Fetch-A-Sketch: Navigating Dependencies in 3D CAD Master Sketches

Kathy Cheng
Mechanical and Industrial Engineering
University of Toronto
Toronto, Ontario, Canada
kathy.cheng@mail.utoronto.ca

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

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

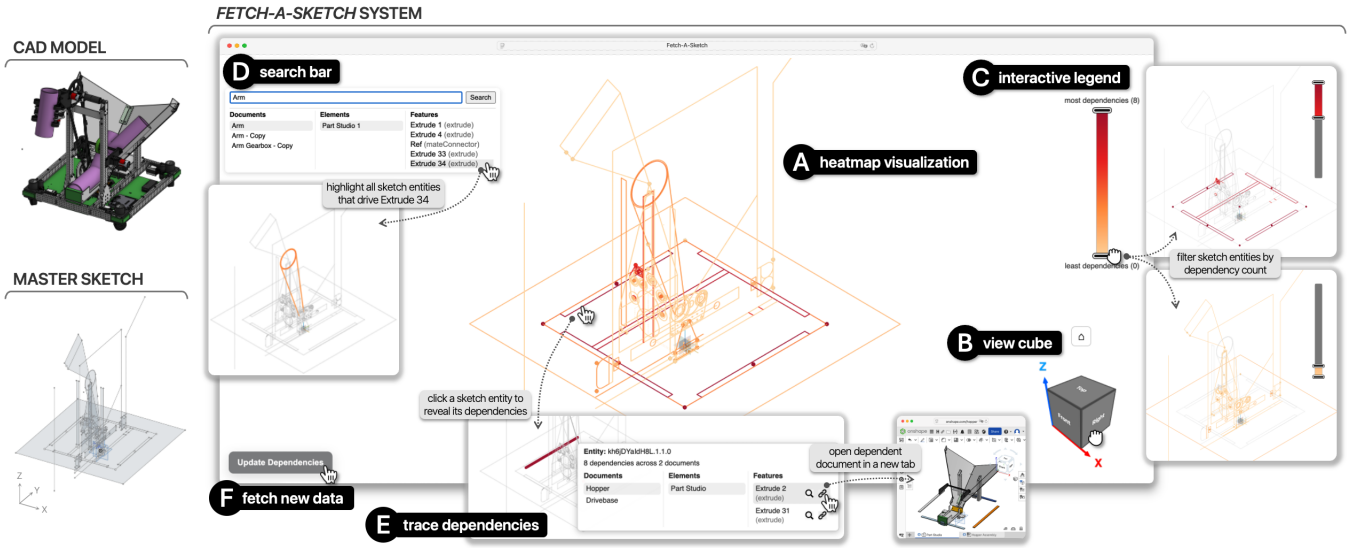

Figure 1: Complex CAD models may use a *master sketch* to control downstream geometry. Fetch-A-Sketch supports master sketch navigation and dependency sensemaking through: (A) a main visualization of the master sketch, where entities are coloured with a heatmap to represent dependency density (darker entities have more downstream dependencies), with interactive zoom, pan, and rotate; (B) a view cube [23] for 3D orientation; (C) an interactive colour scale that serves as both a legend and a filter, allowing designers to threshold sketch entities by dependency count; (D) a search bar to query documents or features, highlighting all entities that drive the selected item; (E) a context menu showing a selected entity's downstream dependencies, with options to open related documents in a new tab or highlight other driving entities (via D's search function); and (F) an update button to refresh dependency data and reconstruct the visualization. Models adapted from [2].

Abstract

Computer-aided design (CAD) models are complex artifacts built from sequences of modelling features (e.g., extrude, fillet), each dependent on previous features. Especially in large-scale projects, these dependencies rapidly proliferate, making them difficult to track, yet designers must understand these relationships to make informed changes. We introduce Fetch-A-Sketch, a prototype system that surfaces and visualizes hidden dependencies in complex 3D master sketches, helping designers anticipate the downstream impacts of changes. In a first-use study, participants reported that Fetch-A-Sketch enhanced their understanding of dependencies, enabling them to navigate unfamiliar models and plan design

CHI EA '26, Barcelona, Spain

ACM ISBN 979-8-4007-2281-3/26/04
https://doi.org/10.1145/3772363.3798403

changes effectively. Feedback also identified opportunities to extend Fetch-A-Sketch to support more complex products and collaborative contexts, which we aim to address in future work. By making dependencies more visible and manageable, Fetch-A-Sketch supports long-term maintainability in large, complex design projects.

CCS Concepts

• **Human-centered computing → Interactive systems and tools**; • **Applied computing → Computer-aided design**.

Keywords

parametric CAD, 3D modelling, information visualization, design maintenance, complex artifacts, change impact analysis

ACM Reference Format:
Kathy Cheng, Zhijing Zhang, Yuanzhe Deng, Shurui Zhou, and Alison Olechowski. 2026. Fetch-A-Sketch: Navigating Dependencies in 3D CAD Master Sketches. In *Extended Abstracts of the 2026 CHI Conference on Human Factors in Computing Systems (CHI EA '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3772363.3798403

1 Introduction

The design of physical products, from smartwatches to bicycles, typically requires parametric computer-aided design (CAD) software [8]. In parametric CAD, geometry is created using dependency relationships that link *CAD artifacts* — like features (e.g., extrude, fillet) or parts (e.g., wheel, frame) — and encode functional interactions (e.g., a wheel that rotates around an axle) or geometric constraints within individual components (e.g., a hole remaining centred on a face).

Since each new feature or part builds on existing geometry, parametric CAD models naturally form complex dependency networks [21]. As designs evolve, geometry is added, removed, or modified, and dependencies are created, deleted, or rerouted. Over time, these accumulating dependencies become difficult to track, obscuring both the model's construction history and its underlying dependency structure. Yet, this understanding is essential for evolving the model aligned with the designer's intent [13].

To help manage this complexity, designers working on large projects commonly adopt a *master sketch* architecture [22], in which a centralized sketch defines geometry that downstream models are built upon [29]. This top-down approach allows components to be edited separately, enabling task parallelization and collaboration [12]; however, master sketches can quickly become dense and cluttered, like *"a haystack"* [11]. As a result, designers expend substantial time and cognitive effort to determine which parts of a sketch are critical, which are fragile, and how changes may propagate through the model [11].

In this paper, we draw on CAD dependency management literature and HCI research on history-rich traces in complex artifacts [27] to derive design goals for supporting CAD dependency understanding. Building on these goals, we introduce Fetch-A-Sketch, an interactive visualization prototype that helps designers perceive, navigate, and trace dependency structures embedded in master sketches. Unlike prior graph-based approaches [7, 24, 25], Fetch-A-Sketch uses heatmap-style visualizations inspired by the concept of *digital patina* [41] to convey dependency density, while interactivity allows designers to surface hidden downstream dependencies.

We make three key contributions: (1) we elucidate design goals for dependency-aware CAD visualization, (2) we introduce Fetch-A-Sketch, a prototype that provides interactive visualization of master sketch dependencies, and (3) we present findings from a first-use study with four experienced CAD designers, illustrating how the system supports sensemaking in complex sketches.

2 Background & Related Work

2.1 Dependencies in Parametric CAD

Parametric CAD systems (e.g., SolidWorks,[1] Onshape[2]) are widely used for hardware development [8]. Here, geometry is defined through parameters and constraints, such that each modelling feature references previously defined features (e.g., extruding a sketched circle to form a cylinder creates a dependency between the extrude feature and the sketch). As products grow in complexity, dependencies proliferate, forming dense, intricate networks. In their formative study of CAD dependency management, Cheng et al. [11] found that designers primarily struggle with dependency *traceability* (understanding artifact origins, histories, and interdependence) and *navigation* (efficiently retrieving, organizing, and making sense of cluttered dependency information).

Recognizing the importance of dependency management [37], researchers have primarily explored graph-based representations of dependency structures [7, 24, 25, 37], finding that supplementing geometric views with dependency visualizations can improve comprehension [24], particularly when interfaces allow users to focus on specific, task-relevant dependency chains [31]. While effective for conveying overall structure at a glance, graph-based approaches abstract rich 3D geometry to nodes and edges, introducing an additional representational layer that makes dependency reasoning less intuitive [18]. This tradeoff motivates the need to explore alternative representations that better preserve the intricacies of parametric relationships. Our work presents a novel approach to visualizing and navigating dependencies in CAD artifacts.

Given the challenges of managing dependencies in parametric CAD models, product development teams commonly adopt a *master sketch* architecture to organize dependencies in a top-down manner [19, 22]. As illustrated in Fig. 1, a master sketch comprises multiple 2D sketches in the XY-, YZ-, and ZX-planes [4, 33], which specify key dimensions and reference geometries that govern relationships for the features and parts throughout the product [29]. By centralizing dependencies, modifications to individual *entities* (e.g., line, vertex) propagate the changes downstream in accordance with the design intent [12]. In practice, however, maintaining and navigating master sketches becomes increasingly difficult as product complexity grows. Designers struggle to identify the downstream dependencies of sketch entities [10], causing unintentional changes or, conversely, conservative over-retention of outdated entities [11]. Over time, such practices lead to cluttered, difficult-to-maintain master sketches, particularly when the intent or relevance of the entity is no longer obvious [42].

[1] https://www.solidworks.com/
[2] https://www.onshape.com/en/products/free

Master sketches are a prevalent approach in product development, yet existing tools offer little support for navigating, understanding, and maintaining them. Our work addresses this gap by providing an intuitive, interactive means to inspect and manage master sketch dependencies, supporting more informed design decisions and long-term maintainability.

2.2 History-Rich Signals for Understanding Complex Artifacts

Complex digital artifacts (e.g., text documents, codebases, CAD models) accrue extensive histories as they are created, edited, and reused. HCI research has explored making digital artifacts *history-rich* by embedding *traces* of past activity directly into the artifact [27]. Traces not only record prior actions, but also shape future activities by helping people recall prior decisions and understand how an artifact reached its current state, supporting the reconstruction of mental context [40].

Hill et al. introduced the concept of *computational wear* [17], in which digital artifacts (e.g., documents, spreadsheets) accumulate visible records of their use. Analogous to physical wear, *edit wear* and *read wear* visually portray a document's modification and readership histories, which can signal the stability of document sections [17], or the actions of collaborators [1]. Computational wear allows users to reason about how artifacts have been used over time, acting as *proximal cues* to help people find useful information [36]. Computational wear is commonly represented with aggregate visual encodings, such as *digital patina* [41], which uses colour to convey intensity of usage data, akin to a heatmap [5, 26]. Researchers have used heatmaps to visualize common and rare commands in applications [32], the location of feedback within documents [45], and time spent interacting with different vertices of 3D models [9]. Patina and heatmaps leverage users' intuitive understanding of "hot" and "cold" zones to support rapid interpretation of accumulated activity [32].

History-rich signals, such as traces, computational wear, and patina, effectively reveal interaction density in digital artifacts. We extend this concept to visualize how dependencies accumulate in parametric CAD sketches, helping designers reason about the stability and criticality of sketch entities, anticipate the downstream impact of changes, and more effectively plan design changes.

3 Design Goals

We derived design goals from the literature on CAD dependency management and master sketch architecture.

DG1: Surface dependencies of master sketch entities. Understanding the downstream dependencies of a sketch is challenging for designers [11, 21], yet it is critical for anticipating the potential impact of edits before making changes [30]. The system should make downstream dependencies of sketch entities visible and inspectable, supporting tracing across multiple granularities, such as high-level documents and low-level features [11].

DG2: Provide lightweight signals of accumulated dependencies. Design artifacts accrue complex dependency structures over time, but detailed logs can be difficult to interpret at a glance. Prior work proposes visual approaches (e.g., colour-based highlighting) to indicate which artifacts require attention [11, 12]. The system should surface accumulated dependencies through lightweight visual signals without overwhelming users, to support rapid sensemaking.

DG3: Support selective and on-demand exploration. Because CAD models can contain thousands of dependencies [14], tools should progressively reveal information rather than exposing everything at once [25], enabling designers to investigate specific dependency chains as needed [11, 31]. By supporting selective, on-demand exploration, designers can focus on the relevant subset of information for their immediate task.

DG4: Preserve and complement designers' existing workflows. To maintain workflow continuity, the system should complement and work alongside familiar native CAD features rather than replacing or interfering with them [24].

4 Fetch-A-Sketch

Based on our design goals, we developed Fetch-A-Sketch, a prototype system that supports the navigation and understanding of master sketch dependencies through interactive visualization. Fetch-A-Sketch takes a master sketch as input, analyzes downstream dependencies, and reconstructs the sketch with a heatmap to represent the number of dependencies — or dependency *density* — of each entity (Fig. 1). Dependency density is operationalized as the absolute downstream dependency count: the number of features that reference a given sketch entity. Heatmap colours range from the highest dependency count in the master sketch to the lowest, providing a simple and interpretable indicator of relative impact. By surfacing dependency information often hidden in typical CAD platforms, Fetch-A-Sketch provides an intuitive way to navigate and understand complex dependency structures in master sketches.

Designers can select individual sketch entities to inspect their downstream dependencies across multiple levels of granularity, from high-level documents to low-level features **(DG1)**. To help designers quickly understand the distribution of dependencies across the sketch, Fetch-A-Sketch draws on prior HCI work on digital patina to display "pressure points" in sketches, allowing designers to rapidly identify areas of heightened dependency complexity **(DG2)**. To support selective, on-demand exploration, a search function allows designers to query documents or features by name and to highlight their driving sketch entities, streamlining the dependency-tracing process **(DG3)**. Finally, Fetch-A-Sketch complements rather than replaces the native CAD interface **(DG4)**, running in a separate workspace for visualization, while leaving all geometry creation and modification to the CAD environment.

4.1 Example User Scenario

To illustrate how Fetch-A-Sketch supports CAD workflows, we describe a scenario in which Alex, a mechanical designer, is tasked with modifying the claw of a pick-and-place robot to handle larger objects. As a newcomer to the project, Alex uses Fetch-A-Sketch to understand the current design state and implement the required changes.

Removing unused sketches. Because the claw has undergone multiple iterations, Alex first examines the heatmap to spot low-density sketches, noticing a few old profile sketches in the claw subassembly that are no longer referenced downstream and can

safely be removed (Fig. 1A,C). Deleting these unused sketches simplifies the master sketch and reduces unnecessary visual clutter.

Identifying critical sketches. Next, Alex examines the heatmap for high-density sketches — those with the most downstream dependencies (Fig. 1A,C). Because changes to these sketches could propagate widely across the entire robot, Alex treats them as "sensitive" and plans to avoid editing them when possible, focusing instead on lower-density sketches where modifications are less likely to impact other components.

Planning and validating structural changes. To safely enlarge the claw design, Alex inspects the downstream dependencies of the claw sketch entities using the dropdown view (Fig. 1E), to see whether other subsystems are affected. While the claw profile is expected to drive features within the claw subsystem, Alex is surprised to see dependencies from the motor mount subsystem. To investigate this, Alex uses the search function (Fig. 1D) to highlight all entities in the master sketch that also drive the motor mount, allowing Alex to determine whether the claw profile is intentionally shared geometry or an accidental or legacy dependency.

To further understand how changes to the claw would impact the motor mount, Alex opens the linked CAD documents in new tabs for detailed examination (Fig. 1E). This allows Alex to anticipate which constraints or features will need adjustment as the claw geometry changes, minimizing the risk of breaking downstream features unintentionally or unknowingly. Once edits to the claw models are complete, Alex fetches updated dependency data, and Fetch-A-Sketch reconstructs the visualization (Fig. 1F), providing a rapid, clear overview of the updated master sketch.

4.2 Implementation Details

Fetch-A-Sketch is implemented as a web-deployable application that processes data from the Onshape CAD platform. The backend, built with Python Flask, retrieves geometry data via REST API calls, and the frontend renders interactive 3D sketches using Three.js. Further implementation details are in Appendix A.

5 User Study

To evaluate Fetch-A-Sketch, we conducted a first-use remote study with four designers experienced with Onshape. The goal of a first-use study is to capture the strengths and limitations of a system during initial exposure [16].

We employed a within-subjects design in which participants first completed CAD dependency exploration tasks using native Onshape features (Round 1), and then with additional support from Fetch-A-Sketch (Round 2). The fixed order was intentional: conducting the Onshape-only condition first allowed us to observe participants' typical dependency sensemaking strategies using existing tools, and avoided priming them with the additional representations introduced by Fetch-A-Sketch. Because Fetch-A-Sketch exposes dependency information not otherwise available in CAD tools, counterbalancing the order would have introduced irreversible learning effects. Although participants gain familiarity with the model during Round 1, we do not treat this as a confound; our goal is not to assess or compare task performance, but to understand how access to additional dependency visualization shapes how designers reason about CAD dependencies. Because Fetch-A-Sketch is designed to supplement rather than replace native CAD tools, providing access to both tools reflects the intended usage context (see Appendix B for further study design details).

We asked the following research questions: **RQ1:** How does Fetch-A-Sketch support designers' dependency sensemaking?; **RQ2:** What benefits do designers perceive?; and **RQ3:** What challenges do designers encounter?

Participants. We recruited four designers from our professional networks, all with at least two years of Onshape experience and familiarity with master sketches (see Appendix C for descriptive information).

Procedure and Tasks. The study was conducted over Zoom. To allow participants to access Fetch-A-Sketch remotely without installation, we hosted it on the cloud platform, Netlify.[3] Participants ran both Onshape and Fetch-A-Sketch on their own computers, and we recorded their screen and audio.

Participants worked with a moderately complex robot project comprised of eight CAD documents, 350 features, and 492 dependencies from a shared master sketch. The study consisted of two rounds of three CAD tasks each: (A) identifying sketches with the greatest downstream impact; (B) identifying sketches with least downstream impact; and (C) predicting downstream effects of changes to a specific subsystem (Claw or Arm) in the master sketch. These tasks aligned with the user scenario (Section 4.1), and CAD workflows reported in prior work [11]. Participants were given five minutes per task, and were asked to think aloud [43].

In Round 2, participants were introduced to Fetch-A-Sketch, received a brief walkthrough of its features, and had a few minutes to explore the tool. Once comfortable with Fetch-A-Sketch, participants then repeated the same three tasks, with Task C focusing on the alternate subsystem to counterbalance exposure.

Following both rounds, we conducted a post-task interview (approximately 15 minutes) to understand general task strategies, how Fetch-A-Sketch supported their work, and any suggestions for improvement (questions in Appendix D). The study lasted approximately one hour, and participants received a $15 gift card.

Data Analysis. Participants' think-aloud verbalizations and interview responses were automatically transcribed using Zoom. Following recommendations for early-stage systems evaluation [28, 34], we analyzed multiple complementary data sources. To understand how designers approached the tasks using Fetch-A-Sketch, we conducted a video interaction analysis [3] of the screen recordings, coding participants' actions in Onshape (e.g., opening documents) and Fetch-A-Sketch (e.g., adjusting the slider). Post-task semi-structured interviews elicited participants' experiences, perceived benefits, and challenges, allowing us to go beyond surface-level usability and probe questions of utility [15]. Interview questions focused not only on Fetch-A-Sketch itself, but on which aspects of its representations participants found valuable and worth carrying forward in future designs. We conducted reflexive thematic analysis [6] of the think-aloud and interview transcripts, following an inductive coding process, and developed themes through affinity diagramming using Dovetail qualitative data analysis software.[4]

[3] https://www.netlify.com/
[4] https://dovetail.com/

5.1 Findings

Supporting dependency sensemaking (RQ1). Without Fetch-A-Sketch, participants relied on heuristics and manual tracing to explore dependencies. To determine which sketch entities were referenced, participants used visual inspection, flipping between the subsystem and master sketch documents: *"I was playing spot-the-difference"* (P3). When identifying the most and least impactful sketches, participants drew on heuristics, such as assuming that exterior sketches were critical since *"they backbone everything"* (P3), prioritizing subsystems *"positionally close"* (P3) to key components, or inferring from naming conventions: *"any sketch in the master sketch that has a corresponding document [will] have dependencies"* (P1). Another strategy was trial-and-error deletion, in which participants removed entities to see what features would break: *"I rely heavily on the delete function. It's a very, very destructive, but very quick way"* (P3).

With Fetch-A-Sketch, participants expressed that they explored dependencies more efficiently and confidently. All four used the slider (Fig. 1C) to identify sketch entities with the most and fewest dependencies, and all clicked on specific entities to inspect those dependencies (Fig. 1E), which sometimes revealed easily overlooked information: *"wow, so that little circle has six dependencies. I wasn't expecting that"* (P1). Three participants (P1, P3, P4) used the search function (Fig. 1D) to investigate a subsystem (Arm or Claw), inspected the driving entities, and then navigated to the relevant document (Fig. 1E) to confirm Fetch-A-Sketch's output. Participants were able to apply the same heuristics and intuition they would normally use, but in a way that was faster, less error-prone, and non-destructive.

Perceived benefits (RQ2). Participants reported several benefits of using Fetch-A-Sketch, particularly related to its visual and interactive approach to dependency sensemaking. Fetch-A-Sketch's visual, spatial interface made understanding dependencies easier than traditional text-based logs, as geometric information aligns with how designers naturally reason about CAD models: *"you already have spatial awareness of where each part is in the sketch. With [Onshape], the dependency information is text, whereas [Fetch-a-Sketch] is visual"* (P4). P3 emphasized that simply having a dedicated visual interface for dependency tracing is valuable, given the lack of comparable tooling in existing CAD systems: *"the fact that there even is a UI is helpful"* (P3).

Participants found the heatmap encoding intuitive and *"super quick"* (P2) for identifying critical areas within a sketch. By visually highlighting regions with many or important dependencies, the heatmap helped designers immediately see where changes might have the greatest impact: *"visually, I love seeing the gradient of the hot areas"* (P1), allowing participants to focus on the most consequential areas without manually tracing relationships.

Finally, participants shared that Fetch-A-Sketch made it easier to identify where a sketch entity was used and which features depended on it, which are typically tedious tasks: *"the processes take an extremely long time to the point where [my colleagues and I] don't want to do them. Seeing what specific features are using [the entity] is very helpful"* (P3). To illustrate the practical impact of this support, P3 described a recent debugging scenario that required several days of engineering and review time, which could be significantly reduced with Fetch-A-Sketch: *"I, as a person who knows [our CAD] inside and out, and a colleague took three days of engineering time because we had to fix anything that did break. And then approximately two days of checking time by another engineer"* (P3).

Ongoing challenges (RQ3). Participants voiced challenges with dependency sensemaking that persist even when using Fetch-A-Sketch. All participants noted that entity IDs were not semantically meaningful: *"apart from the people at Onshape, nobody cares about this number"* (P4), which made it hard to understand their purpose: *"because [these entities] were very similar, I don't know what I'm selecting"* (P1). Participants suggested entity labels to reflect their purpose or relevant subsystem: *"it would be nice if the entity [ID] was actually 'Chain Plan"'* (P3).

Participants noted that absolute dependency count is not the only indicator of a sketch entity's dependency density, and that other factors also determine which entities are most critical or high-risk. In particular, the most impactful sketch entities can be interpreted as those that affect multiple subsystems, because changes may propagate unpredictably. P1 explained: *"if I change Claw, yes, a lot of things are going to change, but only within Claw. So that's not as bad as if everything around Claw — Arm, Grabber, Drivetop, whatever — are all going to change. It's a controlled bomb."*

6 Discussion & Future Work

Fetch-A-Sketch is an interactive visualization prototype intended to support designers' reasoning about dependency structure and change impact in CAD master sketches. Participants valued the system's visual interface and intuitive workflow, finding it natural to integrate into their existing CAD sensemaking practices. Participants also highlighted challenges that are important to consider in future iterations. Below, we discuss implications and future work opportunities.

In its current implementation, Fetch-A-Sketch only provides absolute dependency counts as a simple, interpretable measure. Other metrics or operationalizations of dependency density could provide designers with additional context, for example, by surfacing higher-level violations of best practices [39], such as cross-subsystem dependencies, or subsystem-specific sketches that would be better placed in a separate document rather than the master sketch [10]. Fetch-A-Sketch could be extended into a dashboard that monitors multiple aspects of dependency health, to avoid architectural debt [38], and maintain cleaner, more manageable models.

We also note collaboration implications. P3 noted the value of Fetch-A-Sketch for design reviews, since reviewers would be able to quickly verify that changes were made correctly, avoiding *"very long and arduous manual checks."* Beyond design review, improved dependency understanding benefits any collaborative context, as dependency awareness is difficult even for the CAD model's original designer, and becomes even more challenging when many contributors are involved [10], or when exploring an unfamiliar model (e.g., during design handoffs [20]).

Finally, we acknowledge several limitations. Fetch-A-Sketch was evaluated using a single master sketch model, and the resulting visualizations reflect that specific product. While the robot model aligns well with a three-plane, box-like form, other products — those requiring complex surface modelling [44] (e.g., cars, airplanes) —

may not suit this approach. Additionally, dependency sensemaking is only one aspect of CAD design work, and our participants did not modify models in the study; instead, we focused on how visibility of dependencies supports reasoning, which is a necessary prerequisite for making design changes. Finally, as a first-use evaluation, our study captured initial perceptions rather than long-term adoption. Future work will deploy Fetch-A-Sketch with designers' own models in situ to examine how such visualizations support collaboration, maintenance, and decision-making over time.

References

[1] Jason Alexander, Andy Cockburn, Stephen Fitchett, Carl Gutwin, and Saul Greenberg. 2009. Revisiting read wear: analysis, design, and evaluation of a footprints scrollbar. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Boston MA USA, 1665–1674. doi:10.1145/1518701.1518957

[2] Arborbotics. 2025. FRC Team 6657 Arborbotics Lotus Robot. https://team6657.weebly.com/. [Accessed 04-04-2023].

[3] Eric P.S. Baumer and Bill Tomlinson. 2011. Comparing activity theory with distributed cognition for video analysis: beyond "kicking the tires". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. Association for Computing Machinery, New York, NY, USA, 133–142. doi:10.1145/1978942.1978962

[4] Yannick Bodein, Bertrand Rose, and Emmanuel Caillaud. 2013. A roadmap for parametric CAD efficiency in the automotive industry. *Computer-Aided Design* 45, 10 (Oct. 2013), 1198–1214. doi:10.1016/j.cad.2013.05.006

[5] Agnieszka (Aga) Bojko. 2009. Informative or Misleading? Heatmaps Deconstructed. In *Human-Computer Interaction. New Trends*, Julie A. Jacko (Ed.). Springer, Berlin, Heidelberg, 30–39. doi:10.1007/978-3-642-02574-7_4

[6] Virginia Braun and Victoria Clarke. 2013. *Successful qualitative research: a practical guide for beginners* (first published ed.). SAGE, Los Angeles London New Delhi.

[7] Jorge D. Camba and Manuel Contero. 2024. 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 - Volume 0VISIGRAPP*. SciTePress, Berlin, Germany, 16–25. doi:10.5220/0005261500160025

[8] 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. doi:10.1016/j.cad.2016.01.003

[9] Hsiang-Ting Chen, Tovi Grossman, Li-Yi Wei, Ryan M. Schmidt, Björn Hartmann, George Fitzmaurice, and Maneesh Agrawala. 2014. History assisted view authoring for 3D models. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Toronto Ontario Canada, 2027–2036. doi:10.1145/2556288.2557009

[10] 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), 137:1–137:29. doi:10.1145/3579613

[11] 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, CSCW2 (Aug. 2025), 33. doi:10.1145/3757617

[12] Claudio Ciaccioli, Anna Eva Morabito, and University of Salento, Department of Engineering for Innovation, Lecce, Italy. 2021. An investigation on skeleton-based top-down modelling approaches of complex industrial product. *Journal of Graphic Engineering and Design* 12, 1 (March 2021), 11–21. doi:10.24867/JGED-2021-1-011

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

[14] James A. Gopsill, Chris Snider, and Ben J. Hicks. 2019. The emergent structures in digital engineering work: what can we learn from dynamic DSMs of near-identical systems design projects? *Design Science* 5 (Jan. 2019), e28. doi:10.1017/dsj.2019.20

[15] Saul Greenberg and Bill Buxton. 2008. Usability evaluation considered harmful (some of the time). In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. Association for Computing Machinery, New York, NY, USA, 111–120. doi:10.1145/1357054.1357074

[16] Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. 2007. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, San Jose California USA, 145–154. doi:10.1145/1240624.1240646

[17] William C. Hill, James D. Hollan, Dave Wroblewski, and Tim McCandless. 1992. Edit wear and read wear. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*. Association for Computing Machinery, New York, NY, USA, 3–9. doi:10.1145/142750.142751

[18] Raphaël Hoarau and Stéphane Conversy. 2012. Augmenting the scope of interactions with implicit and explicit graphical structures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. Association for Computing Machinery, New York, NY, USA, 1937–1946. doi:10.1145/2207676.2208337

[19] Christoph M Hoffman and Robert Joan-Arinyo. 1998. CAD and the product master model. *Computer-Aided Design* 30, 11 (Sept. 1998), 905–918. doi:10.1016/S0010-4485(98)00047-5

[20] Leila Homaeian, James R. Wallace, and Stacey D. Scott. 2022. Handoff and Deposit: Designing Temporal Coordination in Cross-Device Transfer Techniques for Mixed-Focus Collaboration. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW2 (Nov. 2022), 1–23. doi:10.1145/3555192

[21] 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 (UIST '24)*. Association for Computing Machinery, New York, NY, USA, 1–14. doi:10.1145/3654777.3676353

[22] PTC Inc. 2023. Creo+: Capabilities and Packages. https://www.ptc.com/-/media/Files/PDFs/CAD/Creo/Creo-Plus/Creo-Plus_Capabilities_and_Packages_Brochure.pdf

[23] Azam Khan, Igor Mordatch, George Fitzmaurice, Justin Matejka, and Gordon Kurtenbach. 2008. ViewCube: a 3D orientation indicator and controller. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games (I3D '08)*. Association for Computing Machinery, New York, NY, USA, 17–25. doi:10.1145/1342250.1342253

[24] Siniša Kolarić, Halil Erhan, Robert Woodbury, and Bernhard E. Riecke. 2010. Comprehending parametric CAD models: an evaluation of two graphical user interfaces. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*. ACM, Reykjavik Iceland, 707–710. doi:10.1145/1868914.1869010

[25] 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 *Proceedings of the 16th International Conference on Computer Aided Architectural Design Research in Asia*. CAADRIA, Newcastle, Australia, 133–142. doi:10.52842/conf.caadria.2011.133

[26] Matthias Kraus, Katrin Angerbauer, Juri Buchmüller, Daniel Schweitzer, Daniel A. Keim, Michael Sedlmair, and Johannes Fuchs. 2020. Assessing 2D and 3D Heatmaps for Comparative Analysis: An Empirical Study. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. doi:10.1145/3313831.3376675

[27] Ida Larsen-Ledet, Myriam Lewkowicz, Clemens N. Klokmose, Carol Linehan, and Luigina Ciolfi. 2025. Traces, Breadcrumbs, and Patina: Exploring and Designing with Traces of Activity. In *Companion Publication of the 2025 Conference on Computer-Supported Cooperative Work and Social Computing*. ACM, Bergen Norway, 116–119. doi:10.1145/3715070.3748288

[28] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. 2018. Evaluation Strategies for HCI Toolkit Research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–17. doi:10.1145/3173574.3173610

[29] Borhen Louhichi and Louis Rivest. 2014. Maintaining consistency between CAD elements in collaborative design using association management and propagation. *Computers in Industry* 65, 1 (Jan. 2014), 124–135. doi:10.1016/j.compind.2013.08.003

[30] Maryam M. Maleki, Robert F. Woodbury, and Carman Neustaedter. 2014. Liveness, localization and lookahead: interaction elements for parametric design. In *Proceedings of the 2014 conference on Designing interactive systems (DIS '14)*. Association for Computing Machinery, New York, NY, USA, 805–814. doi:10.1145/2598510.2598554

[31] 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. doi:10.3722/cadaps.2011.435-448

[32] Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2013. Patina: dynamic heatmaps for visualizing application usage. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. Association for Computing Machinery, New York, NY, USA, 3227–3236. doi:10.1145/2470654.2466442

[33] Duhwan Mun, Jinsang Hwang, and Soonhung Han. 2009. Protection of intellectual property based on a skeleton model in product design collaboration. *Computer-Aided Design* 41, 9 (Sept. 2009), 641–648. doi:10.1016/j.cad.2009.04.007

[34] Dan R. Olsen. 2007. Evaluating user interface systems research. In *Proceedings of the 20th annual ACM symposium on User interface software and technology (UIST '07)*. Association for Computing Machinery, New York, NY, USA, 251–258. doi:10.1145/1294211.1294256

[35] Onshape. 2025. API Limits — onshape-public.github.io. https://onshape-public.github.io/docs/auth/limits/. [Accessed 08-01-2026].
[36] Peter Pirolli and Stuart Card. 1999. Information foraging. *Psychological Review* 106, 4 (Oct. 1999), 643–675. doi:10.1037/0033-295X.106.4.643
[37] Ahsan Qamar, Christiaan J. J. Paredis, Jan Wikander, and Carl During. 2012. Dependency Modeling and Model Management in Mechatronic Design. *Journal of Computing and Information Science in Engineering* 12, 041009 (Dec. 2012), 10 pages. doi:10.1115/1.4007986
[38] Larri Ann Rosser and Zakaria Ouzzif. 2021. Technical Debt in Hardware Systems and Elements. In *2021 IEEE Aerospace Conference (50100)*. IEEE, Big Sky, MT, USA, 1–10. doi:10.1109/AERO50100.2021.9438332 ISSN: 1095-323X.
[39] 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. doi:10.1017/pds.2022.176
[40] Adam Rule and Jim Hollan. 2016. Thinking in 4D: Preserving and Sharing Mental Context Across Time. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion (CSCW '16)*. ACM, San Francisco, CA, USA, 389–392. doi:10.1145/2818052.2869108
[41] Ansel Arjan Schütte. 1998. *Patina : layering a history-of-use on digital objects*. Thesis. Massachusetts Institute of Technology. https://dspace.mit.edu/handle/1721.1/62341 Accepted: 2011-04-25T15:43:57Z.
[42] Arina Shakurova. 2019. *SolidWorks Bottom-Up versus Top-Down Approaches. The Capacity of the Top-Down Modelling*. Degree Programme in Mechanical Engineering and Production Technology. Häme University of Applied Sciences, Riihimäki, Finland. https://www.theseus.fi/bitstream/handle/10024/171101/Thesis.pdf?sequence=2
[43] Maarten W. van Someren, Yvonne F. Barnard, and Jacobijn A. Sandberg. 1994. *The think aloud method: a practical guide to modelling cognitive processes*. Number 13 in Knowledge based systems. Academic Press, London.
[44] Rainer Stark. 2022. Major Technology 1: Computer Aided Design—CAD. In *Virtual Product Creation in Industry*. Springer Berlin Heidelberg, Berlin, Heidelberg, 113–138. doi:10.1007/978-3-662-64301-3_7
[45] Chao Zhang, Kexin Ju, Peter Bidoshi, Yu-Chun Grace Yen, and Jeffrey M. Rzeszotarski. 2025. Friction: Deciphering Writing Feedback into Writing Revisions through LLM-Assisted Reflection. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–27. doi:10.1145/3706598.3714316

A Implementation Details

We chose to build Fetch-A-Sketch to be compatible with Onshape[5] data because Onshape's API exposes unique IDs for individual sketch entities, enabling a level of granularity unavailable in other CAD platforms we tested (e.g., Autodesk Fusion[6]).

The application backend is built with the Python Flask[7] framework, which performs REST API[8] calls to retrieve geometry and dependency data. From the master sketch document, it extracts each sketch entity's unique ID and geometric information (e.g., endpoint coordinates, arc radius). For all other documents in the same folder, the backend examines each feature's referenced geometry, then matches these references to master sketch entities to identify dependencies. All extracted data — including master sketch geometry, dependencies, and document metadata — is processed in the backend, and the frontend user interface renders interactive 3D sketches using Three.js.[9] When a user clicks "Update Dependencies", it triggers the backend to fetch new data, and the frontend updates the visualization.

Fetch-A-Sketch was designed under practical constraints imposed by the CAD platform's API limit of 2.5K calls per year [35], as well as the API communication time cost.[10] These constraints informed several design decisions, including restricting dependency search to documents within a single folder and prioritizing on-demand queries over continuously monitoring document changes.

[5]https://www.onshape.com/en/products/free
[6]https://www.autodesk.com/ca-en/products/fusion-360/overview
[7]https://flask.palletsprojects.com/
[8]https://onshape-public.github.io/docs/api-intro/
[9]https://threejs.org/

B Additional Details for the User Study Design

During the study, Fetch-A-Sketch operated on a fixed snapshot of the model's dependency data rather than a live connection to the Onshape API. Because API-based dependency extraction incurs non-trivial computation time, supporting live updates during the session would have introduced latency and disrupted the study flow. Participants therefore did not modify the model or trigger data refreshes; instead, they used Fetch-A-Sketch as an interactive visualization to explore and reason about existing dependency relationships.

C User Study Participants

We conducted a user study evaluation with four experienced CAD designers. Table 1 summarizes descriptive information of our participants.

Table 1: Details of user study participants, including job role, gender (W: woman; M: man), years of CAD experience, years of Onshape experience, and subsystem order for Task C.

ID	Job Role	Gender	CAD (years)	Onshape (years)	Task C Order
P1	Design Engineer	W	8	3	Claw → Arm
P2	Masters Student	M	5	2	Arm → Claw
P3	Senior Manufacturing Engineer	W	10	6	Claw → Arm
P4	Product Architect	M	13	8	Arm → Claw

D User Study Interview Questions

In the post-task interviews, we asked participants the following questions:

- Can you walk me through your general strategy or process for completing the tasks today?
- What intuition or heuristics did you rely on when exploring the model?
- How did you know you were finished with a task, or felt confident in your answer?
- Overall, how helpful was Fetch-A-Sketch for completing the tasks? Which task did you find it most helpful for? Why do you think that task benefited most?
- Were there tasks where it didn't help, or even interfered?
- Are there other situations or workflows where you think a tool like Fetch-A-Sketch would be helpful?
- What did you enjoy about using Fetch-A-Sketch? Were there any specific features you found helpful?
- What improvements could be made to Fetch-A-Sketch? What features were not helpful?
- Could you see yourself using a similar tool to Fetch-A-Sketch in the future? How would it fit into your current workflow?

[10]As an example, retrieving API data for a moderately complex project with eight documents, 114 parts, and 350 features took approximately 20–25 seconds. Retrieval time increases substantially for larger projects with more documents.

- Do you have any other final feedback or thoughts you would like to share?