesThrough* relationships depicted in yellow in Figure 6. Consequently, the hub raises a notification and sends it to the architectural domain. Besides the model increment

Table 1: Extract of the semantic Modifications caused by the HVAC model update

Node ID	Key	ValueInit	ValueUpdt
122	Coordinates	(15301.83, 5216.73, 2850.00)	(15301.83, 6716.73, 2850.00)
48	Coordinates	(8733.55, 5216.73, 2600.00)	(8733.55, 6716.73, 2600.00)

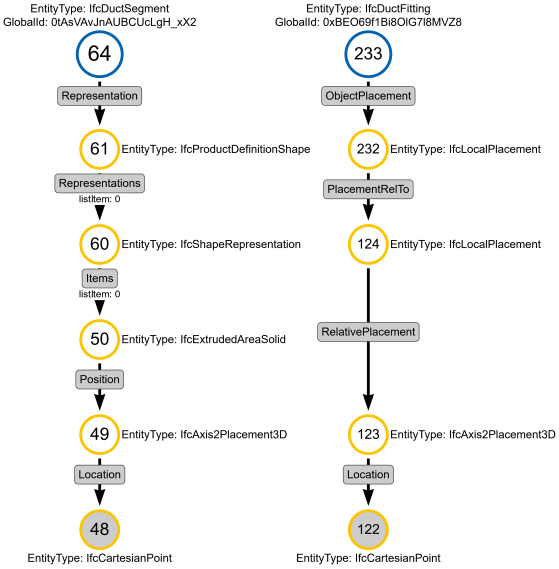

Figure 5: Extract of the UniquePath pattern to identify notes the semantic modifications must modify. The altered nodes are colored in light grey.

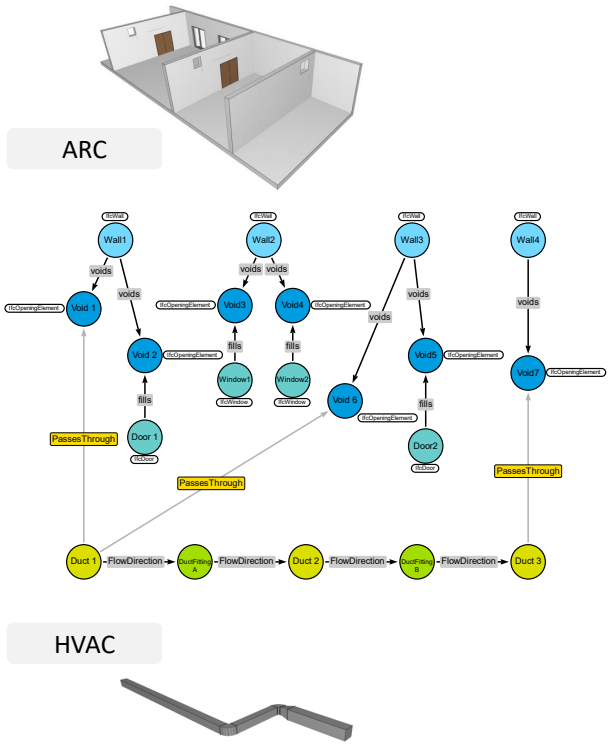

Figure 6: Extract of the Interdisciplinary Conjunction Graph connecting the provisioning voids of the architectural model with the duct system maintained by the HVAC domain.

updating the HVAC model, a list of affected elements from the architectural domain is contained in this notification.

Upon receiving the notification, the architectural model must be edited accordingly to reach an overall consistent state of all discipline models again. In the exemplary system, the natural engineering decision is likely to move the voiding of the left and the middle wall by the distance the duct was moved in the HVAC model. The required shifts in the x and y direction can be extracted from the IfcDuctSegment (node 64) and the IfcDuctFitting (node 233), which only received a positioning update, resulting in $\Delta = (0.00, 500.00, 0.00)$ for the exemplified case. The geometric change in the duct segment perpendicular to the moved element has resulted in a topological modification and is not further depicted here.

As an element shift can be a repetitive operation, an event is added in the event register of the architectural domain, which proposes a corresponding update for the voided elements that provide space for the duct system. Figure 7 illustrates the event processing within the architectural domain. The event information is forwarded from the collaboration hub and includes the foreign increment, along with details about the affected components in the architectural model. This latter information is derived from the Interdisciplinary Conjunction Graph. The condition block defines a filter to determine whether the issued increment involves a semantic modification of conjunct ducts that impacts their Cartesian coordinates. The condition is expressed as a pattern that must be matched against the uniquePath values of all incoming semantic modifications.

As illustrated in Figure 5, a position change of elements can be represented in multiple ways, complicating the definition of robust condition statements. Here, a variable path length is utilized to identify changes in the location of an object. Subsequently, the translation vector is computed to define the corresponding action. The logic required to perform the model alteration in the architectural domain is straightforward, involving the identification of the affected elements and the application of the appropriate geometric operation, expressed in pseudocode in Figure 7.

Limitations

Declarative Patch Formulation

The proposed framework is based on update increments generated by comparing two versions of a discipline model. While the version control approach ensures accurate and lossless updates to outdated models, the content of each increment can vary due to the wide range of possible graph transformations that may arise when comparing

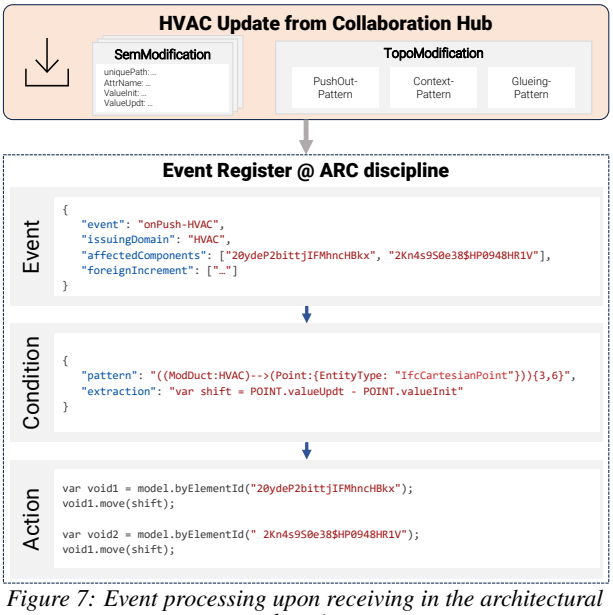

Figure 7: Event processing upon receiving in the architectural domain.

the outdated and latest versions of the model. This variability is primarily driven by the conditions under which nodes and edges are considered modified, leading to their exclusion from the computed Maximum Common Subgraph. The diversity of potential update increments describing one engineering action complicates the efficient definition of corresponding pattern statements in the event condition sections. Even minor object changes, such as shifts in position, can be represented either as semantic modifications, as previously described, or as topological removal and insertion operations within the subgraph representing the object's position. To address this complexity, a more imperative formulation of design changes may be beneficial. Rather than describing graph alterations through graph transformation rules, more model-centric statements—such as 'HVAC Element xy has been shifted'—could expedite the understanding of updates for the consuming discipline client.

While there is no limitation on the number of defined events in the event register, an increasing number of event definitions poses additional challenges in maintaining an overview of all actions triggered by change events. Moreover, as the number of events corresponding to various engineering decisions grows, there is an increased risk that users may define actions that can result in contradictory resolution proposals. This consideration emphasizes keeping human understanding in the loop. The outlined system provides full power to create modification proposals for easy-to-understand model changes, giving users more time to invest in advanced and complex situations.

Integration in Consuming Application

The definition of subsequent actions depends on the selected authoring or simulation application. As a result, recommending a universal language for expressing these actions is challenging. Visual Programming Languages (VPLs) such as Dynamo or Grasshopper offer a suitable approach for design modifications intended for a BIM authoring tool, enabling end-users with minimal programming experience to create the necessary routines and actions. However, as such tools may not always be available, more advanced considerations are required based on the programming interfaces provided by the software tools used for project design and associated simulations. Figure 8 depicts the implemented action to move the architectural model's voidings according to the pipe's extracted shift, whose movement has been issued in the HVAC domain.

Summary and Outlook

The proposed approach supports the integration of incoming updates from external disciplines within a model-based collaboration environment. At its core, the approach focuses on incremental model updates to synchronize federated models based on the contained objects. To streamline the overall communication inside the project, users can register for events on the central collaboration hub for which they wish to receive notifications. In addition to the hub's event register, users can define custom events and associated actions for their client-side design environment.

In both cases, events function as triggers that activate specific actions, while conditions provide additional filtering and fine-tuning capabilities. Actions defined on the collaboration hub are primarily used to send detailed notifications to disciplines that have subscribed to specific activities requiring stakeholder interaction. Additionally, each connected client features a local event register to define custom workflows triggered by incoming updates and notifications from external disciplines. The primary purpose of this local event register is to propose updates to discipline models based on incoming changes from other disciplines.

The system primarily targets the processing of straightforward events, such as object movement or rotation, modifications to object attributes or material definitions, and basic geometric changes. These events facilitate transparent and manageable coordination of updates across disciplines. The framework is built for scalability, accommodating the timeliness and complexity of the events and actions involved. In future considerations, the system could potentially be operated in near-real-time, enabling one domain to issue updates that are processed and reflected in the design of another domain. This approach would allow potential changes to be tested with immediate feedback from other domains, alleviating the need for basic negotiations among model authors. Moreover, this scalability ensures the system can adapt to various project sizes and requirements while maintaining efficient coordination and consistency. Ultimately, any proposal the system may provide to the user, the proper authority and responsibility remain always with the model author.

Looking ahead, more complex update actions will be necessary to propose advanced model alterations. Addressing these more intricate scenarios and prioritizing compet-

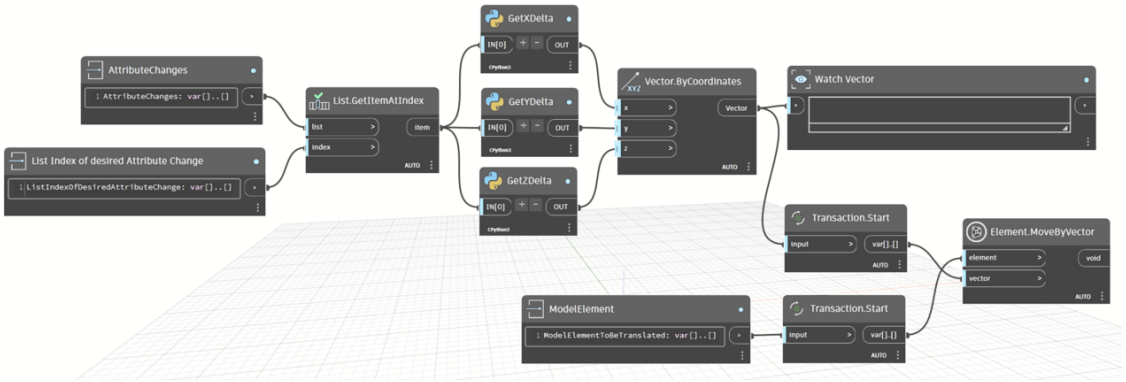

Figure 8: A Dynamo Script to move voids in the architectural model based on the incoming HVAC model update.

ing design paradigms may require sophisticated methods, which the authors plan to explore in future publications.

Use of AI tools

While preparing this work, the authors used ChatGPT and Writefull to enhance the language and detect spelling errors. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

Acknowledgments

The Federal Ministry of Education and Research in Germany (Deutsches Bundesministerium für Bildung und Forschung) funded the work underlying this publication under grant number 02K23A096 (BauPuls360). The responsibility for the content of this publication lies with the authors.

References

Cugola, G. and Margara, A. (2012). Processing flows of information: From data stream to complex event processing. ACM Computing Surveys, 44:1–62.

Dittrich, K. R., Gatziu, S., and Geppert, A. (1995). The active database management system manifesto: A rulebase of ADBMS features. In Rules in Database Systems, pages 1–17, Berlin, Heidelberg. Springer Berlin Heidelberg.

Esser, S. and Borrmann, A. (2025). Enhancing Design Coordination Across Disciplines Through Incremental Model Updates and Inter-Discipline Conjunction Graphs. In Francis, A., Miresco, E., and Melhado, S., editors, Advances in Information Technology in Civil and Building Engineering, pages 77–90, Cham. Springer Nature Switzerland.

Esser, S., Vilgertshofer, S., and Borrmann, A. (2022). Graph-based version control for asynchronous BIM collaboration. Advanced Engineering Informatics, 53:101664.

Esser, S., Vilgertshofer, S., and Borrmann, A. (2023). Reference framework enabling temporal scalability of object-based synchronization in BIM level 3 systems. In 2023 European Conference on Computing in Construction, page 177, Heraklion.

Forth, K., Abualdenien, J., and Borrmann, A. (2023). Calculation of embodied ghg emissions in early building design stages using bim and nlp-based semantic model healing. Energy and Buildings, 284:112837.

Garg, N. (2013). Apache Kafka. Packt Publishing. ISBN: 9781782167938.

Geisler, S. (2013). Data Stream Management Systems. In Kolaitis, P. G., Lenzerini, M., and Schweikardt, N., editors, Data Exchange, Integration, and Streams, volume 5 of Dagstuhl Follow-Ups, pages 275–304. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany.

Milosevic, Z., Chen, W., Berry, A., and Rabhi, F. (2016). Real-Time Analytics. Big Data, pages 39–61.

Shapira, G., Palino, T., Sivaram, R., and Petty, K. (2022). Kafka: The Definite Guide: Real-Time Data and Stream Processing at Scale. O'Reilly Media, 2nd edition edition. ISBN: 9781492043089.

Wang, Z., Ouyang, B., and Sacks, R. (2023). Graph-based inter-domain consistency maintenance for BIM models. Automation in Construction, 154:104979.

Wu, J., Nousias, S., and Borrmann, A. (2025). Design Healing framework for automated code compliance. Automation in Construction, 171:106004.

Zhu, J., Wu, P., and Lei, X. (2023). IFC-graph for facilitating building information access and query. Automation in Construction, 148:104778.