Technical University of Munich

TUM School of Engineering and Design

Chair of Computational Modeling and Simulation

An Object and History-based Approval method for MEP Slot and Opening planning in openBIM projects using a Database-driven workflow

Master Thesis

for the Master of Science program in Civil Engineering

Author: Ganga Surendran Sanila

Student number: [redacted]

Supervisors: Prof. Dr.-Ing. André Borrmann

Fritz Beck M.Sc.

Alexander Hofbeck M.Sc.

Florian Gschwind M.Eng.

Date of issue: July 26, 2021

Date of submission: February 9, 2022

Preface

At this point, I am glad to say that it was with great motivation and pleasure to have worked on this master thesis. I would like to extend my gratitude to all those who supported me during the preparation of my master's thesis.

I would like to thank Prof. Dr.-Ing. André Borrmann and the Chair of Computational Modelling and Simulation, and my supervisor Mr. Fritz Beck who always provided me with timely and valuable suggestions.

I would like to thank my supervisors, Mr. Alexander Hofbeck and Mr. Florian Gschwind, for their consistent support and for patiently answering my countless questions regarding practical aspects of slot and opening coordination. Working on this thesis helped me to enhance my domain knowledge and was an important piece to the puzzle of improving this coordination process.

I would like to thank Mr Fabian Matschinsky (HL-Technik) for explaining the process and its challenges and evaluating my concept and prototype from the viewpoint of an MEP planner. Also, I would like to thank Mr. Nico Neumann (Vielmo Architekten), who explained the process and challenges and provided me valuable suggestions to the concept from an Architect's perspective. Furthermore, I would like to thank Ms. Liis Köster and Ms. Angela Feldmann (Bollinger und Grohmann GmbH) for explaining the process and giving valuable insights from the perspective of a structural designer.

Furthermore, I would also like to thank Bollinger+Grohmann GmbH for supporting me and giving me the chance to write this thesis topic.

Finally, I would like to thank my family and friends who were always there with support and care.

Abstract

The planning of slots and openings for MEP services is one of the most inefficient, time-consuming and highly co-dependent tasks in the AEC industry. This communication-intensive coordination process is often conducted between architectural, structural and MEP disciplines either in a plan-based or model-based approach. The existing approval methods for slots and openings coordination require resending and comparing files or models to communicate a single void object (‘Provision for Void’ object or PfV object). This leads to inconsistency in planning, causes confusion, and limits reliable history tracking of objects. The thesis developed an object-based openBIM concept for slot and opening coordination using a database-driven workflow with a ‘single source of truth’ approach. This eliminates the need for resending files or models to communicate a single PfV object. In addition, it overcomes the limitations of pure file-based versioning for tracking previous approval decisions and changes of PfV objects. The concept facilitates object-based coordination with the most up-to-date information of PfV objects and object-level history tracking. Thus, the concept can speed up the slot and opening coordination and helps to maintain consistency through a supported approach. This results in improving design quality and providing planners with a more efficient method for slot and opening planning in BIM projects.

Table of contents

List of Figures

List of Tables

1 Introduction

The building service systems, including fresh water and wastewater piping networks, electrical installation networks, air conditioning systems, fire prevention and protection systems, and internal and external communication systems, can be summarized as Mechanical, Electrical, and Plumbing or MEP (Abdelhameed and Saputra, 2020). During the design of any construction project, these systems are critical components with tremendous impacts on the other design disciplines such as architectural, structural, and more (regarding aesthetics, accessibility, loads, fire regulations etc.). Moreover, defining MEP components' location and routing often requires intensive coordination between the involved disciplines to avoid clashes and comply with diverse design, construction, and operational criteria (Barton, 1983). BIM Method has significantly transformed how this coordination task is being carried out from a traditional drawing-based approach (Akponeware & Adamu, 2017). However, many professionals from the AEC industry have cited MEP coordination as one of the most challenging and least defined aspects of the design task encountered in a project delivery process (Korman et al., 2003; Riley et al., 2005; Ashuri et al., 2014). With advancements in technology and higher functionality requirements from building users, MEP systems have grown in scope and complexity, demanding effective multidisciplinary coordination.

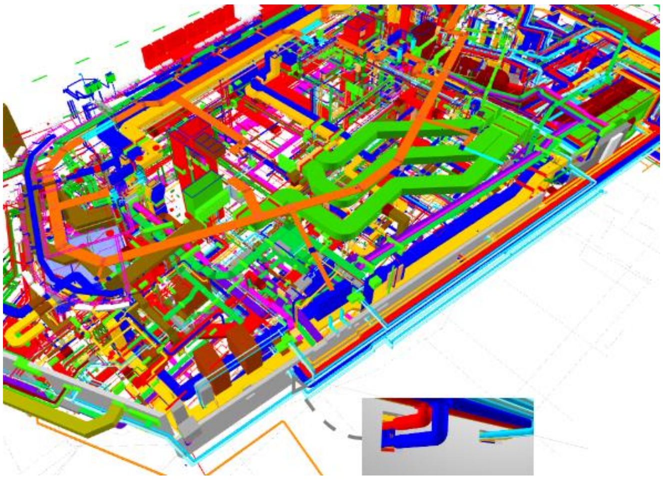


Figure 1.1: The MEP systems in part of a complex BIM Project (Bollinger und Grohmann GmbH, 2022)

In order to install MEP elements such as pipes or ducts, electrical service elements and more in building components, it requires slots and openings as cut-outs in the host elements (see Figure 1.2). An opening or penetration is needed when a reference MEP element fully intersects a host element; it includes wall penetrations, ceiling penetrations, floor penetrations, core drilling etc. On the other hand, a slot or recess is needed when it does not fully intersect with a host element; it includes the wall recesses, ceiling recesses etc. Even though these elements (cut-outs) are not physical components in a building, planning their location and size depends on many factors, such as structural requirements of the host elements, thermal, acoustic, fire protection, and constructability requirements. Therefore, it demands a comprehensive decision-making process before the execution of cut-outs in building components; in other words, it needs an effective coordination process.

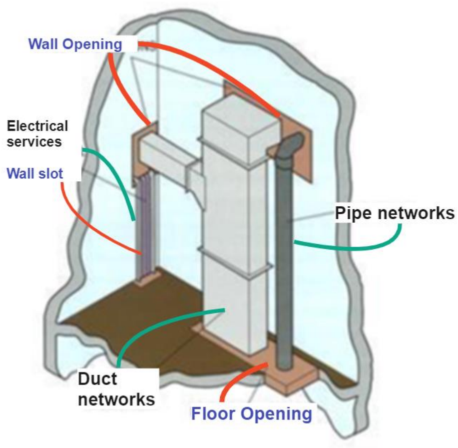


Figure 1.2: A sketch Slot/ openings for MEP services (Flamebar GFS 1000, 2021)

If the slots and openings for hosting MEP elements in building components are not pre-planned or coordinated during design phases, they must be coordinated during the installation process. This coordination process will be time-consuming, laborious, and expensive. Furthermore, it requires the removal of host element materials (typically through core drillings) that were not designed with cut-outs in consideration, resulting in additional issues. For example, core drilling in floor slabs with dense reinforcements may mistakenly cut the designed reinforcements; so, undertaking cut-outs in slabs requires coordination and approval from the responsible structural engineer (Zamani, 2020).

An unplanned opening can easily take up a cubic meter of concrete. Also, if openings are not specified correctly in the case of planned core drillings, it would be challenging to take on-site measurements due to space restrictions; this can cause errors. Therefore, all pre-planned slots and openings must be specified correctly at the right location beforehand; installation must be modelled at a high level early on in the construction project (Trimble, 2017). Furthermore, pre-planning of slots and openings could save material, especially in the case of larger openings.

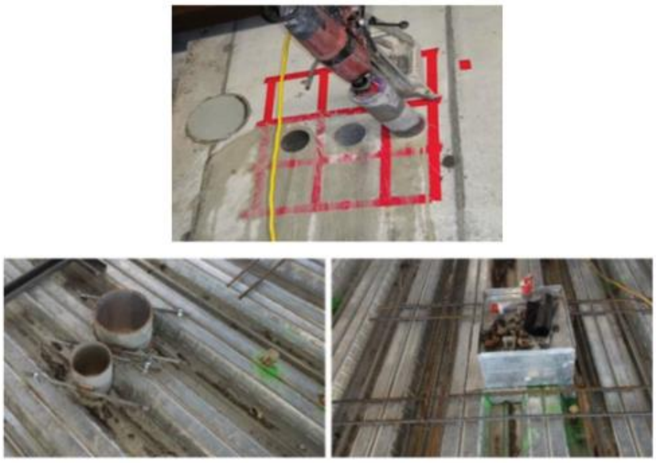


Figure 1.3: Example of execution of core drillings to make openings in concrete floor (top) and execution of pre-planned openings in concrete slabs avoid cutting (bottom) (Zamani, 2020)

In the case of service elements running through steel elements, it is necessary to model and coordinate the openings during the design phases. If they were modelled precisely in 3D models, then the steel fabrication team could quickly take over the openings from the BIM models for the steel construction (Trimble, 2017).

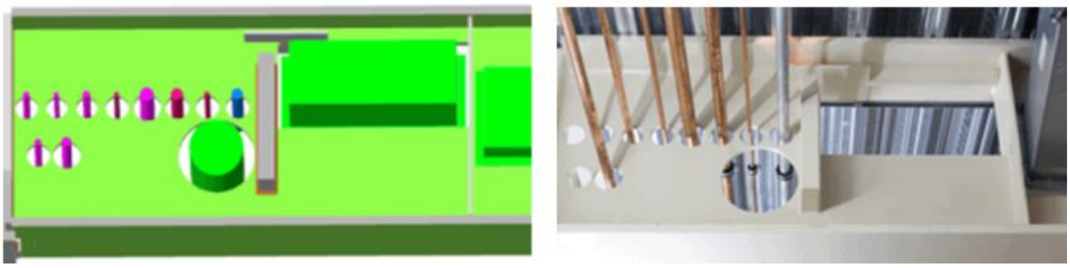


Figure 1.4: Pre-planned openings in BIM model (left) and corresponding openings in steel beams (right) (adapted from (Trimble, 2017)

Eastman (2011) mentioned that BIM methods provide various technical benefits to improve design coordination at different stages of construction projects. Since the IFC 2x3 version, there is a proxy object for ‘void proposal’ with the predefined type ‘ProvisionForVoid[1]’, termed as provision for void object or PfV object. These objects can be used to request the amount of volume of space required for the installation of building service elements as an opening or a slot. That is, any wall or slab that would be cut to pass through or accommodate MEP elements gets a provision for void object

[1]https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD1/HTML/schema/ifcsharedbldgelements/lexical/ifcbuildingelementproxy.htm

or PfV object. The request (as PfV objects) will be checked and approved by other involved decision-making parties or specialist planners, usually through a coordination process. Typically, coordination takes place between MEP, architectural, structural disciplines, and some cases with fabrication/builders as well, with each discipline marking their approval decision as illustrated in Figure 1.5. After specialist planners approve the request through coordination, these will be converted to actual cut-outs for slots or openings (see illustration in Figure 1.6).

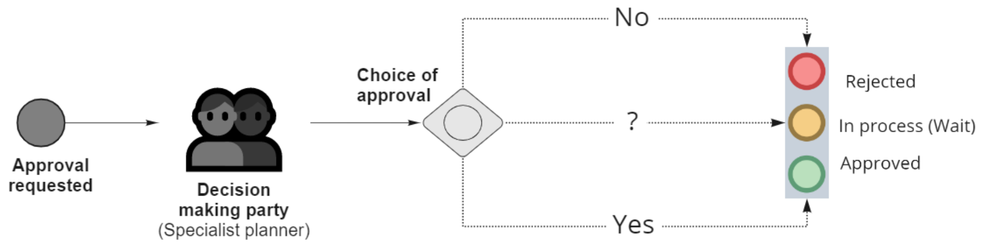


Figure 1.5: The basic concept of the decision-making in slot and opening planning coordination process

Figure 1.6: MEP Provision for void (PfV) objects (left) and cut-out as opening in the wall (right) (adapted based on the 'Guide to BIM Modeling in Revit (2020)'[2]

1.1 Motivation

In order to conduct efficient coordination for slots and openings in construction projects, it demands effective and transparent data exchange (slots and opening approval data) methods among the involved disciplines. Although there are many different methods currently in practice, the coordination of slots and openings is still an unresolved workflow; it needs to be reinvented in any new project. Additionally, timely coordination, model workability, and consistency between discipline models are crucial for bringing efficiency to this process and preventing costly errors. The coordination usually happens either document-based (PDF or 2D file) or model-based (3D BIM model). Coordination via PDF files is, by its nature is, a manual undertaking, which needs

[2] https://blogs.autodesk.com/bimblog/leitfaden-fur-die-modellierung-in-revit/

human input, interpretation, and transfer. The chance of misinterpretation of information from a 2D data representation is high. In addition, as these approval files are disconnected from the CAD file, the approval information usually flows only in one direction. Therefore, it needs human interaction to get back into the design environment.

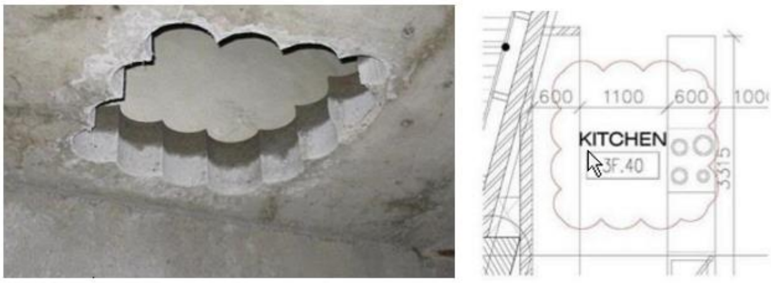


Figure 1.7: Example of a popular[3] misinterpreted core drilling based on a revision cloud

In model-based coordination of slot and opening planning, the most used method of communication is undertaken via BIM Collaboration Format (BCF). However, in design reality with BCF file-based communication, the approval issues can be created multiple times 'round-tripped' for each object under discussion (buildingSMART International, 2021); hence a loss of overview is inherent. Other 'file-based (native or IFC) coordination' approaches also require re-sending and comparing the complete void instance model to communicate and track a single void object. This leads to multiple sources of information and causes inconsistency in design and confusion among the involved planners. This leads to loss of control over the approved or rejected voids requests. Moreover, all other existing methods such as excel-based, native or IFC file-based approvals also need to transfer entire files back and forth to communicate about a single void object.

Additionally, during design development, the change in MEP elements (e.g., change in size/shape/location of pipes) must be updated with corresponding 'provision for void' objects in all related discipline-specific models to maintain consistent design. In such cases, change management, change tracking, and maintaining consistency in the approval of thousands of voids in large projects is challenging, time-consuming, and prone to errors. Moreover, keeping an approval history of who agreed or rejected, when and with what remarks is always a challenge.

[3] https://www.constructionknowledge.net/blog/?p=3742

Keeping a history of changes is necessary so that decisions are trackable and understandable to involved planners. It is essential for the following cases:

- Large and complex projects with thousands of provisions for voids
- Projects with longer design period; change of designers during project phases

An inconsistency in voids and cut-outs across discipline-specific models could lead to costly errors in the project execution phase. An example of such a situation is provided in Figure 1.8.

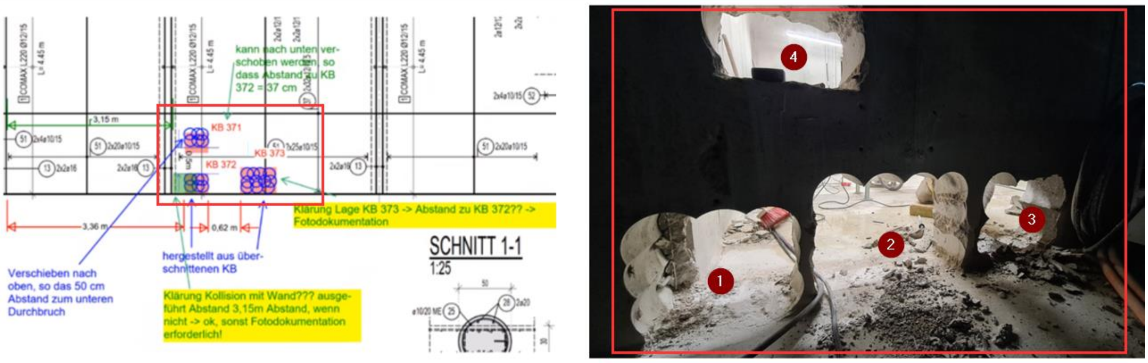


Figure 1.8: Error/inconsistency in the execution design; it was intended to have three openings (left) and drilled four incorrect openings (right) (Bollinger und Grohmann GmbH, 2022)

Recently, the BuildingSMART German chapter[4] provided a guideline for conducting IFC-based slots and opening planning in construction projects. However, the approval information exchange is still open, which is the core part of the coordination. Generally, it can happen in any mode of communication, including uncountable e-mails and telephone calls, BCF issues. So, the slot and opening coordination in the construction projects require a simple and pragmatic BIM-based workflow regardless of the level of experience of the end-user.

1.2 Approach

The thesis investigates a concept for improving the existing slot and opening coordination method in BIM-based construction projects in a practice-oriented approach. The thesis' approach was inspired by the Design Science Research (DSR) methodology described in Peffers et al. (2007). The DSR methodology, in principle, seeks out user insights first, then develops a solution based on the findings, rather than the other way around. Furthermore, throughout the thesis, objectives and

[4] https://ucm.buildingsmart.org/use-case-details/2376/de

proposed solutions were re-evaluated based on the insights from industry professionals to make the solution more practical.

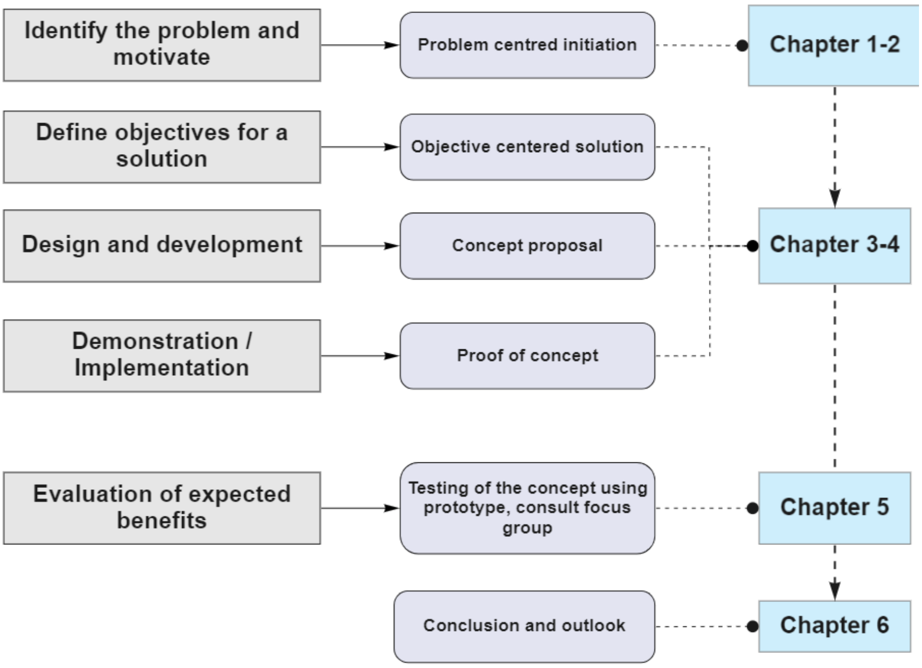


Figure 1.9: Thesis approach; inspired based on Design Science Research (DSR) methodology

Objective-oriented tasks

Based on the thesis approach, objective-oriented tasks were identified and are as follows:

- Identify the existing challenges and limitations in the slot and opening coordination in BIM-based construction projects by reviewing the state-of-the-art methods and processes.
- Develop a simple and pragmatic concept for effectively conducting slot and opening coordination by analyzing the necessary functionality requirements to overcome existing challenges identified.
- Implement a prototype to validate the concept; also, gather opinions (in the given period of time) about the proposed concept from industry professionals to check adaptability in real projects.
- Propose future directions from the evaluation results of the concept for further improvement related to the slot and opening planning process in BIM projects.

2 Theoretical background

Building design can be considered a holistic solution from a collaborated and coordinated multi-disciplinary process involved by designers, engineers, builders, etc. One of the key problems in conventional 2D CAD drawing based construction design is to keep consistency between diverse technical drawings created by experts from different design disciplines and across multiple companies. It requires intensive manual checking. Furthermore, if design changes are indicated with revision clouds, the adaption of changes to keep consistencies needs continuous tracking in all related drawings. If inconsistencies remain undiscovered, it will lead to a high extra cost to solve the issue on the construction site. Also, the limited information depth of technical 2D drawings causes unnecessary additional works. On the other hand, the most obvious feature of a Building Information Model (BIM) is the 3D geometry. In which the objects combine a parametrized 3D geometry representation with additional descriptive properties and their relationships to other elements in the model. The concrete information content of BIM models depend heavily on the use cases it is being used. The most widespread BIM use cases are listed in Figure 2.1 (Borrmann et al., 2018).

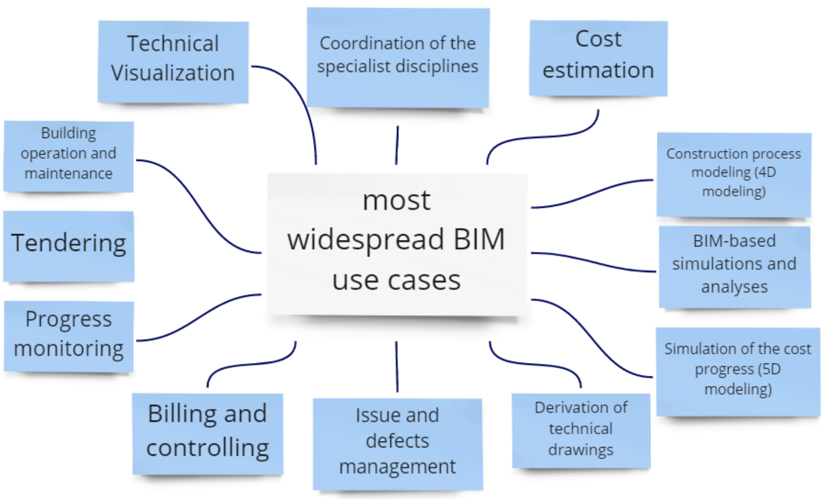


Figure 2.1: Most widespread BIM use cases (adapted from Borrmann et al., (2018))

In BIM-based collaboration projects, Employer Information Requirements (EIR) is the contractual document. Additionally, there is generally a predefined guideline called the BIM execution plan (BEP), which can be considered as the answer to the EIR. It defines the organizational structures and responsibilities, sets out the framework for the BIM services and defines the processes and the requirements for collaboration between the individual participants. The domain-specific planning participants are required to submit their digital models at predefined, periodic times in IFC file format. The required level of information in the specialized models corresponds to the phase-specific services and goals defined in the BEP. Then, the discipline specialist models are compared with each other to check for their data consistency and conflicts/clashes. If BEP is also part of the contract, the client has control over the BIM execution methods. That means, if project participants did not follow the method, it would be considered as the contract won't be fulfilled; in the worst case, project participants can't get paid for the work (Ikerd, 2020).

2.1 BIM-based design coordination

Trade and system coordination is a critical work process for every contractor. Traditionally the coordination of construction design was conducted using 2D drawings through a "sequential comparison overlay process". In this procedure, the potential design conflict and constructability were reviewed sequentially by comparing and overlapping drawings (or layers) of the same scale using a light table physical overlapping of drawings (Korman et al., 2003). This manual process is expensive, time-consuming, and inefficient. When computer-aided design (CAD) was first developed, designs were created as 2D drawings. The CAD drawings-based coordination can be treated as a direct replacement for paper drawings and the light table as it does not significantly change the process. Furthermore, the 2D drawings can only convey limited information, and it is hard to detect conflicts and clashes. It demands a significant reliance on intuition, imagination, technical expertise, and human judgment (Victor, 2019).

The shortcomings of 2D-based coordination and the necessity to improve coordination efficiency have resulted in utilizing the capabilities of BIM technology for design coordination. The BIM-based coordination enables project participants to integrate 3D models from different design disciplines into a single integrated or coordination model. Moreover, this coordination model can be used for the automatic detection of system interferences (Yarmohammadi & Ashuri, 2015).

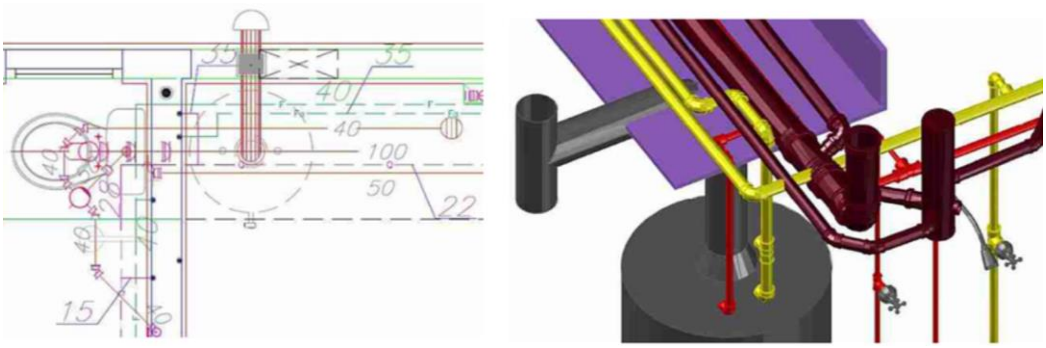


Figure 2.2: Comparison of 2D-CAD model and BIM model; 2D is not enough to identify the interferences between plumbing and ceiling (purple) or between sewage (brown) and cold water (yellow) pipes (left); and they are relatively evident in the BIM model (right) (E. Santos & R. C. Ferreira, 2008)

BIM offers various technical benefits to improve design coordination at different stages of construction projects; mainly, it assists in creating high-quality integrated models at construction-detail levels. Furthermore, if BIM tools that facilitate coordination are applied appropriately, it can improve communication and reduce the cost and time of construction (Eastman, 2011).

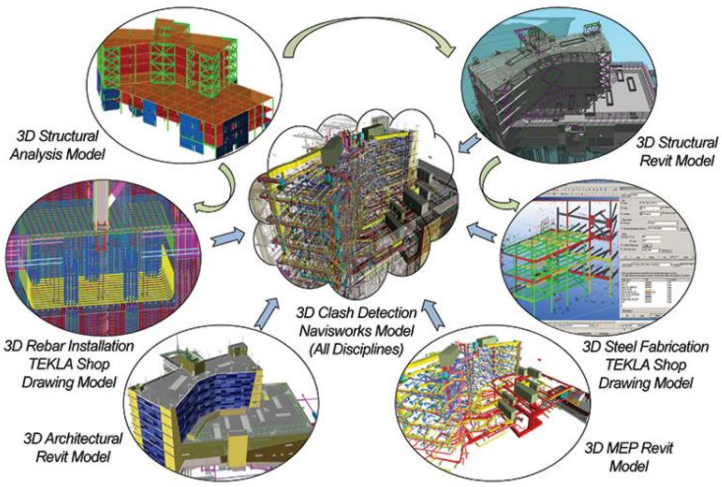


Figure 2.3: Example of an integrated BIM coordination model (TMAD Taylor & Gaines)

A typical BIM coordination can be explained in three critical steps: issue identification, issue resolution, and issue documentation, as shown in Figure 2.4. In the issue identification step, the responsible coordination person (usually BIM coordinator) integrates the federated BIM models from relevant design disciplines using BIM coordination tools (e.g. Navisworks, Solibri Model checker) and performs automatic and manual clash detection. The automatically identified clashes will be analyzed based on coordination relevant knowledge. In issue resolution steps, responsible stakeholders review, discuss and develop solutions to resolve the identified clashes. Finally, the BIM coordinator will document the issues once the relevant clashes are solved. Also, documented issues could be tracked separately for the subsequent issue identification stages (Mehrbod et al., 2019).

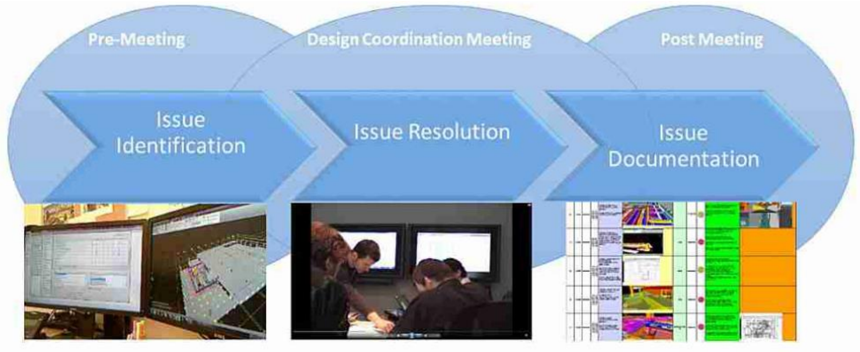


Figure 2.4: BIM-based design coordination process critical steps: issue identification, issue resolution, and issue documentation (Mehrbod et al., 2019)

Issue identification - clash detection and clash avoidance

Clash detection is not a new concept; only the clashes' detection approach has changed from construction sites to digital BIM models during design phases. BIM-based tools detect geometry-based clashes combined with semantic and rule-based clash analysis. Also, to detect the clashes accurately, the BIM models should have an appropriate level of detail. Therefore, the subcontractors or other project team members responsible for designing the systems must participate in the model development process as early as possible. Before the next clash detection phase, agreed-upon changes can be entered into the relevant design model. Space clashes are a significant source of building site problems that can be largely avoided by using an accurate and detailed model for clash detection (Eastman, 2011).

There are three types of clashes in general:

- Hard or physical clash
- Soft clash
- Logical clash

A hard clash occurs when two elements occupy the same physical space. Soft clashes occur when elements are not in physical contact but lack adequate clearance requirements. Logical clashes include issues in constructability and operational aspects. Due to current technical limitations, the corrections to clashes cannot be made in the clash detection environment. Each trade needs to make the necessary changes inside their BIM authoring tools once the team has determined a solution (Eastman, 2011).

Clash avoidance is a proactive effort to prevent spatial and semantic conflicts across BIM Models produced by various design disciplines. Clash avoidance is a type of predictive spatial analysis that aims to identify and address possible problems before they occur. To achieve clash avoidance in construction design projects, planners use relevant interdisciplinary knowledge while designing and coordinating (Eastman, 2011).

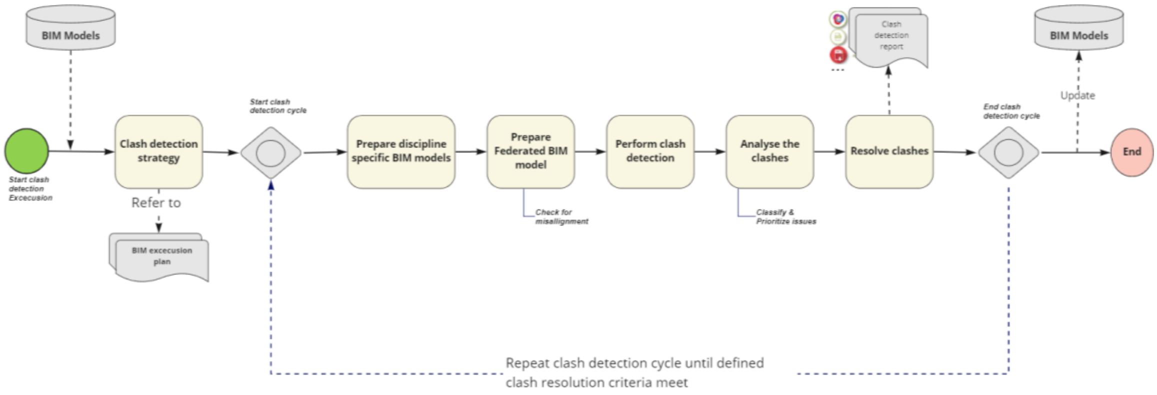


Figure 2.5: The iterative clash detection workflow in the BIM projects (adapted from Machado & Mota, (2020))

Coordination hierarchy

A coordination hierarchy is usually established by prioritizing the objects that are the most difficult or expensive to move. It includes information about priorities and check dates. This enables the team to quickly determine which components in a given discipline may need to change in a clash scenario, depending on the other trade they are clashing. Coordination hierarchy can be established through the clash matrix, a table showing the checks to be carried out in the different coordination phases. It should indicate what needs to be analyzed and what will not be analyzed. Based on which disciplines are furthest to the top and left of the matrix, a clash matrix illustrates which disciplines take precedence. However, an object with a higher hierarchy may also be moved or otherwise updated in certain limitations. For example, consider a situation where the design team determines that a ceiling height cannot be changed, and the plumbing roof drain pipe collides with a structural beam. If the ceiling cannot be lowered, the structural engineer can design a penetration around the mid-span of the beam for the pipe to pass through. Moreover, integrating the clash matrix in the BIM execution plan is better for successfully practising the BIM coordination tasks (Ikerd, 2020).

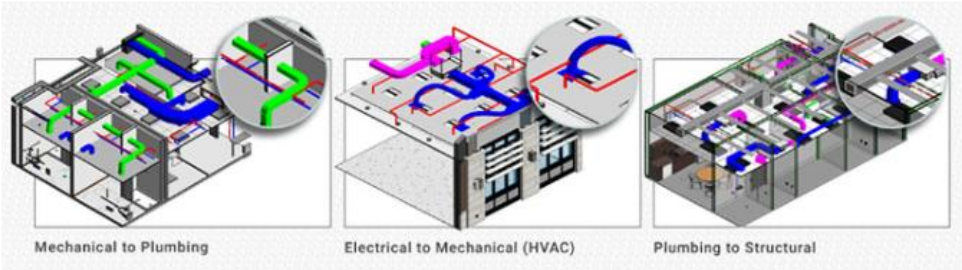


Figure 2.6: Example of clashes between different disciplines (Victor, 2019b)

ALL ZONES	Structure	Ductwork	Plumbing	Electrical	Fire Protection	Tech	Food Service	
Structure (Highest priority)	X							
Ductwork	0	X						
Plumbing	0	0	X					
Electrical	0	0	0	X				
Fire Protection	0	0	0	0	X			
Tech	0	0	0	0	0	X		
Food Service	0	0	0	0	0	0	X (Lowest priority)	
BUILDING TOTAL:	0							

Figure 2.7: Sample layout of a clash matrix. Here structure would take precedence over ductwork, and ductwork over plumbing etc. (Ikerd, 2020)

Coordination-relevant knowledge

BIM coordinators and specialist planners use discipline-relevant and interdisciplinary knowledge to achieve the goals and objectives of BIM coordination tasks. This is crucial to solving the clashes and clash avoidance. In an early attempt, the study conducted by Korman et al., (2003) concluded that the three broad knowledge domains such as design criteria and intents, construction issues, and operation and maintenance significantly influence the outcome and activity of the MEP coordination process. There, the design engineers and modellers bring design knowledge to ensure the performance requirements for the project according to the codes and standards. In addition, superintendents, supervisors, and engineers familiar with field operations contribute construction knowledge, including fabrication, installation, sequencing considerations, and requirements for start-up, testing, and safety aspects for efficient field operations. Also, to minimize the cost of operation and maintenance, coordination must consider the knowledge of facility life cycle aspects, which comes from facility managers, building engineers, and the maintenance staff. Wang and Leite (2015) argue that a significant portion of design and construction knowledge is generated and used in the coordination process, usually lost afterwards, but it can be utilized if systematically documented.

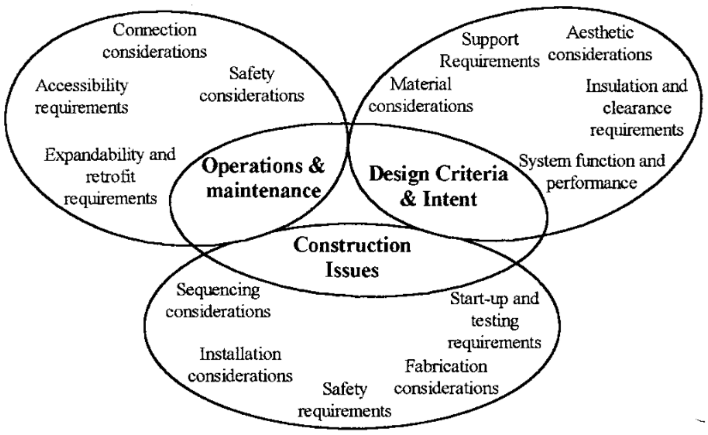


Figure 2.8: Knowledge bases required for MEP coordination (Korman et al., 2003)

The BIM tools that perform automatic clash detections of BIM models using the rule-based reasoning approach consider some of these aspects from well-defined rule sets such as clearance, interference, support requirements, etc. However, the necessity of knowledge acquired by people cannot be replaced with automation. Therefore, a tool-assisted approach would be ideal for achieving successful coordination tasks.

Communication of issues

After identifying the clashes through clash detection, the project partners solve the clashes through coordination meetings via proper communication and data exchange methods. This includes emails, phone calls and more. One of the most common methods to analyze and document the clashes is the BIM collaboration format or BCF (Borrmann et al., 2018). BCF is a vendor-neutral openBIM data standard for comments, inquiries, collision reports or general information on BIM models. BCF works by transferring XML formatted data from one application to another, which is contextualized information about an issue or problem directly referencing a view, captured by PNG and IFC coordinates, and elements of a BIM, as referenced by their IfcGUID (buildingSMART International, 2021). This can be imported into BIM authoring software and will help the partners to identify and solve the clashes. An example of BCF-based issue management is given in Figure 2.9.

BCF can be used in two different ways: as a file-based exchange or as a web service. The file-based exchange workflow is simple and straightforward. A BCF file (.bcfzip) is passed between users, edited, and then returned. Unlike IFC file workflows, BCF files can be ‘round-tripped’ as long as everyone maintains the shared BCF file's integrity and no duplicate copies are circulated. The web service-based (RESTful) API option for BCF is an alternate to the file-based workflow. This involves the implementation of a BCF server, which can also act as a BIM server that store all BCF data and allows project participants to collaborate on the creation, editing, and management of BCF topics from a centralized location (buildingSMART International, 2021).

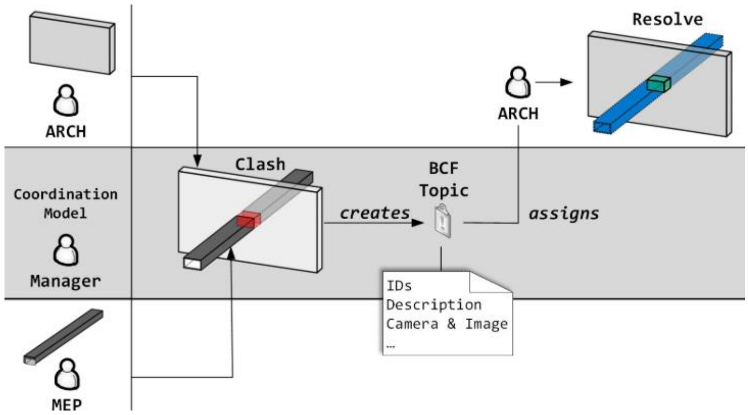
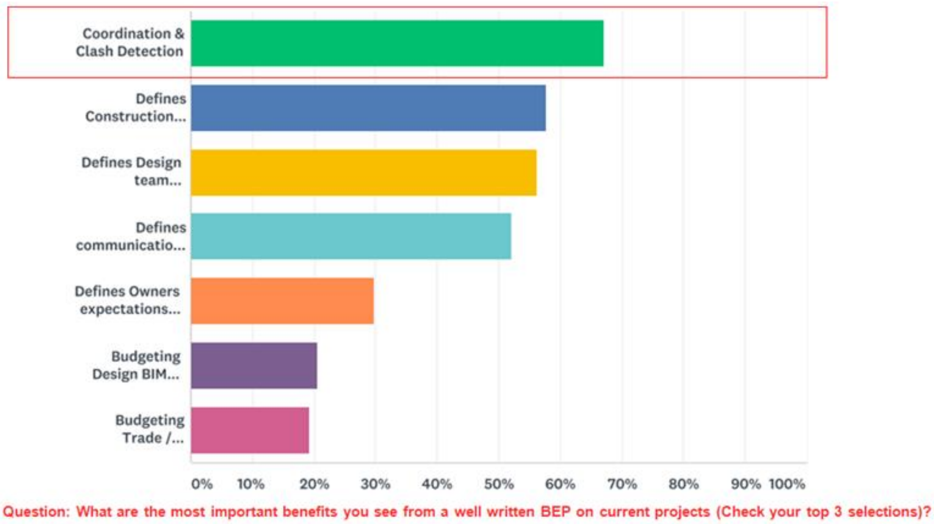


Figure 2.9: Exemplary BCF-based communication process (Borrmann et al., 2018)

Significance of clash detection

A survey conducted by Ikerd (2020) showed that the respondents (professionals from various construction disciplines) perceive coordination and clash detection as the most benefit in utilizing BIM in their construction projects; the results are shown below:



ANSWER CHOICES	RESPONSES	
Coordination & Clash Detection	67.15%	327
Defines Construction team expectations, scope & process with BIM	57.70%	281
Defines Design team expectations, scope & process with BIM	56.26%	274
Defines communication plan for the BIM team.	52.16%	254
Defines Owners expectations with BIM for FM	29.77%	145
Budgeting Design BIM effort	20.53%	100
Budgeting Trade / Fabrication BIM effort	19.30%	94
Total Respondents: 487		

Figure 2.10: Result from the questionnaire survey for the question ‘what is the most important benefit you see from a well written BEP on current projects (check your top 3 selections)’ (Ikerd, 2020)

Association of Construction and Development[5] estimated that, on average, construction industry-wide, in a project, every identified clash saves around $17,000. In the case of large projects, 2000-3000 clashes are not unusual; it could save $34,000,000. In addition, many studies, such as Leite et al. (2010), Mcgraw-Hill (2012), Bockstael & Issa, Cao et al.(2015), considered clash detection and clash avoidance through BIM coordination as critical areas that influence cost and time reduction in construction projects.

2.2 Design changes and change tracking in BIM Projects

Due to the iterative and exploratory nature of design itself, changes are inevitable in construction design. Even after construction has started, the content and structure of the design are susceptible to frequent changes or revisions. However, the real impacts of changes are often overlooked and may only be revealed while making adjustments or identifying alternative solutions. Therefore, timely identification and analysis of the consequences of design changes are essential for the successful delivery of construction projects, mainly when projects are executed using fast-track approaches. Moreover, several previous researchers assume that having a change history would benefit in analyzing the impact of changes (Isaac & Navon, 2013).

Any alteration of current design data that affects the project team's earlier decisions is considered a design change. (Isaac & Navon, 2013) summarize the leading causes of design changes in construction projects as:

- Change orders: Changes initiated by the client due to omissions in the requirements, changes in the clients' activity, or a better understanding of the client's needs.
- Rework: Changes made by the project team as a result of issues/errors or solutions discovered during the development of design and planning stages, as well as during construction.
- Changes brought on by suppliers due to technological changes or the inability to meet the original planning targets.
- External causes from new regulations or standards.

[5] https://www.associationofconstructionanddevelopment.org/

BIM-based design change management

Even though BIM implementation in construction projects claims to reduce design errors leading to reduction of the design changes, design revisions inevitably occur in construction projects. The BIM-based design change management can be treated as a dynamic process. That includes identifying needs and reasons for changes, implementing changes, information flow about changes, analyzing and assessing the consequences of changes, and minimizing adverse effects of changes while maintaining interoperability. The development of BIM tools that allow the efficient implementation of BIM-based design change management appears to be an undeniable and challenging task. Quantifying, visualizing, and analyzing changes made to BIM models would benefit all parties involved in the construction project (Juszczyk et al., 2016).

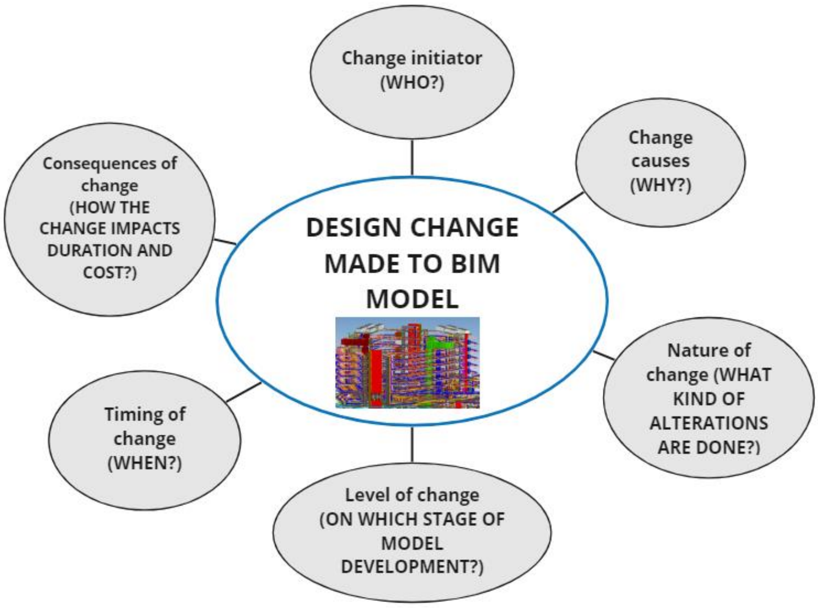


Figure 2.11: Idea of classification of information related to design change made to BIM model (Juszczyk et al., 2016)

A study conducted by Pilehchian et al., (2015) examined change management by tracking design changes in multi-disciplinary construction projects using Autodesk Revit, Navisworks, and Solibri Model Checker. One of the tests was conducted to identify changes in the position of openings, as shown in Figure 2.12 from addition, deletion, and modifications of attributes. However, many of the detected changes were incorrect and unrealistic to the intended targets. Also, it was found that the Solibri Model checker was unable to link the characteristics of recreated, merged, or split components to the characteristics of the old openings, limiting change tracking between models.

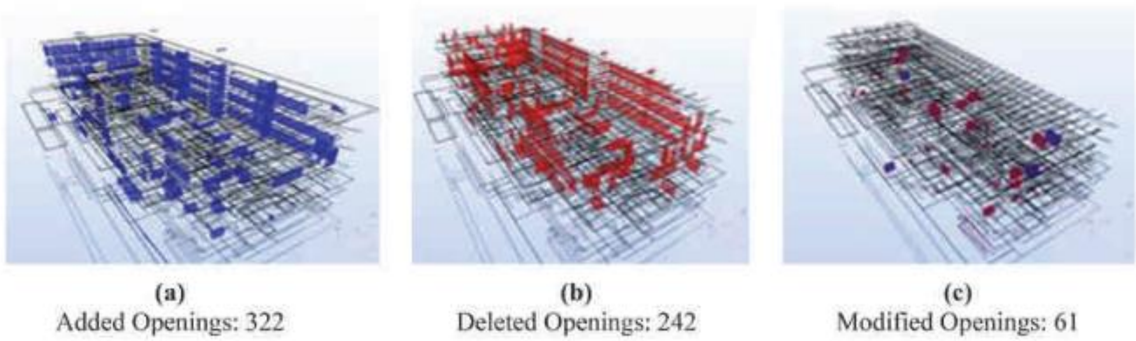


Figure 2.12: Tracking of changes in openings using Solibri Model Checker (a. new openings added, deleted openings, c. modified openings' attribute), (Pilehchian et al., 2015)

In addition, the study found that state-of-the-art BIM tools have limitations in recognizing a wide range of logical dependencies of design changes. The test summarizes three requirements from BIM tools to track changes between different revisions of BIM models, as given below.

- **Recreated** components should inherit the characteristics of the deleted component.
- **Merged** components should inherit the characteristics of their parent components.
- **Split** components should inherit characteristics of the original component.

The BIM data exchange using the open BIM file exchange format IFC causes information redundancy, which is mainly associated with two aspects: sequential identifiers of STEP format (STEP #-IDs) in each line and cross-referencing of IFC objects' GUIDs. Occasionally, the STEP #-IDs for IFC objects are randomly generated, resulting in significant byte-level inconsistencies in the .ifc files. Modern data structures, such as tree-like JavaScript Object Notation (JSON) and eXtensible Markup Language (XML), provide better computational efficiency and explainability than the line-by-line STEP structure. This paved the way for the introduction of new IFC formats by buildingSMART (2022)[6] , such as IFC-XML, which is based on the STEP-XML standard, IFC-ZIP, IFC-OWL, IFC-JSON, and IFC SQLite. STEP #-Id discrepancy has been removed to an extent in several modern IFC formats, such as IFC-XML. The GUID of an IFC object should be unique and constant throughout the BIM model data exchange. But on popular BIM authoring tools, many such GUIDs are

[6] https://technical.buildingsmart.org/standards/ifc/ifc-formats/

randomized. For instance, Autodesk Revit keeps the GUIDs of IFC objects that have a unique' ElementID' such as IfcOpeningElement, but randomizes GUIDs of objects such as object's properties (IfcPropertySet). This causes the creation of a different IFC file with new GUIDs while slightly changing or updating the BIM model. Due to this issue of randomization of some GUIDs, it becomes difficult to trace and compare BIM objects using IFC file exchange (Xue & Lu, 2020).

3 Slots and openings coordination: State of the art review

This chapter aims to formulate the research problem by reviewing the existing methods of 'slot and opening coordination' in construction projects. The chapter is divided into three sections. First, section 3.1 provide a general overview of slot and opening planning. Then section 3.2 outlines the conventional coordination method without the technical aspects. After that, in section 3.3, the model-based coordination methods are described in detail. Finally, the existing challenges and limitations are listed in section 3.4.

3.1 Overview

As mentioned in chapter 1, the planning or coordination of the slots and openings is one of the most difficult and widely discussed topics[7] in construction projects. This is because several specialized planning parties are always involved and must be coordinated with each other. Generally, the coordination is conducted between architectural, structural and MEP planners; but in some cases, contractors and fabricators are also involved. The Architect leads the coordination process in most of the cases.

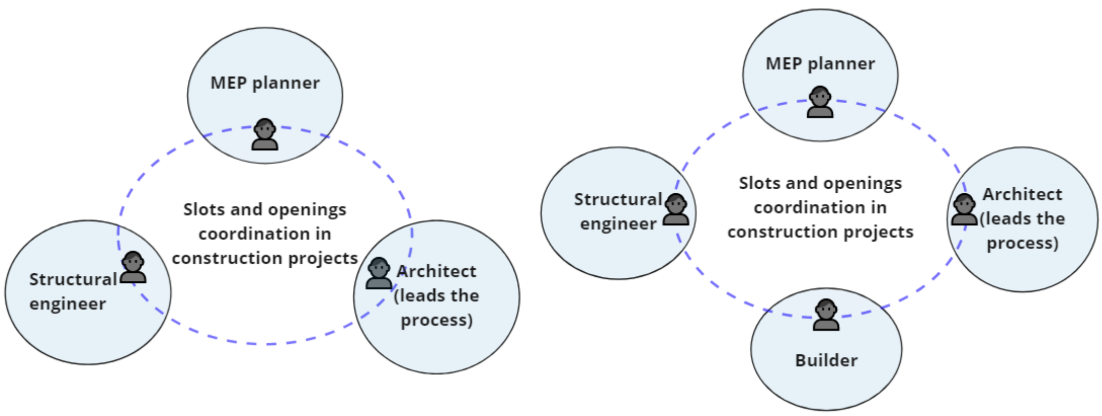


Figure 3.1: The frequently involved parties in the slots and openings coordination task

Also, it should be noted that the planning of large openings such as staircases, elevator shafts, and installation shafts are often agreed upon in the preliminary design phases of construction projects, independent of this slot and openings process.

[7] Some relevant discussions from forums are given in Annex A for further insights.

Design phase and Execution phase coordination

The planning of slots and openings in construction projects usually starts in the design phase. Depending on the contract (a full contract over all phases or limited contract until the design phase), there are two types of slot and opening coordination rounds: design phase coordination and execution phase coordination. In early design phases[8], the MEP planner usually provides openings' sketches based on the architectural drawings. Then, the Architects integrate these sketched openings manually into the architectural design phase drawings; generally, and with the degree of BIM implementation in Germany, these are not done in 3D models. Usually, the maturity of the MEP design will be in an early stage of the project, and only the main routes are designed/sketched; hence not all openings are usually provided in the design phase. These openings will be taken into account as structural relevant openings in the structural design for service phase 4 (LP4: Genehmigungsphase acc. to HOAI[9]) and are also part of the position drawings. Meanwhile, the MEP and architectural discipline start modifying their design. A change in architectural geometry will lead to a change in MEP routings and eventually the structural design. Usually, detailed and comprehensive MEP planning (route planning) occurs in the execution phase (LP5 acc. to HOAI). In addition, a definition of structural relevant openings can vary from project to project. The openings in all load-bearing components can but not have to be relevant for structural design (Hofbeck et al., 2021). A proper definition for the project needs to be communicated from the beginning.

As per HOAI regulations, the MEP discipline does not need to provide opening drawings until service phase 5 (LP5), the execution planning phase. The whole procedure opposes the vision of the BIM methodology in projects where the project should achieve already an early maturity and stability, hence preventing cost-intensive coordination issues in the late stages of the design. In phase 5, the MEP discipline starts creating separate slots and openings drawings for the coordination process. The fully integrated decision-making process as the approval process for slots and openings starts only after phase 4; that is the execution phase coordination. In order

---

[8] https://www.hoai.de/hoai/leistungsphasen/
https://kinoplanung.de/en/how-we-work/8-service-phases/

[9] HOAI: Honorarordnung für Architekten und Ingenieure; The German Official Scale of Fees for Services by Architects and Engineers (translated by the Author)

to speed up the slot and opening coordination, planners define discipline-specific rules or rule-of-thumb from an early phase of coordination.

- **Restrictive (No-Go) zones and minimum clearance:** The restrictive zones in slots and opening planning indicate the regions in a building geometry where slots or openings cannot be made/ or need coordination with structural engineers. This sets boundary/exclusion zones in the geometry (drawings/model) for MEP planners preventing the creation of slots or openings. Therefore, it simplifies the future coordination effort and increases efficiency in the slot and opening planning process, which will speed up the future checking process. No-go zones should be included as early as possible in the models or drawings to obtain a preventive character; before the MEP discipline starts planning the routes and slots/openings. Additionally, defining the minimum clearance between openings ensures the feasibility of the execution of openings in the construction site. In design reality, various rules will be defined for the quick review of openings according to various design criteria. An example of general rules defined for a quick review of structural relevant and construction relevant openings during the slot and opening coordination are provided in Annex A for further insights.

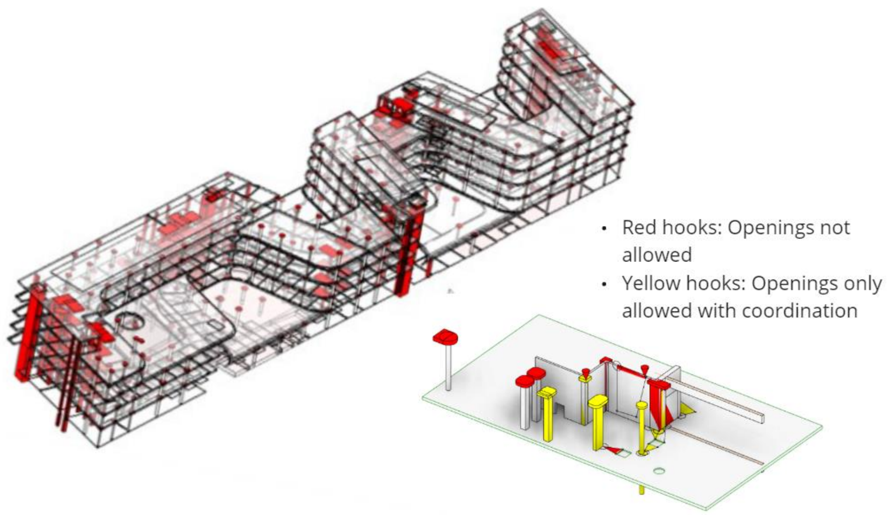


Figure 3.2: No-go zones defined in BIM model during the design phase coordination for slot and opening planning (adapted from Hofbeck et al., (2021))

Core-drilling zones: The core-drilling zones are the region in the architectural drawings marked by the MEP discipline as an area where openings as core drills can occur post-construction. This will be considered as a provision for the probable openings and will be decided later during the detailed route planning. This information will be taken into account for the structural design of that specific region by the structural engineer. For example, reinforcement design can be done with the assumption that if an opening is drilled in future in this core drilling zone, the structure can withstand it with enough load-carrying capacity.

Coordination methods

The general concept of slot and opening coordination includes three subprocesses, as illustrated in Figure 3.3. First, the coordination usually starts with the MEP planner creating void proposals for slots and openings. Then it needs to be approved by involved specialist planners. Finally, after getting approval from all involved specialist planners via coordination, specialist planners will integrate the voids into individual discipline-specific models as slots or openings. The method of coordination depends mainly on how data is exchanged between the concerned parties in the process. This can be done either as conventional 2D drawings or as 3D models. The model-based coordination makes sense only if all slots and openings are modelled in 3D; if any of the involved parties use 2D files for exchanging approval information, then the conventional method of PDF- or DWG-based approach must be used[10].

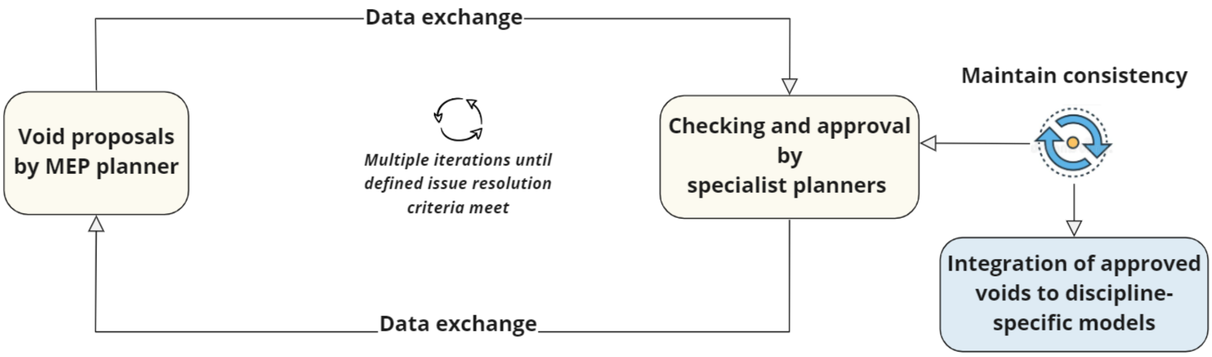


Figure 3.3: Simplified concept of slots and openings coordination with the main subprocess

3.2 2D Drawings-based method

The conventional way of planning and approving slots and openings in building elements is based on 2D drawings with standard representations and annotations. Slots and openings are simple symbols consisting of lines, hatching, label, and dimensions in conventional 2D CAD plans without any reference to the associated component. The symbols must be manually adjusted if the position or dimension changes (Deutsches Architektenblatt, 2021). This approach imitates the centuries-old

[10] https://blogs.autodesk.com/bimblog/leitfaden-fur-die-modellierung-in-revit/

way of working using transparent overlays on a drawing board described by Korman et al. (2003). In 2D-based planning of voids, planners use standard representations for model objects so that they can be understood by specialist planners involved from other disciplines. For example, ÖNORM A 6240-2[11] or DIN 1356-1[12] specify uniform specifications for slot and opening representation in CAD drawings. Also, in practice, the openings will have standard designation with a unique number/text as codes representing the related parameters of void instances. These codes vary from project to project and across companies.

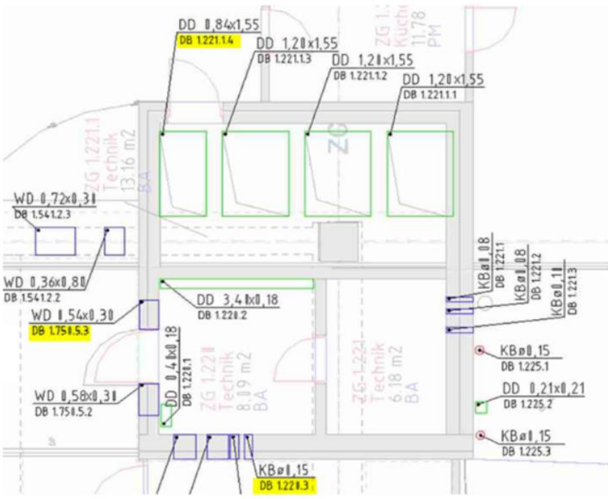


Figure 3.4: Example of encoding in CAD layer (Flughafen Düsseldorf GmbH, 2020)

A 2D-based approach requires several exchanges of paper 2D-plans, PDF or DWG files, with coded slots and openings, sometimes colour-coded, as shown in Figure 3.4. The architect and structural engineer use these predefined codes to verify the slots or openings, accept/reject them with a comment, and return the drawings to the building services engineer. Those involved in the planning send plans or files back and forth until everyone agrees on the void schedule. Phone or email will be used to respond to questions. The drawing-based approval process necessitates a strong spatial imagination and a detailed verbal description of the subject at hand; moreover, in complicated situations, it demands schematic hand-sketches. Additionally, in project reality, the architects are generally not obliged to create additional elevations and sections at all slots and opening positions. Openings are usually only entered in the 2D floor plan; without geometrically checking the third dimension. Therefore, even

[11] https://www.bauberufe.eu/images/doks/Plandarstellung_Oenorm_A_6240.pdf
[12] https://www.beuth.de/de/norm/din-1356-1/2463916

after defining uniform specifications for representing openings in a project, errors, and misunderstandings regarding the type, position, and dimensions are inevitable.

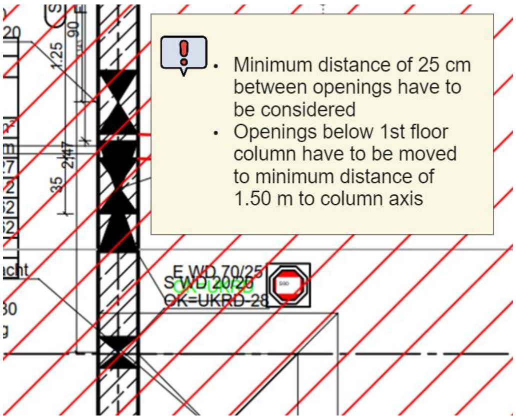


Figure 3.5: Example of communication of slots and openings with 2D drawings (Bollinger und Grohmann GmbH, 2022)

Usually, there can have multiple drawings from the MEP discipline with multiple design firms being involved in the design. In such cases, there will be multiple drawing sets that need to be coordinated by the architect as the lead in the process. If the MEP disciplines don't coordinate prior to this, there will be clashes between 2D routes and 2D openings. Generally, architects do this coordination. In this approach, a change in opening position or geometry comes out of any late design changes are marked with revision clouds in the drawings, which can be hard to detect and ambiguous; due to the unavailability of the third dimension (section at every point). Therefore, this approach can be treated as a ‘disconnected and interpreted’ method. Slot and opening planning in the conventional planning process without the use of BIM methods or IFC has weaknesses in many areas that repeatedly lead to errors, problems or disputes between the parties involved. The most well-known difficulties are (BuildingSMART-Regionalgruppe Mitteldeutschland, 2020) :

- Diverse means of communication
- No clear understanding of roles, different expectations
- Drawings-based work
- Partly different reference heights and systems of units
- High time pressure for many iterative processes
- Overlapping of the design phases

Therefore, it is necessary to have a new approach to the slots and opening planning workflow with the help of BIM methods.

3.3 3D BIM model-based method

There is no standard BIM use case description available for the BIM-based slot and opening coordination process. However, it can be defined as the "process of conducting coordinating MEP penetrations between MEP and various specialist planners in the construction project, particularly architectural design and structural design". BIM model-based slots and opening planning is somewhat complex (technology-wise and it is new) than the conventional 2D-plan based approach; however, it provides many advantages such as:

- Slots or openings in the wall, floor, or ceiling are displayed in three dimensions, eliminating numerous spatial misunderstandings and making errors more clearly recognizable.
- Slots and opening proposals can be inserted manually or automatically using available plugins; they are inherited with reference to the associated component.
- All necessary information such as shape, dimensions, and type related to void objects can be added as parameters (semantics); that is, the void knows its properties. Therefore, they can automatically adapt to modifications such as changes to the wall or ceiling thickness, the displacement of walls or changes to the pipe dimensions etc.
- BIM-based clash detection method eliminates the century-old 'overlay method' in special coordination.
- Void proposal objects can be filtered out with distinct colours by assigning different material or parameter properties. This helps quick identification and review of the proposed void objects. A typical example for parameter-based colour coding is based on void status parameters such as 'new', 'in process', 'old', 'deleted'.

Standard process for slot and opening planning

The BuildingSMART guidelines VDI 2552 Part 11.2[13] recommend that 'IFC-based slots and openings planning' is feasible under HOAI[14] specifications. Furthermore, the first step in using a digital building model to plan the slots and openings is to organize the exchange process with all those involved in the planning and to document it in writing,

---

13 https://ucm.buildingsmart.org/use-case-details/2376/de
14 https://www.hoai.de/hoai/volltext/

preferably in the BEP[15]. The general contract requirements are illustrated in Figure 3.6. If this does not occur, the collaborators will produce a 3D model that, in the worst-case scenario, will not be enough for the purpose and are not interoperable or compatible.

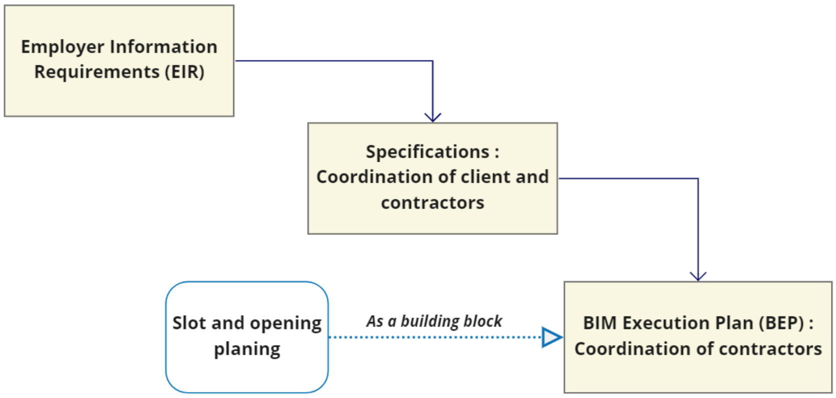


Figure 3.6: Contractual requirements (translated by the author from VDI 2552 Part 11.2)

In addition, the VDI 2552 Part 11.2 recommends that there must always be one responsible or contact person (BIM coordinator). Furthermore, the discipline/role of the BIM overall coordinator, who is in charge of control and merging, is supplemented. The general BIM roles required to conduct IFC-based slot and opening planning in construction projects regardless of the contractual obligations are given in Table 3.1.

Table 3.1: BIM roles required for the process of slot and opening planning (translated by the author from VDI 2552 Part 11.2)

Discipline/ Roles	Description
Coordination	Tasks of coordination in the opening planning. Must provide a contact person for the models.
Creation of slots and openings as void provisions	Tasks of creating opening proposals in the opening planning. Must provide a contact person for the models.
Examiner/Approval person	Tasks of the audit in the opening planning. Must provide a contact person for the models.
BIM overall Coordination	Overall data coordination, merging and checking of the models.
BIM management	Coordination of BIM applications in projects.

[15] https://www.designingbuildings.co.uk/wiki/BIM_execution_plan_BEP

The guideline VDI 2552 Part 11.2 provides a BPNM diagram for the ‘openBIM based slot and opening planning’ workflow (see Figure 3.7). In addition, it describes the minimum requirements concerning the model elements to be used and the associated information requirements for the slot and opening planning, irrespective of the service phase. However, the guideline contains only a methodical concept, and other technical aspects (how to communicate approval information) of slots and openings coordination is open and depends on the projects.

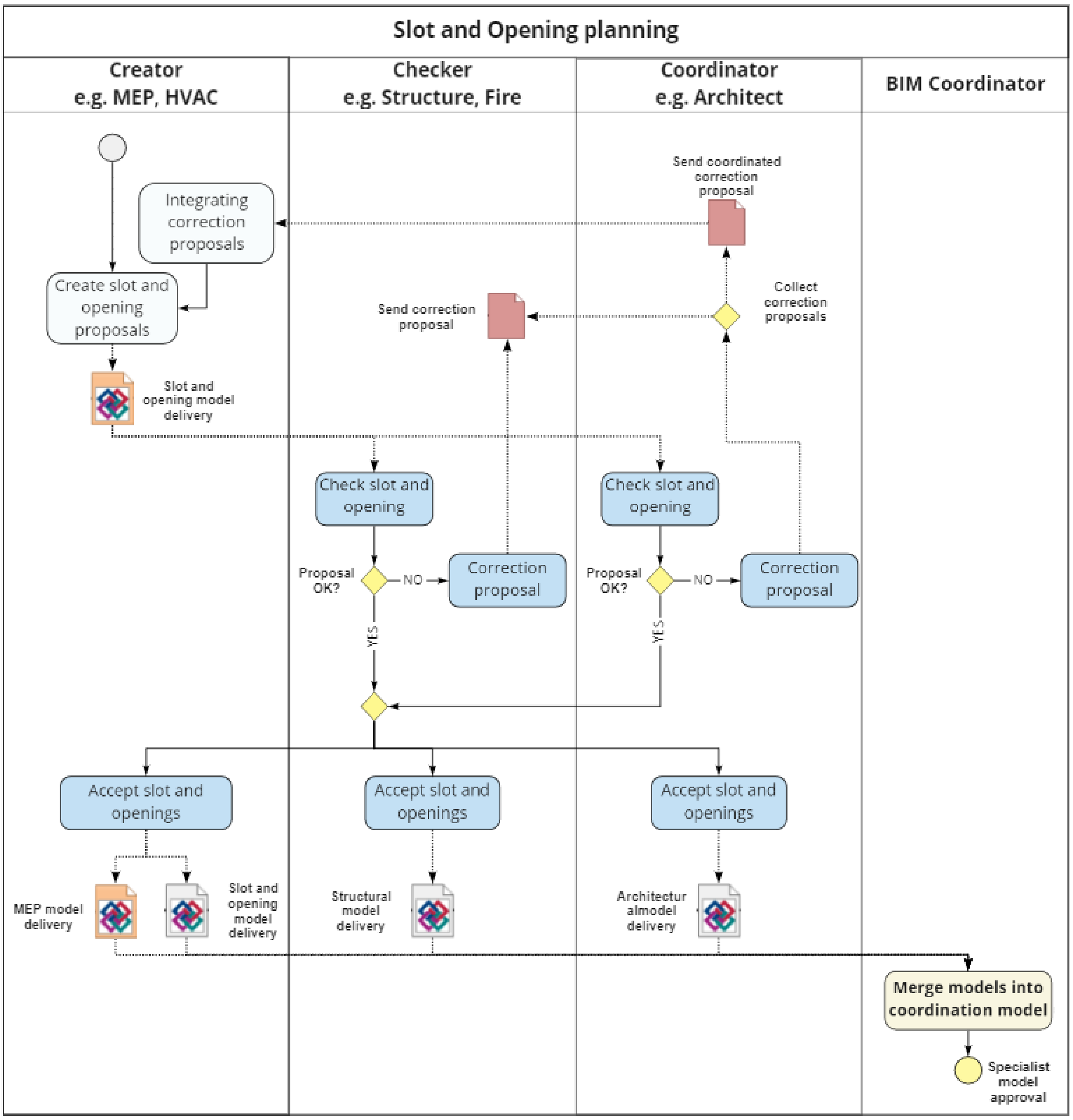


Figure 3.7: Slot and opening planning BPNM (translated by the author from VDI 2552 Part 11.2 guideline for slots and opening planning)

As per the BuildingSMART guideline (VDI 2552 Part 11.2) and according to the guideline provided by Hofbeck et al. (2021), the overall slot and opening planning process can be summarized in three main subprocesses:

1. The creation of the separate provision for void model (or PfV model)
2. Checking and approval of voids from PfV Model
3. Integration of approved voids into discipline-specific BIM models

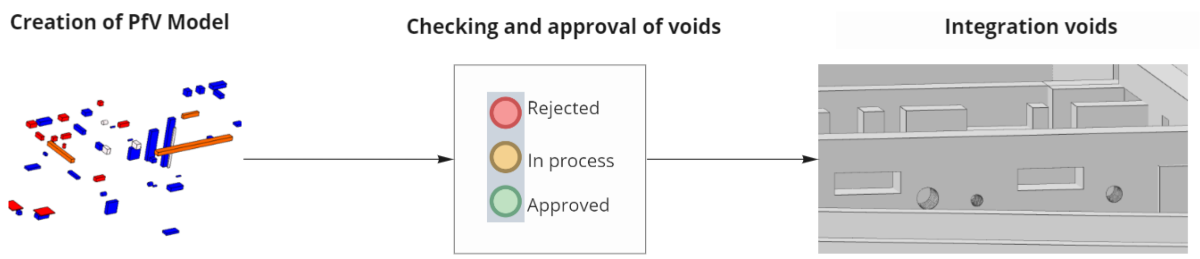


Figure 3.8: Key subprocess in model-based slots and openings coordination

3.3.1 Creation of PfV model

In this step, the slots and voids are defined in a separate 3D model, usually by the MEP planner. The void objects can be created manually or automatically using additional plugins. This is done based on the physical clashes/interference between the architectural and MEP models. The interference regions between MEP-element and architectural elements need to be carefully analyzed in the case of automatic void creation. In open BIM workflows, an IFC file is generated so that it will be interoperable in the design environment of other involved planners. The clash hierarchy defined in BEP will assist the MEP planner to check relevant clashes at interference regions, as shown in Figure 3.10. Generally, this includes all structurally relevant and construction relevant slots and openings.

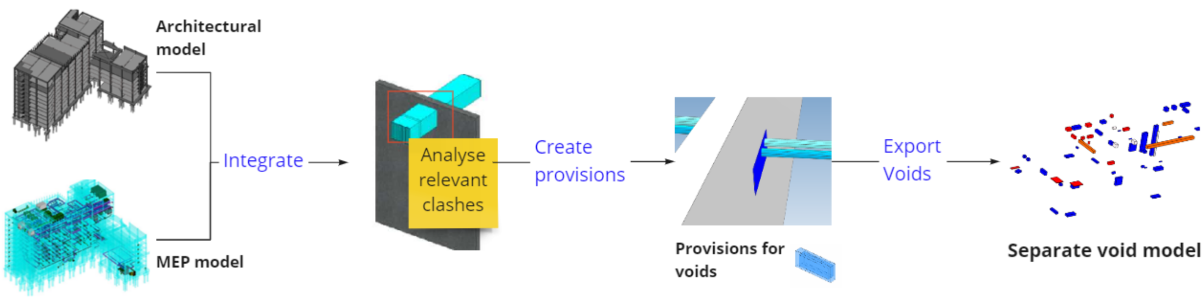


Figure 3.9: General method of creation of void model

		tragende Bauteile OBJ und TWP							nichttragende Bauteile OBJ												HLSK		ELT	GA	AV		
		Gründung	Decke+Dach	Wand	Stütze	Balken	Treppe+Rampe	SuD Rohbaurelevant	Wand	Dämmung	Dachaufbau	Fußbodenaufbau	Abgehängte Decke	Bekleidung	Fenster	Freiraum vor Fenster	Tür	Freiraum vor Türen	Fassade	Einrichtungen/Möbel	Komponente	Dämmung	Elektro	Küchentechnik	Medientechnik	Außenanlage	SuD Ausbaurelevant
tragende Bauteile OBJ und TWP	Gründung	WP2																									
	Decke+Dach	WP2	WP2																								
	Wand	WP2	WP2	WP2																							
	Stütze	WP2	WP2	WP2	WP2																						
	Balken	WP2	WP2	WP2	WP2	WP2																					
	Treppe+Rampe	WP2	WP2	WP2	WP2	WP2	WP2																				
SuD Rohbaurelevant		*	*	*	*	*	*	*																			
nichttragende Bauteile OBJ	Wand	WP3	WP3	WP3	WP3	WP3	WP3	*	WP3																		
	Dämmung	WP3	WP3	WP3	WP3	WP3	WP3	*	WP3	WP3																	
	Dachaufbau	WP3	WP3	WP3	WP3	WP3	WP3	*	WP3	WP3	WP3																
	Fußbodenaufbau	WP3	WP3	WP3	WP3	WP3	WP3	*	WP3	WP3	WP3	WP3															
	Abgehängte Decke	WP3	WP3	WP3	WP3	WP3	WP3	*	WP3	WP3	WP3	WP3	WP3														
	Bekleidung	WP3	WP3	WP3	WP3	WP3	WP3	*	WP3	WP3	WP3	WP3	WP3	WP3													
	Fenster	WP3	WP3	WP3	WP3	WP3	WP3	*	WP3	WP3	WP3	WP3	WP3	WP3	WP3												
	Freiraum vor Fenster	WP3	WP3	WP3	WP3	WP3	WP3	*	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3											
	Tür	WP3	WP3	WP3	WP3	WP3	WP3	*	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3										
	Freiraum vor Türen	WP3	WP3	WP3	WP3	WP3	WP3	*	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3									
	Fassade	WP3	WP3	WP3	WP3	WP3	WP3	*	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3								
	Einrichtungen/Möbel	WP3	WP3	WP3	WP3	WP3	WP3	*	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3							
HLSK	Komponente	WP2	WP2	WP2	WP2	WP2	WP2	*	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP2	WP3	WP2	WP3	WP3	WP3						
	Dämmung	WP3	WP3	WP3	WP3	WP3	WP3	*	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP3					
ELT	Elektro	WP2	WP2	WP2	WP2	WP2	WP2	*	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP2	WP3	WP2	WP3	WP3	WP3	WP3	WP3				
GA	Küchentechnik	WP2	WP2	WP2	WP2	WP2	WP2	*	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP2	WP3	WP2	WP3	WP3	WP2	WP3	WP2	WP2			
AV	Medientechnik	WP2	WP2	WP2	WP2	WP2	WP2	*	WP3	WP3	WP3	WP3	WP3	WP3	WP3	WP2	WP3	WP2	WP3	WP3	WP2	WP3	WP2	WP2	WP2		
	Außenanlagen																										
SuD Ausbaurelevant									*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Die SuD Planung wird durch mehrere Methoden/Prüfungsregelsätze geprüft, um die Integration der Durchbruche in allen Gewerken zu gewährleisten

Figure 3.10: Example of clash matrix incorporated with slots and openings planning in BEP (Bollinger und Grohmann GmbH, 2022)

3.3.2 Checking and approval of voids

Decision-making process

After step 1, as described in section 3.3.1, the next step is the ‘checking and approval of voids’ by responsible parties, generally by architectural and structural discipline specialists. Typically, the approval process is an iterative process starting with data drop (void model) from MEP. Similar to any other approval process involved with multiple parties, this process also takes time and effort to finish. This subprocess can be carried out either in parallel or in the sequential approach.

Sequential checking and approval: In this approach, the approval process is carried out one after the other, as shown in Figure 3.11. Generally, a preliminary check is carried out for all void proposals from the architectural planning point of view. Then, the structural discipline specialist checks already approved voids by the architectural discipline. Therefore, voids that are already being rejected by architectural planning initially will not be checked by the structural design; it will lead to additional loops during the overall inspection. This approach is well suited for the parameter-based approval method (Hofbeck et al., 2021).

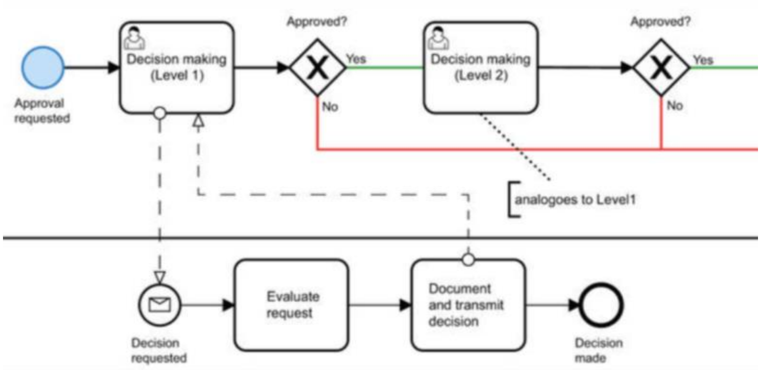


Figure 3.11: Concept of sequential checking and approval process (liNear, 2021)

Parallel checking and approval: In this approach, a specialist planner from both architectural and structural disciplines independently and concurrently checks the void proposals based on their design criteria. The main advantage of this approach is the possibility of a quick review. However, a void object rejected by one planner (e.g. structural) might be approved by the other planner (e.g. architectural); in such cases, the approval information from different parties is more comprehensive. The BCF-based approval method is suitable for this approach (Hofbeck et al., 2021). The process diagrams provided by the BuildingSMART (VDI 2552 Part 11.2) is based on a parallel approach.

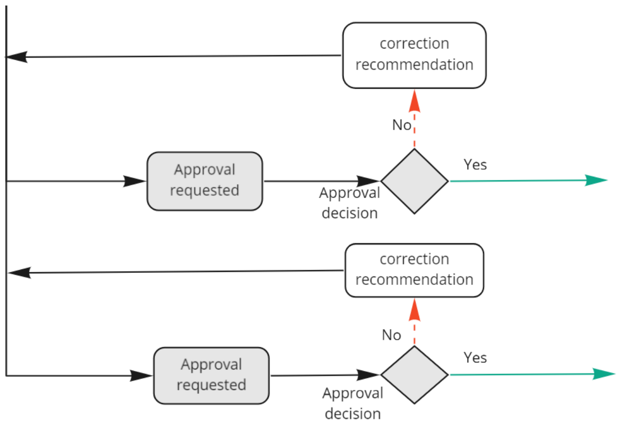


Figure 3.12: Concept of parallel checking and approval process (adapted from Hofbeck et al., (2021))

Data-exchange methods

Many data exchange methods are available for the communication of approval information between the involved planners. The most common approach for exchanging approval information from specialist planners to MEP planners is either issue-based or parameter-based; file-based approaches. The issue-based approach is done based on BCF-issues. The parameter-based approach is based on the predefined project parameters to each void object. According to Hofbeck et al. (2021), during the approval process, voids can have one of the following states:

- IN PROCESS or IN COORDINATION or ON HOLD: When void objects are not yet checked and just created for the first data drop by the MEP planner or when one of the specialist planners reject the void, and it is in coordination back again
- APPROVED: When specialist planners check and approve the void
- REJECTED: When specialist planners check and reject the void
- NEW: Void proposals that come after the first coordination iteration
- DELETED: when a void proposal is no longer needed for the MEP elements

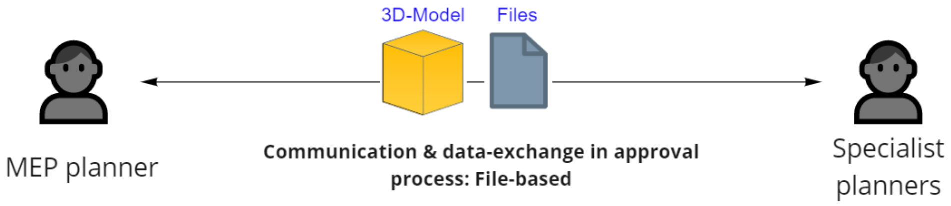


Figure 3.13: File-based communication and data exchange in the approval process

3.3.2.1 BCF issue-based approval method

In this method, the MEP distribute void models via data drop to all involved planners for approval, with all voids having status 'on hold' or 'pending' or 'in coordination'. Then specialist planners check each void in their design environment (BIM authoring tool), and when a void object has an issue, then a BCF issue is crated with necessary information. These issues are communicated back to the MEP planner, either as a file-based exchange or as a web service. Issues are solved and exchanged based on the comments provided in the BCF-issues. All voids with solved issues are set to be approved. Additionally, all voids without an issue communication are assumed to be also approved. In the next iteration, MEP updates the void model with approved/ coordinated, deleted, and new parameters. Ideally, this can be done via parameter or material texture, where the parameter-based view filtering can be utilized for a quick review of the approval status (e.g. traffic light colour coding) (Hofbeck et al., 2021).

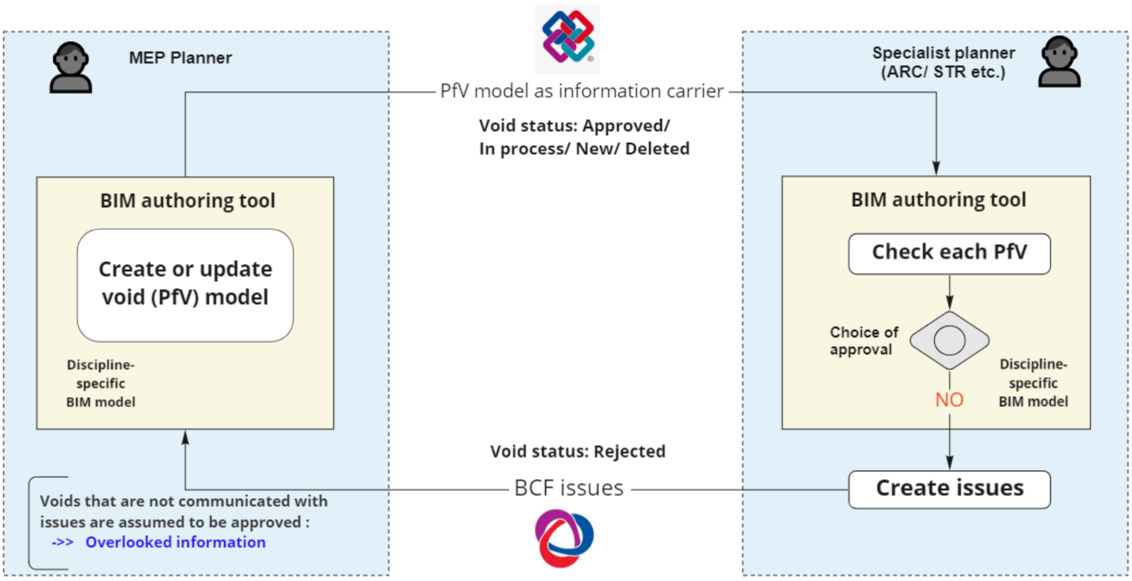


Figure 3.14: Illustration of void information flow: BCF-based approval method

3.3.2.1 Void parameter-based method

This approach is done by sharing data in the predefined attributes or parameters associated with each void object in the PfV model. Therefore, the parameter-based approval process requires precise coordination of the workflow in advance. Rules should be defined as how void proposals go through the review process and is finally transferred as cut-outs to the native models of the specialist planners. In this method, the communication takes place via parameter values. It must be coordinated whether these are transmitted via the exchange of IFC models or other approaches. A predefined definition of required parameters and their values is required for this approach. Approval parameters and their values are the data in this process, which should be maintained throughout the planning process (Hofbeck et al., 2021).

In order to ensure consistency, the MEP Planer generally creates the parameters. Typically this includes three parameters that correspond to each specialist planer, and they are:

- Approval status (mandatory)
- Approval date (mandatory)
- Approval comment (optional)

First, the MEP planners will provide the PfV model with empty or updated void parameters. Then, the specialist planners will check the void and write/choose the relevant approval parameters related to each void in the void model. Finally, these data will get back to the MEP planner in a file-based manner. This can be done in many ways, such as 3D models (native or IFC) in a central file-based or local file transfer or as excel sheets or via external plugins and more. Generally, the approval data from specialist planners are merged by the Architect. The final approval parameter will be the combination of individual approval data. After the first iteration, all approval information will be in the PfV model as parameters. In the parameter-based approval process, the complex issues, which could not be possible to solve via comment parameter, are communicated using BCF files. Some of the commonly used parameter-based approval methods are discussed in the following sections.

- **IFC or Native file-based approach (local file transfer)**

In this approach, MEP planners provide PfV model either as IFC or as a native file with all void objects having associated predefined approval parameters. This will be imported into the BIM authoring tool of the specialist planners, and it will be checked according to the required design criteria concerning the discipline-specific BIM model. The approval information (status, date, comment) will be entered into the relevant approval parameter for the specific discipline. After that, the model will be sent back to the MEP planner in native/ IFC format (see Figure 3.15).

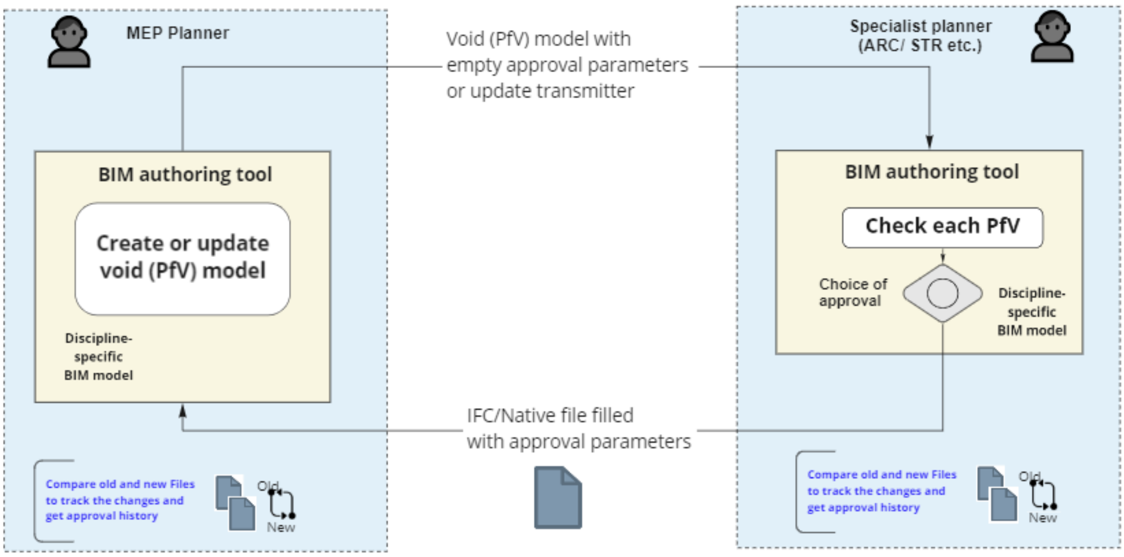


Figure 3.15: Illustration of void information flow: Void parameter-based method

Furthermore, in this approach, in order to track changes that happened in the approval status/ position/ dimension related to void objects in different approval iterations, it needs file comparison. Rule-based model comparison is an option for tracking the changes between the PfV models. For example: in the case of comparison using the

Solibri Model checker[16], the global (X, Y, Z) coordinates, property sets, dimensions can be used to create rule sets. However, change tracking and comparing BIM tools are often prone to errors (Pilehchian et al., 2015).

- **Central file-based method**

The central PfV model will be the master file for the shared project, located in an environment with the same software and central storage. In this method, checking and approval of the PfV model will be done in the shared central model, accessible to all involved disciplines such as MEP planner, Architect, and structural engineer. Project participants will share this model and work in a local copy. In the local copy, provide their choice of approval as parameter values associated with each void object in the PfV Model. Then, this will be merged with the central file. Generally, parameter-based filtering will be employed for quick review. The shared central void model will be considered as the single source of information. There is no file transfer among the involved discipline except the MEP uploading the PfV model to the server. However, the risk of damaging the central file is high. Most importantly, change tracking at the object level is not possible in this approach. The case can happen, such as someone else will edit the parameter of a void object that is not intended for their discipline in the approval process. Having exclusive ownership of properties of void objects is missing here. Employing a checking routine would be an option to manage the model.

One of the most common examples for this approach in Autodesk-based closed BIM projects is using Revit central file, where the participants share the same Revit central database. All team members create their own local copies of the central PfV model, work locally, and then synchronize their choice of approval to the central PfV model using the ‘Synchronize with Central’ command. It also can be done by Revit Server Accelerator configuration via WAN or using BIM360. The ‘workset control’ in the case of the Revit central file provides ownership at the object level, not parameter/ property level. One of the referred projects from B+G Ingenieure Bollinger und Grohmann GmbH was based on Revit central file-based approval process, where Revit central file was used for the coordination process.

[16] https://www.solibri.com/de/

- **Excel-based method**

The excel sheets act as an information career or messenger for the approval process in this method. Here, the GUID of voids must be consistent in each iteration loop, and also, it should be possible to import data from excel sheets in the design authoring tool of MEP planner. In this approach, the MEP planner delivers the PfV model or models in the data drop as IFC or native file depending on the BEP. Then, the specialist planners will extract all void information, including approval data from the void model, to an external excel file. This can be generally achieved via APIs provided by the BIM authoring tools. This approach identifies each void based on an element ID (IfcGUID or native element ID). Checking and approval of voids can be done in this approach, either in BIM authoring software or in Excel sheet itself. Then, the excel file with all approval information will be sent back to the MEP planner. Finally, the MEP planner will import the approval data from the excel sheet to their BIM authoring tools using API functionalities and change the status of the voids in the void model for the next iteration. In reality, projects with excel oriented slots and opening planning will have a standard template for excel sheets and parameter names agreed upon from the beginning of the process.

The choice of approval as ‘approved’ or ‘rejected’ can be given by the respective checking planner to the component ID/GUID of the respective void proposal and imported into the creator's authoring application. Each iteration creates a new excel file corresponding to the provided IFC or native void file. Also, keeping consistency between excel files and the void model by tracking changes in BIM models can be challenging, especially when there is a change in IfcGUID. The main problem is that the overall approval data is in multiple files, and they are disconnected. Nevertheless, one interviewee mentioned that this approach is reasonable in their current closed BIM projects.

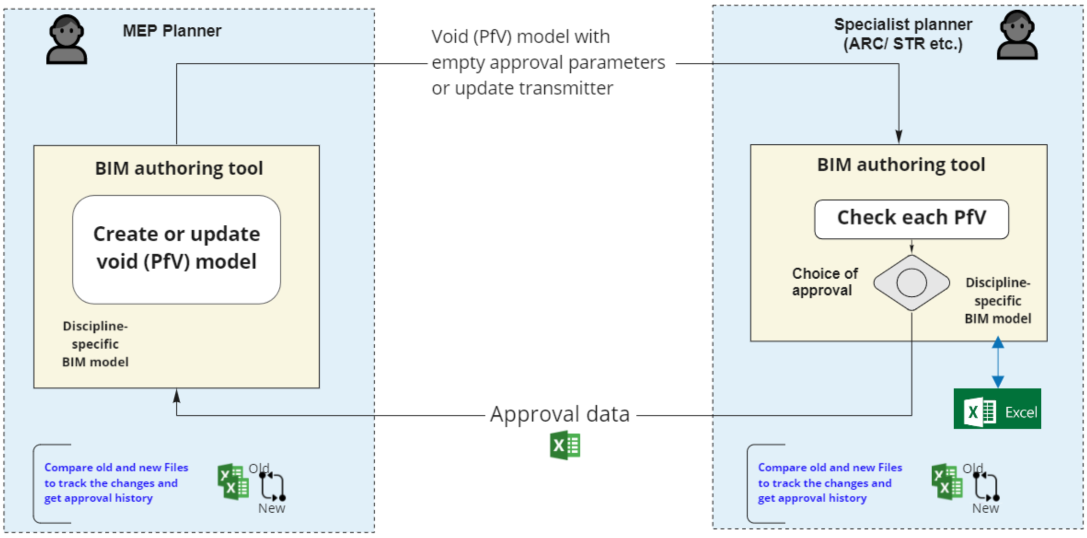


Figure 3.16: Illustration of void information flow: Excel-based approval method

An example of this approach in the case of Revit as one of the BIM authoring tools in the coordination is given below. The Dynamo scripts can be created to extract the necessary void information from a PfV model. All data except the centroid of void objects (global X, Y, Z) can be directly read from parameters of the PfV objects using Dynamo (see Figure 3.17). The centroid position needs to be calculated at the center of gravity point.

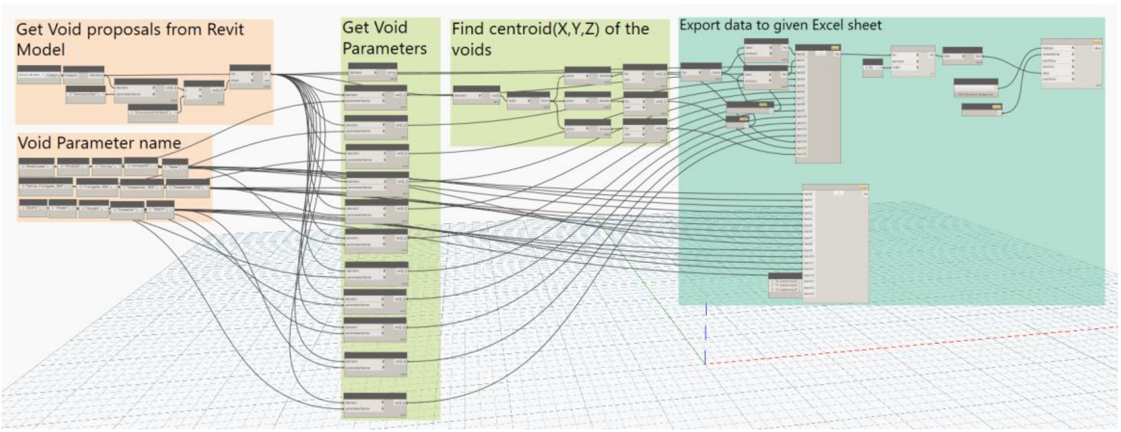
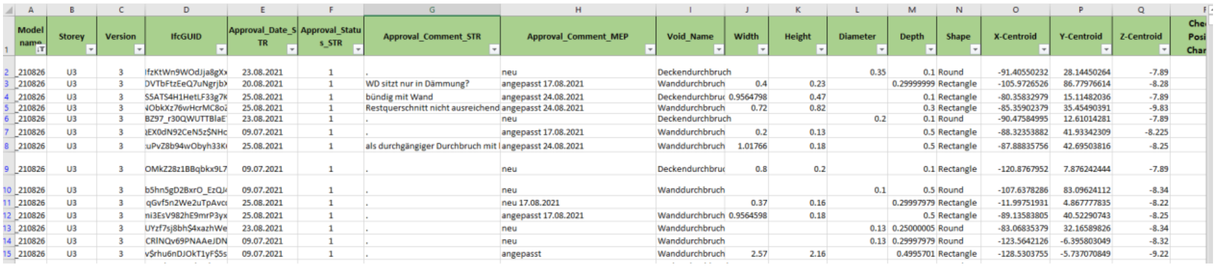


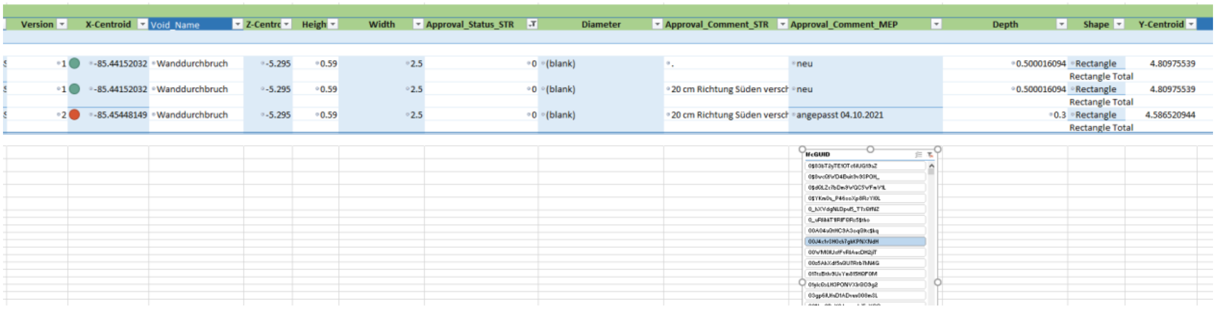
Figure 3.17: Dynamo script created to extract void data from Revit

After importing all void information to an excel sheet, it acts as a data container (see Figure 3.18); this can be further used for object-level change tracking of voids. The different excel sheets exported in approval iterations can be merged to get all approval data in one source. Then, by using Excel's pivot functions in the merged datasheet, an approximated versioning of objects can be achieved. Generally, the native element ID (closed BIM)/ IfcGUID (open BIM) of the void will be used to match the voids for versioning (see Figure 3.19). Additionally, further insights such as the number of pending approvals, number of deleted voids, and more can also be inferred using Pivot functions for a quick progress assessment.



	Model name	Storey	Version	IfcGUID	Approval_Date_STR	Approval_Status_STR	Approval_Comment_STR	Approval_Comment_MEP	Void_Name	Width	Height	Diameter	Depth	Shape	X-Centroid	Y-Centroid	Z-Centroid	Che Posi Char
2	_210826	U3	3	IfzKtWn9WOdJja8gXx	23.08.2021	1	.	neu	Deckendurchbruch			0.35	0.1	Round	-91.40550232	28.14450264	-7.89	
3	_210826	U3	3	DVTbFtzEeQ7uNgrjbX	20.08.2021	1	WD sitzt nur in Dämmung?	angepasst 17.08.2021	Wanddurchbruch	0.4	0.23		0.29999999	Rectangle	-105.9726526	86.77976614	-8.28	
4	_210826	U3	3	S5ATS4H1HetLF33g7K	25.08.2021	1	bündig mit Wand	angepasst 24.08.2021	Deckendurchbruc	0.9564798	0.47		0.1	Rectangle	-80.35832979	15.11482036	-7.89	
5	_210826	U3	3	NObkXz76wHcrMC8oz	25.08.2021	1	Restquerschnitt nicht ausreichend	angepasst 24.08.2021	Wanddurchbruch	0.72	0.82		0.3	Rectangle	-85.35902379	35.45490391	-9.83	
6	_210826	U3	3	BZ97_r30QWUTTBlaE	23.08.2021	1	.	neu	Deckendurchbruch			0.2	0.1	Round	-90.47584995	12.61014281	-7.89	
7	_210826	U3	3	2EX0dN92CeN5z$NHc	09.07.2021	1	.	angepasst 17.08.2021	Wanddurchbruch	0.2	0.13		0.5	Rectangle	-88.32353882	41.93342309	-8.225	
8	_210826	U3	3	:uPvZ8b94wObyh33K	25.08.2021	1	als durchgängiger Durchbruch mit	angepasst 24.08.2021	Wanddurchbruch	1.01766	0.18		0.5	Rectangle	-87.88835756	42.69503816	-8.25	
9	_210826	U3	3	OMkZ28z1BBqbkx9L7	09.07.2021	1	.	neu	Deckendurchbruc	0.8	0.2		0.1	Rectangle	-120.8767952	7.876242444	-7.89	
10	_210826	U3	3	b5hn5gD2BxrO_EzQJ	09.07.2021	1	.	neu	Wanddurchbruch			0.1	0.5	Round	-107.6378286	83.09624112	-8.34	
11	_210826	U3	3	qGvf5n2We2uTpAvc	25.08.2021	1	.	neu 17.08.2021		0.37	0.16		0.29997979	Rectangle	-11.99751931	4.867777835	-8.22	
12	_210826	U3	3	ni3EsV982hE9mrP3yx	25.08.2021	1	.	angepasst 17.08.2021	Wanddurchbruch	0.9564598	0.18		0.5	Rectangle	-89.13583805	40.52290743	-8.25	
13	_210826	U3	3	UYzf7sj8bh$4xazhWe	23.08.2021	1	.	neu	Wanddurchbruch			0.13	0.25000005	Round	-83.06835379	32.16589826	-8.34	
14	_210826	U3	3	CRlNQv69PNAAeJDN	09.07.2021	1	.	neu	Wanddurchbruch			0.13	0.29997979	Round	-123.5642126	-6.395803049	-8.32	
15	_210826	U3	3	v$rhu6nDJOkT1yF$5s	09.07.2021	1	.	angepasst	Wanddurchbruch	2.57	2.16		0.4995701	Rectangle	-128.5303755	-5.737070849	-9.22	

Figure 3.18: Void information exported to excel file from Revit using Dynamo



Version	X-Centroid	Void_Name	Z-Centr	Heigh	Width	Approval_Status_STR	Diameter	Approval_Comment_STR	Approval_Comment_MEP	Depth	Shape	Y-Centroid
1	-85.44152032	Wanddurchbruch	-5.295	0.59	2.5	0	(blank)	.	neu	0.500016094	Rectangle	4.80975539
											Rectangle Total	
1	-85.44152032	Wanddurchbruch	-5.295	0.59	2.5	0	(blank)	20 cm Richtung Süden versch	neu	0.500016094	Rectangle	4.80975539
											Rectangle Total	
2	-85.45448149	Wanddurchbruch	-5.295	0.59	2.5	0	(blank)	20 cm Richtung Süden versch	angepasst 04.10.2021	0.3	Rectangle	4.586520944
											Rectangle Total	

Figure 3.19: Object-level versioned history of approval information of voids using the Pivot function

3.3.3 Integration of voids into BIM models

After the approval iteration process, all approved 'provision for voids' are integrated into the domain-specific BIM models as cut-outs in this subprocess by specialist planners in their design environment. However, in the case of open BIM projects, the voids that are transferred in the IFC data drop will be simple 3D bodies with the IfcBuildingElementProxy type. Therefore, these provisions must be adapted to corresponding cut-out objects as subtractions in the host elements. Nevertheless, this is not a straightforward process (BuildingSMART-Regionalgruppe Mitteldeutschland, 2020). There is no direct connection between provisions for voids and cut-out objects. The void provisions will be in a separated PfV or void model, and cut-outs need to be adapted and integrated into the discipline-specific model created in various BIM authoring tools. This is usually achieved with external plugins or using visual programming scripts such as Dynamo scripts in the case of Autodesk Revit. In addition, it is essential to keep consistency in all cut-out objects across the discipline-specific BIM models. Therefore, all late changes to already integrated voids must be communicated with the involved planners, ideally based on BCF-issues. An example of using Dynamo scripts to integrate the approved voids into the specialist model is given in Figure 3.20. In this approach, the voids are cut by creating an already defined family instance at the intersection of horizontal pipes with walls.

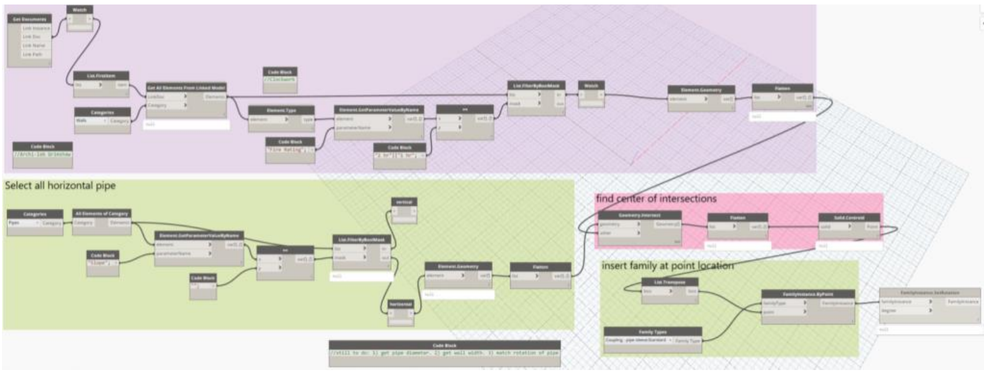


Figure 3.20: An approach to integrating approved voids to discipline-specific BIM model using Dynamo scripts

Checking routine

In each coordination iteration, coordinators conduct necessary model checks as a checking routine to ensure effective void planning. Few necessary checks are given below:

- Check completeness of the voids: All the interference regions between MEP elements and host elements must be provided with adequate provisions for the void. This can be checked and analyzed based on physical clash detection. The necessary parameters of voids such as dimensions, shape, and position must be verified.

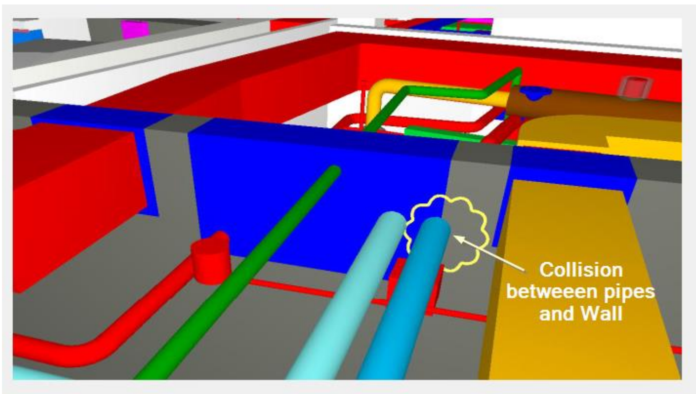


Figure 3.21: The provision of the void proposed is incomplete (pipes do not fit 100%) (Bollinger und Grohmann GmbH, 2022)

- Check the consistency of voids in discipline-specific BIM models: All void proposals in walls/ceilings/floors/beams must be adapted to the latest state in the architectural and structural models.
- Visual inspection: Check if all deleted provisions for voids have already been deleted with corresponding slots and openings in the models. This visual checking

can be done concerning the latest MEP design. For example, if an opening is located in a wall, but there is no MEP route going through it, then such unnecessary openings must be communicated with planners and deleted. Additionally, the parameter-based view (colour) filtering would help planners to do visual checking efficiently. A visual inspection is also needed to ensure the alignment of the openings in the model.

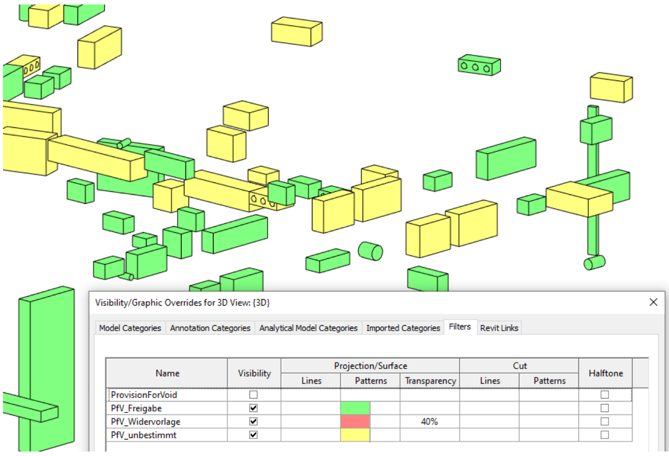


Figure 3.22: Colour-filters employed for approval parameters in Revit for a quick review of the status of voids

3.4 Existing challenges and limitations

After analyzing the most common methods in the slot and opening coordination process, the summary of the comparison is listed in Table 3.2.

Table 3.2: Comparison of existing methods

Approval method	Opportunities	Limitations
2D-drawing based	• No additional files and workflows are needed • Changes are marked with change clouds	• Incomplete interpreted information • Disconnected workflow from the design environment • Duplication of information • The time lag between the approval process in crucial phases • Difficult to track the history of changes using revision clouds. • Change clouds can be overlooked either by checking or by integrating them.
BCF(issue)-based)	• Connected to design environment • Open BIM method • Quick issue management	• Overlooked approval information; predictive approach • Lack of overview over approved and rejected openings • Duplication of information • Hard to track the history of changes for every void

Native central file (parameter)-based	• Connected to design environment • Filtering is possible for quick review	• Limited to closed BIM. • Hard to track the history of changes • The possibility to manipulate/damaging the central file is high. Coordination overview may get lost
Native file-based (local file transfer) (parameter)-based	• Connected to design environment • Closed BIM method • Filtering is possible for quick review	• Hard to track the history of changes • Information of voids are scattered across multiple models.
IFC file based (parameter)-based	• Connected to design environment • Open BIM method • Filtering is possible for quick review	• Hard to track the history of changes • Information of voids are scattered across multiple models.
Excel (parameter)-based	• Files can be linked to the design environment • Possible in open BIM as well as closed BIM workflow • Easy review with excel sheets • Easily adaptable by planners • Filtering is possible for quick review	• It is not a direct approach and demands much effort to manage the process. • Hard to track changes and to merge and keep void data in a versioned manner

Summary

The technological and non-technical challenges are listed below, which includes the results from the personal interviews as well.

- Co-dependency (the nature of the process):
  - The slots and openings can be considered as one of the most relevant BIM model objects as it has equal influence on architectural, structural, MEP disciplines.
  - The choice of approval of voids from one discipline affects other disciplines. Therefore, intensive and timely communication is required.
  - The design freeze between phases limits consistent coordination and leads to model inconsistencies.
- Contractual issues:
  - The full coordination happens only during the execution phase as per the HOAI regulation. This leads to multiple change requests, and planers do not get enough time to adapt it before the next phase, which in turn causes the processes to overlap. This is also applicable for late change requests that happen even during construction.

   - The HOAI regulations are independent of the method and technology. This HOAI contractual condition contradicts the BIM methodology, where there is a need to provide detailed information in the earlier phases.
   - The linear nature of existing guidelines and standards conflicts with design reality. Therefore, fast-tracked BIM projects in a competitive market underestimate the time and effort required in the iterative design process, which results in a lower fee structure.
- Multiple sources of information:
   - In all file-based approaches, except the central file-based method, the communication of voids via files lead to multiple sources of information. Information about approval voids is distributed across several files. An accessible history is missing; there is a danger of losing control over the sound and correct source of information.
- Predictive approval information:
   - In the case of the BCF issue-based approach, only issues are communicated back to the MEP planner. So, it is clear that the voids which are not communicated are predicted as approved by the involved specialist planners. But, the chances of error in this approach is high. If one of the parties forget to create issues to a rejected void, it will be considered as approved, and the issue will be missing in this case.
- Current practices in notifying changes or updates lie in transferring IFC models between the Specialist planners and MEP planners. This file-based data exchange produces massive data volume and redundancy in information lying in BIM models. The information in the BIM models is constantly being changed and updated by the stakeholders, and saving as a new file for such changes causes Information redundancy. This causes the stakeholders to spend time and effort to synchronize with the new BIM model

4 Design and Development

This chapter proposes a solution for the existing problem identified in chapter 3. The chapter is divided into two sections. In the first section, 4.1, a concept is proposed. The developed concept is intended to reduce the effort required to conduct this intensive approval process and to solve some of the crucial challenges identified in chapter 3. Later in section 4.2, the implementation aspect of the prototype to demonstrate the developed concept is explained.

4.1 Concept development

4.1.1 Objectives for a solution

In order to propose a solution to overcome the existing problems in the slot and opening coordination process, the objectives were identified from the beginning as:

Open and federated BIM

As this coordination process involve multiple disciplines with the possibility of not working in one native software environment, an open BIM concept is required. The overall concept should be in accordance with the BuildingSmart standard process for slot and opening coordination (VDI 2552, part 11.2), as illustrated in Figure 4.1. The requirement is to have an approach that is easier to handle than any other method currently in practice. Additionally, it is desired that the end-user does not have to acquire knowledge in the technical and procedural details of the BuildingSmart standard process. Also, all approval, change management, and communication tasks need to be localized in a model-based fashion in the preferred BIM authoring tool of the involved planners. This would make BIM models clean and provide freedom to work in separate or discipline-specific BIM models.

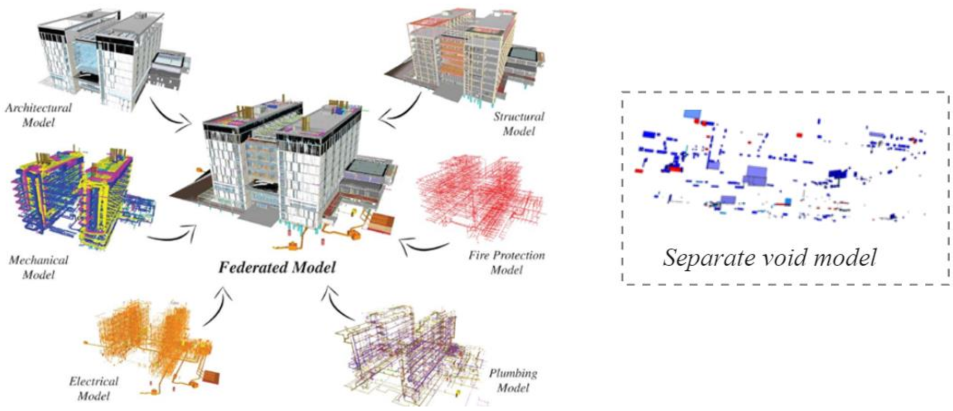


Figure 4.1: Federated BIM data model (adapted from Beach et al., (2017))

Single source of Truth (SSOT)

One of the main problems in the existing methods is the multiple sources of truth; due to file-based communication. The void information and corresponding approval information in the different iterations are scattered in different files across multiple disciplines. Therefore having a single source of truth concept will enhance the transparency of the process and brings efficiency in the process by bringing control over 'approved'/'rejected'/'on hold' voids to all involved disciplines in the process. As explained in section 3.3, using standard data exchange formats like IFC or BCF files would lead to duplication and confusion if all related data is not stored in one easily accessible common location. A single source of truth concept can be achieved by a collaborative database. A cloud-based collaborative database would provide the possibility to get the up-to-date information of the voids in real-time.

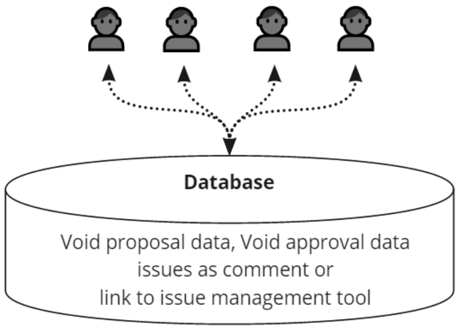


Figure 4.2: Concept of a single source of truth (SSOT) with a collaborative database

Object-based communication: Sender-receiver approach

As the process of slot and opening coordination itself is highly communication-intensive, exchanging complete files would not make sense. Therefore, if the discipline specialist could communicate void (object)-wise, then sending files back and forth, comparing files, and versioning of the whole model could be eliminated.

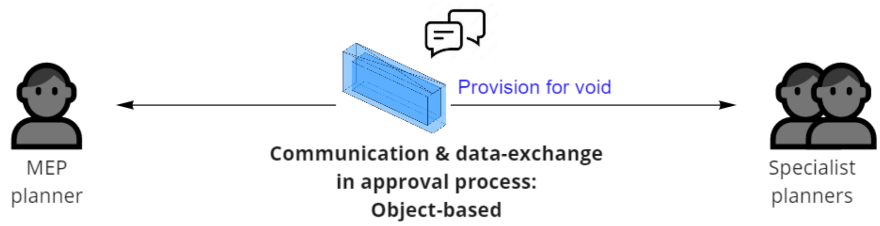


Figure 4.3: Concept of object-based communication

Keeping track using temporal versioning for void objects

The change tracking at an object level is required in the slot and opening process due to the following main reasons.

- To have an overview of each void regarding its position, geometry, and approval data, versioning of void objects is necessary.
- To integrate the coordination process into a rule-based environment.
- Enhance the Knowledge bases required for MEP coordination; the planner could see the past/present decisions in order to decide for the future

The requirement of having versioned data in the single source of information will assist the involved discipline to track the changes related to each object in the latest real-time. Additionally, it allows assisted change management which would not be possible without versioning. The vision is to have independent states of voids in each iteration and its corresponding approval information, as shown in Figure 4.4. The void state indicates the proposed void's state in the iterative approval process in a given time. Each void state will have position, shape, dimension and approval data (empty or given by specialist planners) associated with it. For every new iteration of a void object proposed by the MEP planner, a new void state needs to be created.

All approval data provided by specialist planners need to be linked with the corresponding void iteration state. This can be achieved through the collaborative database-oriented workflow, with each void object having a unique history. It must be possible to version voids based on their state and should be possible to track the changes of each void, or an accessible history should be available for each void object. Moreover, it is not unusual that there are up to 5 iterations or more for a single void in a complex BIM project.

Also, it needs to remove or reduce file comparison to track the previous iterations' changes and approval data related to void objects. In a smooth approval process, changes mean a reset of the status of the proposed voids. All other changes due to lack of communication can be avoided. Since the import and export of files across multiple files change the data structure, it is also prone to error to track the changes through the file comparison (Pilehchian et al., 2015). However, the file comparison method can be taken as an optional approach to double-check the results (for example, using Solibri Model checker). If object-level history tracking is possible, the specialist planners (as well as MEP planners if interested) do not need to compare the files to track the changes related to each void object.

In addition, having an overview of the previous decisions from the involved planners based on their discipline-specific design requirements would enhance the Knowledge bases required for MEP coordination. Knowledge management is one of the most important factors in reducing the coordination error and the need for many iterations (Korman et al., (2003); Juszczyk et al., (2016)). In other words, the planners could learn from the reasons for previous rejections and thus reduce the chances of having similar situations in the future.

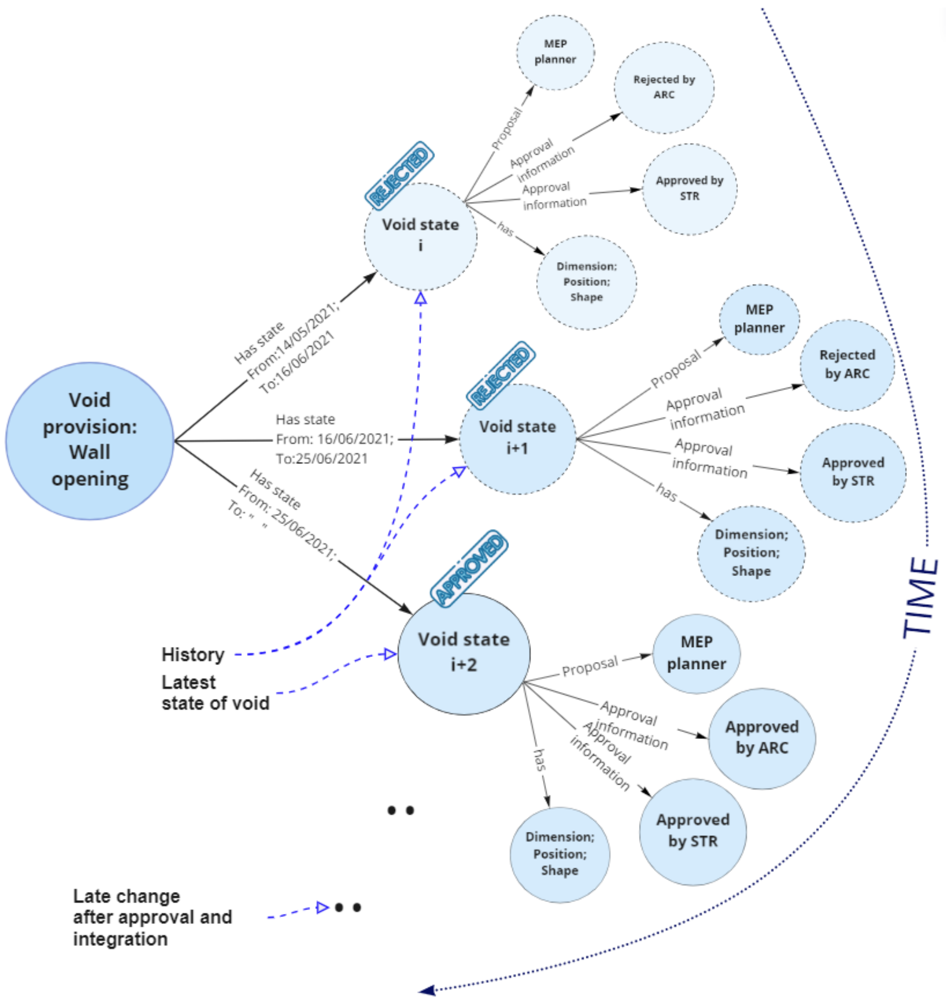


Figure 4.4: Concept of versioning for a void object states - example of a simple approval process in three iterations with planners from MEP, architectural (ARC), structural (STR) design disciplines

4.1.2 Concept proposal

In the concept, a collaborative cloud database is the central part of the process that ensures the single source of truth concept. All the information flow starts with an entry to the database, which determines the truth. This facilitates real-time object-based communication of voids with the most up-to-date approval, position and dimension information. Moreover, all approval, change management and communication tasks are localized in a model-based fashion in the preferred design tool of specialist planners. The connection between the design environment and the cloud database can be established via Web APIs. Therefore, it is possible to track the history of changes related to each void in a given time interval. In this proposed concept, the IFC void model acts as a geometry container to check and approve the voids between the involved disciplines and facilitate openBIM workflow. The information flow in the proposed concept is illustrated in Figure 4.5.

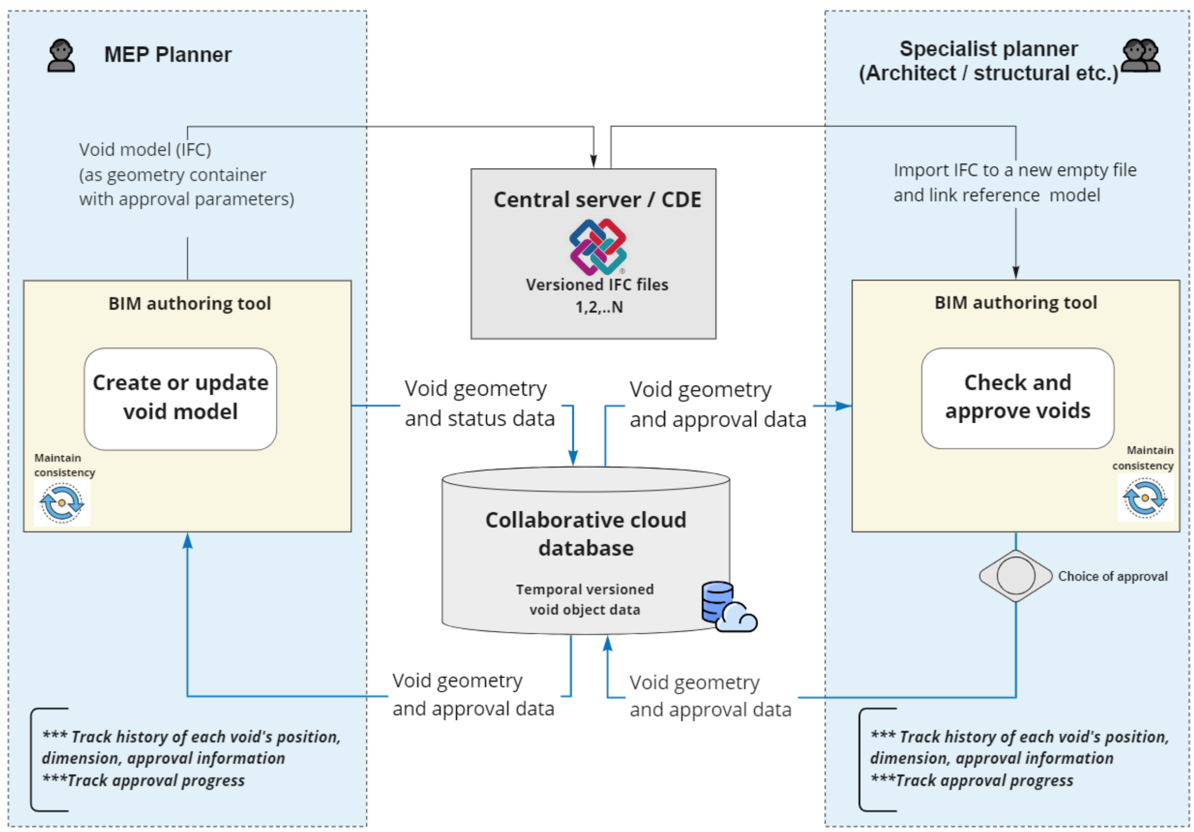


Figure 4.5: Information flow in the approval process of proposed slot and opening coordination concept

Provision for void proposal/ request

MEP discipline starts the coordination process by creating provisions for voids. This can be done according to the MEP planners' preferred method, either automatically or manually. Each void will contain the necessary project parameters. The commonly defined project parameters need to be agreed upon prior to the process with involved planners. A test scenario is recommended to conduct at the beginning to ensure all relevant parameters and scenarios have been considered in the parameter definition. Additionally, the elements need to have the same parameter GUID in every application MEP planner updates all void object data into the collaborative database with the necessary information. This includes void proposal information and time-based versioning. In addition, the MEP planner exports all void objects to a separate void instance model as IFC with a shared project origin. The shared project origin must be verified in this process to eliminate misplacement of the voids. The void IFC file is shared with empty/updated approval information in each iteration (as a data drop) to a central server or Common Data Environment (CDE). Generally, the shared parameters include the following information about each void and the project origin.

- **Void object parameters:** Shape, Width, Height, Diameter, Depth, Storey Number, Void name, IfcGUID
- **Shared approval parameters:** MEP approval status, MEP comment, MEP approval date, ARC approval status, ARC comment, ARC approval date, STR approval status, STR comment, STR approval date
- **Shared project origin:** The general project notes of the design drawings and specifications should clearly define the local relative building XYZ coordinates that other disciplines would use. The architect generally determines the model origin in coordination with the other specialist disciplines and should be located near the program origin. All specialist models must have the same model origin and be identically aligned in their XYZ axes. A 3D insertion point, inserted at previously agreed coordinates (e.g. XYZ = 0,0,0), helps check that all compartment models visually have the same origin.

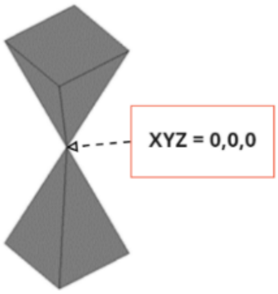


Figure 4.6: Example of a 3D coordination body for the shared project origin

Checking and approval of voids

The checking and approval of voids are done in specialist planners' preferred BIM authoring tool. The specialist planner gets the IFC PfV Model from a central server/ CDE in the data drop. Then, they import the IFC data to their BIM authoring tool, and voids are checked object-by-object according to discipline-specific design criteria. After checking, the approval data is updated to the database, where it is also versioned in a time-based manner according to the predefined schema. The history can be made available in the design environment as a history log or visualised with temporary indicators. The quick availability of history information assist planners to track the underwent changes of voids in highly complex projects.

Figure 4.7: Concept of visualization of history indicators of a void object with its previous version (V0, V1) in BIM authoring tool

Furthermore, this history-based concept allows the planners to do an assisted consistency check in the respective discipline-specific BIM model. The following checks can be done with the help of change information:

- Check if the position has changed
- Check if the status has changed
- Check if the size has changed

Communication of issues

In the proposed parameter-based approval process, communication of approval information from specialist planners to MEP through the database has the date, comment related to the object as text, discipline name, object ID (as IfcGUID).

Additionally, in most cases, it is enough to describe the issues in a few words (Hofbeck et al., 2021); for example, “move 19.6 cm towards north” or “move to original position or move 56.2 cm towards south”, etc. But, in case it is difficult to explain the issue through a few words (for example, issues in MEP service rooms), there is a need for additional visual communication. This can be provided either with an Issue Link (e.g. link to BIMcollab etc.) or create a snapshot and provide the link address (source of the images with the issue) instead of the verbal text. Thus, these issue links will be referenced to the void states, and they can be tracked similarly to other approval parameters.

Real-time Overview of the coordination progress

It is possible to have a coordination overview for the existing voids (as dashboards). This can be achieved by inferring the computer-readable data stored in the cloud database. An example case would be: get the overview of the status of the existing provision for voids in complex construction projects. This would show the involved MEP planer and specialist planners the progress of the coordination task, especially the approval process. A conceptual illustration is shown in Figure 4.8.

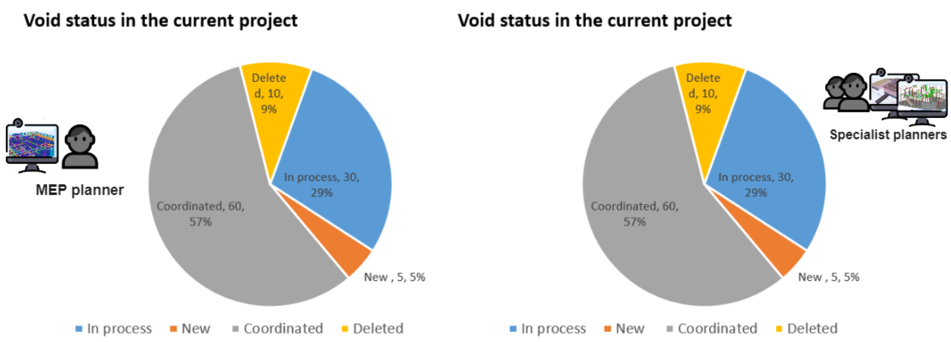


Figure 4.8: Conceptual illustration of Pie-chart showing the overview of the void provision’s status in BIM projects

4.2 Implementation

In order to validate the proposed concept in section 4.1, a prototype was implemented. This section describes the implementation aspects of the developed prototype in detail. At first technological choices are being analysed. Then the system architecture is provided. Later on, the implemented functionalities of the prototype are mentioned briefly. The scope of the prototype will be limited to a single BIM authoring tool. The integration of this tool with the BIM authoring tool can be achieved by using API programming. The requirement of the prototype is to act as a gatekeeper between the design environment and the database employed for storing the void information, as illustrated in Figure 4.9. The computer-readable data will be communicated (read/write) between the database and the BIM authoring tools based on predefined checks and rules.

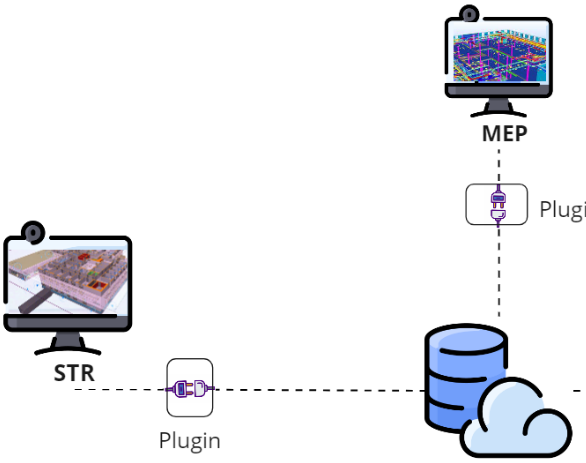


Figure 4.9: Concept of a prototype plugin as a gatekeeper between the cloud database and BIM authoring tool

4.2.1 Technological choices

In order to develop the prototype, necessary technologies and available software tools and digital solutions were checked as an initial step with regards to its acceptance in the AEC industry, affordability, intuitiveness, and easiness to handle.

4.2.1.1 BIM authoring software

As part of the concept, it was planned to have the planner's specific design environment be a local one. Therefore, all involved planners in this coordination process will be working locally at their preferred BIM authoring tool. Therefore the most-used BIM authoring tools by MEP/ Structural/ Architectural discipline professionals were checked for the prototype implementation. According to the results from the survey conducted by National Building Specification (NBS) (2020)[17], Autodesk Revit (2021) was found to be the most used design tool, as shown in Figure 4.10. Also, from the analysis of the BuildingSMART certified product list[18] (provided by openBIMInternational) for the BIM application, it was found that Autodesk Revit (MEP, Architecture, Structure) has certified for export/import of IFC2x3 schema. This was important while using the void model as IFC2x3 as a geometry container for the

[17] https://architecturaltechnology.com/static/3f388415-32f9-408d-85cc2c1adf13d012/TheNBSBIMReport2020.pdf
[18] https://www.buildingsmart.org/compliance/software-certification/certified-software/

provision for voids. Therefore, the Autodesk Revit 2021 was considered for the thesis prototype from the aspects mentioned above.

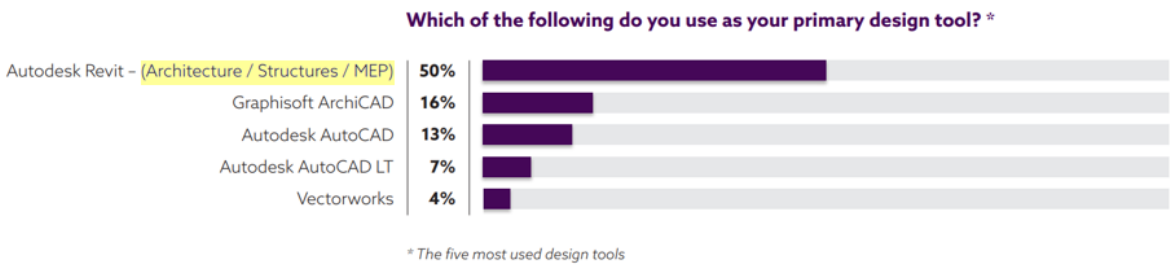


Figure 4.10: The five most used design tools (National Building Specification [NBS], 2020)

4.2.1.2 Collaborative cloud database

A database-oriented concept has been proposed to achieve the single source of truth concept. Database Management systems (DBMS) are essential for performance, accessibility and usability and are inevitable for almost all considerable software applications. Conventional relational databases have been the primary storage structure in most data management and retrieval applications. A relational database is a collection of data items organized as a set of tables with columns and rows with pre-defined relationships between them. The tables are used to store information about the objects that will be represented in the database. Each table represents one entity type, with columns representing attributes and rows representing instances of that kind (AWS, 2022). The connections among tables are defined using unique primary keys of the relevant tables. Interactions with relational databases require the use of query languages, and the most popular query language is structural query languages (SQL). One of the main limitations of using relational databases in applications is that if the data contains many relationships, it requires numerous joins of the large tables. Another significant drawback is the inflexibility of such data systems, which must be set in advance and with considerable effort. Any later change requests require the redesign of the data schema (AWS, 2022). It becomes inefficient to store, query, and manipulate such large interconnected data using relational databases, which paved the way to new alternatives in the form of non-relational databases. A non-relational database does not use a tabular schema as in relational databases for storing data; instead, the data is being stored in the form of simple key-value pairs, JSON documents, or graphs with edges and vertices. The term NoSQL (Not only SQL) is also used in general to denote non-relational databases (Microsoft Azure, 2022).

Typically, people sketch example data on a whiteboard and connect it to other data to show how things are connected when designing a data model. Afterwards, the whiteboard model is restructured and re-formatted to fit normalized tables for a relational model. In graph data modelling, a similar process is used. Instead of modifying the data model to fit a table structure, the graph data model stays the same as it was drawn on the whiteboard. This is one of the nice aspects of graphs- it is "*white-board friendly*" (Robinson et al., 2015). Therefore the objective of the slots and openings approval process sketched in Figure 4.4 could be better and intuitively represented in the graph model. Along with this fact, the graph database opted for the prototype implementation.

Property graphs are intuitive and easy to understand while providing the means to model most use cases. In addition, this would enable to store the information as nodes and relationships for better data structure and easy querying. In the thesis, Neo4j was employed for prototyping as it is the most popular solution on the market among Graph DBMS with native graph storage and processing (DB-Engines, 2022). Graph models are highly effective for representing and describing complex relationships such as between building elements and data in BIM models (Isaac et al., 2013)

Graph database

Graph databases are databases that query and store information in the form of nodes and edges using graph models. Graphs are made up of vertices and edges, which are represented by nodes and relationships respectively in graph DBMS. The nodes represent entities in the domain of interest, interconnected via various relationships. The labelled property graph, which is the most popular form of graph model, has the following characteristics such as (Robinson et al., 2015):

- It consists of nodes and relationships
- Nodes contain properties (key-value pairs)
- Nodes can be labelled with one or more labels
- Relationships are named and directed and always have a start and end node
- Relationships can also contain properties

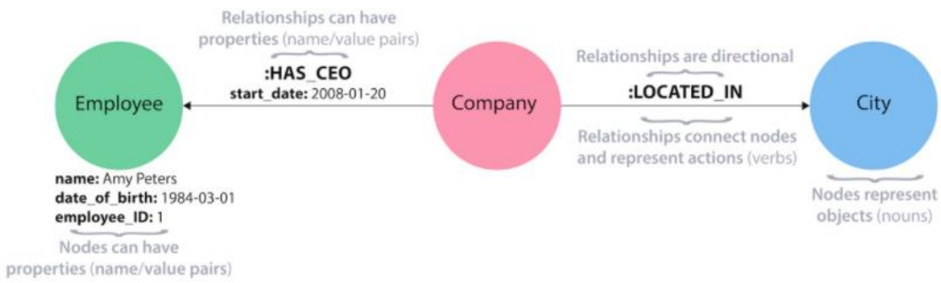


Figure 4.11: An example of a property graph model (Neo4j, 2022)

The inflexibility and difficulty in the conception of data schema in relational databases can be solved to an extent using graph databases. They are flexible in nature, and new information can be added easily compared to relational databases. Furthermore, the ability of graph databases to quickly traverse between data objects distinguishes them from other databases, such as SQL. The experiment of Partner and Vukotic further verified it, which tried to locate friends-of-friends in a social network containing one million people to a maximum depth of five. The results from the experiment imply that graph databases perform better queries than relational databases for interconnected data, as shown in Table 4.1.

Table 4.1: Benchmark results of finding extended friends in a relational database vs Graph DB (Vukotic, 2015)

Depth	Execution time (in seconds)	
	Relational DB (MySQL)	Graph DB (Neo4j)
2	0.016	0.010
3	30.267	0.168
4	1543.505	1.359
5	> 1 hour	2.132

Neo4j database

Neo4j is used for the data storage in the prototype in order to store the void information centrally as a single source of truth. It is a NoSQL graph database management system released in 2010 and uses graphs to store and process data. It is implemented in Java and uses Cypher query language for interacting with the graph data. Neo4j offers many advantages such as quick and efficient transactions and processing for data relationships, flexible data schema- which allows the data to be added and updated easily, and better performance in querying, especially for interconnected data. In addition, Neo4j supports the labelled property graph model, which is the basis for the consideration in the thesis.

Neo4j consists of four basic elements: nodes, properties, relationships, and labels. Nodes generally represent entities in a system such as physical and virtual objects, whereas relationships describe a connection between a source node and a target node. Data can be stored directly through nodes and relationships and the properties attached to them. These are key-value pairs where the key is a string, and the value is either a number, string, Boolean, geographic type, or temporal type. Following the installation of Neo4j either on a local device, server or using the existing cloud platform Neo4j Aura DB, HTTP calls could be employed to access and alter the underlying database. Driver libraries simplify such queries, which Neo4j provides for popular programming languages such as C#, Java, JavaScript, and Python. These libraries provide a collection of functions for simple authentication and query execution. Furthermore, Neo4j includes a built-in browser that may be accessed through a different port. This browser provides access to documentation, tutorials, sample graphs, an overview of the underlying database and the ability to run queries in Cypher, as shown in Figure 4.12.

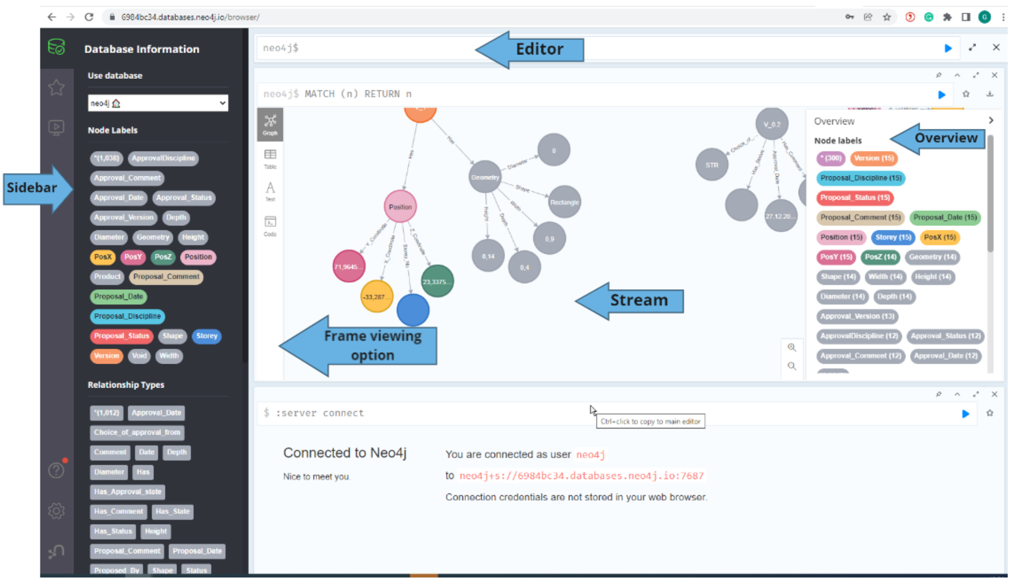


Figure 4.12: Neo4j browser interface with example query and result visualization

The Neo4j Aura DB is used for the thesis, which is a fully managed cloud graph database service provided by Neo4j. It is a fully automated graph database service that allows building graph applications without worrying about database administration. The Neo4jDotNetDriver is used to query and manage the information in the Neo4j database from the .NET application. Cypher was used as the query language, which is a declarative graph query language developed by Neo4j for expressive and efficient querying as well as updating the graph. Cypher was not only designed to be easily read and understood by developers, but also domain professionals and business stakeholders as well. As a result, the patterns are written using ASCII-art syntax, which is easily readable by most people. It allows users to specify what they want to select, insert, update, or delete from the graph data without knowing the specific steps (Robinson et al., 2015).

Modelling in Graphs

In Cypher, queries are made up of different clauses. Clauses can be linked together, and intermediate results can be passed to each other. Five basic statements commonly used in the query sentence are MATCH, WHERE, SET, CREATE, and RETURN. In addition, aggregation functions are used to calculate aggregated data; for example, “COLLECT” is used to return a list and “MIN” returns the minimum value (Robinson et al., 2015).

- ‘MATCH’ identifies certain connection patterns in the network and is used to retrieve data from the graph by describing the pattern or properties.
- ‘WHERE’ constrains the properties of nodes and edges and behaves as a filter.
- ‘SET’ consists of changing existing or adding new properties,
- ‘CREATE’ creates new nodes or relations.
- ‘RETURN’ defines the information that a user is interested in or return the queried result.

A sample query using some of the basic clauses is provided in Code 1. The corresponding graph pattern where the nodes are connected using the ‘*Relationtype’* relationship is illustrated in Figure 4.13. The query can be done as follows: First, MATCH two nodes with given labels that are connected by a single relationship. Then, filter the retrieved subset using the WHERE clause. Finally, it returns the queried result using the RETURN clause. The RETURN clause lists or identifies the node (*Node 2*) which is related to *Node 1* by *Relationship1* and has ‘*property1’* as ‘*value’*.

Code 1: A sample cypher query

```
MATCH (node1 : Label1)-[relationship1 : RelationType]->(node2 : Label2)
WHERE node1.property1 = {value}
RETURN node2
```

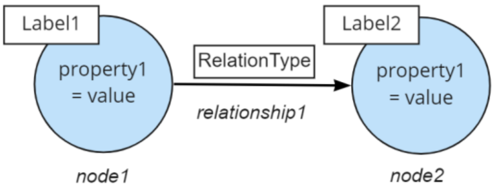


Figure 4.13: Corresponding graph expressed using a diagram

Temporal versioning in graphs

The versioning using graphs can be intuitively done using the nodes, relationships and labels. An example scenario and its corresponding graph model are given below, where objects are separated into different state nodes and are linked with relationships. Additionally, changes in states are captured within the relationship property linking the two nodes. This way, it is possible to track the changes that happened to the state nodes in a given period of time. Also, the associated nodes to the state nodes can be queried (Lazarevic, 2019). An example of a time-based versioning scenario and corresponding graph model is given below:

'*a company has a product with the name 'Widget'. On the 4th of May 2016, a couple of decisions were made: The company name was changed from 'Widget' to 'MiniWidget' and reduced the price of this product down to 3.99 from 4.29 (Lazarevic, 2019).*'

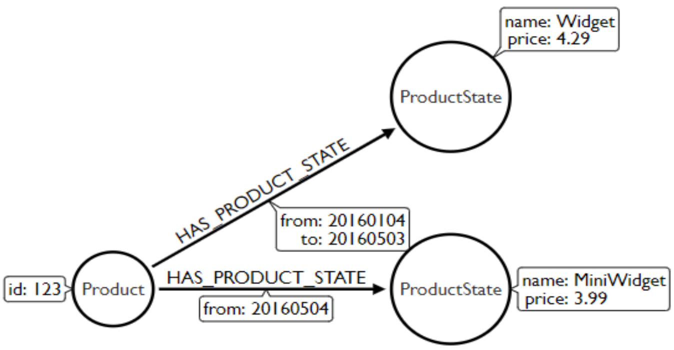


Figure 4.14: The graph representing the track of changes with time-based versioning (Lazarevic, 2019)

4.2.2 Prototype development

As illustrated in Figure 4.9, a plugin as a gatekeeper tool between Revit and cloud database was implemented in .NET framework using C# programming language. This was further integrated and interfaced within the Autodesk Revit application by employing the Revit APIs. The architecture of the developed prototype consists mainly of two parts: the developed plugin (gatekeeper tool) on the BIM authoring tool Autodesk Revit, cloud database- Neo4j Aura DB. The basic architecture used to prototype the gatekeeper tool is depicted in Figure 4.15. The user (specialist planners) interacts with the prototype using the implemented user interface (UI) for coordinating the slot and openings. The prototype interacts with the Revit database for accessing the data concerned with the voids using Revit APIs. The connection to the cloud database (Neo4j Aura DB) was established over the internet using various APIs provided by Neo4j. The APIs provided by Neo4j helps to develop applications that can create, read, update, and delete information from the graph database.

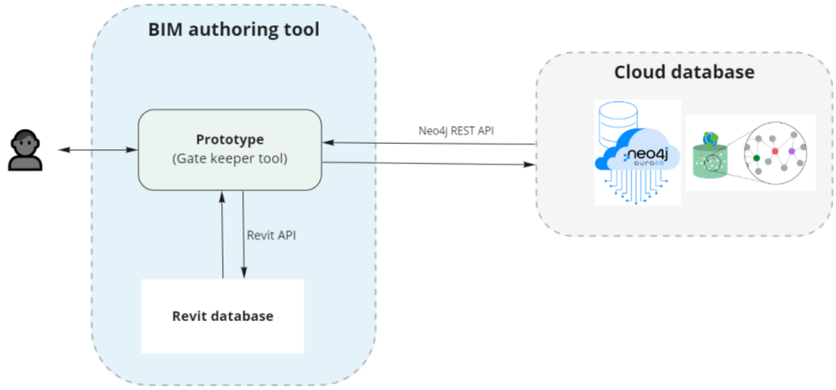


Figure 4.15: Basic prototype architecture

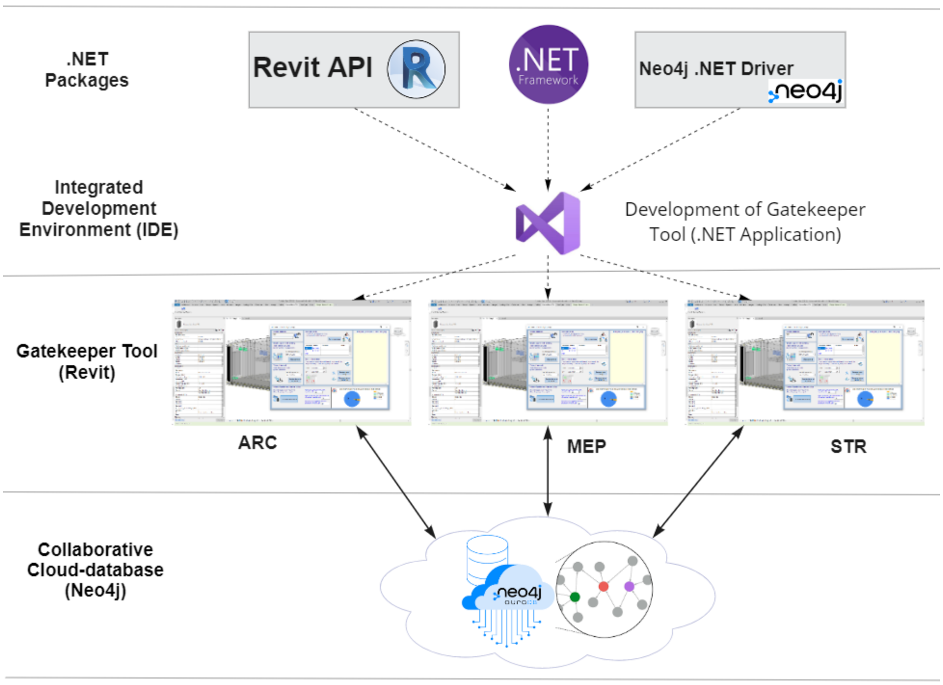


Figure 4.16: Software ecosystem of the developed gatekeeper tool

4.2.2.1 Model requirements

As mentioned in section 4.1.1, all review processes will be carried out in the local BIM authoring tool. For the thesis prototype, it was agreed upon to use Revit 2021. Some of the pre-requirements for the prototype implementation are given below.

Predefined approval parameters

The necessary parameters (commonly defined) related to void objects need to be defined from the beginning of the implementation. This is usually defined in the MEP software as the origination of the void objects. All void objects were planned to have three approval parameters for each involved discipline in the void model: approval status, approval date, approval comment.

The planners from MEP/ARC/STR disciplines were considered in the parameter definitions. These parameters were created in Revit as shared parameters, as shown in Figure 4.17. If the project parameters are different from the database parameters (by name), a mapping process is necessary to connect database parameters to locally user-defined parameters.

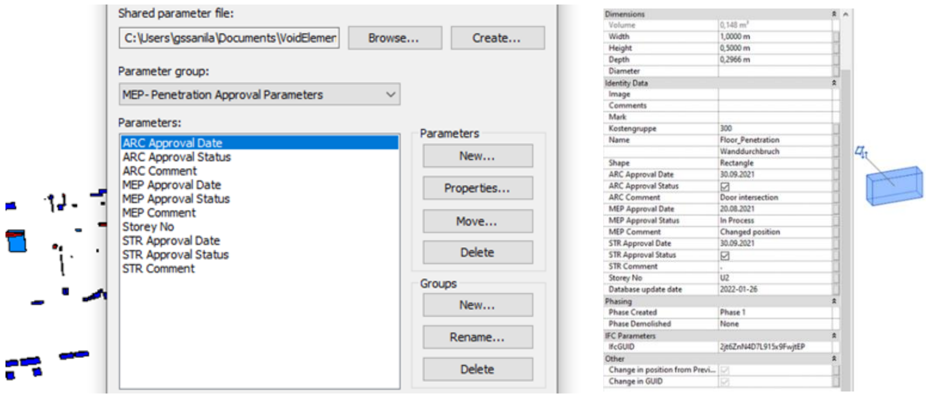


Figure 4.17: Approval parameters defined in Revit as shared parameters (left) and a void object with shared approval parameters (right)

History indicator family and parameters

In order to visualize the history of changes in the authoring tool, parametric (in dimensions and version number) void objects corresponding to previous proposals have to be recreated. This includes voids corresponding to cuboidal and cylindrical voids. Therefore, Revit families were created first, corresponding to voids hosted in floor/wall elements. In addition, the material parameters (texture) were adjusted to make better visualization, as shown in Figure 4.18. For the testing (validation of concept) purpose, the focus was given to cuboid void objects.

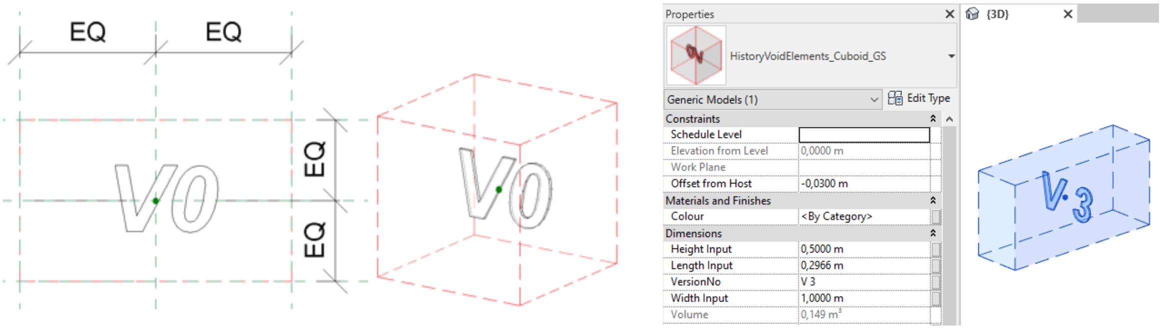


Figure 4.18: Parametric cuboid family created in Revit for history indicators (left) and corresponding instance (right)

4.2.2.2 Modelling the graph –Temporal versioning in Neo4j

The graph was modelled primarily using the concept of temporal versioning, as illustrated in Figure 4.4. This whiteboard model is then re-formatted to the graph model. Many different situations in the design reality of slot and opening planning were also taken into account. Eventually, the graph was made in the way in which designers explain the slot and coordination approval processes. This method simplifies the queries and makes them easier to read and understand. Additionally, the concept of Lazarevic (2019) was also considered. Moreover, the following rule of thumb was presumed:

'*Model the graph how one can explain the slot and opening approval process in a talk. Where nouns become nodes, and verbs become relationships.'*

The main question addressed during the development are given below:

- How can the three versions (MEP/ARC/STR) of the same void be created? All three elements with the same GUID with different parameters such as location (x,y,z), approval status (accepted/ rejected) and comments.
- How to show three realities (MEP/ARC/STR) of the same opening proposal without losing order (GUID)?

Approach: When the MEP planner proposes a new/modified void, a new iteration version will be created with a specific position (coordinates and storey number) and geometry (shape, dimensions) parameters. In addition, each proposed void will have a status (e.g., 'in progress'), a related comment, and the proposal date. When a specialist planner checks and provides an approval choice for the requested void, a new approval version will be created with the approval status, comment, and date of approval parameters.

While modelling the graph as part of the implementation, emphasis was given to prove the proposed concept. As a result, optimizing the number of nodes was not taken into account. Moreover, modelling data in nodes enable connecting them with numerous relationships. This is crucial for the planners to deduce or infer the information needed in future. After creating the graph data model, corresponding queries were made directly in the Neo4jAura browser and tested with sample scenarios. After that, this was included in the plugin. An example void approval scenario is given below, and the respective graph data model is given in Figure 4.19.

Sample approval scenario: "*MEP starts the process with requests of a void for approval. The architect then rejected the void and requested to move up. In contrast, the structural engineer accepted the void. The MEP immediately received approval information, changed the position, and proposed the void for further review and approval (proposal version 1).*"

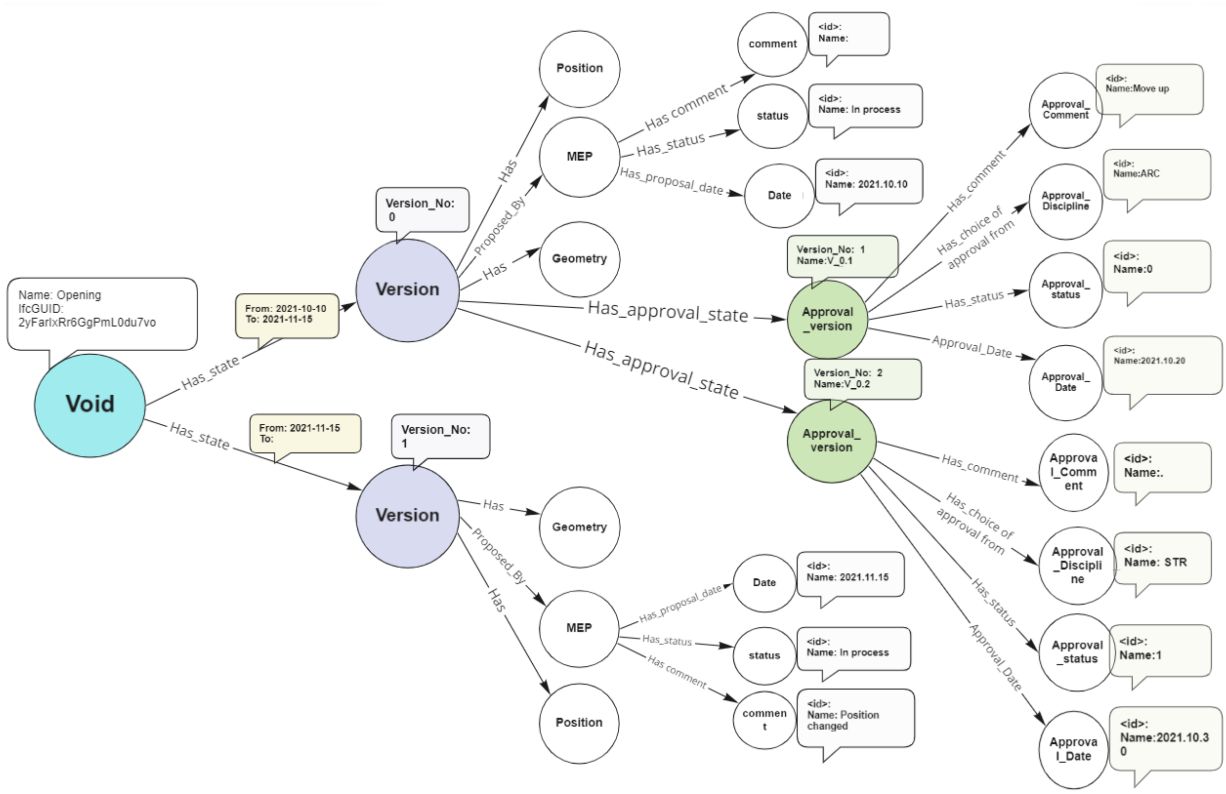


Figure 4.19: Labelled property graph model for slot and opening coordination process (nodes and labels associated with position and geometry are not shown in this figure)

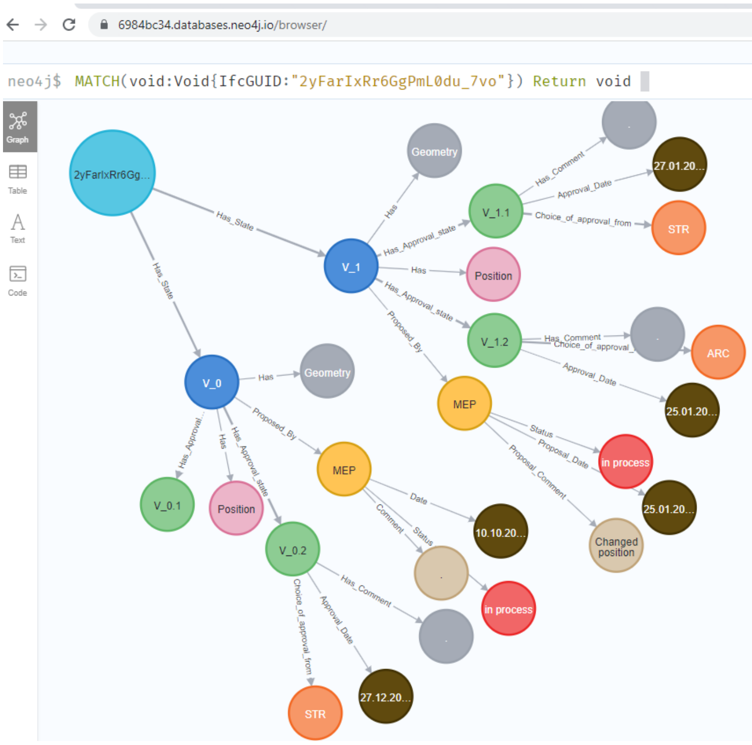


Figure 4.20: Labelled property graph model developed for slot and opening coordination approval process based on Figure 4.19

4.2.2.3 Functionalities

The necessary functions were incorporated into the prototype, aiding the objective requirements in the proposed concept for the slot and opening coordination process. The functionalities included in the prototype are described below. For the reader's preference, the cypher queries used to model the graph for a few features are provided in Annex B.

Propose void objects:

This functionality enables to propose/ update a void object data into the database for further checking and approval during the design iterations. This is usually done by the MEP-planner. The predefined parameters of the selected void will be added to the database, which includes:

- Dimensions, shape, name, type, centroid coordinates
- Void status, comment, date

The centroid coordinates were also calculated as it was not possible to read directly from the void parameters. Each void proposal will create a new iteration version (the dark blue node) and link to the void node with date properties.

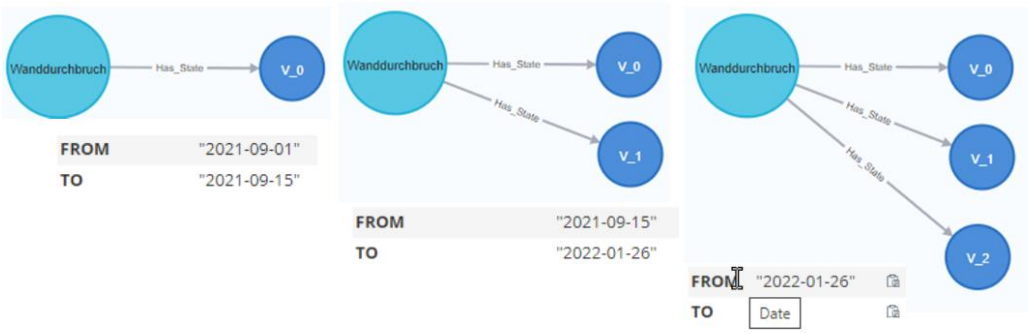


Figure 4.21: Temporal versioning of void states in each iteration (proposed by MEP planner)

Provide choice of approval:

This functionality is used to provide the choice of approval to the database from the specialist planner side. It was planned to have separate approval states (approval version) for involved discipline. Whenever a planner provides the choice approval to a void object, a new approval state will be added with the date property (green node). Furthermore, all the approval information will be integrated into this approval state node (brown node). This includes approval choice (Yes/No), approval comment, approval date, respective planner discipline name (e.g. STR (structural)). The related cypher queries are included in Annex B.

Figure 4.22: Temporal versioning of approval information related to the void object in each iteration states

Get the current status of the selected void:

The planners can utilize this functionality to know the current approval status of the selected void in the most up-to-date time. First, the IfcGUID of the selected void is accessed from the Revit database and is used to match the concerned void node in the graph database. Then, the latest void state is queried from the database and listed in the prototype UI. This includes information such as:

- approval status from the involved discipline (approved/rejected or Yes/No)
- approval comment from the involved discipline
- date of approval

Read the history of the void object:

This functionality allows the users to get the history of changes related to a selected void object in a provided period of time in the project. History includes all the previous void states (in the design iterations). The history of information can be read as text in the prototype UI, which includes:

- Previous position, dimension and shape parameters
- Previous approval information (status, comment, date) from the involved discipline

This is achieved by reading the data saved in the cloud database through various queries. Initially, the IfcGUID of the selected void is accessed from the Revit database and is used to match the concerned void node in the graph. Then all information related to this selected void in the provided time interval is queried from the graph. Afterwards, the results are made available as quickly readable text in the History-log of the prototype UI.

Visualize the history of the void object:

This functionality enables the planners to visualize the history of the selected void in the BIM model (in Revit UI). The previous states of the void will be recreated as temporary void instances with the previous dimensions at former locations. The Revit history indicator families (e.g. see Figure 4.18) will be used for this function. The dimensions and position data will be read from the cloud database through various queries. A mapping between feet to the meter was necessary to create the void instance model using Revit API; Revit uses imperial units internally-feet for length (The Building Coder, 2011). Similar to previous functionality, the IfcGUID of the selected void is accessed from the Revit database and is used to match the concerned void node in the graph. After that, the instances will be created with an iteration number as a multiline text inside the history indicator family instance for quick review.

Get an overview of the coordination process:

This functionality enables the involved discipline specialists to have an overview of coordination progress, especially the approval process. It was achieved by creating a simple dashboard in the GUI. For the prototype aspect, the focus was given to show the further possibilities of having a single source of information in the slot and opening coordination process. Furthermore, the progress can be reviewed by inferring the labelled property graph in the cloud database. The implemented prototype provides the following inferences at the most up-to-date time:

- Overview of the existing status of the void provisions:
  - Total number of already coordinated voids (or voids that are approved by all involved disciplines): Void status "Coordinated".
  - Total number of voids in-process (or voids that are not communicated/updated with choice of approval by involved disciplines): Void status "In process".
  - Total number of newly proposed voids by MEP-planner (after the first iteration): Void status "New".
  - Total number of deleted voids (the voids which are not needed anymore): Void status "Deleted".
  - Total number of voids with unclear status, if the status is not updated to the database with one of the predefined terms): Void status "undefined”.
- Overview of approval progress
  - Total active voids
  - The total voids with approval pending from the Architectural discipline
  - The total voids with approval pending from the Structural discipline

The system architecture of the implemented prototype with functionalities mentioned above is illustrated in Figure 4.23. The connection to the cloud database can be established at the beginning of the coordination process by providing the necessary credentials in the user interface. Some relevant cypher queries used to achieve the functionalities and results in the graph are provided in Annex B. The user interface (UI) of the developed plugin with the functionalities mentioned above is shown in Figure 4.24.

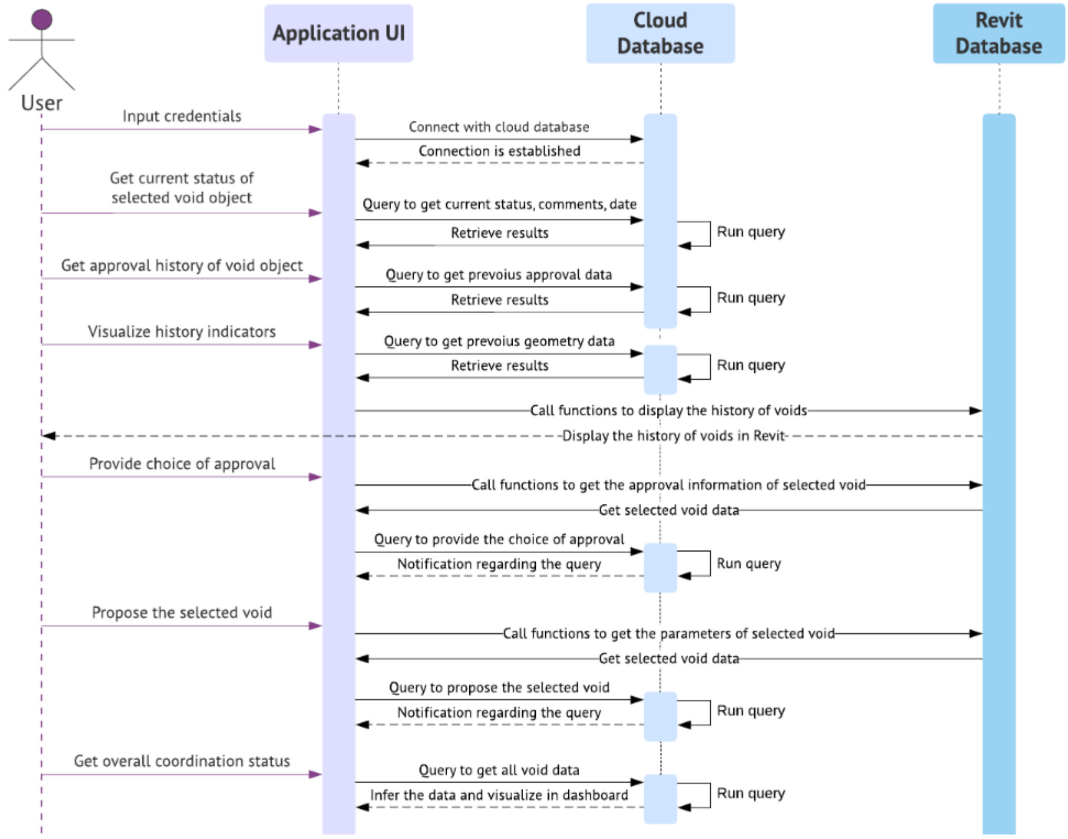


Figure 4.23: System architecture of the implemented prototype

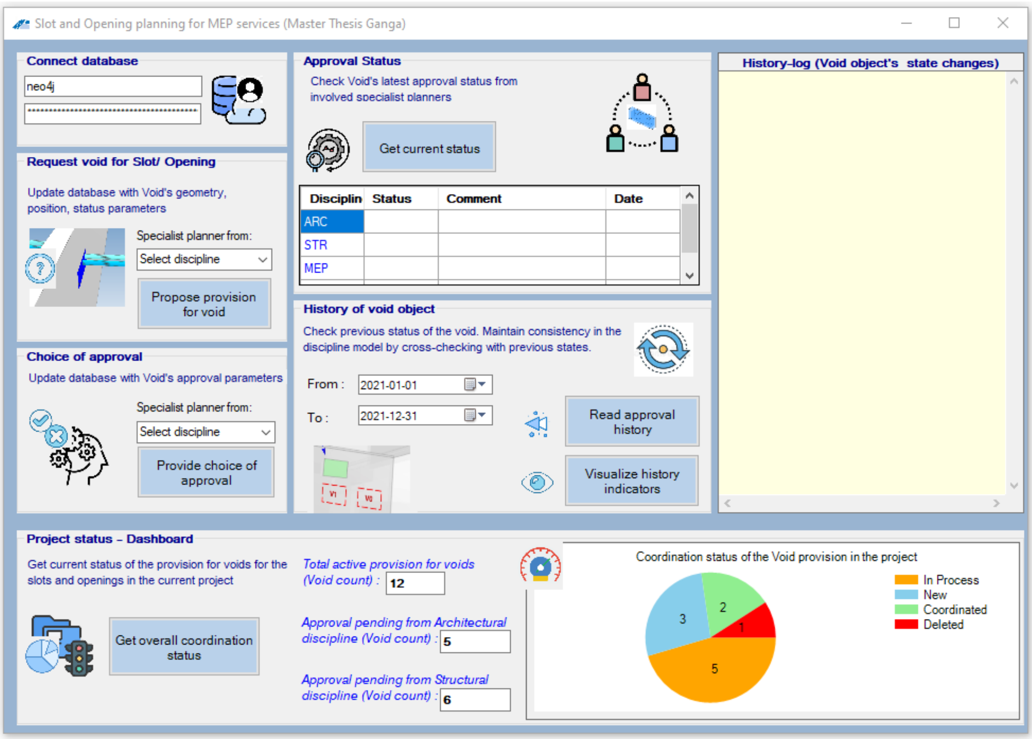


Figure 4.24: The user interface (UI) of the developed prototype plugin (as a gatekeeper tool)

5 Evaluation

This chapter discusses how the concept proposed in Chapter 4 was validated using the implemented prototype. The concept was tested with a reference project. In addition, the concept and implementation were reviewed with industry professionals through personal interviews.

5.1 Testing

5.1.1 Reference projects

In order to test the proposed concept for the slots and openings coordination process, an existing complex openBIM project from B+G Ingenieure Bollinger und Grohmann GmbH[19] was used. The slots and opening coordination task has been done by using a model-based approach in this ongoing project. In this project, for the approval process, a parameter-based method was adopted. The MEP planner distributed the void model with the required void parameters. These parameters were commonly defined and provided by the MEP planner. PfV models (or void models) were distributed floor-by-floor and approved floor-by-floor as separate IFC2x3 files (with ProvisionForVoid objects). The files were transmitted by the MEP designers via e-mail. The architect was responsible for leading the slot and coordination task. The predefined void parameters considered architectural, structural, fire protection aspects. It was agreed upon that the void objects need to have the same parameter for IfcGUID in every BIM authoring application. The coordination-relevant parameters related to void objects included the following information such as:

- Void dimensions and shape
- Void type (openings/core drills, wall openings/floor openings etc.)
- Void status (new, coordinated, in process etc.)
- Void ID (IfcGUID)
- Approval parameters for each specialist planner (status, comment, date)

[19] https://www.bollinger-grohmann.com/de.home.html

Furthermore, the reason for a void's rejection was planned to be communicated through comment parameters. Also, it was planned to communicate the late changes and complex issues in a BCF-based approach.

5.1.2 Test set-up

From the reference project, Void models delivered for two floors from the MEP planner in three approval iterations were taken for testing (see Figure 5.1) concept. The predefined approval parameters' names in the reference project were different from the prototype parameters. Therefore, a parameter mapping was done between project-specific approval parameters and predefined database parameters, as shown in Figure 5.2. The Architectural and structural IFC models were referred for the approval process to compare the discipline-specific decision criteria that are relevant for the approval process.

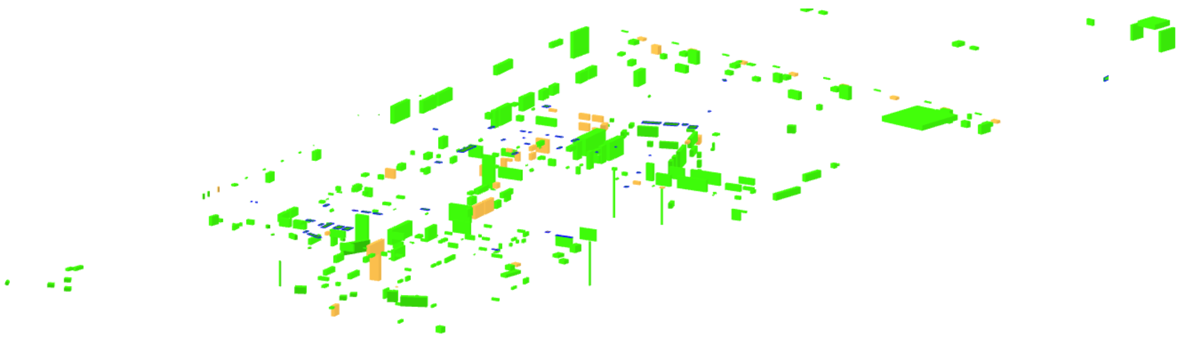


Figure 5.1: Void (PfV) model (Bollinger und Grohmann GmbH, 2022)

Mapping

Test model (Project specific) parameters		Prototype parameters		
Parameter name	Parameter type	Parameter name	Parameter type	
Name	string	Name	string	Necessary
Shape	string	Shape	string	
Height	double	Height	double	
Depth	double	Depth	double	
Width	double	Width	double	
Diameter	double	Diameter	double	
Kommentar_TGA	string	MEP Approval Status	string	
Kommentar_TGA	string	MEP Approval Date	string	
Kommentar_TGA	string	MEP Comment	string	
Freigabe_AR	string	ARC Approval Status	Yes/No	
Kommentar_AR	string	ARC Comment	string	
Datum_Freigabe_AR	string	ARC Approval Date	string	
Freigabe_TWP	string	STR Approval Status	Yes/No	
Kommentar_TWP	string	STR Comment	string	
Datum_Freigabe_TWP	string	STR Approval Date	string	
IfcGUID	string	IfcGUID	string	
Mark	string	Type	string	Optional
Storey No	string	Storey No	string	
Database update date	string	Database update date	string	

Status, date, comment were added to the parameter "Kommentar FCG" as single text : (Kommentar_TGA rows)

Figure 5.2: Mapping of project-specific parameters to prototype parameters

5.1.3 Functionality testing

The implemented prototype (see section 4.2) was used for testing the workflow with the above-described test model. The attention was given to the scope of the slot and opening process in the design reality, which usually involves the disciplines architecture, structure, and MEP. Therefore, the author had to go through the tasks of MEP, architectural/ structural specialist planners. The focus was given to the information flow between involved disciplines in a sender-receiver approach. Based on the proposed concept, test-relevant coordination steps were defined from the perspective of the MEP planner and specialist planners. This is illustrated in Figure 5.3 and Figure 5.4.

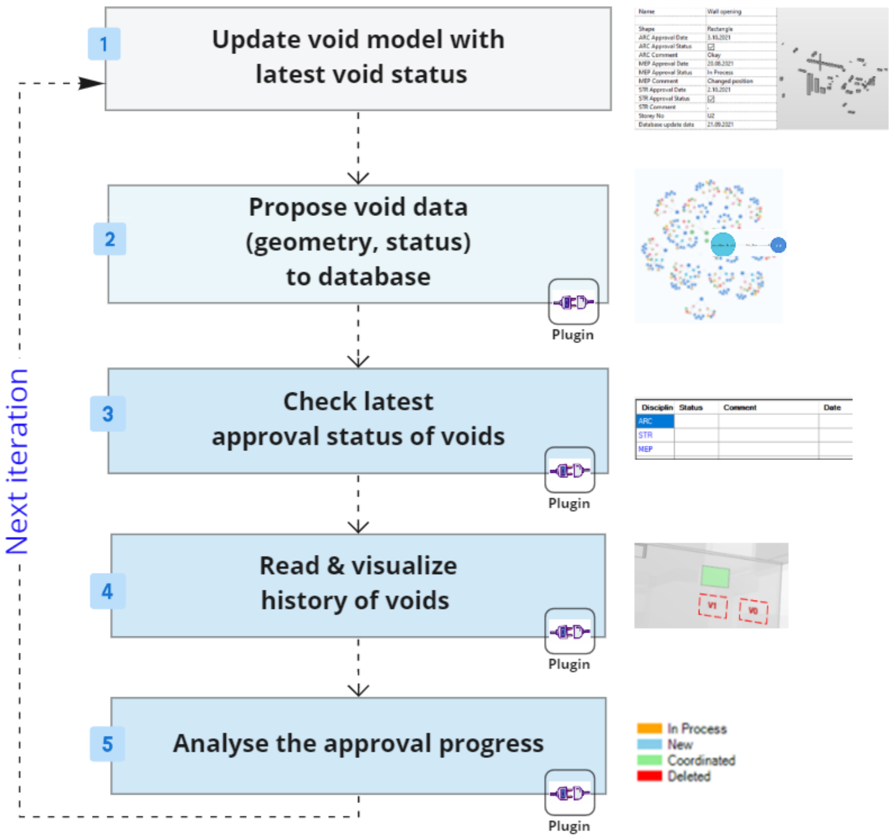


Figure 5.3: Steps identified to test the proposed concept from the perspective of MEP planner

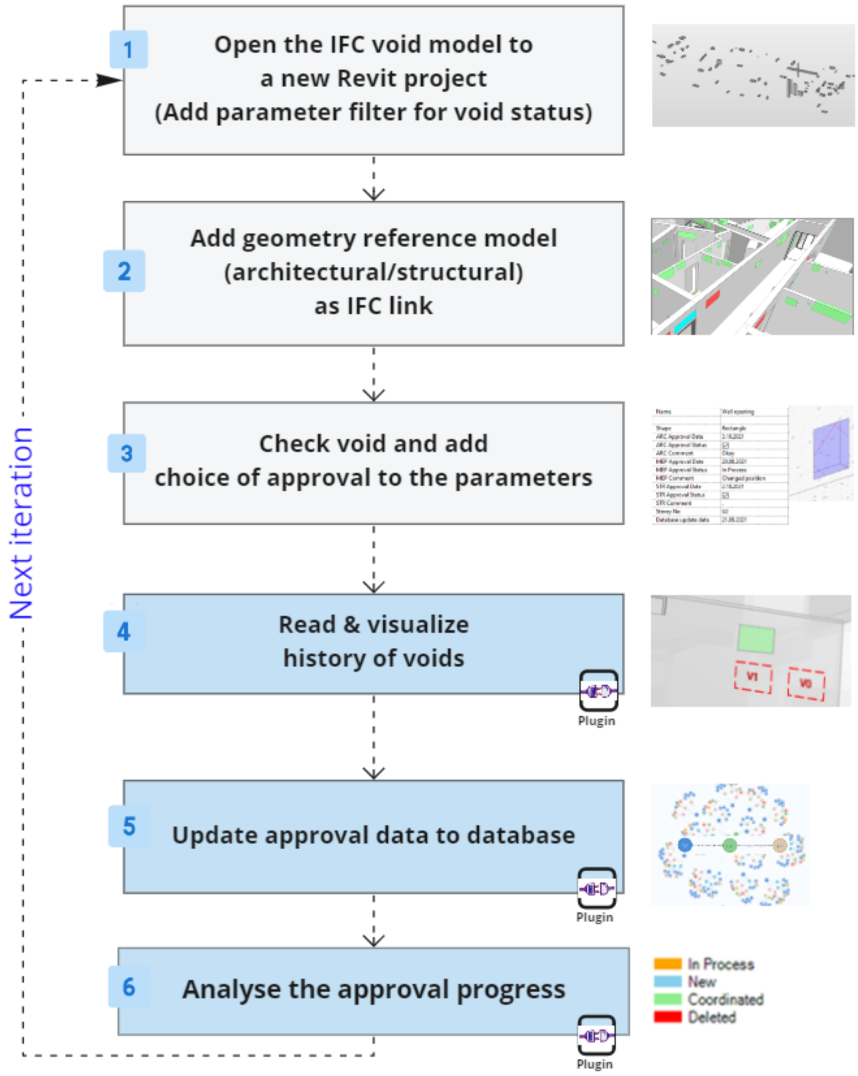


Figure 5.4: Steps identified to test the proposed concept from the perspective of specialist planners (ARC, STR etc.)

After analyzing these two conditions, distinct steps were selected and grouped into two categories, as given in Table 5.1. Before starting the test, the cloud database authentication credentials (username and password) were entered into the prototype UI, where it was set to default throughout the testing.

Table 5.1: Testing of the developed concept

Selected test case	Category
• Propose void data to the database • Update approval data to the database	Send information in real-time
• Get the latest approval status of selected void in the plugin UI • Get approval history of selected void in the plugin UI • Get position and geometry history with history indicators • Get inference about overall coordination progress status in the plugin UI	Receive information in real-time.

- Test: Propose provision for void (discipline: MEP)

The void model was opened in a new Revit project template for this test. Then a few voids were selected, and necessary parameters were checked. Additionally, missing shared project parameters were added (MEP approval date, MEP approval status). Also, the MEP status parameter was changed according to the void approval condition (in process, new, deleted, coordinated). Then, the proposal discipline name was selected from the prototype list box to ‘MEP’. Afterwards, selected void’s data (geometry, position, status, date) were updated to the database using the prototype's functionality ‘Propose void’. Furthermore, each successful update was notified in a message box with the current iteration version.

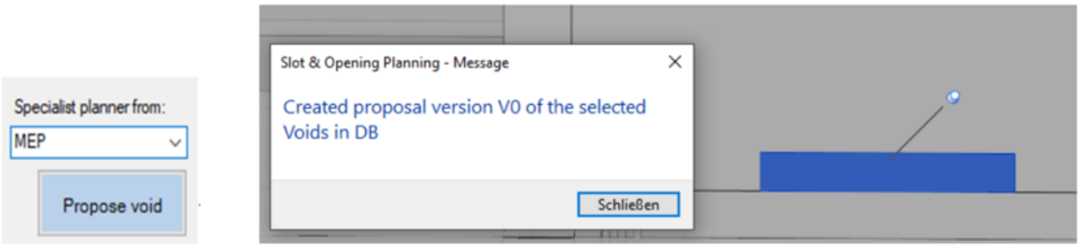


Figure 5.5: Updating of void information of the selected void to the database

- Test: Check and provide the choice of approval (discipline: ARC/ STR)

The geometrical reference model (Architectural IFC file) was linked to the opened void model with proper alignment. Also, the shared coordination bodies from the opened and linked models were double-checked for correct placement. View filters (parameter-based) were added for efficient handling of the 3D model. In order to prevent position change, the reference model and voids were pinned.

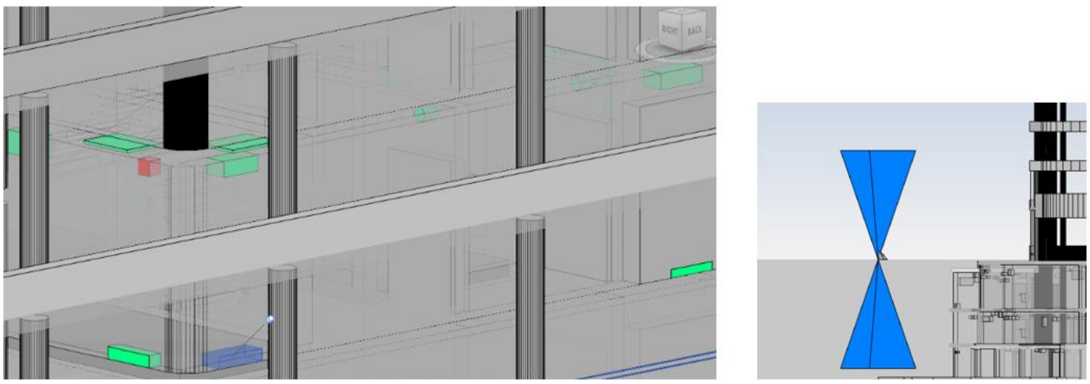


Figure 5.6: A view of void model opened in Revit (with colour filters) and linked reference (structural) model (left); Aligned coordination bodies with shared project origin (0, 0, 0) (right)

Then, through a void-by-void approach, the checking and approval of provision for voids according to the discipline-specific design criteria were tested- concerning the reference model. The approval data in the model was kept the same, but comments were translated to English. Then void approval information was updated to the database using the prototype's functionality ‘Provide choice of approval’. Furthermore, each successful update was notified in a message box with the current proposal version number.

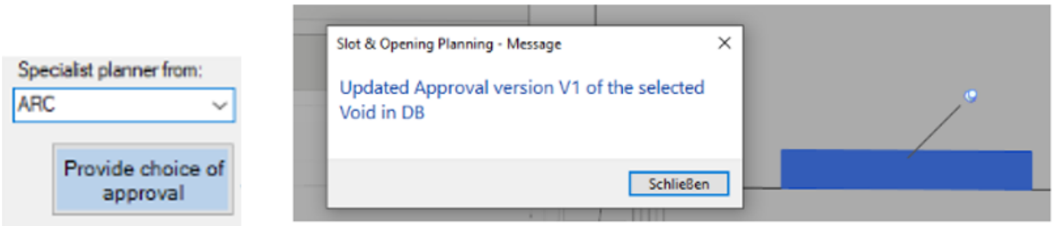


Figure 5.7: Updating approval information to the database

- Test: Get the latest approval status of each void (discipline: MEP)

The latest approval status of each selected void was checked using the prototype functionality ‘Get current status’. In addition, a schedule was created for better reviewing the approval information of the model. The void object was selected from the schedule/ the model directly. The approval data queried from the database was also cross-checked with approval parameters in the model.

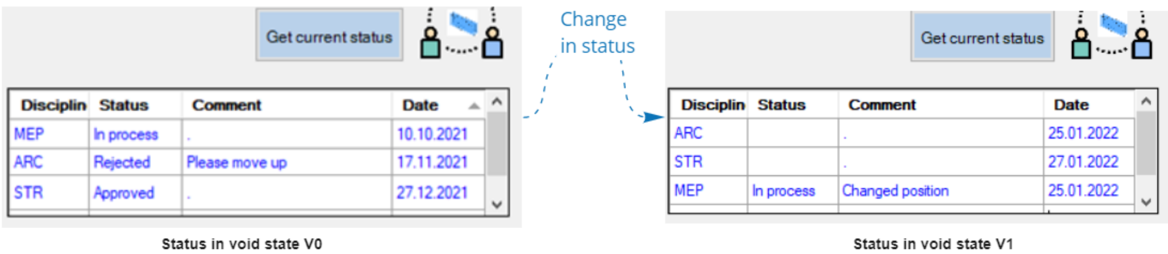


Figure 5.8: Reading the latest status of the selected void object in prototype UI at different times

The above tasks were repeated for 2-4 iterations. Furthermore, the status, dimensions, and positions were modified by checking the reference project’s different void models. After the first iteration, a few voids were added with the status: ‘new’. If ARC and STR approved the void, the status was changed to ‘coordinated’. If one of the parties rejected the void, the comment was checked for issue resolution. These voids were modified and set the void status ‘in process’.

- Test: Get approval history of each void (discipline: MEP, ARC, STR)

The history of approval was then checked and tested using the functionality ‘Read approval history’. The required time period was chosen from the UI of the prototype. Results about the change in approval were displayed in the History-log of the prototype UI. This was also verified with the previous approval parameters provided in the PfV model. In the prototype, the database update date was set to be taken automatically to prevent data manipulation (that is why From and To dates will be the actual update date to the database).

Figure 5.9: Approval data from the given time period tracked using the prototype

- Test case: Get position and geometry history with history indicators (discipline: MEP, ARC, STR)

The change in geometry and position during the selected period of time was tracked using the functionality ‘visualize history indicators’. The results were displayed as text, and history indicators were created in the Revit model. The centroid position and alignment of the created history indicators were verified by using the free Add-in ‘Revit Lookup’[20].

[20] https://github.com/jeremytammik/RevitLookup/

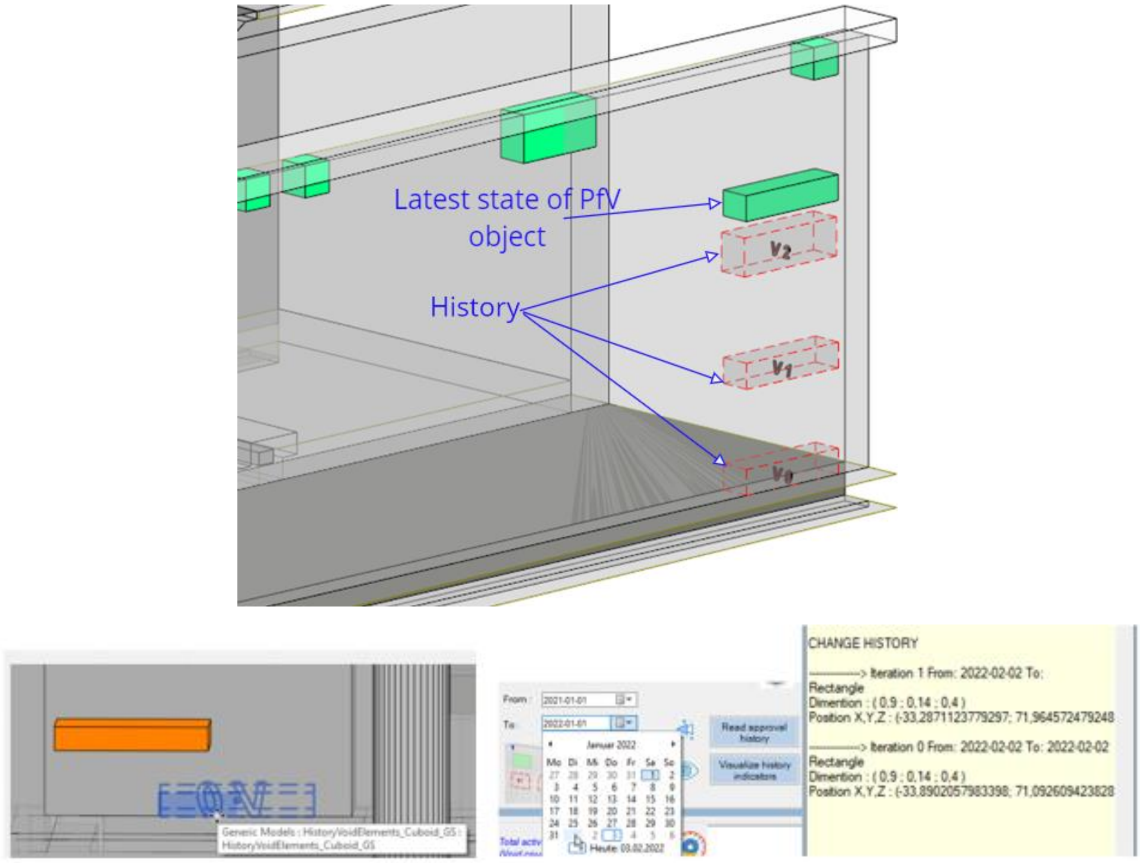


Figure 5.10: Visualizing history with history indicators and corresponding position and geometry parameters.

- Test case: Get inference about overall coordination progress status(discipline: MEP, ARC, STR)

The approval process progress was tested using the prototype functionality 'Get overall coordination status'. After finishing each void approval, this test was done to verify the results. Some mistakes were intentionally added in the testing to check the results. PfV without IfcGUID were not taken into account in these test results. Moreover, the database was checked in parallel by querying required void data using the IfcGUID.

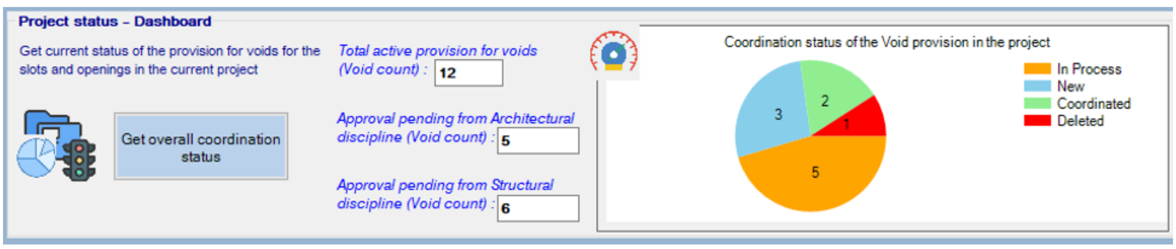


Figure 5.11: Overall coordination status viewed in the UI of the prototype

5.1.4 End-to-end testing

The developed prototype was tested with an experienced MEP- planner as a key player in the slot and opening coordination workflow. The concept was explained in 20 to 25 minutes with prototype functions. The person was quite interested to know the database-oriented approach and was curious about the new concept. The focus was given to test the feasibility of the approach in real conditions and by adding the feedback of the designer to discuss possible scenarios. The test was done for a few voids with sample conditions, which were updated to the database first and then provided random approval parameters. In parallel, the graph database was checked to see the void states. The latest approval status was then checked using the prototype. The position change in voids and corresponding history tracking was also checked for a few voids. This was done by changing the position of existing voids in the same model. Then using the history tracking functionality of the prototype, history indicators were created at the previous positions of the void.

After the testing, the dependency of IfcGUID as an identifier to track the history in this process was discussed. In his opinion, in general, it is possible to keep the IfcGUID of voids throughout the process. But, there can be exceptions like accidentally deleting a provision for void (PfV object) element. If an additional parameter for change in GUID is also included in the process, it would not be an issue as the MEP planner can tick this boolean operator to indicate the loss of the GUID reference. So, based on the previous parameters (position, dimension, void name), void could be traced in the graph and can be connected with additional relationships. Overall, the interviewee had a positive impression of the approach and database-oriented concept. Also, he mentioned that such an approach could solve the current pain in their coordination process.

5.2 Personal interviews

A qualitative evaluation of the concept and overall workflow were checked through personal interviews; with specialist planners from Architectural, structural, MEP disciplines. The evaluation was given focus to user experience and acceptance to real project scenarios, not technological aspects. The details of the interviews are provided in Annex A. The main aspects from the interview are provided below:

- Comparing multiple files and checking multiple sources of information costs a lot of effort and money. In such cases, the database, which acts as the single source of information, will save time and effort.
- It is essential to have an overview of the coordination process in complex projects with longer design periods and ‘breaks in projects due to change iterations and cost-down rounds’. Bringing the versioning aspects to the process will be highly beneficial. There are many cases where voids get deleted accidentally or were forgotten to be integrated into the discipline-specific model in the execution design phase. In such cases, a versioned history of information would be helpful to avoid disputes and conflicts among the involved specialist planners.
- The cloud database-oriented workflow with the sender-receiver approach will help to communicate information in real-time or at the most up-to-date time. This is very important in the change management aspect; planning based on the old files or models (delivered in previous data drop; could be weeks ago or months ago) results in late changes. Therefore timely communication is important.
- The functionality to view the progress and history with temporary indicators will help the planners to keep an overview of the Void objects in a better way.
- The history-based approach would be helpful for the coordination process as planners could see their previous decisions; why a void got rejected, or where it was before. They don't have to compare the drawings to track the design changes; they could focus on the design.
- The dashboard concept can help to keep track and visualize the synchronization between all discipline models.
- In order to execute a new concept or a new workflow in a multi-disciplinary process in design reality, it has to be simple and user-friendly. A plugin approach to the already familiar design environment will be very much feasible to execute in the real scenario.
- “Industry should evolve in this way”, said an interviewee. He also mentioned that there is a need for planners to find better ways to reduce repetitive and time-consuming tasks. Planners could save a lot of time by changing processes like slots and opening coordination in projects. The designers and planners have to reflect on other disciplines to make a collaborative process more meaningful.
- The availability of geometry data also connected to the database through a collaborative platform would be ideal.

- History-based aspects should not be used for spying on other parties decisions. Trust should not be misused.
- A possibility to add a relationship between cut-outs in the void model will be beneficial, which is also a time consuming, tedious task.
- There was a different viewpoint in starting time of the slots and openings coordination process. From the perspective of the MEP planner, the late starting of the planning of voids is better; because then they will be already decided on the routing of service systems. On the other hand, from the perspective of the architect and structural engineers, the earliest starting would be ideal as they get the position of voids, and it would help them with precise design and calculation from an earlier phase itself. Also, the late changes could be avoided.

6 Conclusion and Outlook

6.1 Summary

The coordination of slots and openings for MEP services is one of the most inefficient and time-consuming tasks in the AEC industry. The co-dependency between various disciplines in the planning of the slots and openings often leads to multiple iterations of decision making through the approval process, which in turn demands intensive and timely communication between the involved parties. The use of BIM methods and technologies in slots and opening coordination has enhanced the process from a conventional 2D drawing-based method. However, the existing BIM-based approval approaches lie on file-based communication, which limits the design consistency through multiple sources of information. In addition, the tracking of changes in BIM models at the object-level in a file-based communication approach is time-consuming and often prone to errors. Furthermore, the lacking of object-level versioning causes a loss of overview to the approval process, which is essential in fast-paced, complex projects with thousands of provisions for voids.

After analyzing the existing challenges and limitations in the slot and opening planning, the thesis proposed a database-driven open BIM concept for slot and opening planning that enables planners to access the most up-to-date information of voids from their convenient BIM authoring tool. Also, the approval information can be transparently communicated in an object-based fashion via APIs. The geometry and information of voids in the BIM authoring program are linked to the cloud database in a temporal-versioned manner to help designers track the underwent changes of voids. This makes it easier for all involved parties to adapt the latest void’s state to the discipline-specific model. Thus, human error could be reduced to a greater limit. History tracking visually as temporary void objects in the design tool or as text in the history-log assists change management and provides better control of the process. In addition, having a history of decision making enhances the interdisciplinary coordination-relevant knowledge of involved specialist planners. They could learn and infer from their (and others) previous decisions and thus reduce the need for multiple decision-making loops in the future. Furthermore, the plugin-based approach helps the users work in their preferred modelling environment, so additional technical knowledge is not needed to adopt this concept in their daily BIM workflow. In addition, the end-user does not have to acquire knowledge in the technical and procedural details of the BuildingSmart standard process (VDI 2552, part 11.2) for slot and opening coordination.

In order to evaluate the proposed concept, a prototype was implemented. The prototype is primarily a simple proof of concept which can be developed in various aspects. The concept was assessed by running through the slot and opening coordination task in a reference openBIM project. In the test, it was able to obtain the latest approval/ void status/ geometry, position data, history of information related to the void object, and track the coordination progress. Furthermore, the concept was presented to specialist planners from architectural, structural and MEP disciplines. The interviewees were very much interested in having a convenient solution for the current issues of slot and opening coordination in their daily BIM workflows. The proposed concept of timely and transparent coordination with history tracking is well suited to bring efficiency to their current methods. In addition, as the process of slots and opening planning starts from the MEP discipline, additional end-to-end testing was done with an MEP specialist planner.

6.2 Conclusion

An object and history-based approach for slot and opening coordination through a database-driven openBIM workflow eliminates the need for comparison and resending of files or complete void instance models among the involved planners. In addition, it can overcome the limitations of pure file-based versioning for tracking previous decisions and changes of Void provisions (PfV objects). The developed cloud-database concept provides the planners with the most up-to-date information of PfV objects in their preferred BIM authoring tool. In addition, it enables change tracking of void objects regarding the geometry, position, and approval parameters and provides an overview of the coordination progress. Thus, the concept can speed up the approval and change management process with transparent and timely communication in the slot and openings planning for MEP-services in construction projects. This will aid designers in keeping consistency by an assisted automated approach, which improves the design quality and provides the designers with a more efficient method for slots and openings coordination. In conclusion, the developed concept has the potential to save a lot of time and effort for the specialist planners, which they could have utilized more on their productive and creative tasks.

6.3 Outlook

During the development of this practice-oriented thesis, some challenges along with new concepts and ideas arose conceptually and at the implementation level. Some of the relevant challenges and future aspects are described in the following section, which has to be addressed and explored to enhance the subject at hand.

6.3.1 Challenges

One of the main issues identified was the lack of resources to understand the solid background, knowledge requirement, and process. There were relatively few research publications regarding MEP coordination and addressing the topic of void planning or slot and opening coordination. It was mainly through intensive interaction with industry professionals that developed the solution's objective. Furthermore, the solution concept was modified and streamlined to make it more adaptable for the end-user. However, during the study and the development of the prototype, a few shortcomings were identified, which are listed below:

Dependency on the IfcGUID: The current prototype works based on IfcGUID and uses it for identifying the void in concern. Each void is assigned with an IfcGUID during the IFC export of the PfV models. Problems arise when a void is accidentally deleted by the MEP. Recreation of the void results in a new IfcGUID and causes an error in the long run. To mitigate this issue, the previous IfcGUID should be provided manually for the newly recreated void as well, or a mapping between the old and new IfcGUID of the concerned void needs to be performed. This can be done by having additional parameters for the void object. Further research on improving IfcGUID and, if not possible, further investigations should be done for identifying the voids to make the process more efficient and effective.

Modelling of the graph: In the thesis, the graph database was selected as the collaborative cloud database to achieve a single source of information concept and to perform object-level versioning and object tracking for the approval process. The focus was given to finding a way for object-based communication and history tracking. Therefore, the optimization of graph storage was not in the mainstream of the prototype development. Also, while modelling the graph directly from the whiteboard data model approach, there was a challenge in naming the relationship and nodes. One question was where to assign data; in nodes/ in labels or in two nodes/ in one node with more property and so on. This aspect of graph modelling (storage optimization) has to be further investigated in future.

Testing and evaluation: The testing of the concept was done in an existing reference project. The test was done with the developed plugin in the Autodesk Revit software. In order to investigate more solid aspects of the proposed approval method for slot and opening coordination, it needs to be done in cooperation with multiple disciplines in multiple approval iterations with various BIM authoring tools. This is important as the process is a highly co-dependent and multi-disciplinary decision-making task. Therefore more existing projects in various levels of complexity need to be tested and evaluated for refining the proposed concept in future.

Integration of approved voids into the discipline-specific model: After a Provision for Void (PfV object) is approved through coordination, the specialist planner integrates this void as a cut-out in their discipline-specific model. In openBIM workflows, there is no direct approach to convert approved voids to the cut-out in the discipline-specific BIM models. The thesis was focused on the approval process. Therefore adaptation of approved voids to cut-out was not checked in detail. So, this problem must be addressed in future to make the proposed openBIM workflow more efficient.

6.3.2 Further opportunities

Automatic synchronization and checking routine: As per the proposed concept, designers can access the most up-to-date information of each Void proposal (PfV object) from their BIM authoring software. This information, along with history tracking, assist designers to keep consistency and change management of integrated PfV objects as cut-out objects. This process can be further developed to a fully-automated approach through a rule-based algorithm that checks each cut-out with their corresponding void status in the database. The vision is:

*"When the MEP designer changes the position of the already approved PfV object, it could be possible to notify the specialist planners in their BIM authoring tool during the checking routine and ask for a new approval iteration" (*see Figure 6.1*).*

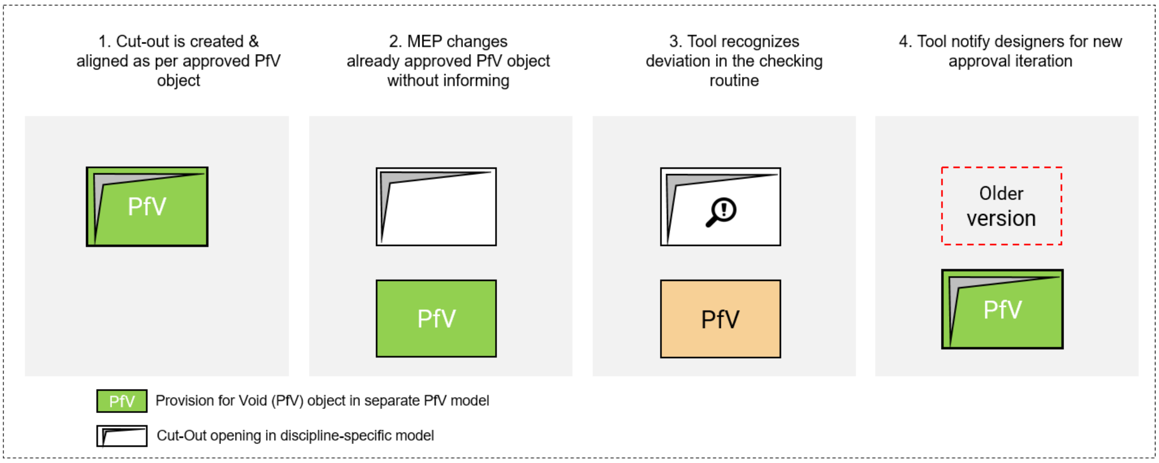


Figure 6.1: The concept of change management through rule-based checking routine

This could be achieved by further investigating and developing the proposed concept object-based versioning for void objects given in Figure 4.4 and extending the functionalities of the gatekeeper tool (prototype). One approach would be that, after approval of each void through the coordination process, the gatekeeper tool can be used to generate the cut-out object in the discipline-specific model. This could be achieved in a similar way of recreating history indicators discussed in the thesis (see Figure 4.7 and Figure 4.18), as the database could store the latest geometry, position, dimension data. After creating the cut-out object, the tool would update the native object ID, host element ID etc., of this cut-out to the labelled property graph; thus, a new relationship could be created between the latest provision for void data and the corresponding cut-out object. A 'white-board modelling of the case' is illustrated in Figure 6.2. Then, by employing a rule-based algorithm could infer from the graph for deviations and changes. This way, it would be possible to keep consistency between Void provisions and corresponding cut-out objects in the discipline-specific model in a quick way, where human error can be reduced. However, this needs to be further studied in detail regarding its technical and conceptual aspects.

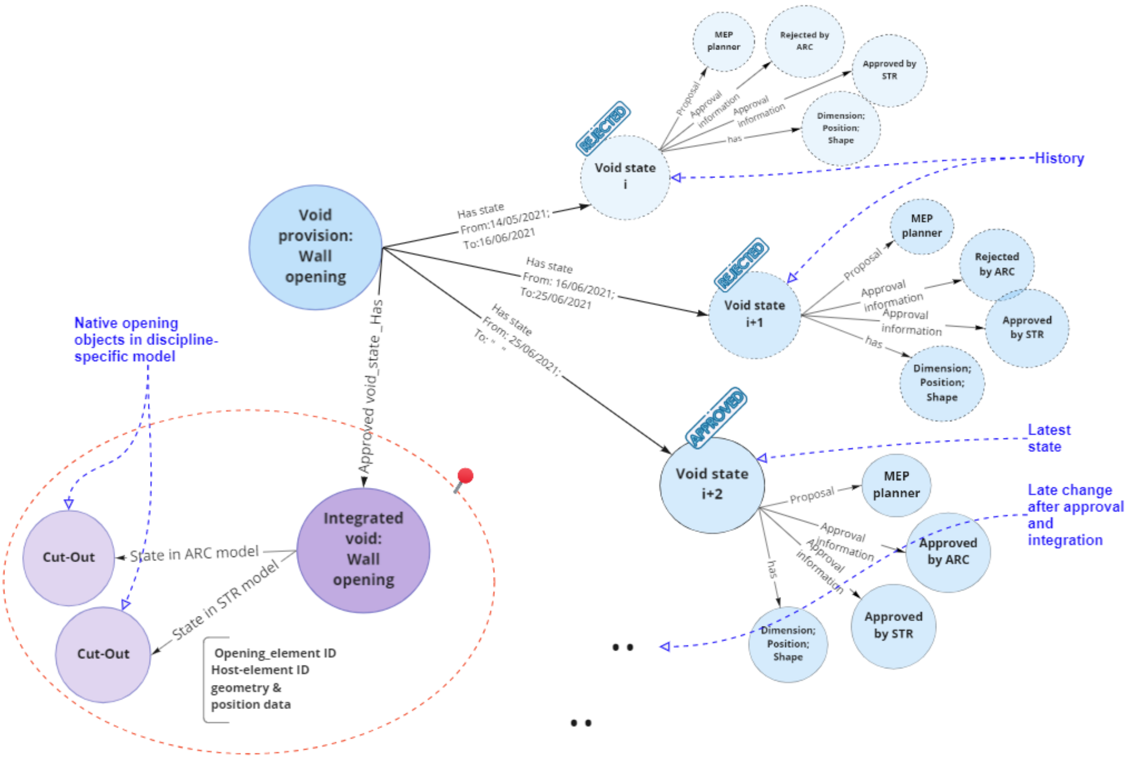


Figure 6.2: The concept for integrating provision for voids and corresponding discipline-specific cut-out objects in a labelled property graph

Cloud-based solution: The implemented gatekeeper tool is integrated with the BIM authoring tool (Autodesk Revit) for performing slot and opening coordination. This on-premise-/desktop solution might be extended to a cloud platform which could make the coordination task more accessible everywhere. Thus, the desktop-based nature and the requirement of the running BIM authoring tool for the operation of the gatekeeper functionalities can be avoided. For example, a web application can be created that could perform the coordination process online without the need of a BIM authoring tool by employing emerging cloud development platforms and libraries such as Autodesk Forge (Reynolds, W., 2018), Speckle etc. Such a cloud-based platform could extend the functionalities of the implemented prototype to a web-based manner, such as providing proposals and approvals for void objects, history tracking, history visualizations etc. Furthermore, updating and managing the information in the cloud graph database can be achieved from the cloud platform itself (such as utilizing the driver libraries for Neo4j). Coupling this platform with functionalities such as viewing the BIM/IFC models online, extracting and updating the information from the BIM/IFC model using Autodesk Forge APIs (Reynolds, W.,2018), Speckle, IFC.js etc., might improve the possibilities of the gatekeeper tool. It might also be possible to update the changes to the BIM model on the BIM authoring tool by implementing the connectors or plugins to the authoring tools for the cloud-based platform. However, further aspects of this cloud platform-based approach need to be studied and evaluated in the real scenario.

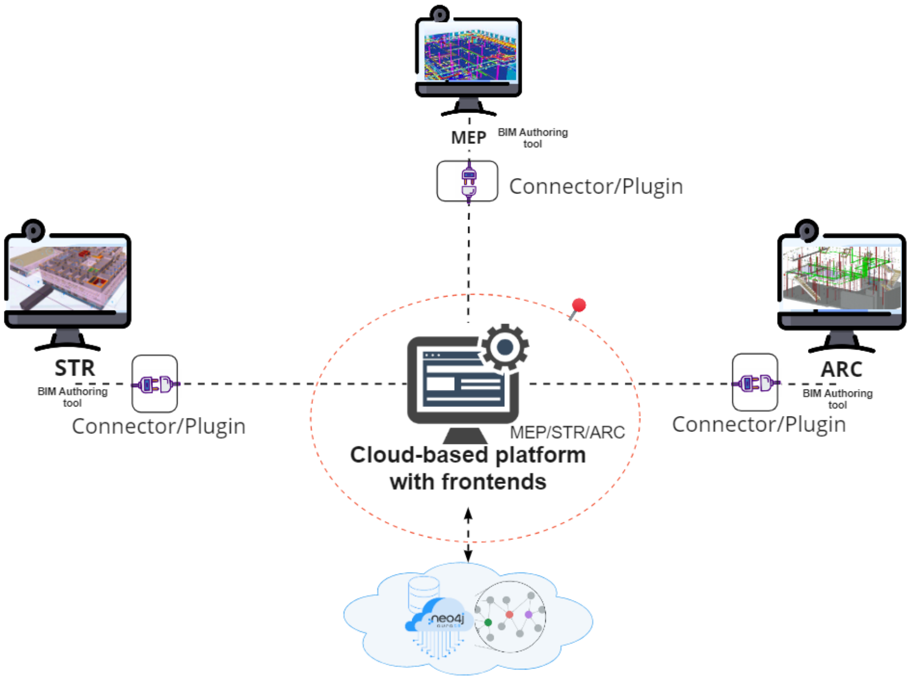


Figure 6.3: Vision for a cloud-based platform for slot and opening coordination with object tracking

References

Abdelhameed, W., Saputra, W. (2020). Integration of Building Service Systems in architectural design. Journal of Information Technology in Construction (ITcon). Retrieved November 1, 2021, from https://www.itcon.org/paper/2020/7.

Akponeware, A., & Adamu, Z. (2017). Clash Detection or Clash Avoidance? An Investigation into Coordination Problems in 3D BIM. Buildings, 7(3), 75. https://doi.org/10.3390/buildings7030075

Ashuri, Baabak & Yarmohammadi, Saman & Shahandashti, Mohsen. (2014). A Critical Review of Methods Used to Determine Productivity of Mechanical, Electrical, and Plumbing Systems Coordination. 777-786. 10.1061/9780784413517.080

AWS. (2022, January 19). What is a Relational Database? – Amazon Web Services (AWS). https://aws.amazon.com/relational-database/

Barton, P.K. (1983). Building Services integration, E & FN Spon, London

Beach, Thomas & Petri, Ioan & Rezgui, Yacine & Rana, Omer. (2017). Management of Collaborative BIM Data by Federating Distributed BIM Models. Journal of Computing in Civil Engineering. 31. 04017009. 10.1061/(ASCE)CP.1943-5487.0000657

Bhatt, M., Borrmann, A., Amor, R., and Beetz, J. (2013). "Architecture, computing, and design assistance." Automation in Construction

Bollinger+Grohmann. (2022). Retrieved February 1, 2022, from https://www.bollinger-grohmann.com/

Borrmann, A., König, M., Koch, C., & Beetz, J. (2018). Building Information Modeling: Why? What? How?, 1–24. https://doi.org/10.1007/978-3-319-92862-3_1

The Building Coder. (2011). Internal Imperial Units. https://thebuildingcoder.typepad.com/blog/2011/03/internal-imperial-units.html

buildingSMART International. (2021). BIM Collaboration Format (BCF) - buildingSMART International. buildingSMART International. https://www.buildingsmart.org/standards/bsi-standards/bim-collaboration-format-bcf/

BuildingSMART-Regionalgruppe Mitteldeutschland (2020). Leitfaden für die Schlitz- und Durchbruchsplanung auf Basis von IFC, 3

BuildingSmart and VDI. (2021). VDI/BS 2552 blatt 11.2 - building information modeling - informationsaustauschanforderungen - schlitz- und durchbruchsplanung. VDI 2552 Part 11.2. Retrieved November 10, 2021, from https://www.vdi.de/richtlinien/details/vdibs-2552-blatt-112-building-information-modeling-informationsaustauschanforderungen-schlitz-und-durchbruchsplanung

DB-Engines. (2022, January 23). DB-Engines Ranking. https://db-engines.com/en/ranking/graph+dbms

Deutsches Architektenblatt. (2021). BIM mit Fachplanern Teil 3: Durchbrüche und Aussparungen. https://www.dabonline.de/2021/11/18/bim-server-modell-fachplaner-teil3-durchbrueche-aussparungen-schlitze-tga-kolllisionen/

DIN 1356-1 Bauzeichnungen – Teil 1: Arten, Inhalte und Grundregeln der Darstellung. Berlin: Beuth Verlag, Februar 1995

E. Santos, & R. C. Ferreira (2008). BUILDING DESIGN COORDINATION: COMPARING 2D AND 3D METHODS. Undefined. https://www.semanticscholar.org/paper/BUILDING-DESIGN-COORDINATION%3A-COMPARING-2D-AND-3D-Santos-Ferreira/21b365976b9de3f76dd0ac46c5e81c87e2bd6fca

Eastman, C. M. (2011). Bim handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors (2. ed.). Wiley.

Flamebar GFS 1000: Fire Stopping Mortar. (2021). https://www.newkem.com/product/passive-fire-protection/fire-stopping/flamebar-gfs-1000/

Flughafen Düsseldorf GmbH. (2020). Richtlinie zum CAD Datenaustausch Version Version 5.1.06 - Anhang A "HochBau". https://www.dus.com/-/media/dus/businesspartner/konzerneinkauf/downloads/02_anhang_a_hochbau_5-1-06.ashx

Hausknecht, K.; Liebich, T.: BIM-Kompendium. Building Information Modeling als neue Planungsmethode. Stuttgart: Fraunhofer IRB Verlag, 2016

Hofbeck, A., Gutena, T., & Matschinsky, F. (2021). Modellbasierte Schlitz- und Durchbruchsplanung in der openBIM Methode. https://www.rug-dach.de/images/bilder/durchbruchsdokumentation/Modellbasierte_Schlitz-_und_Durchbruchsplanung_in_der_openBIM_Methode_V01.pdf

Ikerd, W. (2020). BIM Project Execution Plan (BxP) Guide: An Introduction For Those New to BIM. BIMForum. https://bimforum.org/resources/Documents/BIMForum_BXP_2020-11_pub-draft.pdf

Isaac, S., & Navon, R. (2013). A graph-based model for the identification of the impact of design changes. Automation in Construction, 31, 31–40. https://doi.org/10.1016/j.autcon.2012.11.043

Isaac, S., Sadeghpour, F., & Navon, R. (2013). Analyzing Building Information Using Graph Theory. In F. Hassani, O. Moselhi, & C. Haas (Eds.), Proceedings of the International Symposium on Automation and Robotics in Construction (IAARC), Proceedings of the 30th International Symposium on Automation and Robotics in Construction and Mining (ISARC 2013): Building the Future in Automation and Robotics. International Association for Automation and Robotics in Construction (IAARC). https://doi.org/10.22260/ISARC2013/0111

Juszczyk, M., Tomana, A., & Bartoszek, M. (Eds.) (2016). Current Issues of BIM-based Design Change Management, Analysis and Visualization. : Vol. 164. Elsevier. https://www.researchgate.net/publication/311357373_Current_Issues_of_BIM-based_Design_Change_Management_Analysis_and_Visualization

Kamara, John & Augenbroe, Godfried & Anumba, Chimay & Carrillo, Patricia. (2002). Knowledge management in the architecture, engineering and construction industry. Construction Innovation: Information, Process, Management. 2. 53-67. 10.1108/14714170210814685.

Korman, T. M., Fischer, M. A., & Tatum, C. B. (2003). Knowledge and Reasoning for MEP Coordination. Journal of Construction Engineering and Management, 129(6), 627–634. https://ascelibrary.org/doi/10.1061/%28ASCE%290733-9364%282003%29129%3A6%28627%29

Lazarevic, L. (2019, December 16). Keeping track of graph changes using temporal versioning. Neo4j Developer Blog. https://medium.com/neo4j/keeping-track-of-graph-changes-using-temporal-versioning-3b0f854536fa

liNear. (2021). Collaborative working in Bim Processes. Retrieved December 15, 2021, from https://www.linear.eu/en/collaborative-working-in-bim-processes/

Machado, Fernanda & Mota, Paula & Moreira, Lorena & Ruschel, Regina. (2021). Template class to teach clash detection. ENCONTRO NACIONAL SOBRE O ENSINO DE BIM. 3. 1. 10.46421/enebim.v3i00.315.

Mehrbod, S., S. Staub-French, Narges Mahyar, & Melanie K. Tory (2019). Beyond the clash: investigating BIM-based building design coordination issue representation and resolution. Undefined. https://www.semanticscholar.org/paper/Beyond-the-clash%3A-investigating-BIM-based-building-Mehrbod-Staub-French/edea38e255b9ab7b26bb1f35f422be8c23ad15bf

Microsoft Azure. (2022, January 23). Non-relational data and NoSQL - Azure Architecture Center. https://docs.microsoft.com/de-de/azure/architecture/data-guide/big-data/non-relational-data

Neo4j. (2022). Graph Modeling Guidelines - Developer Guides: Describing a Domain. Neo4j. https://neo4j.com/developer/guide-data-modeling/

Peffers, Ken & Tuunanen, Tuure & Rothenberger, Marcus & Chatterjee, S.. (2007). A design science research methodology for information systems research. Journal of Management Information Systems. 24. 45-77

Pilehchian, B., Staub-French, S., & Nepal, M. P. (2015). A conceptual approach to track design changes within a multi-disciplinary building information modeling environment. Canadian Journal of Civil Engineering, 42(2), 139–152.

Reynolds, W. (2018). Representing MEP systems as graph data for rapid simulation and analysis. Autodesk University. Retrieved January 25, 2022, from https://www.autodesk.com/autodesk-university/de/forge-content/au_class-urn%3Aadsk.content%3Acontent%3A32085624-55e9-4e4e-9603-d10da3cf5825#handout

Riley, David & Varadan, Priya & James, Jennifer & Thomas, H.. (2005). Benefit-Cost Metrics for Design Coordination of Mechanical, Electrical, and Plumbing

Systems in Multistory Buildings. Journal of Construction Engineering and Management- asce - J CONSTR ENG MANAGE-ASCE. 131. 10.1061/(ASCE)0733-9364(2005)131:8(877)

Robinson, I., Webber, J., & Eifrem, E. (2015). Graph databases: New opportunities for connected data (Second edition). O'Reilly.

Trimble. (2017). Recesses in the BIM workflow. https://www.stabiplan.com/en-us/news/news-and-blogs/recesses-in-the-bim-workflow/

Victor, L. (2019a, January 25). MEP Coordination- Everything You Need to Know | Importance & Benefits. United-BIM Inc. https://www.united-bim.com/mep-coordination-bim-importance-benefits-outsourcing/

Victor, L. (2019b, January 25). MEP Coordination- Everything You Need to Know | Importance & Benefits: Approaches for Design Coordination of a Construction Project. https://www.united-bim.com/mep-coordination-bim-importance-benefits-outsourcing/

Vukotic, A. (2015). Neo4j in action. Manning Publications. https://learning.oreilly.com/library/view/-/9781617290763/?ar

Wang, Li & Leite, Fernanda. (2015). Process Knowledge Capture in BIM-Based Mechanical, Electrical, and Plumbing Design Coordination Meetings. Journal of Computing in Civil Engineering. 30. 04015017. 10.1061/(ASCE)CP.1943-5487.0000484

Xue, F., & Lu, W. (2020). A semantic differential transaction approach to minimizing information redundancy for BIM and blockchain integration. Automation in Construction, 118, 103270. https://doi.org/10.1016/j.autcon.2020.103270

Yarmohammadi, Saman & Ashuri, Baabak. (2015). Exploring the approaches in the implementation of BIM-based MEP coordination in the USA. 20. 347-363

Zamani, N. (2020). Structural Considerations for Openings in Composite Floor Decks. National Council of Structural Engineers Associations (NCSEA). https://www.structuremag.org/?p=16271

Annex A

Personal interview with industry professionals

Five BIM professionals from Architectural, structural and MEP backgrounds with sufficient knowledge in MEP coordination were interviewed as part of the thesis. The interviews were scheduled for 60 to 90 minutes and were conducted in English via video meetings/offline. An excerpt of the questionnaire is attached below. Moreover, the exact questions of the individual interviews were adaptive according to the respective discipline, answers, and situation. The interviews were solely focused on the context of the slot and opening coordination for MEP services in construction projects.

The goal of the interviews was as follows:

- Understand the existing challenges in the slot and opening in the design reality of planning/ coordination tasks
- Obtain the perspective of the proposed concept from the viewpoint of specialist planners from an Architectural/ Structural/ MEP background.
- How do you see and evaluate the process of ‘the slot & opening coordination tasks in your projects’? Do you agree that "The slot & opening coordination task is one of the most inefficient and time-consuming BIM coordination tasks”?
- Which approval methods/ tools are you using for conducting the slot & opening coordination in your BIM projects?
- Could you explain the main obstacles you face during the coordination of slots and openings in BIM projects?
- Are you using any standard approval procedure? For example, are you following BuildingSMART‘s proposal process for slot & opening coordination?
- Are you using BCF based approach for conducting the slot & opening coordination process?

- How beneficial would be Model + Database driven workflow compared to Model only workflow? (I see that it is helpful to validate the BIM Model elements over the database to avoid confusion and conflict)
- What is your vision on the history tracking or change tracking of BIM Model objects? For example, do you think having an option for tracking a void object's history might help the process?
- In your openBIM projects, have you had issues with "keeping objects' IfcGUID" over the project period?
- How do you see the future potential, and how do you evaluate the proposed concept?

Topic related discussions from Forums

In order to understand more details regarding the existing challenges in the slot and opening planning in construction projects, some popular forums were searched. Discussions among industry professionals on this topic contributed to gaining a greater understanding of the problem's significance and depth in daily design workflows. For the reader's convenience, links to some of the relevant discussions are provided below:

- https://forums.buildingsmart.org/t/provision-for-voids-in-ifc4x3-rc1/2998
- https://www.xing.com/communities/posts/durchbruchsplanung-die-2-1002608695
- https://www.hoai.de/forum/viewtopic.php?TopicID=2995
- https//www.tektorum.de/praesentation-darstellung/8895-deckendurchbrueche-bodendurchbrueche.html
- http://forum.cad.de/foren/ubb/Forum329/HTML/002438.shtml
- https://forums.autodesk.com/t5/revit-bim-360-deutsch/durchbruche-bei-phasenplanung/td-p/9694487
- https://connect.allplan.com/de/forum/themen/topic/topics/bim-benutzer-forum/ifc-import-nicht-nutzbar-bimplus-dadurch-weiterhin-unbrauchbar.html
- https://forums.autodesk.com/t5/revit-bim-360-deutsch/durchbruch-darstellung/td-p/10237114
- https://www.vectorworksforum.eu/topic/16373-durchbr%C3%BCche-in-wand-und-decke-nach-ifcwrapper-ifc-durchbruchsplanung-erweiterung/
- https://www.bosch-professional.com/de/de/community/category/durchbruch-in-beton/10911422-t#main https://forum.graphisoft.de/viewtopic.php?f=3&t=27476
- https://forum.freecadweb.org/viewtopic.php?t=25491
- https://www.tektorum.de/praesentation-darstellung/8895-deckendurchbrueche-bodendurchbrueche.html
- https://blogs.autodesk.com/bimblog/durchbruchsvorschlage-in-revit-mit-dynamo-ubernehmen/

- https://forums.autodesk.com/t5/revit-architecture-forum/ifc-void-coordination/td-p/6263981
- https://forums.autodesk.com/t5/revit-bim-360-deutsch/in-revit-mit-dynamo-basisbauteil-hsl-nach-verschneidung-erkennen/td-p/9328230
- https://bim2.ngbailey.co.uk/wiki/index.php/NGB_Standards:Builderswork#Adding_holes_retro-spectively

General rules for accelerating slot and opening coordination

- Construction relevant openings: The clearance and proximity to construction relevant openings must be checked during the slots and openings approval process to avoid later issues at the construction site. (The definition can vary from project to project and scenario)

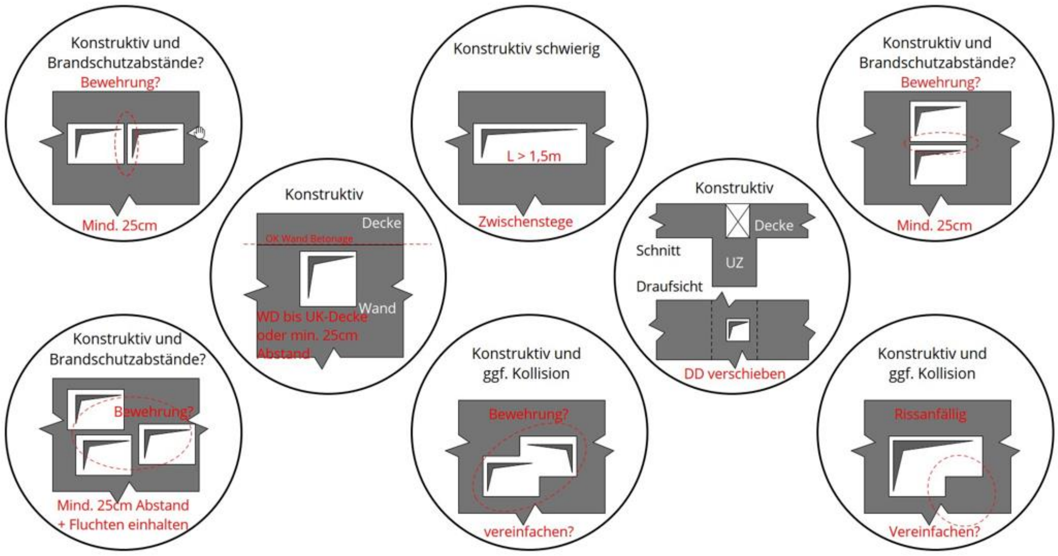


Figure: Construction relevant openings (General guidelines at Bollinger und Grohmann GmbH, 2022)

- Structural relevant openings: The openings must be checked during the slots and openings approval process with the approval from the structural disciplines to avoid issues regarding structural requirements (The definition can vary from project to project and scenario)

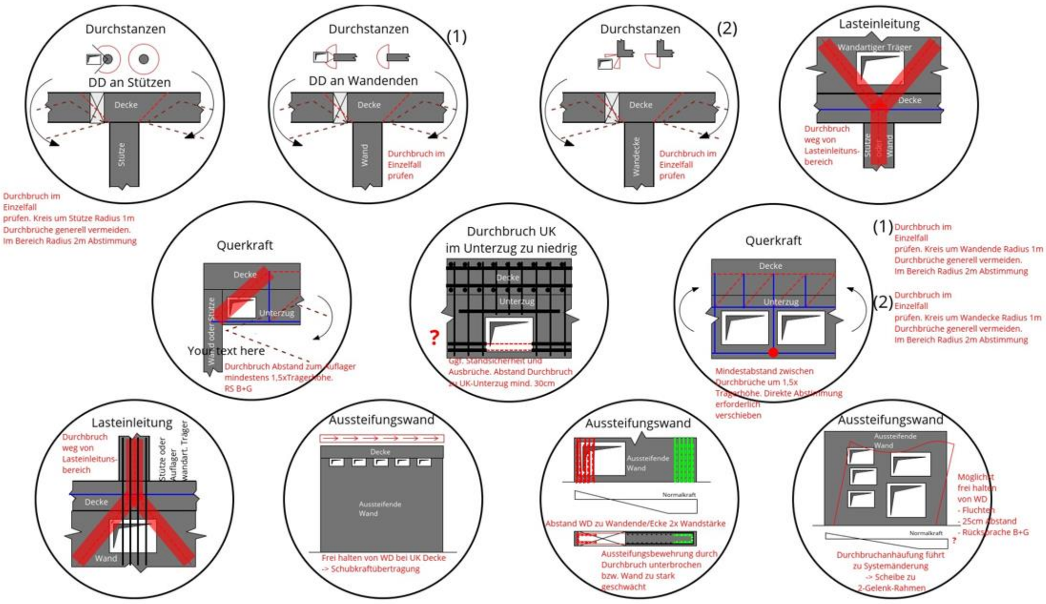


Figure: Structural relevant openings (General guidelines at Bollinger und Grohmann GmbH, 2022)

Annex B

This section includes the some important cypher queries created during prototype implementation, according to the defined graph model as illustrated in Figure 4.4.

1. Proposing new voids to the Graph DB

Code 2: Cypher query for proposing new voids to the Graph DB

```
CREATE (v0:Void {name: $void_name, IfcGUID: $IfcGUID})
CREATE (vn:Version {name: $void_VersionNo_Name, VersionNo: $VersionNo})
CREATE (v0)-[state_n:Has_State {FROM: date($Date), TO: ''}]->(vn)
CREATE (mep:Proposal_Discipline{name: $ProposalDiscipline})
CREATE (vn)-[:Proposed_By]->(mep)
CREATE (mepstatus:Proposal_Status {name: $ProposalDiscipline_Status})
CREATE (mep)-[:Status]->(mepstatus)
CREATE (mepcomment:Proposal_Comment {name: $ProposalDiscipline_Comment})
CREATE (mep)-[:Comment]->(mepcomment)
CREATE (mepapprovaldate:Proposal_Date {name: $ProposalDiscipline_Approval_Date})
CREATE (mep)-[:Date]->(mepapprovaldate)
CREATE (Pos:Position {name: $Position})
CREATE (vn)-[:Has]->(Pos)
CREATE (srn:Storey {name: $StoreyName})
CREATE (Pos)-[:Storey_No]->(srn)
CREATE (xcoord:PosX { name: $X_coord})
CREATE (Pos)-[:X_Coodinate]->(xcoord)
CREATE (ycoord:PosY { name: $Y_coord})
CREATE (Pos)-[:Y_Coodinate]->(ycoord)
CREATE (zcoord:PosZ { name: $Z_coord})
CREATE (Pos)-[:Z_Coodinate]->(zcoord)
CREATE (dim:Geometry {name: $Dimensions})
CREATE (vn)-[:Has]->(dim)
CREATE (shp:Shape {name: $VoidShape})
CREATE (dim)-[:Shape]->(shp)
CREATE (wth:Width {name: $VoidWidth})
CREATE (dim)-[:Width]->(wth)
CREATE (ht:Height {name: $VoidHeight})
CREATE (dim)-[:Height]->(ht)
CREATE (dia:Diameter {name: $VoidDiameter})
CREATE (dim)-[:Diameter]->(dia)
CREATE (dpt:Depth {name: $VoidDepth})
CREATE (dim)-[:Depth]->(dpt)
RETURN *
```

The parameters starting with the '$' sign in the code are input arguments for the query. These are assigned during the run time of the prototype according to the selected voids' parameter values from the Revit. This can be replaced with specific values while querying directly in the Neo4j browser. An example case of using this query to propose a new void (Version 0 as it is the first proposal) with the name ‘Wall penetration’ and other parameters associated with the selected void in the PfV void model is shown in Figure below.

Figure: Resulting Data visualization in Neo4j browser

2. Provide choice of approval for selected Void into Graph DB

Code 3: Cypher query for providing choice of approvals by the specialist planners to the Graph DB

```
MERGE (v0:Void {IfcGUID : $IfcGUID})-[state_old:Has_State {TO: ''}]-(v_old:Version)
CREATE (vn:Approval_Version { name: $void_VersionNo_Name, VersionNo: $VersionNo })

CREATE (v_disc:ApprovalDiscipline  { name: $ApprovalDiscipline })
CREATE (vn)-[:Choice_of_approval_from]->(v_disc)

CREATE (v_old)-[state_approval:Has_Approval_state { FROM: ''}]->(vn)
SET state_approval.FROM = date($new_Date)

CREATE (spec_status:Approval_Status{ name: $ApprovalDiscipline_Status })
CREATE (vn)-[:Has_Status]->(spec_status)
CREATE (spec_cmt:Approval_Comment { name: $ApprovalDiscipline_Comment })
CREATE (vn)-[:Has_Comment]->(spec_cmt)
CREATE (spec_date:Approval_Date { name: $ApprovalDiscipline_Approval_Date })
CREATE (vn)-[:Approval_Date]->(spec_date)

RETURN *
```

Similar to the above-mentioned functionality, the parameters starting with the ‘$’ sign in the code are input arguments for the query. These are assigned according to the selected voids’ parameter values from the Revit. An example case of using this query to provide a choice of approval to the latest version (Version 0 in this case) of the void with the name ‘Wall penetration’ with a specific IfcGUID and its data visualization is shown in the Figure provided below.

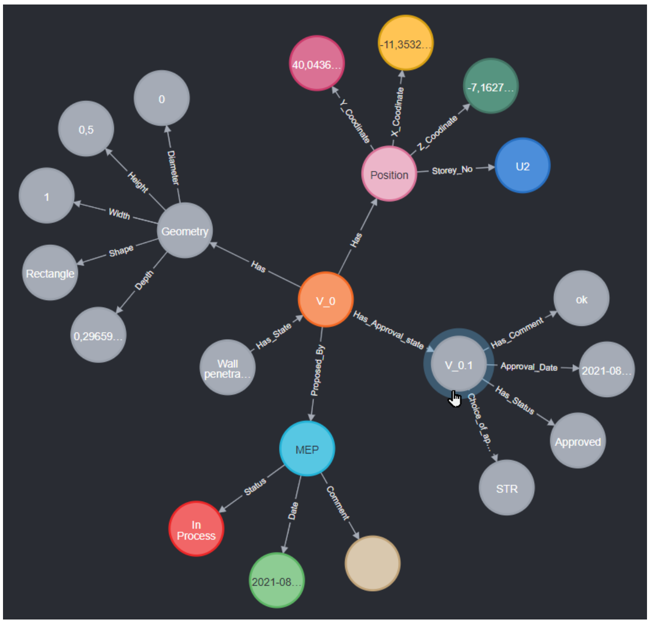


Figure: Resulting Data visualization in Neo4j browser

Declaration

I hereby declare that I have written the Master thesis independently. Only the Sources and tools explicitly named in the thesis were used. I have marked as such any ideas that have been taken over literally or analogously.

Further, I assure that the thesis has not yet been the subject of any other examination procedure.

München, 9. February 2022

Ganga Surendran Sanila

Ganga Surendran Sanila
[redacted]