CADModelScope: Revealing the Dependency Structure Behind Parametric Computer-Aided Design Models

Yuanzhe Deng
Mechanical and Industrial Engineering
University of Toronto
Toronto, Ontario, Canada
yuanzhe.deng@mail.utoronto.ca

Zhijing Zhang
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada
sallyz.zhang@mail.utoronto.ca

Shurui Zhou
ECE&CS
University of Toronto
Toronto, Ontario, Canada
shurui.zhou@utoronto.ca

Alison Olechowski
Mechanical and Industrial Engineering
University of Toronto
Toronto, Ontario, Canada
a.olechowski@utoronto.ca

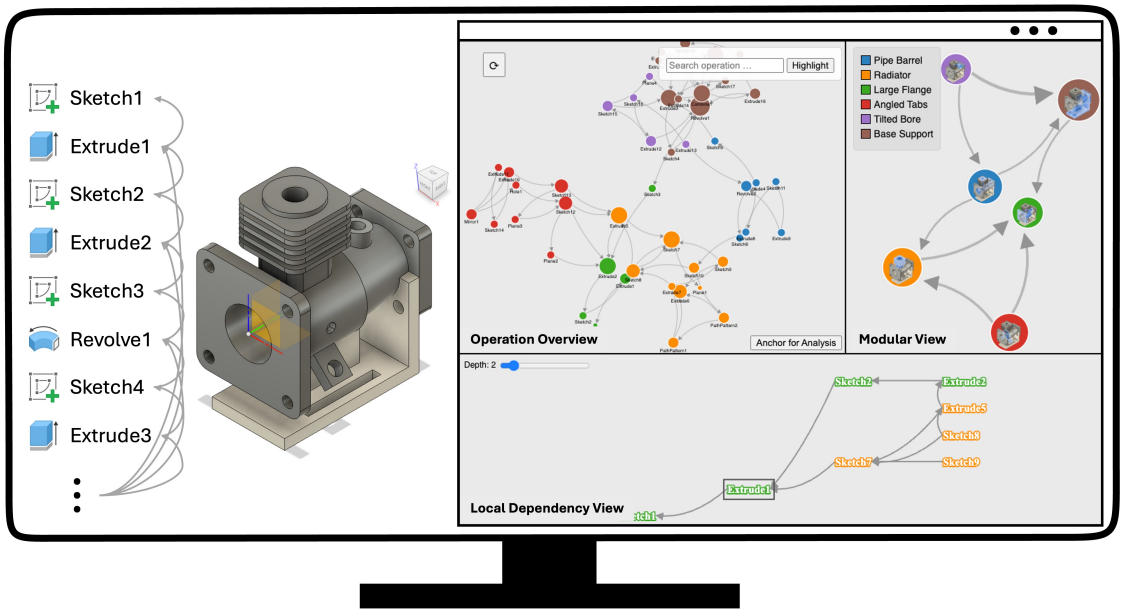


Figure 1: While parametric CAD models are constructed using a linear sequence of modelling operations, the true structure is a complex network of hidden dependencies among operations (left). Our assistive tool, *CADModelScope*, explicitly visualizes this complexity (right): (1) an *Operation Overview* reveals the full dependency structure, (2) the *Modular View* clusters closely interdependent operations into functional units, and (3) the *Local Dependency View* supports localized dependency tracing for any selected operation. All graph nodes are coloured in consistence with clustering results in the *Modular View*.

Abstract

Parametric computer-aided design (CAD) models are constructed by a sequence of operations, where each operation may reference geometries created by earlier operations. This network of dependencies enables efficient modelling of complex geometry but also results in fragile models, where small modifications can trigger cascading errors. These interdependencies are obscured in commercial CAD systems, leaving users to rely on trial and error when navigating, modularizing, and debugging unfamiliar and complex models. In this paper, we motivate, present, and pilot *CADModelScope*, a multi-level graph-based visualization of operation dependencies integrated into a commercial CAD platform. In a qualitative lab study, we observed how participants locate and interpret operations, and how *CADModelScope* enhances awareness of hidden

CHI '26, Barcelona, Spain

ACM ISBN 979-8-4007-2278-3/26/04
https://doi.org/10.1145/3772318.3791070

interdependencies and supports more structured navigation. Our findings highlight the potential of using the network of operation dependency as an effective representation for understanding and interacting with parametric CAD models, and we discuss implications for future tool design.

CCS Concepts

• **Applied computing** → **Computer-aided design**; • **Human-centered computing** → **Information visualization**; **Graphical user interfaces**.

Keywords

Parametric CAD, Modelling Operation, Dependency Management, Program Navigation, Awareness, Information Foraging

ACM Reference Format:

Yuanzhe Deng, Zhijing Zhang, Shurui Zhou, and Alison Olechowski. 2026. CADModelScope: Revealing the Dependency Structure Behind Parametric Computer-Aided Design Models. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3772318.3791070

1 Introduction

Computer-aided design (CAD) is an essential tool for modern engineering and product development. By enabling the digital modelling of a product's detailed geometry, CAD supports rapid design iterations with kinematic analysis and computer simulations before physical prototyping or manufacturing [56]. In the widespread history-based parametric modelling approach to CAD, users construct 3D models through a sequential series of *modelling operations*[1] (e.g., sketch, extrude, fillet). These operations form a linear *operation sequence*[2] that records the model's construction process (see Figure 2), where each operation generates new, or removes existing, geometric entities (e.g., vertices, edges, faces) in the order they appear in the sequence. Every operation is defined by a set of numerical parameters and references to existing geometric entities, which are generated by earlier operations in the sequence, creating a network of dependencies that supports the efficient creation and modification of complex geometry [15]. Although these interdependencies among modelling operations play a critical role in supporting accurate yet flexible model generation, they are concealed within the linear view of the operation sequence, and limited support is currently available in commercial CAD software for visualizing these hidden relationships.

The interdependency between modelling operations enables modifications made to one operation to automatically propagate to downstream dependent operations. However, poorly defined dependencies can be a common cause of model regeneration errors [10, 34, 60]. Building a model with clear *design intent* is a complex challenge for the CAD modeller; building design intent into the model involves dimensioning the operations in such a way that modifying one will result in appropriate parametric changes to other operations [28, 51]. Mastering effective modelling strategies to develop design intent requires substantial training and experience [23, 58]. These challenges are further magnified in collaborative settings. Without disciplined adherence to unified modelling conventions, teams often struggle to interpret each other's design intent and frequently run into editing errors and inefficiencies [3, 17, 60]. As models grow with more operations and deeper dependency chains, the difficulty of navigating and managing this network of operation dependency also escalates, ultimately undermining the model's reusability [22].

Despite the importance of operation dependency in CAD workflows, software's built-in support for navigating through these complex structures remains limited, and dependency management in CAD remains challenging [18]. As shown in Figure 2a–c, when a user inspects an operation, commercial CAD software user interfaces (UI) typically surface only the operations that the operation directly depends on, and the operations that directly depend on it (i.e., one-hop dependency). Yet in practice, operation failures often arise from modifications made to a much earlier operation in the sequence that lead to unintended changes that propagate through the dependency chain [34]. While some CAD software, such as CATIA V5-6, offer a complete dependency graph (a schematic representation shown in Figure 2d) to support dependency tracing, these visualizations have been reported to become overwhelming in large models due to the densely intersecting links [46, 60], limiting the effectiveness in conveying the high-level model structure. Ultimately, when a user needs to trace operation failure dependency issues on modern CAD platforms, they are required to manually inspect multiple operations along the complex dependency chain step by step, which can be time-consuming and cognitively demanding. It is this challenge that we aim to address in this work: **the need for more accessible and interactive visualizations of hidden modelling operation dependencies to support understanding and navigation of large and unfamiliar CAD models**.

We begin by identifying and analyzing the challenges users face when navigating operation dependencies within a single CAD model in Section 3, building on prior work by Cheng et al. [18]. Our analysis reveals common barriers to understanding and navigating modelling operations, stemming from commercial CAD platforms' lack of support for visualizing the hidden operation dependencies. Informed by our analysis, we developed *CADModelScope* (see Section 4), an interactive, multi-level graph-based visualization tool implemented as an add-in for Autodesk Fusion.[3] *CADModelScope* supports dependency-aware operation navigation through three key features: (1) an *Operation Overview* to build high-level awareness of operation dependency, (2) a *Modular View* to expose the structural organization of operations, and (3) a *Local Dependency View* to enable efficient dependency tracing and editing.

To evaluate the effectiveness of *CADModelScope*, we conducted a user study to examine how participants navigate large and unfamiliar CAD models with and without *CADModelScope* (Section 5). Our findings show that *CADModelScope* fosters greater awareness

[1] Also commonly known as "modelling features" on many CAD platforms. We use the term "operations" in this paper to distinguish from functional attributes of a CAD model, which we call "features" here. As an example of usage in this paper, a seat is a functional feature of a bike, and the seat is created using multiple modelling operations in CAD.

[2] Also known as the "design timeline" in Autodesk Fusion or the "feature list", "design tree", etc. in various CAD platforms.

[3] https://www.autodesk.com/products/fusion-360/overview

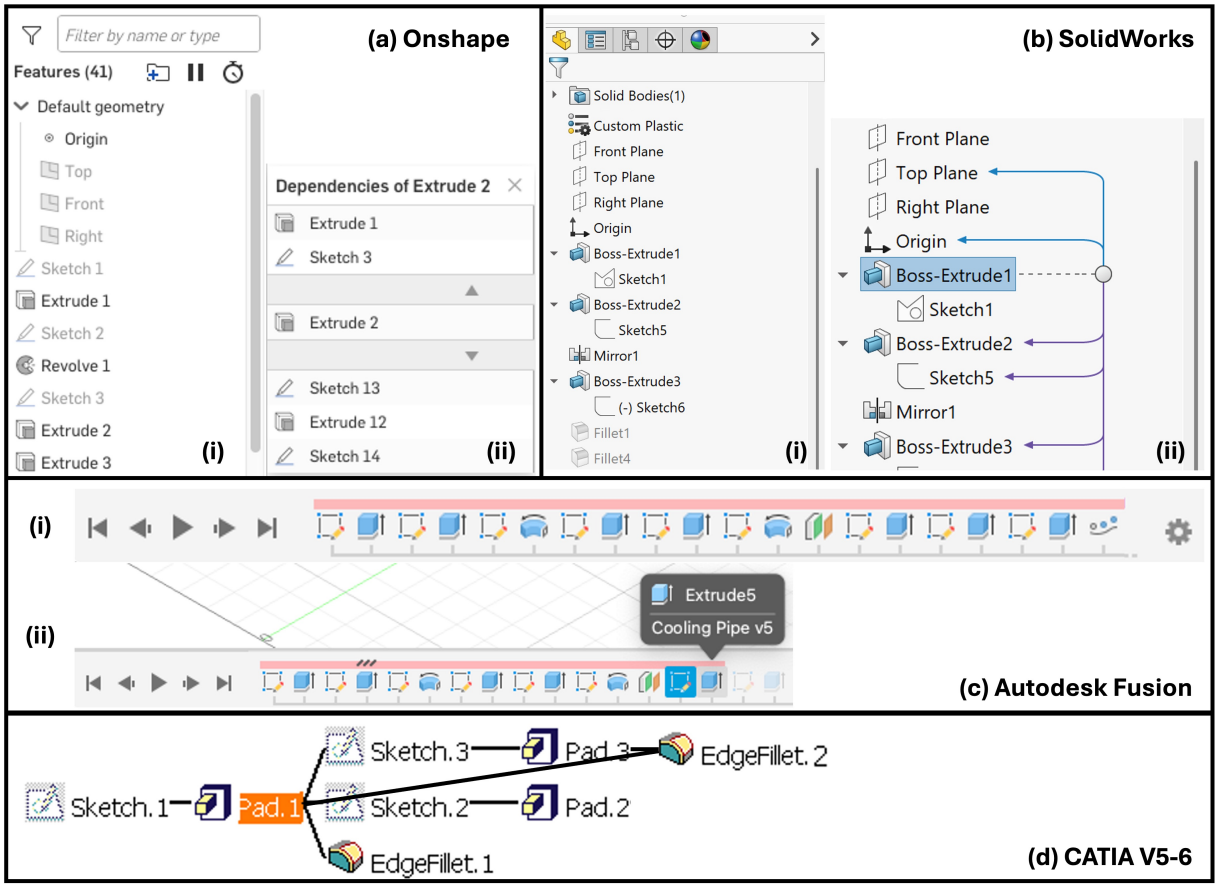


Figure 2: Representative visualizations of the modelling operation sequence and the dependencies among operations in commercial CAD software. (a) (i) Onshape presents modelling operations in a vertical list. (ii) Dependencies of a selected operation are shown in list view, presenting operations that it depends on (at the top) and operations that depend on it (at the bottom). (b) (i) SolidWorks presents operations hierarchically in a vertical tree, with referenced sketches absorbed into solid modelling tools. (ii) Dependencies of a selected operation are presented with coloured arrows directly pointing to upstream and downstream operations with direct dependency. (c) (i) Autodesk Fusion presents operations in a horizontal sequence. (ii) When a modelling operation is in editing mode, the sequence highlights sketches that it depends on, and markers are added on top of solid modelling tools that generated any of the referenced geometries. (d) CATIA V5-6 presents operations using a similar vertical tree like SolidWorks (not shown), but a graph can be generated to show all upstream and downstream operations of a selected operation. Screenshots and schematic representations of software interfaces are presented for comparison purposes only.

of the underlying dependency structure and enables more structured navigation compared to exhaustive trial-and-error strategies commonly observed in commercial CAD. In particular, the *Modular View* proved especially valuable for contextualizing information foraging in large and complex operation sequences. We further synthesize these findings into design guidelines and implications for future research and tool development in Section 6.

In summary, our main contributions are:

- A characterization of user challenges in navigating modelling operations within large and unfamiliar CAD models, highlighting the limitations of current commercial CAD systems (Section 3).
- The design and implementation of *CADModelScope*, an interactive, multi-level graph-based visualization tool that exposes hidden operation dependencies and supports dependency-aware navigation in CAD (Section 4).
- An empirical evaluation of user workflows with and without *CADModelScope*, demonstrating how dependency-aware visualizations reshape navigation strategies and support more structured interaction with complex CAD models (Section 5).
- Design guidelines and implications for future research and tool development, highlighting broader insights into how making underlying system structures visible can improve sense-making and tool usability in complex software systems (Section 6).

2 Background and Related Work

In this section, we first introduce the detailed background of dependencies in CAD in Section 2.1. We then review earlier attempts to visualize these dependencies (Section 2.2) and tools that assist users in dependency-aware model editing (Section 2.3) from the literature.

2.1 Dependencies in CAD

Parametric CAD software regenerates the affected model every time a modelling operation is added, removed, or modified. This regeneration re-processes all operations in the order they are listed in the sequence and rebuilds the model. To prevent circular dependencies among modelling operations, an operation can only reference geometric entities (bodies, faces, edges, vertices) generated by preceding operations in the sequence (establishing *upstream* dependencies), not those created by later operations. Thus, when editing a modelling operation, only geometries generated by its preceding operations are visible to the user for reference. Conversely, an operation may also possess *downstream* dependencies, established when subsequent operations reference the geometry it generates. When a user selects a geometric entity, the CAD program typically highlights the operation that last modified the selected entity, but tracing its full network of upstream and downstream dependencies must be done manually in most CAD platforms. Furthermore, reordering existing operations in the sequence is only valid if the resulting dependency structure remains acyclic, ensuring the model can still regenerate without errors.

When upstream operations are modified, the referenced geometric entities may be altered in shape, split into multiple new entities, or even removed entirely. For a model built with poorly structured design intent, upstream modifications may result in errors or unintended geometric behaviours in dependent downstream operations, either because a referenced entity is removed or because its altered form no longer matches the original design intent [16, 34]. Different CAD programs respond differently in this instance. For illustration, we built a simple model in Autodesk Fusion as shown in Figure 3 to demonstrate how upstream changes can cause issues downstream. In this example, "Extrude 1" references "Sketch 1" to generate an edge that "Sketch 2" references, but modifying "Sketch 1" removes the edge that "Sketch 2" refers to. As Fusion silently updates "Sketch 2" with the invalid constraint removed, "Sketch 2" becomes under-constrained that lacks enough information to determine its shape uniquely, causing the hole's position to be decoupled from future geometric updates. If the same model is built and modified in Onshape, in comparison, the program will report an error in "Sketch 2" with a warning of missing references. The trade-off between enabling flexible geometric updates and ensuring robust parametric models leads to the inconsistency in how different CAD programs resolve reference updates, but the end users are mostly left out of the resolution process.

2.2 Navigating Modelling Operation Dependency

For complex designs composed of highly interdependent modelling operations, revealing the hidden structure of the CAD model is essential in both to identify the appropriate operation to edit and to understand the broader implications of that edit [63]. Given the interdependent nature of modelling operations in CAD, visualizing these relationships as a network graph naturally reveals their dependency structure [13]. In such representations, each modelling operation corresponds to a graph node, with graph links indicating dependency relationships. However, as the model complexity increases, as would be the case in a professional design setting, the number of operations grows rapidly, leading to densely populated graphs that are hard to navigate [46, 60]. To address this issue, various research studies tried to develop more effective presentation of the CAD model, and ultimately better user awareness, by supporting interactivity, reducing clutter, and better visualizing the graph nodes, links, and the overall layout [41].

For instance, Camba and Contero observed that sketches are often referenced by only a single non-sketch operation, which solely transforms all the sketch entities into 3D solids [14]. Following this logic, and in line with how some CAD software already visually group these connecting sketches under their directly dependent operations in the operation sequence (e.g., Figure 2b(i)), graph representations can merge these sketch nodes with their associated operations. Eventually, this technique reduces the number of nodes being presented and simplifies the graph. However, while this improved representation helps reduce visual clutter, it only partially mitigates complexity and may fail to scale well with increasingly large models.

An alternative approach was proposed by Marchenko et al., who introduced an augmented view of the traditional tree-form operation sequence (e.g., Figure 2b(i)) [46]. Their method presents two parallel copies of the tree-like representation of a CAD model, with graph-like links added between corresponding operations across the trees to indicate dependencies. This layout provides more space for visualizing the dense interconnections, reducing visual clutter and easing cognitive load in some areas. Nevertheless, this approach does not resolve the core limitation of the linear sequence itself. As the number of operations increases, the sequence can become excessively long, making it difficult to locate and follow dependencies between operations that are far apart.

While these studies present important concepts for reducing complexity and improving the effectiveness of the visualization, they focus primarily on demonstrating the algorithmic feasibility of these proposed visualizations. Out of all the reviewed studies, only Kozlova et al. [41] took a user-centric approach, which would involve conducting user studies to assess how, or whether, these visualizations actually help users better navigate complex CAD models. Responding to this gap in the literature, in this study, we conducted a controlled experiment to closely observe how users navigate the modelling operations in a large, unfamiliar CAD model with and without our proposed visualization tool.

2.3 Editing Modelling Operation Dependency

Beyond visualizing the complete network of dependencies among modelling operations, users also constantly modify existing operations to iterate the design and reorder operations for organizational purposes. After every modification made to the model, users need to ensure the validity of all hidden operation dependencies, or the model may fail to properly regenerate and cause an error due to

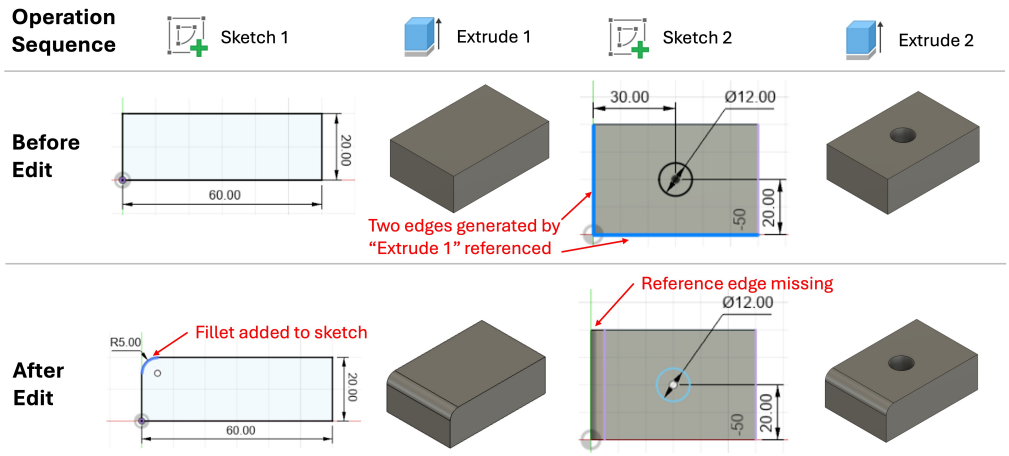


Figure 3: Impact of upstream operation modification to downstream dependencies in a CAD model designed with four modelling operations (top row - Operation Sequence) in Autodesk Fusion. Initially (middle row), the hole's position is defined in "Sketch 2" by referencing two straight edges created by the "Extrude 1" operation. After "Sketch 1" is modified with an added fillet (bottom row), one of the edges referenced by "Sketch 2" is replaced by a curved surface. As a result, the constraint for the hole's horizontal position becomes invalid, leaving the circle in "Sketch 2" unconstrained and decoupled from future geometric updates.

broken dependencies. In this section, we review other research efforts that focused on supporting specific modelling actions that involve editing and restructuring these existing dependencies.

To assist users in resolving dependency-related errors following operation edits, Hähnlein et al. approached the problem through the lens of software debugging [34]. They prototyped a graphical debugging tool that compares *model states* before and after an operation is modified. This tool helps narrow the user's attention to a small subset of upstream operations that the failed operation depends on, especially those that introduce significant geometric changes after the edit. However, this solution assumes access to a CAD platform with version control, which records the full model state after every user action, enabling fine-grained comparisons across the operation sequence. In practice, this capability is generally not available in conventional product data management (PDM) systems, which typically centralize versioning and only preserve model states at major milestones, therefore not recording every individual edits [61].

Beyond passively debugging errors as they arise, another line of work considers how operations can be deliberately restructured to improve the model's clarity and resilience. While multiple modelling workflows may lead to the same final geometry, some workflows are better at communicating the design intent and producing models that are more robust to modifications [60]. A key goal when the user is structuring the modelling operations is to decompose a complex model into smaller modular subsystems, enabling collaborators to concentrate their attention on individual parts of the model with reduced complexity [59]. To support the analysis and restructuring of large CAD models, Bhaskara introduced the use of a design structure matrix (DSM) to represent the interdependency between modelling operations [8]. For a model constructed with n operations, an $n \times n$ matrix effectively encodes these dependencies, where a matrix element a_{ij} in row i and column j is set to 1 if operation j depends on operation i, or set to 0 otherwise [30]. Using matrix partitioning algorithms, closely interdependent operations are grouped into subsystems, and heavily referenced operations are repositioned earlier in the operation sequence. Reordering and grouping modelling operations in modular structures have the advantages of enabling better documentation of the design intent, more effective impact analysis, and more efficient design reuse [8]. While this approach is promising, it has only been demonstrated manually on a small-scale model, and its integration into practical real-world workflows remains largely unexplored.

Unlike previous work that either focuses on localized error debugging or high-level overview, our work aims to develop a comprehensive solution that addresses all the above-mentioned challenges. Further, our approach is grounded in conventional, commercially available CAD software, and we evaluate the effectiveness of our tool with complex CAD models, not limited to simple demonstrations.

3 Design Motivations

Modern CAD software enforces the validity of operation dependencies to ensure that models can regenerate without errors following user edits. However, as the volume of dependencies grows in a model, these tools provide limited support for users to navigate this intricate network of interdependent operations. As a result, users often lack awareness of the underlying dependencies among operations when making design decisions (see Figure 2 for examples of dependency visualizations in commercial CAD systems).

To better understand these challenges, Cheng et al. [18] conducted a thematic analysis of online forum discussions and semi-structured interviews, identifying three overarching categories (traceability-, navigation-, and consistency-related) that span nine

key challenges users face when managing dependencies in CAD. To contextualize these challenges, they discussed them as dependency management across multiple granularity levels in three different ways: (1) between multiple part models and assemblies of parts, (2) between modelling operations within a single part model, and (3) between a sketch operation and multiple other part models that reference the shared sketch.

Recognizing the complexity of dependency management in CAD and the absence of universal solutions, as a starting point, our work focuses on a subset of four challenges that specifically concern the *dependencies within a single part model*: (1) difficulty in tracing dependency chains, (2) broken dependencies, (3) lack of overview of project structure, and (4) disorganized design history. As summarized in Table 1, we reinterpret these challenges as *gaps in user awareness* of operation dependencies at three different levels of abstraction:

- **Global level: Limited awareness of the overall dependency structure.** Users lack visibility into how individual modelling operations fit within and impact the overall structure of the model (Challenge 1 in Section 3.1);
- **Local level: Difficulty tracing local dependencies.** It is challenging to follow how dependencies form and propagate from specific operations within a localized scope (Challenge 2 in Section 3.2); and
- **Modular level: Lack of dependency-aware modularization support.** Existing tools do not effectively support identifying and organizing closely related operations into meaningful modular units (Challenge 3 in Section 3.3).

These challenges are particularly pronounced when navigating large, complex, or unfamiliar models. This is a commonly encountered scenario in engineering design as engineers often work on large-scale projects with multiple collaborators [52], and they often revisit legacy models to reuse part of the design after a long time since they were first built [35].

To address the three levels of challenges summarized above and discussed in detail below, we establish three design goals (DG) that guide the development of our add-in tool, *CADModelScope*:

DG1 ***Build global awareness of hidden operation dependencies.*** The tool should present the complete network of hidden dependencies across all modelling operations, and reveal the structural significance of operations based on their degrees of downstream dependencies (Challenge 1).

DG2 ***Support localized operation dependency tracing.*** The tool should assist users in tracing upstream operations to diagnose errors, and in identifying downstream operations that may be unintentionally impacted by changes (Challenge 2). The tool should also support users to anticipate and avoid dependency violations during operation reordering (Challenge 3).

DG3 ***Reveal the modular structure of modelling operations.*** The tool should help users recognize the model's high-level functional units by grouping closely interdependent operations (Challenge 1). It should also guide the reordering of operations to better reflect and preserve this modular structure (Challenge 3).

We do not propose to replace the existing operation sequence with our new tool; it is expected that the proposed solution would augment the current design workflow as an assistive add-in to a commercially available CAD software.

To motivate our envisioned usage scenarios, consider a hypothetical mechanical engineer, *Nova*, who works on mechanical design using CAD software at a medium-sized engineering firm. The company develops custom mechanical parts and frequently collaborates with external suppliers, a common practice in the industry. In Nova's daily work, they regularly engage with a high volume of complex CAD models that may be developed solely by them, co-developed with their colleagues, or received from external partners.

3.1 Challenge 1: Limited Global Awareness of Dependency Structure

CAD models of complex mechanical parts often involve long sequences of interdependent modelling operations, which users must understand to make informed design decisions. However, modern CAD systems typically present these operations in a simple linear sequence (see examples in Figure 2a(i) and c(i)), offering limited visibility to their underlying dependencies and how modifications may propagate through the model [47], like the ones visualized in Figure 2d.

Many professional CAD users and teams follow top-down design workflows, where users begin by establishing their design with one or more key operations such as master sketches or layout sketches, which act as reference points for subsequent features [3, 21, 68]. These foundational operations embed essential design intent and underpin large portions of the model, which will be realized via extensive downstream dependencies. Despite the importance of these early sketches, as shown in Figure 2, modern CAD interfaces present all operations of the same type with identical icons, without any additional information to distinguish their structural significance or dependency-related characteristics (e.g., the number of downstream dependent operations). Therefore, in current practice, users are forced to manually inspect each operation to infer which ones carry substantial downstream influence, imposing a significant cognitive burden. Especially when working with an unfamiliar model, this lack of global structural awareness makes it difficult for users to understand how different operations are interrelated and to reason about the potential consequences of their actions [18, 57, 60].

Usage scenario. When Nova is asked to modify a legacy model that they created long ago in response to a new regulation, they cannot easily recall the details of the CAD model's structure nor identify the relevant operations. Instead of manually searching through the long linear sequence of operations that were not meaningfully renamed when the model was first constructed, *CADModelScope* streamlines this navigation process by automatically grouping operations into high-level modules (**DG3**), helping Nova quickly narrow their focus to a smaller relevant set of operations in the sequence. As made apparent by **DG1**, Nova can also identify operations with many downstream dependencies (i.e., ones more likely to carry core design intent) and prioritize them for closer inspection. Overall, *CADModelScope* is expected to significantly reduce the number of operations that Nova needs to examine to understand the overall model structure.

Table 1: Mapping of key challenges in dependency management from Cheng et al.'s review [18] to our study. As we focus exclusively on dependencies between operations, only relevant challenges are mapped to a level of abstraction; a dash (-) indicates challenges that fall outside the scope of our study.

Category [18]	Challenge [18]	Dependency between			Level of abstraction
		Parts	Operations	Operations & parts	
Traceability	Difficulty in tracing dependency chains		✓	✓	Local
	Poor impact analysis	✓			-
	Broken dependencies	✓	✓	✓	Local
Navigation	Lack of overview of project structure	✓	✓	✓	Global
	Difficulty reorganizing models within the hierarchy	✓		✓	-
	Disorganized design history		✓		Modular
	Messy navigation of the master sketch			✓	-
Consistency	Ambiguous dependency freshness	✓			-
	Dependency conflicts across versions	✓		✓	-

3.2 Challenge 2: Difficulty Tracing Local Dependencies

When editing a CAD model, a user's modification of a single operation can ripple through the network of downstream dependencies, leading to errors or unintended geometric updates, as described in Section 2.1. In practice, modifying an operation may alter the topology or geometry of another operation, either in an intended or an unintended way. There is then the risk that subsequent operations that depend on the altered geometry may fail due to missing references, resulting in errors. But not all dependency-related issues result in explicit errors (see Figure 3 for example). The modification could also propagate unintended geometry due to reference changes [34]. These unintended changes are often subtle, and can result in consequential geometric changes to multiple downstream operations [15].

What exacerbates the cognitive load of users when troubleshooting these issues is that all these affected modelling operations are not necessarily positioned and visually appear to be in close proximity in the operation sequence — they are often separated by many unrelated operations (e.g., one operation that "Extrude5" in Figure 2c(ii) depends on is 10 operations apart). Yet, most CAD software offers limited support for screening only the affected operations based on dependency. As a result, in order to trace dependency issues, users are often forced to manually sift through a large number of unaffected operations [18], making operation editing an inefficient and cognitively demanding process.

Usage scenario. After Nova modifies an existing modelling operation, instead of repairing the broken model by manually tracing the failure based on error messages, *CADModelScope* allows Nova to visually trace the model's dependency chain. They can quickly identify both the upstream operations that the failed operation depends on and the downstream operations that are impacted by their edit in an unintended way (**DG2**). This focused view dramatically narrows the scope of investigation, allowing Nova to effectively diagnose issues and correct them, transforming a tedious and cognitively demanding process into a more structured and manageable task.

3.3 Challenge 3: Lack of Dependency-Aware Modularization Support

In the design of complex products, CAD modellers are typically encouraged to follow best modelling practices (e.g., modular design strategy – grouping modelling operations that relate to individual features of the product in distinct modules) to enhance maintainability and facilitate future modifications [15, 21]. Before such grouping can be created, however, related operations must be arranged in successive positions in the operation sequence. While the same CAD model can usually be constructed through multiple valid sequences of operations [63], reordering existing operations after the fact is risky and error-prone, as explained in Section 2.1. Yet to effectively convey design intent and improve model structure, reordering operations is sometimes necessary. For instance, repeated sets of operations can be effectively reused through modelling operations such as pattern and mirror to reduce dependency complexity [4], and operations that contribute to the same functional feature should ideally be grouped together to support modular reasoning [57].

However, accurately identifying all operations that are relevant to a functional unit of the model (rather than simply listing all dependent operations, as in Challenge 2), requires considerable mental bookkeeping, as these operations are often scattered throughout the operation sequence. Attempting to reorder them into modular groups can inadvertently violate implicit dependencies, leading to model regeneration failures as described in Section 3.2, a common source of frustration for CAD users [18]. In the operation sequence, an operation can be reordered to an earlier position without error if and only if it does not depend on any operations between the new position and its original position in the sequence. Yet, the dependencies that constrain this re-ordering movement are not made visible in current CAD interfaces, and users are thus forced to rely on trial-and-error to manually explore safe reordering options, making it a time-consuming and error-prone process.

Usage scenario. Consider a scenario when Nova is tasked to improve a model's maintainability and comprehensibility to support future reuse and collaboration by modularizing the operation sequence. This task involves reordering operations so that operations contributing to the same functional feature appear together and are structured into groups (or folders in some CAD software) that reflect the model's high-level design intent. *CADModelScope* significantly reduces the mental bookkeeping required by displaying a view that

clusters closely interdependent operations – offering a dependency-aware recommendation for forming modular groupings (**DG3**). Then, as Nova reorders operations, *CADModelScope* continuously visualizes and updates the relevant dependency chains, enabling them to restructure the operation sequence while maintaining a clear understanding of the underlying dependencies (**DG2**). This support enables Nova to modularize the model with confidence and efficiency, turning an error-prone and cognitively demanding task into a guided and dependency-aware design activity.

4 Prototype: *CADModelScope*

Based on the three design goals outlined in Section 3, we have designed an application for a commercially available CAD package. Our application implements three complementary graph visualizations to present the hidden dependencies of modelling operations in CAD models:

- An **Operation Overview** showing all modelling operations and their connections through dependency (**DG1**);
- A **Local Dependency View** showing a structured view of upstream and downstream operations presented for a selected operation (**DG2**); and
- A **Modular View** showing closely interdependent operations grouped in clusters (**DG3**).

Details of each of the three graph representations are presented in Sections 4.1 to 4.3, and the implementation of *CADModelScope* and its interaction with the Fusion UI is described in Section 4.4. The CAD models presented in this section are used solely for demonstration purposes, and applications of the *CADModelScope* prototype do not limit to any specific models.

4.1 Global Overview of Operation Dependency

As previously introduced in Section 2.1, parametric CAD models are constructed through a sequence of modelling operations, where every operation may reference (i.e., *depend on*) one or more geometric entities generated by upstream operations. As shown in Figure 4, we build the *Operation Overview* by representing these dependencies with a directed acyclic graph, where each graph node is a modelling operation, and each graph link indicates the dependency between two modelling operations, with the link direction representing directional dependence (reading as "X depends on Y"). Additionally, as shown in the *Operation Overview* in Figure 1, the size of all nodes is scaled by the nodes' in-degree (i.e., the number of incoming links to a node), where a bigger node visually indicates the fact that more modelling operations depend on it. Nodes that are more closely connected (i.e., operations that are more interdependent on each other) are displayed in the same colour, which will be described in greater detail in Section 4.3.

In order to implement this graph representation, we iterate through each modelling operation in the sequence to identify and map its geometric references to their parent operations. For an operation sequence with n modelling operations, every operation o_i may depend on one or more operations o_j, where $j < i \leq n$. We first query the set of geometric entities (including faces, edges, and vertices) generated by each operation o_j at the model state immediately before the creation of o_i. This model state includes all the entities that are available for reference in the definition of o_i, and we map each entity to the modelling operation that generated (or last modified) it. Any reference to these existing geometric entities establishes a dependency. This procedure is repeated for every operation o_i in the operation sequence, as geometry IDs are reassigned after the addition of every operation to accommodate the added, removed, and modified geometric entities (see Section 2.1). Overall, constructing the complete dependency graph for a CAD model requires $\frac{n(n-1)}{2}$ queries of geometric entities created by individual modelling operations.

4.2 Local View of Operation Dependency Chains

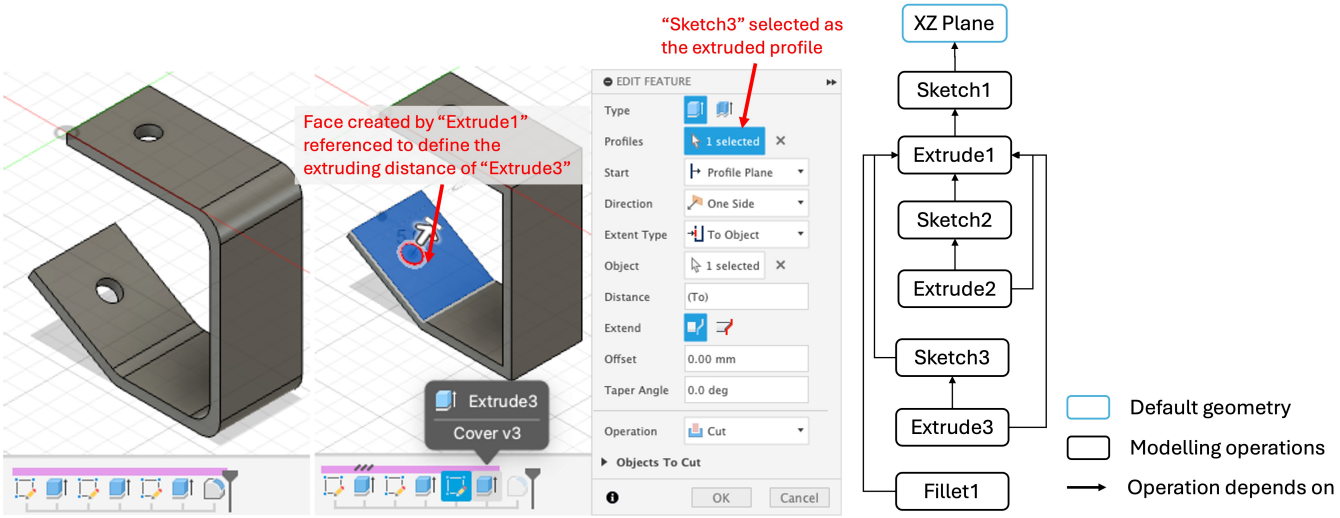
Using the complete network of operation dependencies built in Section 4.1, we further support localized dependency tracing through a *Local Dependency View*. This interactive tree view visualizes both upstream and downstream operation chains relative to a user-selected operation in the CAD model. Conceptually similar to the parents-children view in CATIA (see Figure 2d), this view provides a focused linear layout of relevant operations. Leveraging the *Operation Overview* introduced in Section 4.1, we extract the minimal relevant subset of operations directly and indirectly connected to the selected operation. As illustrated in Figure 5, upstream and downstream operations are laid out to the left and to the right of the user-selected operation, respectively. When multiple operations directly depend on the same operation concurrently, they are arranged in parallel to reflect their equivalent level in the dependency hierarchy.

When an operation o_i is selected in the *Operation Overview* and "anchored" for analysis, operations that o_i directly depends on and operations that directly depend on o_i form its one-hop neighbourhood (i.e., nodes located at a shortest distance of one link from o_i). Users can expand the view to include multi-hop neighbourhoods by controlling the depth of exploration through a slider in the *CADModelScope* UI (see Figure 1). While adjusting the hop size reveals longer dependency chains, it may also introduce additional branches (increasing the graph's height) and more intersecting links (increasing visual complexity) in the layout. Instead of enforcing a fixed or theoretically optimal hop size, or displaying all dependencies at once, we enable users to adjust the hop size (i.e., the dependency "*Depth*" shown in Figure 1) dynamically.

Although our *Local Dependency View* does not eliminate the challenge of visual clutter caused by densely intersecting links as seen in CATIA, users can always re-anchor the view on any operation to continue tracing dependencies with a lightweight exploratory workflow that balances clarity with depth. In addition, we incorporate interactive features to mitigate this issue. When users select an operation, its directly connected dependencies are highlighted to support focused analysis with reduced clutter, as shown in Figure 5b. Also, text labels retain consistent cluster-based colouring across views, reinforcing contextual awareness and coherence with the *Operation Overview* and the *Modular View* of *CADModelScope*.

4.3 Modular View of Operations in Clusters

In addition to localized dependency tracing (Section 4.2), we also offer a high-level modular view of the network of operation dependencies. Using the full operation dependency graph constructed as described in Section 4.1, we apply clustering algorithms (also



(a) A bracket designed with six operations

(b) Definition of operation "Extrude3"

(c) Graph representation of operation dependency

Figure 4: Graph representation of interdependency among modelling operations in CAD. (a) A sample bracket model created in Autodesk Fusion using six modelling operations. (b) When an operation is edited, Fusion provides visual cues in the operation sequence: the directly transformed profile is highlighted, and markers indicate upstream operations that generated referenced geometry. (c) A schematic graph illustrates this interdependency. The sequence of operations in (a) and (b) (from left to right) corresponds to the order of graph nodes in (c), from top to bottom.

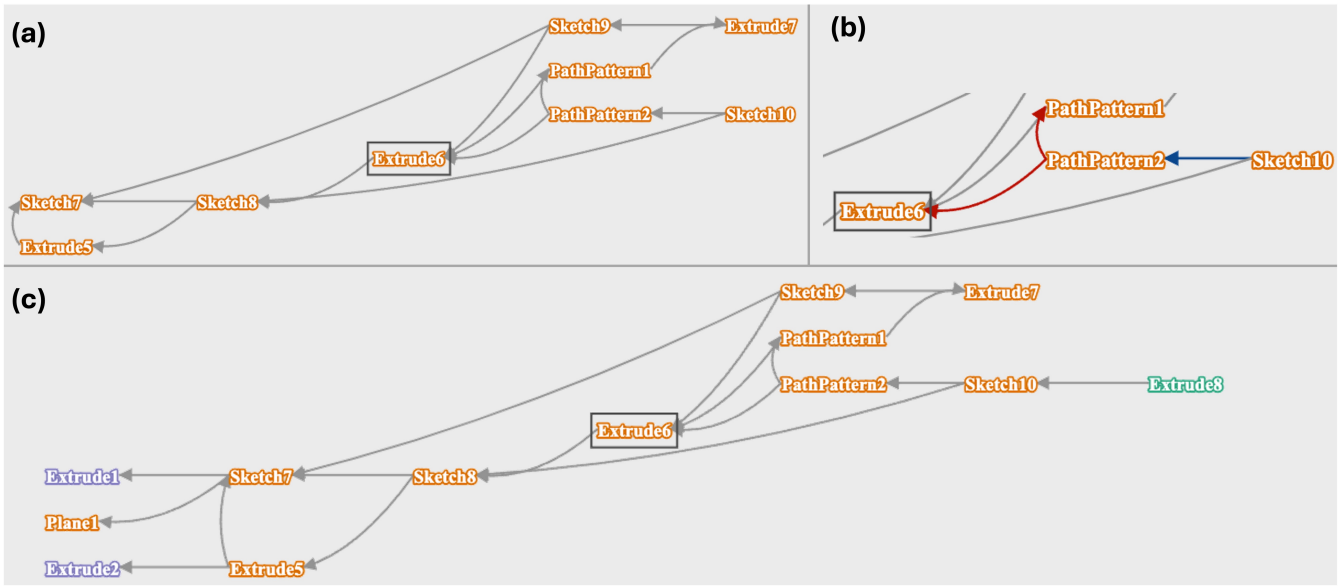


Figure 5: The *Local Dependency View* of modelling operation "Extrude6" of the model shown in Figure 6. Users can dynamically adjust the dependency depth of the view: e.g., a depth of 2 in (a) and 3 in (c). Individual operation labels may be selected to highlight in-edges (blue) and out-edges (red), as shown in (b). The black box around "Extrude6" denotes the source of dependency in the graph, and the colours of operation labels correspond to the clustering results from Section 4.3.

known as *community detection* in network analysis [2, 25]) to generate a *Modular View* of the model. Through community detection, the algorithm identifies groups of closely related graph nodes (as clusters or communities) with dense intra-cluster connections but sparse inter-cluster connections. In the context of CAD, we interpret each resulting cluster of modelling operations as a *feature* of the model. As illustrated in Figure 6, this approach allows the automatic identification of meaningful features of a CAD model based on the underlying interdependency between modelling operations. Notably, tested with the sample model shown in Figure 6, a general-purpose clustering algorithm is capable of generating clusters that closely align with the original design intent of the sample model, as annotated in a prior study [23].

To achieve automatic identification of operation clusters, we adopt a popular clustering algorithm, namely the Louvain algorithm [9]. This algorithm has been shown to be capable of performing fast, yet high-quality, community detection, even for large graphs [2]. The algorithm starts with assigning a different cluster to every graph node, and clusters are then iteratively aggregated to increase the graph *modularity*, a metric that quantifies the quality of a particular partition by comparing the density of links within clusters against links between clusters [9]. Once the modelling operations are grouped into clusters, we reconstruct the link connections between clusters. In the *Modular View*, a directed graph link is added from cluster A to B if at least one operation in cluster A depends on at least one operation in cluster B. The thickness of inter-cluster links is scaled according to the number of links between cluster members in the *Operation Overview*, while node size indicates the number of operations contained within each cluster.

Further, clustering results do not come with labels for individual clusters. As we do not assume rich semantics from user input, we do not perform keyword extraction to label clusters based on user namings of modelling operations in the CAD model. Instead, for every cluster of modelling operations, we capture a screenshot of the CAD model which highlights the geometries generated by operations within the cluster (see Figure 6), providing a contextualized view for the graph nodes. While we do not automatically extract labels from the models, users are still encouraged to rename individual clusters of nodes as part of the design comprehension process.

4.4 Implementation

We implemented *CADModelScope* as an add-in for Autodesk Fusion (V.2602.1.25), using its Python application programming interface (API).[4] We chose Fusion for our prototype implementation because of the availability of API support for scalable querying of operation dependencies, as outlined in Section 4.1, and a free research license. Through the API, any CAD model created in Fusion can be algorithmically processed, allowing us to extract both its geometry and the sequence of modelling operations for analysis. The add-in can be assessed through an icon added to Fusion's existing UI, which triggers the backend analysis of the currently opened CAD model and launches a pop-up window with the three interactive graph visualizations of the modelling operations described in earlier subsections. The pop-up window is displayed as an in-app palette,[5] implemented as an interactive HTML page that communicates with the Fusion backend API via JavaScript functions, and all graph components are visualized using the D3.js JavaScript library [11]. This setup enables real-time updates and interaction between *CADModelScope* and the CAD model in Fusion. Throughout the duration of the user session in *CADModelScope*, all information retrieved from Fusion is stored temporarily as in-memory variables, and no database or persistent storage is required.

Importantly, the three views provided by *CADModelScope* are designed to work in coordination with each other and with Fusion's native design workspace. User interactions are synchronized across all components, as shown in Figure 7. For example, when a modelling operation is selected in the operation sequence at the bottom of the Fusion's design workspace, or when a sketch or reference plane is selected in the native side-panel browser, Fusion automatically highlights the geometric entities generated by the selected operation in blue, visually indicated on the model. At the same time, nodes that are relevant to the selected operation in all three views of *CADModelScope* are also selected and highlighted. This interaction works both ways, as activities in our add-in palette also trigger backend messages to be sent to Fusion's native UI and update the selection of operations. However, if the CAD model is modified with operations added, deleted, or edited with altered dependencies, *CADModelScope* needs to be refreshed to fetch the updated data from Fusion.

5 Evaluation

To evaluate the effectiveness of *CADModelScope*, we conducted a user study with 10 participants, all with mechanical engineering backgrounds and prior CAD experience (see Section 5.1 for details). Participants were asked to complete tasks similar to the envisioned usage scenarios described in Section 3, which were designed to reflect realistic and complex design challenges. Following task completion, we collected structured feedback from the participants to better understand the tool's impact. In general, our evaluation was guided by the following three research questions (RQ):

- **RQ1** What challenges do users face when navigating modelling operations in commercial CAD software? How do these align with challenges reviewed in Section 3, and what additional challenges emerge in practice?
- **RQ2** How does *CADModelScope* address current limitations in CAD modelling? What new workflows or strategies does it enable that are not supported by existing CAD platforms?
- **RQ3** What are the trade-offs between efficiency, accuracy, and cognitive effort when using *CADModelScope* compared to relying on native features in commercial CAD software?

5.1 Participants

Through the authors' professional network, we recruited 10 participants for the user study via LinkedIn.[6] Two participants were

[4] https://help.autodesk.com/view/fusion360/ENU/?guid=GUID-A92A4B10-3781-4925-94C6-47DA85A4F65A

[5] https://help.autodesk.com/view/fusion360/ENU/?guid=GUID-6C0C8148-98D0-4DBC-A4EC-D8E03A8A3B5B

[6] https://www.linkedin.com

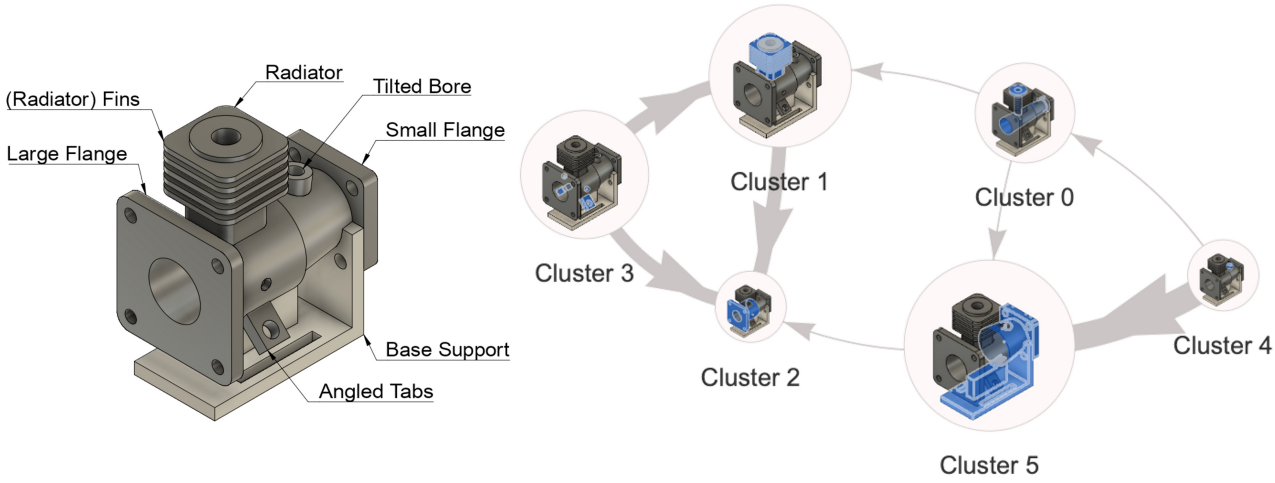


Figure 6: For a model composed of 48 operations in Fusion (left), applying the Louvain algorithm [9] yields six clusters (right). Every graph node represents a cluster of modelling operations and is labelled with a model screenshot, highlighting the geometry generated by the cluster in blue. Link thickness reflects the number of inter-cluster dependencies, and node size reflects the number of operations within each cluster. Labelling of the model features is reproduced from Ref. [23]. Background colours of graph nodes are omitted here for clarity; see Figure 1 for the implemented version.

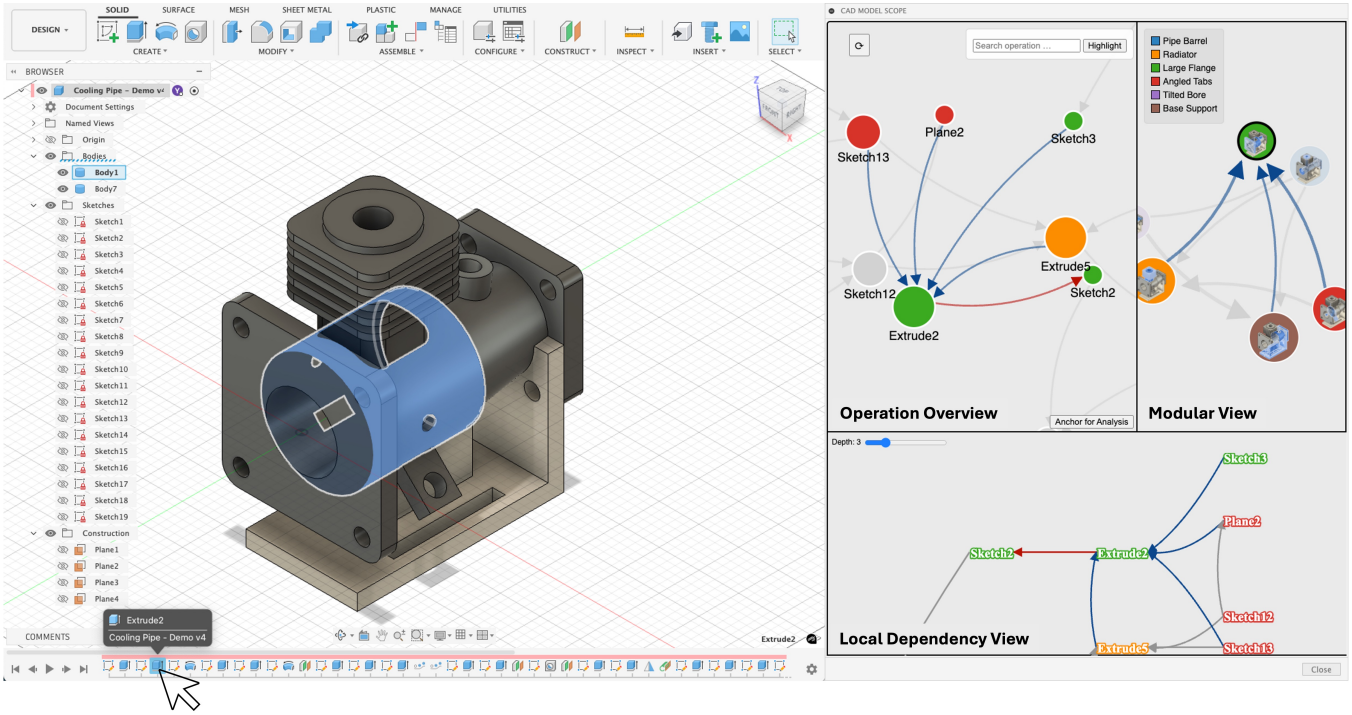


Figure 7: The *CADModelScope* interface synchronizes with the Fusion UI. When a modelling operation is selected in Fusion (as indicated by the cursor icon), the corresponding geometry is highlighted in blue on the model (left). Simultaneously, *CADModelScope* highlights the corresponding graph node in the *Operation Overview*, the cluster that contains it in *Modular View*, and its text label in the *Local Dependency View* (right).

excluded from analysis due to their observed inability to meaningfully approach Task 2 and 3 without the tool. Table 2 summarizes the background of the remaining eight participants. All participants either had an academic background in mechanical engineering or were working as a mechanical engineer in industry, with real-world project experience in CAD modelling.

As we will describe in Section 5.3, the final sample of eight participants was sufficient to address all three research questions and to reach saturation in observed behaviours. Specifically, the participants collectively demonstrated all the user challenges reviewed in Sec. 3 (**RQ1**), demonstrated a diverse range of interaction strategies and workflows when using *CADModelScope* (**RQ2**), enabled meaningful comparisons with the control conditions (**RQ3**), and provided rich qualitative feedback. While additional participants may surface further edge-case behaviours or novel strategies, such data would be unlikely to substantially alter the patterns and conclusions reported in this paper.

Table 2: Participants' background information.

ID†	Participation	CAD experience (years)	
		Fusion	Other CAD
A1	In-person	0	SolidWorks (3)
A2	Virtual	5	Inventor (4), SolidWorks (3)
A3	In-person	3	CATIA (2), SolidWorks (1)
A4	Virtual	5	Onshape (5), SolidWorks (2), Creo (1)
B1	Virtual	3	Inventor (4), Onshape (2), SolidWorks (<1)
B2	In-person	1	SolidWorks (3)
B3	In-person	0	SolidWorks (2), Onshape (2)
B4	In-person	3	None

† The first letter indicates the participant group described in Section 5.2.2.

5.2 Study Design

The study was carried out in a controlled laboratory environment using a Windows desktop computer with Autodesk Fusion installed locally. The experimental set-up consisted of a 27-inch monitor, a keyboard, and a mouse. Three out of the ten participants participated virtually through Zoom,[7] where they were asked to remote control the experimenter's computer screen on a 27-inch monitor. Throughout the study, all verbal communication between the participant and the experimenter was audio recorded, and participants' on-screen activities were recorded for later analysis. The user study lasted approximately one hour, and each participant received compensation of $15. This study was approved by our University's Research Ethics Boards. The study consisted of three main phases: (1) a brief tutorial to convey key features of both Fusion and *CADModelScope*, (2) the main task session where participants performed a series of modelling tasks, and (3) a post-study interview to gather qualitative feedback.

5.2.1 Pre-study tutorial. Before beginning the main tasks, participants received a short tutorial introducing the core features and modelling operations available in Fusion relevant to our experiment. Then, the key features of *CADModelScope* were demonstrated to the participants using a sample CAD model. This ensured that participants were adequately familiar with both the CAD environment and our tool. After the tutorial, participants were also given time to freely explore the interface, ask clarifying questions, and interact with the tutorial model to gain hands-on experience.

5.2.2 Experiment procedure. Next, participants were asked to complete three modelling tasks using two publicly available CAD models adapted for the study (see Table 3 for a summary and Section 5.3 for more details). These tasks aligned with the three envisioned usage scenarios described in Section 3, and were presented to all participants in the order listed in Table 3. Prior to the experiment, we confirmed that none of the participants had prior familiarity with the models used in the experiment. Participants were randomly assigned to one of two groups:

Group A Completed Tasks 1 and 2 with *CADModelScope*, and then completed Task 3 without it;

Group B Completed Tasks 1 and 2 without *CADModelScope*, and then used *CADModelScope* for Task 3.

This crossover design ensured that each participant served as both a control and a treatment case, while also allowing all participants to experience and provide feedback on *CADModelScope*.

Throughout the experiment, participants had unrestricted access to all features of Fusion and could ask the experiment administrator questions about either Fusion or *CADModelScope*. However, no direct guidance was given on how to solve the tasks. Based on two pilot studies, we set a 10-minute time limit per task. If participants were unable to complete a task within this allotted time, they were asked to stop and proceed to the next task.

5.2.3 Post-study interview. Following the experiment, we conducted a semi-structured interview to gather qualitative insights into participants' experiences with *CADModelScope*. The discussion was guided by the following open-ended questions:

- In your day-to-day design work, how often do you encounter scenarios similar to those in this study, and how do the study tasks compare in complexity?
- What were your overall impressions of our tool?
- Were there any specific features that you found useful or non-useful? How would you suggest improving the tool?

5.3 Results

In this section, we present our findings by first summarizing observations of the participants' task completion processes in Section 5.3.1 to 5.3.3 in relation to the three RQs. Then, we discuss general user feedback in Section 5.3.4.

5.3.1 Navigate to identify specified operations. In Task 1, participants were tasked to identify the sketch that outlines the guitar's tuning head geometry, as shown in Figure 8b and c. This target sketch was the 61st in the sequence of 191 operations, one of 69 total sketches.

*Exhaustive navigation without tool support (**RQ1**).* With Fusion's built-in features only, three Group B participants relied on browsing through every sketch sequentially or at random, with Fusion highlighting the corresponding entities as they clicked. This exhaustive search without a clear initial point of exploration proved error-prone: two overlooked the target sketch on their first pass,

[7] https://www.zoom.com

Table 3: Tasks assigned to participants for evaluation.

No.	Context	Task Description	Model	No. of Operations	No. of Bodies†
1	Overview	Identify the operation that defines a specified key dimension of the model (scenario in Section 3.1)	Guitar‡	191	64
2	Restructuring	Identify all operations that contribute to a specified model feature and reorder them to appear next to each other for modularization (scenario in Section 3.3)	Same as above		
3	Editing	After editing a specified dimension, correct all errors and unintended changes in the model without removing or adding operations and without changing the original edit (scenario in Section 3.2)	Cooling pipe [23]	50	6

† A body is a solid defined by a set of boundary surfaces that enclose a watertight volume. While a large number of operations indicates a complex modelling process, a large number of bodies indicates a complex product.
‡ A publicly available model found from the Autodesk Community Gallery: https://www.autodesk.com/community/gallery/project/77980/1940-gibson-l-00.

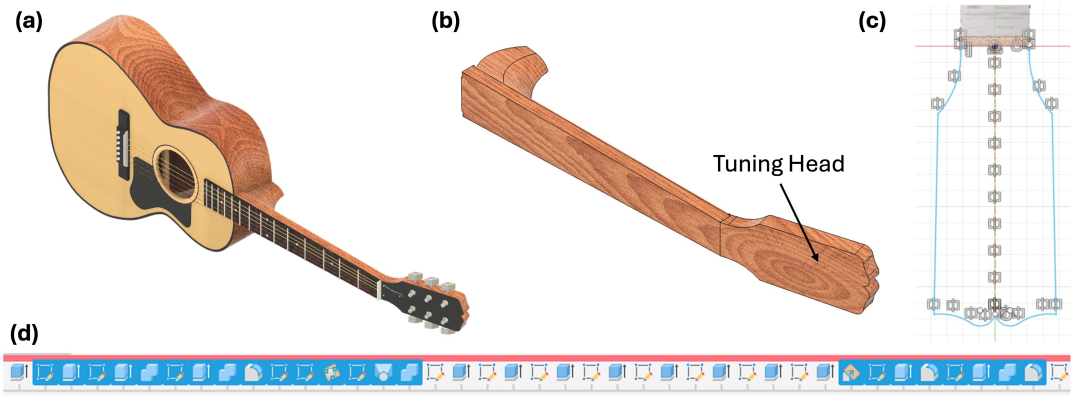


Figure 8: CAD model of a guitar used in Tasks 1 and 2. (a) Full guitar model. (b) Body renamed as the guitar's "neck", with the tuning head labelled for Task 1. (c) Target sketch participants were tasked to locate in Task 1. (d) All operations contributing to the guitar's neck highlighted for Task 2.

and visual clutter often made highlighted geometry difficult to interpret (Challenge 1 in Section 3.1). Participant B3 attempted a more targeted approach by selecting a face on the tuning head, prompting Fusion to mark the operation that last modified it, and inspecting nearby sketches in the sequence. While this method is an improvement from brute force search, it is unreliable since geometric adjacency does not necessarily align with operation order.

*Context-aware navigation with CADModelScope (**RQ2**).* With access to *CADModelScope*, three Group A participants instead used *Modular View* to narrow the search to the cluster that included the tuning head (**DG3**), then traced operation dependencies in the *Local Dependency View* (**DG2**). Notably, participants adopted a slightly different strategy in selecting the initial anchored operation: A1 chose the operation that last modified a geometric entity near the tuning head; A2 anchored a random sketch within the cluster; and A3 selected the operation with the largest node size in the *Operation Overview* (**DG1**). However, all three participants ultimately converged on similar exploration patterns, suggesting that *CADModelScope* can accommodate diverse entry points while still guiding users toward consistent and reliable results. A4 completed the task without using *CADModelScope*, adopting a strategy similar to those observed in Group B.

*Workflow comparison (**RQ3**).* These field observations suggest that *CADModelScope* can enable a more structured, context-aware navigation compared to the exhaustive sequential or random browsing typical with traditional tools. While their search was more effective, Group A participants using *CADModelScope* ($\mu = 253$ sec, $\sigma = 144$) generally took more time than Group B without *CADModelScope* ($\mu = 152$ sec, $\sigma = 93.5$) to complete the task ($U = 4.0, p = 0.343$), which might reflect the additional effort required to interpret the clustering representation. These findings suggest that *CADModelScope*'s advantages scale with model complexity, where the overhead of cluster interpretation is expected to be offset by reduced reliance on exhaustive exploration and error-prone search.

5.3.2 Restructuring operation sequence into modular units. In Task 2, participants had to identify and group the 23 operations that construct the guitar's "neck" (Figure 8b). In the original operation sequence, these operations were split into two groups, separated by unrelated operations (Figure 8d). Participants needed to reorder and merge the split operations into one contiguous group without introducing dependency-related errors.

*Manual modularization without tool support (**RQ1**).* Since Fusion lacks built-in features for post-hoc modularization, Group B participants (without access to *CADModelScope*) relied on a stepwise, entity-based search. Specifically, three participants began by selecting a geometric entity on the guitar's neck to then locate the operation that last modified it, before expanding the search outward

to nearby operations in the sequence. However, as they progressed, these participants frequently lost track of their tentative grouping range, and two participants reverted to exhaustive browsing by inspecting nearly every operation. This manual modularization process required considerable mental bookkeeping from the participants (Challenge 3 in Section 3.3). While feasible for our study, this approach would not scale to larger models with relevant operations sparsely located.

*Verification-based workflow with CADModelScope (**RQ2**).* With access to *CADModelScope*, Group A participants instead used the *Modular View* to verify pre-clustered operation groups (**DG3**). A2 and A4 began by directly inspecting the cluster nodes, whereas A1 and A3 started by selecting a relevant geometric entity and then locating the corresponding cluster node. After completing the first grouping, A2 and A3 identified the second cluster node via graph links in the *Modular View*, while A1 anchored the last operation of the first group in the *Local Dependency View* to trace the first operation of the second group (**DG2**).

*Workflow comparison (**RQ3**).* Overall, *CADModelScope* shifted the operation modularization process from an exhaustive search for relevant operations to a verification of clustering results, reducing the navigation overhead. Within the 10-minute time constraint, Group A participants using *CADModelScope* ($\mu = 0.859, \sigma = 0.0188$) achieved higher accuracy on average than Group B without *CADModelScope* ($\mu = 0.761, \sigma = 0.265$), although this difference was not statistically significant ($U = 10.5, p = 0.538$).[8] Notably, three Group A participants were misled by *CADModelScope*'s clustering result, including three irrelevant operations that were clustered with relevant ones and incorporated without sufficient verification. However, since *CADModelScope*'s clustering algorithm is dependency-aware, reordering did not introduce dependency-related errors. This finding highlights the risk of misinterpretation of unsupervised clustering results, but it also demonstrates the resilience provided by dependency-aware clustering in complex models even in the face of errors in interpretation.

5.3.3 Error resolution after operation edit. In Task 3, participants were presented with a CAD model along with a brief description of its design intent, as shown in Figure 9. Participants were then asked to correct any unintended geometric changes that occurred after a dimensional edit (labelled in Figure 9d), without adding or deleting operations and without modifying the edit itself. Unlike reference errors that trigger explicit messages, these unintended updates occurred silently, introducing no regeneration errors and thus requiring participants to first detect the problem before resolving it.

*Exhaustive playback without tool support (**RQ1**).* After visually detecting the unintended geometry by rotating the model, Group A participants, without access to *CADModelScope*, had no direct way to trace the problem's source, since Fusion only identifies the operation that last modified it (Challenge 2 in Section 3.2). As

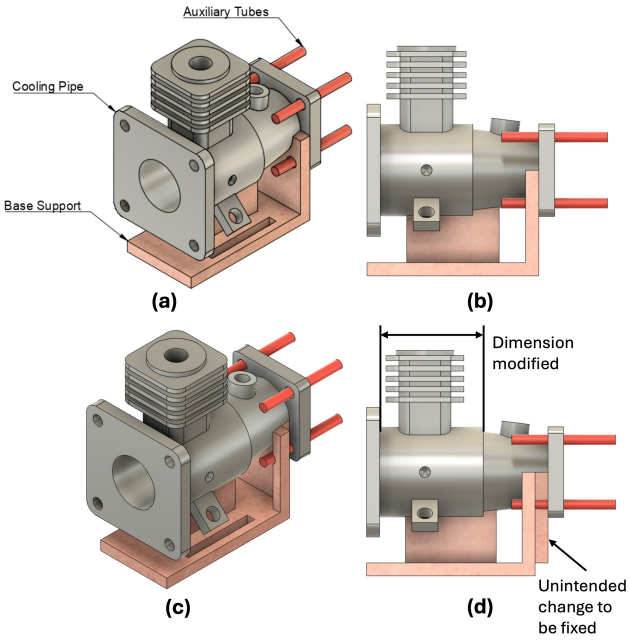


Figure 9: CAD model of a cooling pipe with a base support used in Task 3. The auxiliary tubes illustrate how holes in the support accommodate cylindrical tubes passing through the main section. (a) Orthogonal and (b) side views of the model before editing; (c) orthogonal and (d) side views after editing.

a workaround, all four Group A participants relied on step-by-step playback of the operation sequence,[9] an exhaustive process to visualize how the model changes after every operation. Three participants then misinterpreted an unrelated sketch as the source of the problem, and only two completed the task within 10 minutes.

*Targeted tracing with CADModelScope (**RQ2**).* With *CADModelScope*, participants also began with visual inspection of the model but then used the *Modular View* to narrow their focus to a subset of relevant operations (**DG3**). Within the cluster that included the unintended geometry, they anchored an operation in the *Local Dependency View* to trace upstream and downstream dependencies (**DG2**), avoiding an exhaustive review of the entire sequence. By the end of the 10-minute limit, three participants successfully completed the task.

*Workflow comparison (**RQ3**).* When resolving modelling errors in CAD, *CADModelScope* enabled a targeted, dependency-based workflow, in contrast to the exhaustive playback of the full operation sequence observed from those using native tools. In our study, completion rates were higher with access to *CADModelScope*. Notably, when both groups misinterpreted the same unrelated sketch, Group B participants could still recover by tracing dependencies locally with the support of *CADModelScope*, whereas Group A participants were forced to re-inspect other operations without *CADModelScope*.

[8]The correct final grouping should contain 23 operations, and accuracy was calculated as (23 - number of extra operations - number of missing operations) / 23.

[9]In Fusion's terminology, this corresponds to moving the "timeline marker": https://help.autodesk.com/view/fusion360/ENU/?guid=ASM-USE-TIMELINE

This advantage is expected to become increasingly important as relevant operations grow sparser within longer operation sequences.

5.3.4 General user feedback.

Realism and modelling context. Participants generally reported that the study tasks reflected realistic professional and technical challenges. While some participants noted that the models they typically encounter are less complex than those used in the study (B1, B3), others emphasized that it is common to "*end up having really long [operation sequences] so it gets a bit difficult to keep track of the different operations*" (B4). In particular, participants reported that debugging modelling errors is a routine part of modelling, especially in one's early career before developing the habit of planning ahead (A1, A4). These downstream errors often propagate to the point that "*[the CAD software] will literally not render [and] crash*" (A3), and as one participant admitted, they "*probably don't always fix it the best way*" (B1).

Positive reception of CADModelScope features. Overall, participants responded positively to *CADModelScope*. The *Modular View* was the most frequently described as a useful and intuitive starting point for navigation (A1, A3, B3, B4), particularly compared to "*manually going through every single [operation]*" (A3). The *Local Dependency View* was valued for tracing dependencies backward (A3), and interactive features such as synchronized highlighting across views in *CADModelScope* and Fusion's native UI further supported navigation (A2). Importantly, participants reported that the tool raised their awareness of the hidden operation dependencies they would otherwise overlook when modelling (A3, B3).

Suggestions for refinement. Participants also offered constructive suggestions for improving *CADModelScope*. A recurring concern was that the tool occupied too much screen space from the modelling workspace, where several participants recommended a toggle or collapsible interface, especially for smaller displays (A2, A3, A4). Other usability improvements suggested included listing clusters by their first appearance in the operation sequence (A3), enabling renaming of operations directly in *CADModelScope* (B4), adjusting the relative size of views (B2), and preserving custom cluster names after updates (A3). While not yet implemented in the current prototype, these refinements are all technically feasible. Finally, A4 and B4 suggested that *CADModelScope* is more valuable for large and complex models with many bodies and dependencies, but offers less benefits for small models. At the same time, B2 anticipated that "*it's helpful definitely, but it takes practice*".

6 Discussion

In this section, we first outline design guidelines and implications derived from our findings to inform future tool development (Section 6.1). Building on the positive feedback from our study, we then explore the potential of extending this currently underutilized, yet valuable, approach of representing parametric CAD models through their underlying operation dependencies used in *CADModelScope* for broader applications (Section 6.2), and conclude by reflecting on the limitations of our study (Section 6.3).

6.1 Design Guidelines and Implications

Awareness of the underlying model representation. While parametric CAD offers powerful capabilities for creating complex products, commercial CAD systems provide limited support to help users understand the underlying model representation (i.e., the dependencies among modelling operations). Indeed our participants reported a lack of *awareness* of the model's hidden structures which often remain obscured in the linear sequential presentation of operations. Since a sequential list does not capture the complexity of interdependencies, *CADModelScope* visualizes and juxtaposes related operations, supporting users to navigate relevant information more effectively. This parallels work in HCI on increasing the discoverability of hidden structures to support debugging in both CAD [34] and programming environments [39], and this also highlights how making hidden mechanisms visible is a critical foundation for future innovation.

In collaborative design, CAD models serve as *boundary objects* that embed and mediate shared representations across perspectives [27, 62]. By making operation dependencies visible, *CADModelScope* enhances the communicative power of CAD as a boundary object by presenting the underlying design intent of a CAD model that is otherwise implicit. Explicitly visualizing the design intent as a network of operation interdependencies helps users reconstruct how the model was originally conceived and anticipate how it will be impacted by subsequent modifications [28, 51]. This constructed understanding contributes to a shared mental model of the design, whether across multiple collaborators or by the same individual over time, which can improve collaboration quality [48] and better support decision-making [5].

Clustering as context for information foraging. As observed in our user study, the capability of *CADModelScope* to cluster closely interdependent operations into groups was particularly useful and well received by participants. From the perspective of information foraging theory, the *Modular View* improves *information profitability* – the ratio of valuable information gained to the effort required to obtain it [53, 54]. In the context of CAD, rather than scanning a long linear sequence of similar-looking operations, the *Modular View* in *CADModelScope* enables users to navigate cluster nodes that correspond to functional units of the model. By labelling each cluster with a screenshot of the geometry it produces, the *Modular View* reduces the cost of identifying where relevant operations reside. This shifts the search space from potentially hundreds of indistinguishable operations to a handful of meaningful cluster nodes, enabling users to focus their navigation effort more strategically. In doing so, the *Modular View* increases the efficiency of information foraging and supports more effective navigation in complex CAD models.

Integration, interactivity, and customization as key design principles. Participants in our study emphasized that they valued *CADModelScope* for its seamless *integration* with Fusion's native interface and its interactive features. For advanced applications such as CAD, close integration with users' known CAD system is critical, as it synchronizes new capabilities with existing tools from the CAD system and reduces the friction of transitioning between the two interfaces [34]. At the same time, *interactivity* provides users with

a stronger sense of control [44], which was especially important given that operation interdependencies are otherwise hidden in Fusion.

However, our evaluation revealed a perceived lack of *customizability* of *CADModelScope*. In complex CAD-related tasks, users often develop individualized modelling strategies and interpretations of design intent [23], making it difficult for a single default visualization to meet all needs. For example, users may differ in how they prefer operations to be clustered, a discrepancy also observed in software engineering contexts [69]. While the current prototype does not enable users to edit the clustering results, future development should improve the flexibility in adjusting the *Modular View*. Furthermore, customization extends beyond interface flexibility to include tailoring assistance to different levels of user expertise. For instance, novice users who often struggle with help-seeking in CAD [38] may benefit from more proactive guidance, while experienced users may prefer more lightweight, on-demand assistance that does not interrupt their workflow. Echoing prior work [34], our participants expressed that customizable, on-demand assistance would make the tool more broadly useful.

6.2 Operation Dependency as an Effective Representation of Parametric CAD Models

Through the implementation of *CADModelScope* and our evaluation study, we demonstrated that representing a parametric CAD model as a dependency graph of its constituent modelling operations makes hidden design structures visible. Beyond navigation, this representation opens the potential to apply the same idea to support other CAD-related applications. By foregrounding these interdependencies, *CADModelScope* highlights how HCI-informed visualizations can transform how users interact with complex parametric models.

Documentation generation. The operation dependency graph, which encodes the underlying structure of CAD models, offers a foundation to facilitate the generation of model documentation that conveys a model's function and design intent. Such graphs not only help unfamiliar users to navigate the operation sequence, but they also assist model authors in structuring documentation around clusters of related operations. In the absence of well-maintained documentation, users often rely on reopening the CAD file and manually inspecting the operation sequence and resulting geometry. This approach becomes inefficient for large or complex models, due to long loading time and the cognitive burden of interpreting all the design details [37]. In software engineering, researchers have shown that documentation can be more effectively generated by analyzing graph-based representations of code structure instead of solely processing sequential source code [1, 24, 29, 42]. A similar principle applies in CAD, where prior work demonstrated that extracting low-level design intent from the network of interdependent operations can support the automated generation of meaningful documentation with predefined templates, but these approaches often lack interpretability and flexibility for end users [19, 45, 51]. Our *Modular View* in *CADModelScope* can build on this idea by identifying and grouping closely interdependent operations, providing a high-level abstraction of the model's structure. These clusters can form the basis for concise yet comprehensive documentation, capturing the key functional features embedded in the model while offering greater visibility and control in the documentation generation process.

Semantic version comparison. Modern CAD platforms with version control typically offer comparisons based on the operation sequence, contrasting individual parameters, geometric differences, or a combination of both [12, 70]. However, these comparisons often lack contextual insights into how changes affect the model as a whole. Without effective change summarization, the current best practice of manually reviewing all edits, every time a version is revisited, is tedious and inefficient [17], especially when resolving differences across branched versions as they are merged [12, 26]. By contrast, software engineering has long leveraged *semantic* version comparison, which emphasizes the functional implications of code changes rather than surface-level textual difference [36]. This is often achieved through graph-based analysis of code dependency and hierarchy [6, 55, 69]. Similar graph-based analyses have also been implemented to manage and compare design versions of artifacts that can be represented as a graph of design components [66, 67]. Inspired by these approaches, *CADModelScope*'s graph-based representation of operation dependencies can similarly enable semantic version comparison in CAD, as illustrated in Figure 10. Instead of focusing solely on parameter or geometric differences that may be captured in an *Operation Overview*, the *Modular View* in *CADModelScope* further contextualizes the version differences at the level of functional features, helping users reason about the broader design impact of operation edits.

Model retrieval. Another relevant line of research has focused on the effective querying and retrieval of CAD models from databases to support design reuse. Prior work has attempted to infer design intent and construct semantic representations from modelling operation parameters [7], the network of constraints between assembly components [31], and the model geometry and topology [43, 64]. While these approaches have shown promising results in retrieving similar models from sample user queries, they can lack transparency in how retrieval decisions are made, offering limited interpretability for the end users. Building on our work in this paper, the interactive visualization included in *CADModelScope* helps users explore and understand networks of interdependency within a CAD model. This interpretive layer can be further leveraged to align the retrieval process with the underlying model semantics, enhancing both precision and usability.

6.3 Limitations and Future Work

This paper presents an exploratory study toward visualizing the hidden network of interdependency within a CAD model, and demonstrates how such visualization can enhance design efficiency, particularly when navigating complex and unfamiliar models. While the current work establishes a promising foundation, it has several limitations that offer opportunities for future expansion of the prototyped *CADModelScope*.

Assembly associativity. This study focused exclusively on part modelling. However, real-world CAD workflows also involve assemblies composed of multiple parts [61]. In assemblies, parts are connected through joints or mates that constrain their relative

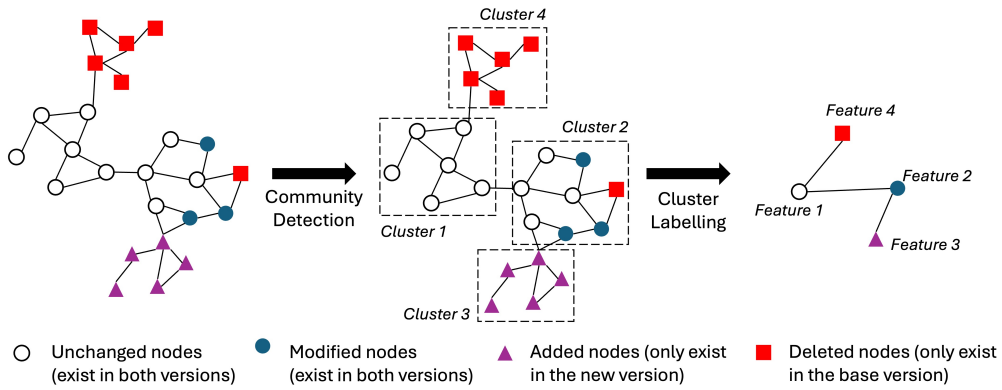


Figure 10: A schematic graph-based analysis for semantic version comparison of a CAD model. In the left and middle graphs, each node represents a modelling operation, and each link represents the dependency relation between nodes. The right graph presents a concise summary of the comparison results.

motion. Although the assembling process differs from part modelling, the associativity formed among components can also be represented as a network of constraints. We hypothesize that *CADModelScope*'s graph-based analysis approach could be extended to visualize assemblies by treating these constraints as links in a network, and performing community detection on such a network could reveal the modular structure of an assembly, supporting user comprehension and navigation of the model. Future work should explore adapting *CADModelScope* to assemblies and empirically evaluating its effectiveness through new user studies.

Inclusion of user-generated semantics. The current implementation of *CADModelScope* generates graph representations exclusively from parametric modelling operations and geometry, without incorporating the semantic information that users sometimes embed during the design process. For example, users may group operations into folders or rename operations to convey design intent. Such semantics could be integrated into the graph structure by introducing weighted links between grouped operations or by leveraging renamed operations for automatic labelling of clusters via keyword extraction. The integration of user-provided metadata can potentially lead to clusters that are more intuitive and interpretable, and advanced semantic analysis techniques may even reveal latent relationships among operations, further enriching the clustering results. Prior work has shown that the incorporation of semantics is effective for expressing design intent in CAD modelling [20, 33], suggesting a promising direction for *CADModelScope*.

Limitations in lab study. In our user study, we evaluated *CADModelScope* with a modest participant sample using publicly available online models, which introduces several limitations. The complexity of these models was constrained by both the limited availability of highly complex models online and the challenge of creating evaluation tasks that realistically reflect professional pain points. Future work could draw on commercial product designs of greater complexity. Participants were confronted with unfamiliar CAD models. While encountering unseen models is a common experience in practice, future studies should also investigate how *CADModelScope* may function when users engage with familiar models where prior knowledge shapes interpretation. Additionally, our tasks were intentionally explicit and isolated to enable clear comparisons, whereas real-world design iterations rarely come with explicit instructions [65] and often involve multiple instances of the challenges we identified. A project-based or longitudinal study could better capture *CADModelScope*'s value in more realistic and iterative workflows. Finally, larger-scale user studies may improve the reliability and generalizability of our findings, surface additional user behaviours, and enable statistical assessments of usability (e.g., readability [49, 50], cognitive load [40], ease of operation [32]) to complement our formative results.

7 Conclusion

In this work, we addressed the challenge of navigating complex CAD models by visualizing the hidden interdependencies among modelling operations. We began with an analysis of how operation dependencies shape critical tasks such as navigation, modularization, and debugging of operations in CAD, as well as the difficulties users face when these dependencies remain concealed in modern commercial CAD systems. Building on this understanding, we developed *CADModelScope*, a multi-level graph-based visualization of operation dependencies that is integrated into Autodesk Fusion. Through a qualitative lab study, we examined how participants located and interpreted operations with and without *CADModelScope*, and how our tool fostered greater awareness of hidden model structures to support more structured navigation workflows. Our findings contribute implications for the design of future CAD support tools, and more broadly, highlight the importance of making underlying system mechanisms visible to support sense-making, collaboration, and effective use of complex software systems.

Acknowledgments

We acknowledge the support of the Government of Canada's New Frontiers in Research Fund (NFRF) [NFRFE-2022-00543] and the Centre for Analytics and Artificial Intelligence Engineering (CARTE) at the University of Toronto.

References

[1] Uri Alon, Shaked Brody, Omer Levy, and Eran Yahav. 2019. code2seq: Generating Sequences from Structured Representations of Code. In *7th International Conference on Learning Representations (ICLR)*. ICLR, New Orleans, LA, USA, 1–22. https://openreview.net/forum?id=H1gKYo09tX

[2] Samin Aref and Mahdi Mostajabdaveh. 2024. Analyzing modularity maximization in approximation, heuristic, and graph neural network algorithms for community detection. *Journal of Computational Science* 78 (June 2024), 102283. https://doi.org/10.1016/j.jocs.2024.102283

[3] Chukwuma M. Asuzu, Kathy Cheng, and Alison Olechowski. 2024. The Personas of Cloud CAD Collaboration: A Case Study of a Team of CAD Professionals. *IEEE Transactions on Engineering Management* 71 (2024), 11225–11237. https://doi.org/10.1109/TEM.2024.3409178

[4] Jing Bai, Haonan Luo, and Feiwei Qin. 2016. Design pattern modeling and extraction for CAD models. *Advances in Engineering Software* 93 (March 2016), 30–43. https://doi.org/10.1016/j.advengsoft.2015.12.005

[5] Gagan Bansal, Besmira Nushi, Ece Kamar, Walter S. Lasecki, Daniel S. Weld, and Eric Horvitz. 2019. Beyond Accuracy: The Role of Mental Models in Human-AI Team Performance. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* 7 (Oct. 2019), 2–11. https://doi.org/10.1609/hcomp.v7i1.5285

[6] Mike Barnett, Christian Bird, Joao Brunet, and Shuvendu K. Lahiri. 2015. Helping Developers Help Themselves: Automatic Decomposition of Code Review Changesets. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. IEEE, Florence, Italy, 134–144. https://doi.org/10.1109/ICSE.2015.35

[7] David Baxter, James Gao, Keith Case, Jenny Harding, Bob Young, Sean Cochrane, and Shilpa Dani. 2007. An engineering design knowledge reuse methodology using process modelling. *Research in Engineering Design* 18, 1 (May 2007), 37–48. https://doi.org/10.1007/s00163-007-0028-8 Publisher: Springer Science and Business Media LLC.

[8] Sreeram Bhaskara. 2011. Analysis and Visualization of Complex Computer Aided Design Models as a Design Structure Matrix. In *13th International Dependency and Structure Modelling Conference, DSM'11*. The Design Society, Cambridge, 61–75.

[9] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (Oct. 2008), P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008

[10] Yannick Bodein, Bertrand Rose, and Emmanuel Caillaud. 2014. Explicit reference modeling methodology in parametric CAD system. *Computers in Industry* 65, 1 (Jan. 2014), 136–147. https://doi.org/10.1016/j.compind.2013.08.004

[11] M. Bostock, V. Ogievetsky, and J. Heer. 2011. D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (Dec. 2011), 2301–2309. https://doi.org/10.1109/TVCG.2011.185

[12] Matthieu Bricogne, Louis Rivest, Nadège Troussier, and Benoît Eynard. 2012. Towards PLM for Mechatronics System Design Using Concurrent Software Versioning Principles. In *Product Lifecycle Management. Towards Knowledge-Rich Enterprises*, Louis Rivest, Abdelaziz Bouras, and Borhen Louhichi (Eds.). Vol. 388. Springer Berlin Heidelberg, Berlin, Heidelberg, 339–348. https://doi.org/10.1007/978-3-642-35758-9_30 Series Title: IFIP Advances in Information and Communication Technology.

[13] Willem F. Bronsvoort, Rafael Bidarra, and Alex Noort. 2002. Feature Model Visualization. *Computer Graphics Forum* 21, 4 (Dec. 2002), 661–673. https://doi.org/10.1111/1467-8659.00624

[14] Jorge D. Camba and Manuel Contero. 2015. Improved Representation of Dependencies in Feature-based Parametric CAD Models using Acyclic Digraphs. In *Proceedings of the 10th International Conference on Computer Graphics Theory and Applications*. SCITEPRESS - Science and and Technology Publications, Berlin, Germany, 16–25. https://doi.org/10.5220/0005261500160025

[15] Jorge D. Camba, Manuel Contero, and Pedro Company. 2016. Parametric CAD modeling: An analysis of strategies for design reusability. *Computer-Aided Design* 74 (May 2016), 18–31. https://doi.org/10.1016/j.cad.2016.01.003 Publisher: Elsevier Ltd.

[16] Dan Cascaval, Rastislav Bodik, and Adriana Schulz. 2023. A Lineage-Based Referencing DSL for Computer-Aided Design. *Proceedings of the ACM on Programming Languages* 7, PLDI (June 2023), 76–99. https://doi.org/10.1145/3591223

[17] Kathy Cheng, Michal K. Davis, Xiyue Zhang, Shurui Zhou, and Alison Olechowski. 2023. In the Age of Collaboration, the Computer-Aided Design Ecosystem is Behind: An Interview Study of Distributed CAD Practice. *Proceedings of the ACM on Human-Computer Interaction* 7, CSCW1 (April 2023), 1–29. https://doi.org/10.1145/3579613

[18] Kathy Cheng, Alison Olechowski, and Shurui Zhou. 2025. It's a Complete Haystack: Understanding Dependency Management Needs in Computer-Aided Design. *Proceedings of the ACM on Human-Computer Interaction* 9, 7 (Oct. 2025), 1–32. https://doi.org/10.1145/3757617

[19] Yuan Cheng, Fazhi He, Xiao Lv, and Weiwei Cai. 2019. On the role of generating textual description for design intent communication in feature-based 3D collaborative design. *Advanced Engineering Informatics* 39, December 2018 (2019), 331–346. https://doi.org/10.1016/j.aei.2019.02.003 Publisher: Elsevier.

[20] Jiin Choi, Seung Won Lee, and Kyung Hoon Hyun. 2025. GenPara: Enhancing the 3D Design Editing Process by Inferring Users' Regions of Interest with Text-Conditional Shape Parameters. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–21. https://doi.org/10.1145/3706598.3713502

[21] Dexin Chu, Xuening Chu, Yupeng Li, Guolin Lyu, and Deyi Xue. 2016. A multi-skeleton modelling approach based on top-down design and modular product design for development of complex product layouts. *Journal of Engineering Design* 27, 10 (Oct. 2016), 725–750. https://doi.org/10.1080/09544828.2016.1227428 Publisher: Informa UK Limited.

[22] Manuel Contero, David Pérez-López, Pedro Company, and Jorge D. Camba. 2023. A quantitative analysis of parametric CAD model complexity and its relationship to perceived modeling complexity. *Advanced Engineering Informatics* 56 (April 2023), 101970. https://doi.org/10.1016/j.aei.2023.101970

[23] Yuanzhe Deng, James Chen, and Alison Olechowski. 2024. What Sets Proficient and Expert Users Apart? Results of a Computer-Aided Design Experiment. *Journal of Mechanical Design* 146, 1 (Jan. 2024), 1–39. https://doi.org/10.1115/1.4063360 Publisher: ASME International.

[24] Patrick Fernandes, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Structured Neural Summarization. In *International Conference on Learning Representations*. ICLR, New Orleans, Louisiana, USA, 1–18. https://openreview.net/forum?id=H1ersoRqtm

[25] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3-5 (Feb. 2010), 75–174. https://doi.org/10.1016/j.physrep.2009.11.002

[26] Jessie Frazelle. 2021. A New Era for Mechanical CAD: Time to move forward from decades-old design. *Queue* 19, 2 (April 2021), 5–16. https://doi.org/10.1145/3466132.3469844

[27] Uri Gal, Kalle Lyytinen, and Youngjin Yoo. 2008. The dynamics of IT boundary objects, information infrastructures, and organisational identities: the introduction of 3D modelling technologies into the architecture, engineering, and construction industry. *European Journal of Information Systems* 17, 3 (June 2008), 290–304. https://doi.org/10.1057/ejis.2008.13 Publisher: Informa UK Limited.

[28] Rajaram Ganeshan, James Garrett, and Susan Finger. 1994. A framework for representing design intent. *Design Studies* 15, 1 (Jan. 1994), 59–84. https://doi.org/10.1016/0142-694x(94)90039-6 Publisher: Elsevier BV.

[29] Shuzheng Gao, Cuiyun Gao, Yulan He, Jichuan Zeng, Lunyiu Nie, Xin Xia, and Michael Lyu. 2023. Code Structure–Guided Transformer for Source Code Summarization. *ACM Transactions on Software Engineering and Methodology* 32, 1 (Jan. 2023), 1–32. https://doi.org/10.1145/3522674

[30] David A. Gebala and Steven D. Eppinger. 1991. Methods for Analyzing Design Procedures. In *3rd International Conference on Design Theory and Methodology*. American Society of Mechanical Engineers, Miami, Florida, USA, 227–233. https://doi.org/10.1115/DETC1991-0052

[31] Zhoupeng Han, Rong Mo, Haicheng Yang, and Li Hao. 2018. CAD assembly model retrieval based on multi-source semantics information and weighted bipartite graph. *Computers in Industry* 96 (April 2018), 54–65. https://doi.org/10.1016/j.compind.2018.01.003

[32] Traci J. Hess, Anna L. McNab, and K. Asli Basoglu. 2014. Reliability Generalization of Perceived Ease of Use, Perceived Usefulness, and Behavioral Intentions1. *MIS Quarterly* 38, 1 (March 2014), 1–28. https://doi.org/10.25300/MISQ/2014/38.1.01

[33] Megan Hofmann, Gabriella Hann, Scott E. Hudson, and Jennifer Mankoff. 2018. Greater than the Sum of its PARTs: Expressing and Reusing Design Intent in 3D Models. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Montreal QC Canada, 1–12. https://doi.org/10.1145/3173574.3173875

[34] Felix Hähnlein, Gilbert Bernstein, and Adriana Schulz. 2024. Understanding and Supporting Debugging Workflows in CAD. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. ACM, Pittsburgh PA USA, 1–14. https://doi.org/10.1145/3654777.3676353

[35] Ganeshram R. Iyer, John J. Mills, Sharon Barber, Venkat Devarajan, and Saurabh Maitra. 2006. Using a Context-based Inference Approach to Capture Design Intent from Legacy CAD. *Computer-Aided Design and Applications* 3, 1-4 (Jan. 2006), 269–278. https://doi.org/10.1080/16864360.2006.10738464

[36] Daniel Jackson and David Ladd. 1994. Semantic Diff: a tool for summarizing the effects of modifications. In *Proceedings International Conference on Software Maintenance ICSM-94*. IEEE Comput. Soc. Press, Victoria, BC, Canada, 243–252. https://doi.org/10.1109/ICSM.1994.336770

[37] D.J. Kasik, W. Buxton, and D.R. Ferguson. 2005. Ten CAD Challenges. *IEEE Computer Graphics and Applications* 25, 2 (March 2005), 81–92. https://doi.org/10.1109/MCG.2005.48

[38] Kimia Kiani, George Cui, Andrea Bunt, Joanna McGrenere, and Parmit K. Chilana. 2019. Beyond "One-Size-Fits-All": Understanding the Diversity in How Software Newcomers Discover and Make Use of Help Resources. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, Glasgow Scotland Uk, 1–14. https://doi.org/10.1145/3290605.3300570

[39] Amy J. Ko and Brad A. Myers. 2004. Designing the whyline: a debugging interface for asking questions about program behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Vienna Austria, 151–158. https://doi.org/10.1145/985692.985712

[40] Thomas Kosch, Jakob Karolus, Johannes Zagermann, Harald Reiterer, Albrecht Schmidt, and Paweł W. Woźniak. 2023. A Survey on Measuring Cognitive Workload in Human-Computer Interaction. *Comput. Surveys* 55, 13s (Dec. 2023), 1–39. https://doi.org/10.1145/3582272 Publisher: Association for Computing Machinery (ACM).

[41] Karine Kozlova, Roham M. Sheikholeslami, Lyn Bartram, and Robert Woodbury. 2011. Graph visualization in computer-aided design: An exploration of alternative representations for GenerativeComponentsTM Symbolic View. In *CAADRIA proceedings*. CAADRIA, Hong Kong, 133–142. https://doi.org/10.52842/conf.caadria.2011.133 ISSN: 2710-4265.

[42] Alexander LeClair, Sakib Haque, Lingfei Wu, and Collin McMillan. 2020. Improved Code Summarization via a Graph Neural Network. In *Proceedings of the 28th International Conference on Program Comprehension*. ACM, Seoul Republic of Korea, 184–195. https://doi.org/10.1145/3387904.3389268

[43] Ming Li, Frank C. Langbein, and Ralph R. Martin. 2010. Detecting design intent in approximate CAD models using symmetry. *Computer-Aided Design* 42, 3 (March 2010), 183–201. https://doi.org/10.1016/j.cad.2009.10.001

[44] Benedikt Loepp, Katja Herrmanny, and Jürgen Ziegler. 2015. Blended Recommending: Integrating Interactive Information Filtering and Algorithmic Recommender Techniques. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, Seoul Republic of Korea, 975–984. https://doi.org/10.1145/2702123.2702496

[45] Qinyi Ma, Lihua Song, Peng Xue, Dapeng Xie, Maojun Zhou, and Junli Shi. 2018. A CAD Model Retrieval System Based on Design Intent. In *Advances in Mechanical Design*, Jianrong Tan, Feng Gao, and Changle Xiang (Eds.). Vol. 55. Springer Singapore, Singapore, 691–706. https://doi.org/10.1007/978-981-10-6553-8_46 Series Title: Mechanisms and Machine Science.

[46] Maxim Marchenko, Bernd-Arno Behrens, Gregor Wrobel, Robert Scheffler, and Matthias Pleßow. 2011. A New Method of Visualization and Documentation of Parametric Information of 3D CAD Models. *Computer-Aided Design and Applications* 8, 3 (Jan. 2011), 435–448. https://doi.org/10.3722/cadaps.2011.435-448

[47] Mahmoud Masmoudi, Patrice Leclaire, Marc Zolghadri, and Mohamed Haddar. 2015. Dependency identification for engineering change management (ECM): an example of computer-aided design (CAD)-based approach. In *Proceedings of the 20th International Conference on Engineering Design (ICED 15) (ICED, Vol. 3)*. The Design Society, Milan, Italy, 199–208.

[48] John E. Mathieu, Tonia S. Heffner, Gerald F. Goodwin, Eduardo Salas, and Janis A. Cannon-Bowers. 2000. The influence of shared mental models on team process and performance. *Journal of Applied Psychology* 85, 2 (April 2000), 273–283. https://doi.org/10.1037/0021-9010.85.2.273

[49] Aliaksei Miniukovich, Antonella De Angeli, Simone Sulpizio, and Paola Venuti. 2017. Design Guidelines for Web Readability. In *Proceedings of the 2017 Conference on Designing Interactive Systems*. ACM, Edinburgh United Kingdom, 285–296. https://doi.org/10.1145/3064663.3064711

[50] Aliaksei Miniukovich, Michele Scaltritti, Simone Sulpizio, and Antonella De Angeli. 2019. Guideline-Based Evaluation of Web Readability. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, Glasgow Scotland Uk, 1–12. https://doi.org/10.1145/3290605.3300738

[51] Jeffrey Otey, Pedro Company, Manuel Contero, and Jorge D. Camba. 2018. Revisiting the design intent concept in the context of mechanical CAD education. *Computer-Aided Design and Applications* 15, 1 (Jan. 2018), 47–60. https://doi.org/10.1080/16864360.2017.1353733

[52] David Xiaosong Peng, Gregory R. Heim, and Debasish N. Mallick. 2014. Collaborative product development: The effect of project complexity on the use of information technology tools and new product development practices. *Production and Operations Management* 23, 8 (2014), 1421–1438. https://doi.org/10.1111/j.1937-5956.2012.01383.x Publisher: Wiley-Blackwell.

[53] Peter Pirolli and Stuart Card. 1995. Information foraging in information access environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '95*. ACM Press, Denver, Colorado, United States, 51–58. https://doi.org/10.1145/223904.223911

[54] Peter Pirolli and Stuart Card. 1999. Information foraging. *Psychological Review* 106, 4 (Oct. 1999), 643–675. https://doi.org/10.1037/0033-295X.106.4.643

[55] Eduardo Rinaldi, Davide Sforza, and Fabio Pellacini. 2023. *NodeGit* : Diffing and Merging Node Graphs. *ACM Transactions on Graphics* 42, 6 (Dec. 2023), 1–12. https://doi.org/10.1145/3618343

[56] D. Robertson and T.J. Allen. 1993. CAD system use and engineering performance. *IEEE Transactions on Engineering Management* 40, 3 (1993), 274–282. https://doi.org/10.1109/17.233189

[57] P. Rosso, J. Gopsill, S. C. Burgess, and B. Hicks. 2022. Does CAD Smell Like Code? A Mapping Between Violation of Object Oriented Programming Design Principles and Computer Aided Design Modelling. *Proceedings of the Design Society* 2 (May 2022), 1737–1746. https://doi.org/10.1017/pds.2022.176 Publisher: Cambridge University Press (CUP).

[58] Anthony Rynne and William Gaughran. 2007. Cognitive Modelling Strategies For Optimum Design Intent In Parametric Modelling (Pm).. In *2007 Annual Conference & Exposition Proceedings*. ASEE Conferences, Honolulu, 12.366.1–12.366.15. https://doi.org/10.18260/1-2--2651 ISSN: 2153-5965.

[59] Salehi Salehi and Chris McMahon. 2009. Development of a generic integrated approach for parametric associative CAD systems. In *Proceedings of ICED 09, the 17th International Conference on Engineering Design (ICED, Vol. 5)*. The Design Society, Palo Alto, CA, USA, 145–156.

[60] Vahid Salehi and Chris McMahon. 2009. Action research into the use of parametric associative CAD systems in an industrial context. In *Proceedings of ICED 09, the 17th International Conference on Engineering Design (ICED, Vol. 5)*. The Design Society, Palo Alto, CA, USA, 133–144.

[61] Rainer Stark. 2022. Major Technology 5: Product Data Management and Bill of Materials—PDM/BOM. In *Virtual Product Creation in Industry*. Springer Berlin Heidelberg, Berlin, Heidelberg, 223–272. https://doi.org/10.1007/978-3-662-64301-3_11

[62] Eswaran Subrahmanian, Ira Monarch, Suresh Konda, Helen Granger, Russ Milliken, Arthur Westerberg, and Then-dim Group. 2003. Boundary Objects and Prototypes at the Interfaces of Engineering Design. *Computer Supported Cooperative Work (CSCW)* 12, 2 (June 2003), 185–203. https://doi.org/10.1023/a:1023976111188 Publisher: Springer Science and Business Media LLC.

[63] Tom Veuskens, Raf Ramakers, Danny Leen, and Kris Luyten. 2023. History in Motion: Interactive 3D Animated Visualizations for Understanding and Exploring the Modeling History of 3D CAD Designs. In *Proceedings of the 8th ACM Symposium on Computational Fabrication*. ACM, New York City NY USA, 1–13. https://doi.org/10.1145/3623263.3623358

[64] Zhansong Wang, Ling Tian, and Wenrui Duan. 2014. Annotation and retrieval system of CAD models based on functional semantics. *Chinese Journal of Mechanical Engineering* 27, 6 (Nov. 2014), 1112–1124. https://doi.org/10.3901/CJME.2014.0815.134

[65] David C Wynn, Claudia M Eckert, and P John Clarkson. 2007. Modelling iteration in engineering design. In *Proceedings of ICED 2007, the 16th International Conference on Engineering Design*, Vol. DS 42. The Design Society, Paris, France, 1–12.

[66] Loutfouz Zaman, Wolfgang Stuerzlinger, and Christian Neugebauer. 2017. MACE: A New Interface for Comparing and Editing of Multiple Alternative Documents for Generative Design. In *Proceedings of the 2017 ACM Symposium on Document Engineering*. ACM, Valletta Malta, 67–76. https://doi.org/10.1145/3103010.3103013

[67] Loutfouz Zaman, Wolfgang Stuerzlinger, Christian Neugebauer, Rob Woodbury, Maher Elkhaldi, Naghmi Shireen, and Michael Terry. 2015. *GEM-NI*: A System for Creating and Managing Alternatives In Generative Design. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, Seoul Republic of Korea, 1201–1210. https://doi.org/10.1145/2702123.2702398

[68] Ibrahim Zeid. 2016. Investigating and Comparing Two Different CAD Methodologies to Create Top-down Assemblies. In *2016 ASEE Annual Conference & Exposition Proceedings*. ASEE Conferences, New Orleans, Louisiana, 25471. https://doi.org/10.18260/p.25471

[69] Shurui Zhou, Ştefan Stănciulescu, Olaf Leßenich, Yingfei Xiong, Andrzej Wąsowski, and Christian Kästner. 2018. Identifying features in forks. In *Proceedings of the 40th International Conference on Software Engineering*. ACM, Gothenburg Sweden, 105–116. https://doi.org/10.1145/3180155.3180205

[70] Linghao Zou, Dongming Guo, and Hang Gao. 2011. A method to analyze the difference of 3-D CAD model files based on feature extraction. *Journal of Mechanical Science and Technology* 25, 4 (April 2011), 971–976. https://doi.org/10.1007/s12206-011-0210-9