CAUSAL-CAD: LEARNING CAUSAL MODELS FOR ROOT-CAUSE ANALYSIS OF PARAMETRIC CAD ERRORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Parametric computer-aided design (CAD) stores both an ordered feature history and a graph of the references consumed by each feature. A local edit can propagate through this dependency graph and prevent distant downstream features from regenerating, while the CAD system often reports only where regeneration stopped. We present a system that learns executable causal models from Autodesk Fusion 360 dependency graphs before and after edits. Each model is a program defining a causal graph and structural functions over its variables. An LLM-guided quality-diversity search evolves these programs for several error families, and a live Fusion agent tests their explanations. Causal explanations from our system achieve a mean score of 0.238 on our repair task testbed, compared with 0.150 for an LLM-only explanation and 0.123 without an explanation. Performance also rises after validating the models in the live Fusion environment, demonstrating the value of integrating learning and acting in a real-world CAD setting.

1 INTRODUCTION

Computer-aided design (CAD) is essential for modern engineering and manufacturing (Robertson & Radcliffe, 2009; Ikubanni et al., 2022; Kyratsis et al., 2025). Software such as Autodesk Fusion 360 represents designs as histories of parametric operations, making each construction history an executable program as well as a geometric model (Willis et al., 2021). Many CAD systems represent solids as boundary representations (B-reps) composed of faces, edges, vertices, and their topological relationships. Features create and modify these entities, while later features in the timeline retain references to them. A local edit can therefore affect geometry referenced elsewhere in the feature history (Bidarra et al., 2005). Some errors remain hidden until a parameter change forces regeneration, further separating the visible symptom from its cause (Otto & Mandorli, 2024). Root-cause analysis is required to identify the underlying causes of these failures (Yang & Han, 2006).

We learn an executable causal model through evolutionary programming (Romera-Paredes et al., 2024; Liu et al., 2024) that predicts an error from changes between design histories before and after an edit. Each model contains a Directed Acyclic Graph (DAG) of variables and structural functions that compute root, intermediate, and outcome variables. Root variables measure changes to design features, references, and geometry; intermediate variables represent how those changes propagate to an error. The Counterfactual Effect Size Model (CESM) estimates how strongly changes to each model variable affect the predicted error (Quillien, 2020; Quillien & Lucas, 2023). These scores and the corresponding causal paths help identify the root cause and a counterfactual change that would prevent the failure.

We evaluate the method on several families of Fusion 360 errors by asking a language-model agent to repair held-out failures through the Fusion 360 API. We compare causal explanations with no explanation and with an LLM-only explanation written without access to a learned causal model. We also measure how repair performance changes as the models are refined through a learning–acting loop. The resulting explanations remain traceable to the underlying CAD design and improve mean downstream repair performance across error families.

Our main contributions are:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

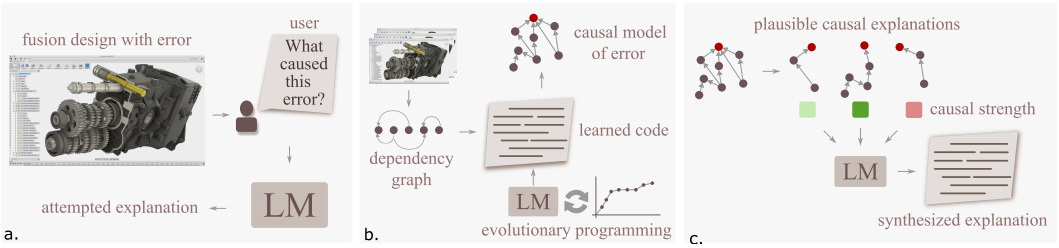

Figure 1: System overview. (a) An edit propagates through a CAD dependency graph and causes a downstream feature to fail. A user requests an explanation for the failure from a LLM, but the response is often unhelpful. (b) Evolutionary program search learns executable causal models from before-and-after dependency graphs and observed errors. (c) CESM ranks causes in a selected model, and an LLM converts the ranked paths and counterfactuals into an explanation.

- We formulate CAD root-cause analysis as causal reasoning over changes to a feature-dependency graph and represent candidate causal models as small, auditable programs.
- We introduce a learning–acting loop that combines LLM-guided evolutionary programming search with live verification, allowing model structure and causal vocabulary to improve together without manually labeled root causes.
- We evaluate causal explanations through downstream repair by LLM agents and the Fusion 360 API.

2 Related Work

Parametric CAD. Feature-based CAD represents a design through semantic features, parameters, and references rather than geometry alone (Bidarra & Bronsvoort, 2000). These references support editable design histories but introduce interactions between distant operations and the persistent-naming problem when topology changes (Cicirello & Regli, 1999; Bidarra et al., 2005). Work on change propagation formalizes how edits spread through product models (Ma et al., 2008), while design-history repair uses expert rules to reconstruct invalid feature sequences (Yang & Han, 2006). The Fusion 360 Gallery and DeepCAD later treated construction histories as programs for learning and generation (Willis et al., 2021; Wu et al., 2021). Our work instead learns causal models over changes in CAD dependency graphs and evaluates them through explanation-guided repair.

Causal explanations. Structural causal models define causes and explanations through interventions and counterfactual outcomes (Pearl, 2009; Halpern & Pearl, 2005). Counterfactual explanations are useful when they identify changes that would alter an outcome (Wachter et al., 2018), while human explanation research shows that people prefer selective, contrastive, and socially useful accounts rather than exhaustive descriptions (Miller, 2019). CESM formalizes causal selection by measuring association across plausible nearby counterfactual worlds (Quillien, 2020; Quillien & Lucas, 2023), with stability-weighted sampling controlling similarity to the observed case (Lucas & Kemp, 2015). We apply this framework to executable causal models learned from CAD dependency data.

Fault diagnosis. Fault diagnosis has long combined model-based, process-history, data-driven, and hybrid methods (Venkatasubramanian et al., 2003; Isermann, 2005; Soleimani et al., 2021). Model-based approaches compare observed behavior with expected system dynamics; data-driven approaches learn signatures from recorded failures. CAD adds a particular challenge because the reported feature may be downstream of the edit that caused the failure. Recent CAD work uses controlled alterations and graph knowledge to expose dormant associativity deficiencies and backtrack to their causes (Otto & Mandorli, 2024). Tool-using LLM agents have also gathered evidence for root-cause analysis in software infrastructure (Wang et al., 2024). We separate these roles: learned causal models propose mechanisms, while a Fusion agent verifies them and attempts repair.

LLM-guided program search. LLMs can propose executable programs as mutations within an evaluate-and-select loop. Evolution through Large Models combines LLM variation with MAP-Elites (Lehman et al., 2022); FunSearch couples LLM program generation with automated evaluation (Romera-Paredes et al., 2024); and EoH evolves both heuristic ideas and executable code (Liu et al., 2024). OPRO similarly conditions an LLM on previous solutions and objective values to generate improved candidates (Yang et al., 2024). Quality-diversity search retains multiple high-performing behaviors rather than a single optimum (Lim et al., 2024). Memery & Subr (2026) synthesize transparent programs that map low-level simulation traces to sparse high-level patterns. We use LLM-guided quality-diversity search to evolve executable causal models and preserve competing explanations for live verification.

3 BACKGROUND

Fusion dependency graphs. Modern CAD models are parametric: designers encode design intent as features, parameters, and references rather than as a static mesh, so that a family of variations can be produced by editing parameters. A design is therefore represented as a feature history, an ordered timeline of operations (sketches, extrudes, fillets, patterns, Boolean operations, and others) that are re-evaluated whenever the model changes (Cicirello & Regli, 1999). While the timeline represents the order in which features were created, each feature also consumes geometric or topological references produced or modified by earlier features. For example, an extrude references a sketch profile, a fillet references a set of edges, and a pattern references a set of faces or bodies. Because a feature can reference geometry from any earlier point in the timeline, these references trace out a second structure that can differ from the creation order.

The reference relationships form a directed acyclic dependency graph (DAG): its nodes represent features, sketches, and construction geometries, and its typed edges record reference relations such as ownership, a required reference, or the Boolean sequence that produces intermediate geometry and topology. The geometric modeling kernel executes this graph, rather than the raw timeline, to generate the resulting B-rep bodies.

As the dependency graph grows, editing one feature can propagate through it and cause distant features to fail to regenerate. Modern CAD systems flag the failed feature in the timeline and report a brief error message, although the root cause often lies elsewhere in the feature history.

Fusion error families. Raw Fusion messages are too coarse for causal-model learning, so we group episodes into eight mechanism-oriented families. `MISSING_REF_GEOMETRY` and `MISSING_REF_SKETCH` separate lost geometric references from lost sketch or support references. `NO_PROFILES` covers sketches that no longer contain a valid closed region. `BOOLEAN_FAILURE`, `PATTERN_FAILED`, and `FILLET_CHAMFER_FAILED` capture operation-specific precondition failures. `SELF_INTERSECTION` and `TOPOLOGY_FAILURE` capture invalid resulting geometry. Appendix A defines each family and its typical trigger.

Quality-diversity search with CVT-MAP-Elites. Quality-diversity (QD) search maintains high-performing solutions that differ along user-chosen behavioral descriptors (Mouret & Clune, 2015; Cully & Demiris, 2018). MAP-Elites is a canonical QD algorithm. CVT-MAP-Elites (Vassiliades et al., 2018) extends it to general descriptor spaces through a centroidal Voronoi tessellation, retaining the best solution (the elite) in each region.

Our search combines CVT-MAP-Elites with LLM-based code evolution (Vassiliades et al., 2018; Lehman et al., 2022). A search run maintains four parallel islands: three index candidates by different pairs of structural descriptors, and one uses all three. The LLM mutates an archived causal-model program using its source code, fitness, and evaluator feedback. Each proposal is parsed, executed on training episodes, scored, and assigned to a CVT cell, where it replaces the incumbent only when fitter. Section 4 defines the descriptors, causal-model representation, and fitness. We will make the implementation code and experiment configurations publicly available.

Counterfactual causal selection. Every failure has many contributing factors, but a useful explanation selects the one that best accounts for it. We adopt the counterfactual theory of causal selection of Quillien & Lucas (2023): when people judge whether a factor C caused an outcome

E, they imagine alternative ways the situation could have unfolded that are both a priori likely and close to what actually happened, and judge C a cause of E to the extent that C and E are correlated across those alternatives. The two ingredients are therefore a sampler over counterfactuals and a measure of causal strength.

Counterfactuals are sampled from a causal model using the stability-parameter scheme of the extended structural model (Lucas & Kemp, 2015). For each exogenous variable X, a sample either preserves its actual-world value x^* or is drawn afresh from the variable's prior:

$$P_s(X = x) = s\,\mathbb{I}[x = x^*] + (1 - s)P(X = x), \tag{1}$$

where $s \in [0, 1]$ trades similarity to what happened against prior plausibility; $s = 0$ recovers prior sampling, whereas $s = 1$ reproduces the actual assignment. Endogenous variables are then recomputed from the model's structural equations. Given these samples, CESM (Quillien, 2020) assigns a candidate cause C a standardized effect on outcome E. In an unconfounded model this reduces to the correlation between C and E across sampled counterfactual worlds: C is strong when changing it reliably changes whether E occurs. Our implementation approximates nearby-world sampling with the type-specific root perturbations described in Section 4.

4 METHOD

Figure 1 summarizes the pipeline, Algorithm 1 gives one complete round of our method, and Appendix B lists the LLM prompts.

Episodes and data splits. An edit episode is

$$z = (d, u, D^{\mathrm{pre}}, D^{\mathrm{post}}, \Delta D, y), \tag{2}$$

where d is the design, u the edit, D^{pre} and D^{post} the CAD dependency graphs before and after u, ΔD their structural difference, and y the observed error family or no-error outcome. Nodes in D represent features, sketches, bodies, and construction geometry; typed edges record references consumed or produced by features. We split episodes into $\mathcal{Z}_c^{\mathrm{tr}}$, $\mathcal{Z}_c^{\mathrm{val}}$, and $\mathcal{Z}_c^{\mathrm{te}}$. Training episodes determine evolutionary fitness, validation episodes are investigated during acting, and test episodes are reserved for repair evaluation.

Executable causal models. A candidate executable causal model is a Python program $M = (V, E, F)$. The ordered variables $V = R \cup I \cup \{Y_c\}$ comprise roots R, semantic intermediate variables I, and the binary outcome Y_c. Each variable v has an executable function $f_v \in F$. Root functions derive measurements directly from the episode graphs,

$$x_r = f_r(D^{\mathrm{pre}}, D^{\mathrm{post}}, \Delta D), \qquad r \in R, \tag{3}$$

while intermediate and outcome functions read previously computed parent values,

$$x_v = f_v\big(x_{\mathrm{pa}(v)}\big), \qquad v \in I \cup \{Y_c\}. \tag{4}$$

An edge $(p, v) \in E$ is induced when f_v reads x_p. Executing functions in declared order makes E acyclic and yields the prediction $\hat{y}_M(z) = x_{Y_c}$.

Evolutionary causal-model search. An LLM mutates model code by adding graph-grounded roots, changing intermediate variables, rewiring parent reads, or rewriting structural functions. A proposal must parse and satisfy the model schema. Let $\mathbf{y}$ be the binary training labels and $\hat{\mathbf{y}}_M$ the model predictions. Candidate quality is

$$q(M) = F_1^{\mathrm{macro}}(\mathbf{y}, \hat{\mathbf{y}}_M). \tag{5}$$

CVT-MAP-Elites preserves code diversity using

$$\phi(M) = \left(|V| - 1,\ \frac{\mathrm{LOC}(M)}{|V| - 1},\ \frac{\mathrm{ParentReads}(M)}{|V| - 1} \right), \tag{6}$$

namely the number of causal variables excluding the outcome, non-empty non-comment lines per variable, and parent-value reads per variable. Parent reads count source accesses through `parents[...]` or `parents.get(...)` and approximate causal connectivity. Four parallel islands use the three pairwise projections of ϕ and the full vector. Descriptors determine archive location, not fitness; a candidate replaces the occupant of its CVT cell only if q is higher.

Algorithm 1 Learning–acting loop for one error family.

Require: Training episodes $\mathcal{Z}_c^{\mathrm{tr}}$, validation episodes $\mathcal{Z}_c^{\mathrm{val}}$, archive $\mathcal{A}_0$, feedback K_0, vocabulary L_0, rounds T

for $t = 0, \ldots, T-1$ **do**
  $P_t \leftarrow \mathrm{Prompt}(\text{schema}, K_t, L_t)$
  $\mathcal{A}_{t+1} \leftarrow \mathrm{CVTMapElites}(\mathcal{A}_t, P_t, q, \phi)$
  $\mathcal{S}_{t+1} \leftarrow \mathrm{SelectDiverseModels}(\mathcal{A}_{t+1})$
  $\mathcal{T}_{t+1} \leftarrow \emptyset$
  for $z \in \mathcal{Z}_c^{\mathrm{val}}$ **do**
    $\mathcal{E}_z \leftarrow \mathrm{ExplainCESM}(\mathcal{S}_{t+1}, z)$
    $\tau(z) \leftarrow \mathrm{InvestigateFusion}(z, \mathcal{E}_z)$
    $\mathcal{T}_{t+1} \leftarrow \mathcal{T}_{t+1} \cup \{\tau(z)\}$
  end for
  $(K_{t+1}, L_{t+1}) \leftarrow \mathrm{Refine}(K_t, L_t, \mathcal{T}_{t+1}, \mathcal{A}_{t+1})$
end for
return $\mathcal{A}_T, K_T, L_T$

Explanation generation. From the archive $\mathcal{A}_t$, we select high-fitness models with dissimilar variable vocabularies. For a validation episode, each model first computes the observed variable values and error prediction. We then generate 250 nearby counterfactual worlds by perturbing all root values: Boolean roots may flip, numeric roots vary around their observed value, and categorical roots sample from their declared alternatives. The model recomputes every downstream variable in each world.

CESM assigns each root variable the fraction of worlds in which changing that root also changes the error prediction:

$$s(r) = P(\text{error prediction changes} \mid r \text{ changes}). \tag{7}$$

Roots are ranked by $s(r)$, and outcome-flipping assignments become counterfactual explanation items. An LLM combines the ranked roots, causal paths, and dependency-graph evidence into one engineering explanation, removing claims unsupported by the observed graph data.

Acting and feedback. The acting agent receives the explanations but is not told that any is correct. With live access through the Fusion MCP service Autodesk (2026); Anthropic (2024), it inspects the feature history, reference state, health messages, bodies, sketches, and geometry. It also makes edits to test hypotheses. For each validation episode it returns

$$\tau(z) = (m^*, C^-, e, c'), \tag{8}$$

where m^* is the best-supported model or null, C^- the contradicted variables, e the observed evidence, and c' an alternative cause when no candidate holds. The acting agent is given up to 16 actions per episode.

The acting agent updates two persistent objects: mechanism feedback K_t and a variable vocabulary L_t. An LLM removes stale claims, adds supported mechanisms, records failed API approaches, and proposes at most a few new dependency-graph-computable variables. We write this refinement as

$$(K_{t+1}, L_{t+1}) = \mathrm{Refine}\big(K_t, L_t, \{\tau(z) : z \in \mathcal{Z}_c^{\mathrm{val}}\}, \mathcal{A}_t\big)\,. \tag{9}$$

The next mutation prompt concatenates the fixed model schema with K_{t+1} and L_{t+1}.

5 Experiments

We use `gpt-oss-120b` (OpenAI, 2025) for causal-model mutation, explanation synthesis, live Fusion investigation, repair, and semantic classification. Unless stated otherwise, the same model is used across all experiments.

Causal explanations for CAD repair. We evaluate on held-out episodes from the eight error families in Section 3. Causal-model search uses training episodes, and the learning–acting loop uses

Error family	No explanation	LLM-only explanation	Causal explanation
MISSING_REF_GEOMETRY	0.098 ± 0.022	0.217 ± 0.021	$\mathbf{0.219 \pm 0.022}$
BOOLEAN_FAILURE	0.111 ± 0.031	0.111 ± 0.031	$\mathbf{0.268 \pm 0.042}$
FILLET_CHAMFER_FAILED	0.199 ± 0.035	0.335 ± 0.033	$\mathbf{0.385 \pm 0.033}$
PATTERN_FAILED	0.049 ± 0.011	0.046 ± 0.013	$\mathbf{0.106 \pm 0.014}$
NO_PROFILES	0.065 ± 0.024	0.067 ± 0.023	$\mathbf{0.095 \pm 0.023}$
SELF_INTERSECTION	0.111 ± 0.035	0.111 ± 0.033	$\mathbf{0.244 \pm 0.036}$
TOPOLOGY_FAILURE	0.068 ± 0.019	0.105 ± 0.016	$\mathbf{0.321 \pm 0.029}$
MISSING_REF_SKETCH	$\mathbf{0.279 \pm 0.035}$	0.208 ± 0.031	0.266 ± 0.034
Mean	**0.123**	**0.150**	**0.238**

Table 1: Held-out CAD repair performance on the $[0, 1]$ scale of our repair metric. Family-level values report mean overall score $\pm$ standard deviation across repeated LLM sampling seeds. The final row is the mean across families.

validation episodes. For each test episode, the repair agent receives live Fusion access and must clear the failure without reverting the edit that caused it.

We compare three inputs to the agent: "no explanation" provides access to the design without any root-cause analysis. The LLM-only explanation is generated directly from the episode and dependency-graph diff. The causal explanation uses a selected causal model's CESM-ranked factors and paths through its causal graph. All conditions use the same LLM, Fusion interface, repair prompt, episode set, and scoring procedure. We average attempts within each family and condition across 25 LLM sampling seeds.

The overall score rewards an error-free repair that preserves the triggering edit and remains structurally close to the clean design. It combines error resolution, continuous similarity, and the root-cause-quality penalty for removing an erroring feature where an in-place repair is expected.

Each attempt receives error resolution $s_{\text{err}} \in \{0, 1\}$, structural similarity $s_{\text{sim}} \in [0, 1]$, and root-cause quality $s_{\text{root}} \in [0, 1]$. The overall score is

$$s_{\text{overall}} = s_{\text{err}} s_{\text{sim}} s_{\text{root}}. \tag{10}$$

Structural similarity compares the repaired design with the clean pre-edit design using body count, sketch count, timeline length, active-feature count, total feature count, and component count. For each available nonzero count a and b, similarity is $1 - |a - b| / \max(a, b)$; s_{sim} is the mean across fields, with an exact match scoring 1. Appendix A gives the full scoring details.

Causal explanations reach a mean score of 0.238, compared with 0.150 for LLM-only explanations and 0.123 without an explanation (Table 1). Family-level standard deviations range from 0.011 to 0.042 across seeds, and causal explanations improve on the LLM-only baseline by 0.088.

The improvement of the causal explanations varies by error family. The largest gains over LLM-only explanations occur for TOPOLOGY_FAILURE (0.321 versus 0.105), BOOLEAN_FAILURE (0.268 versus 0.111), and SELF_INTERSECTION (0.244 versus 0.111). These are cases in which the reported symptom can be separated from the earlier geometric or dependency condition that makes the operation infeasible. Causal explanations provide smaller gains for FILLET_CHAMFER_FAILED, PATTERN_FAILED, and NO_PROFILES. For MISSING_REF_GEOMETRY, causal and LLM-only explanations perform similarly; for MISSING_REF_SKETCH, neither explanation condition exceeds the no-explanation score. The learned models are useful on average, but their benefit depends on the error family. Table 2 pairs two explanations with the triggering edit and an overview of the root and intermediate variables in a representative selected model.

Performance over the learning–acting loop. We test whether later causal models yield more useful explanations than earlier models. For each error family, we select five learning runs spaced across the available history and take the highest-fitness candidate model from each run. A single pinned model generates the causal explanation at each checkpoint. The repair agent evaluates every checkpoint on the same ten episodes within a family.

Action	Example model used in synthesis	Causal explanation
Set the first extrude to a negative distance, reversing its direction.	**Roots:** `pattern_features_before`, `zombie_pattern_features`, removed `MULTIPLICITY_ESSENTIAL`, `ESSENTIAL`, `BODY_EXISTENCE`, and `JOIN_INTERSECT_TARGET` counts, construction-axis loss, and body counts. **Intermediates:** `pattern_source_dependency_broken` → `pattern_geometric_context_broken` → `pattern_intersection_would_fail` → `PATTERN_FAILED`.	The `extrude_negative_distance` edit triggered a `PATTERN_FAILED` error by breaking a **MirrorPattern** feature's dependency on its construction geometry. Raw data confirmed a `MirrorPattern` (`entity_type='mirror'`) existed pre-edit, with no removed `BODY_EXISTENCE` or `JOIN_INTERSECT_TARGET` edges. However, the extrusion severed **ESSENTIAL** dependencies linking the pattern to its construction axis/plane, collapsing its replication context. The error occurred because the pattern lost its geometric reference, despite no bodies being deleted (count remained 342). **Root cause:** The extrusion invalidated the `MirrorPattern`'s dependency on construction geometry. **Recommendation:** Suppress dependent patterns pre-edit or ensure construction references remain intact.
Suppress the first extrude, removing all downstream geometry.	**Roots:** removed node, edge, body, and feature counts; body counts before/after; newly suppressed/zombie counts; removed `ESSENTIAL`, `BODY_EXISTENCE`, `JOIN_INTERSECT_TARGET`, `OWNERSHIP`, and transform dependencies. **Intermediates:** `geometry_reference_edges_lost`, `construction_geometry_orphaned`, `multiple_geometry_refs_broken` → `MISSING_REF_GEOMETRY`.	The `extrude_suppress_first` action suppressed an extrude feature, severing **BODY_EXISTENCE** and **ESSENTIAL** dependency edges—both high causal strength factors. Raw data confirmed removal of at least one **BODY_EXISTENCE** edge, breaking downstream references to the suppressed feature's output body. Additional **JOIN_INTERSECT_TARGET** edges were lost, compounding the failure. The error occurred because orphaned geometry (faces/bodies) lost critical references, triggering `MISSING_REF_GEOMETRY` when Fusion 360 could no longer resolve dependencies. **Recommendation:** Audit suppressed features for **BODY_EXISTENCE** and **ESSENTIAL** edges; rehome references to stable geometry.

Table 2: Examples of learned causal models and explanations. The model column summarizes one input model; each explanation synthesizes evidence from several selected models.

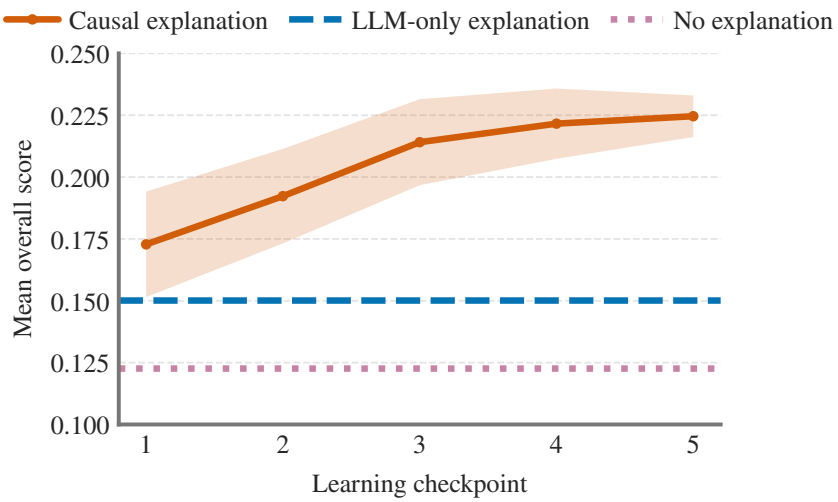

Figure 2: Mean repair performance across learning–acting checkpoints using different LLM seeds on the same tasks. Shading shows one standard deviation across seeds; horizontal lines show Experiment 1 baselines.

Figure 2 reports performance at five learning checkpoints. Every checkpoint evaluates different LLM seeds on the same fixed tasks. The curve shows the mean across seeds and the shaded region shows one standard deviation. Exact numerical values are provided in Appendix A.

Mean repair performance rises from 0.1728 ± 0.0213 at the first checkpoint to 0.2246 ± 0.0084 at the fifth, an absolute improvement of 0.0518, a 29.9% increase. The means increase at every checkpoint: by 0.0195 from checkpoint 1 to 2, 0.0218 from 2 to 3, 0.0075 from 3 to 4, and 0.0030 from 4 to 5. Roughly 80% of the total gain is therefore achieved by checkpoint 3.

The standard deviation decreases from 0.0213 to 0.0084 across the same checkpoints. Later archives produce both higher mean repair scores and less dispersed outcomes, while the smaller final increments indicate diminishing returns after checkpoint 3.

Cross-family semantic similarity by causal depth. Each error family has a separate causal-model archive and no shared graph template. We analyze up to 100 models per family across nine families, giving 14,829 variable occurrences in 678 models. Static analysis reconstructs each model's causal edges from accesses to parent variables. For every variable on a directed path to the error outcome, we compute the shortest path length d. Variables with no path to the outcome are excluded.

Different models often use different names for the same causal concept, so we classify each variable from its name and description into one of 25 fixed semantic classes. Three independent LLM classifications are combined by majority vote. Variables without a valid majority are excluded rather than assigned to a residual class. Appendix Table 7 lists the complete taxonomy and definitions.

After mapping variable names and descriptions to 25 semantic classes, we count occurrences by family, depth, and class. Let $p_{f,d}$ be the normalized class-frequency vector for family f at depth d. With nine error families there are $\binom{9}{2} = 36$ unique family pairs. We retain pairs with at least five classified occurrences per family at depths 1–3. For each retained pair (i, j), semantic similarity is

$$\begin{aligned} m &= \frac{p_{i,d} + p_{j,d}}{2}, \\ S_{ij}(d) &= 1 - \frac{1}{2}\mathrm{KL}_2(p_{i,d} \| m) - \frac{1}{2}\mathrm{KL}_2(p_{j,d} \| m). \end{aligned} \tag{11}$$

where KL_2 uses base-2 logarithms, so $S_{ij}(d) \in [0, 1]$. The matched statistic compares the same pair near the error and one layer farther upstream:

$$\Delta_{ij} = \frac{S_{ij}(d = 1) + S_{ij}(d = 2)}{2} - S_{ij}(d = 3). \tag{12}$$

Positive Δ_{ij} indicates greater similarity near the error. We average Δ_{ij} over the 36 family pairs with support at all three depths. For each of 500 bootstrap replicates, we resample learning runs within each family and then models within each sampled run, recompute the family distributions, and take the 2.5th and 97.5th percentiles of the resulting mean Δ. This preserves shared mutation lineage within a run.

Across family pairs, downstream similarity is 0.551 and depth-3 similarity is 0.437, giving $\Delta = 0.115$ with a 95% interval of $[0.083, 0.175]$. For a null test, we shuffle semantic labels among variables within each model while preserving its depth positions and class frequencies, then recompute the mean Δ 1,000 times. The null mean is 0.036, and none of the permutations reaches the observed value ($p < 0.001$). This provides exploratory evidence for an association between causal depth and cross-family semantic similarity, not shared physical mechanisms. Restricting the analysis to fully classified models gives 0.163; deeper layers are too sparse for inference. Full coverage and sensitivity results are in Appendix A.

6 Discussion

The learning–acting loop. Predictive fitness can reward downstream correlates rather than the mechanism that caused a failure. The acting phase tests learned models against live Fusion feature histories, references, feature health, bodies, sketches, and geometry. Its findings guide later search rounds toward variables and causal paths that are more relevant to the actual failure.

Experiment 2 supports this learning–acting process. Mean repair performance increases from 0.1728 at the first checkpoint to 0.2246 at the fifth, while the standard deviation falls from 0.0213 to 0.0084. Most of the gain is achieved by checkpoint 3, consistent with live Fusion investigations correcting major omissions before later rounds produce diminishing returns.

Each checkpoint includes both further search and acting-derived feedback, so the experiment does not isolate acting as the cause of improvement. A matched ablation with and without acting feedback would be needed for that claim. The current result shows that models become more useful for repair as live Fusion evidence is incorporated into the loop.

Expert feedback on explanations. Three Fusion CAD professionals independently reviewed five designs each. For every design, they saw one LLM-only explanation and one explanation from our method side by side. They valued explanations that traced the dependency graph from the first upstream change through the failure cascade. One reviewer described our system as providing "a more detailed analysis of the dependency graph" and "more reasonable root-cause analysis." Another repeatedly found the more structured explanation "clearer" and "easier to follow." These comments support presenting the triggering edit, first failing feature, propagation path, and recommended fix as separate parts of an explanation.

The feedback also exposed errors that aggregate repair scores do not describe. For one episode, a reviewer wrote that "both explanations are bad" because neither named the modified `Extrude1` nor the failing `Extrude10`. Another found a recommended fix "completely incorrect and cannot produce a valid design", implying that our learned causal models may not provide enough context for accurate repairs. The expert comments favour explanations that identify the earliest relevant dependency change and verify that the proposed repair is geometrically valid, rather than descriptions of downstream graph changes alone. Appendix C reproduces the complete comments quoted here.

A shared downstream semantic vocabulary. Experiment 3 suggests a modular structure in the learned causal models. At depth 3, variables remain tied to family-specific inputs, including `sketch_availability`, `edge_availability`, `body_availability`, and `operation_extent_validity`. Near the error, models more often use shared terminal concepts such as `reference_target_availability`, `dependency_chain_preservation`, `downstream_feature_orphaning`, `operation_target_availability`, and `geometric_degeneracy`. These downstream concepts could be reused across error families, while upstream detectors remain specialized to particular edits and references.

This is a property of our learned representations but shared prompts and a common outcome structure may lead to some similarity. Further evidence is needed before making claims about causal transfer between error families. The shared vocabulary is a promising direction for future work, as it may allow causal models to be learned more efficiently and with less data.

7 Limitations and Conclusion

The episode generator and dependency-graph extractor bound which causal variables the models can represent; mechanisms absent from the extracted graphs cannot be learned. Predictive fitness may also reward correlates rather than physical causes, and live Fusion validation reduces but does not eliminate this risk. Experiment 2 evaluates the combined effect of additional search and acting feedback, so it does not isolate acting through an ablation. The models are learned from one Fusion 360 dataset, separately for each error family, and transfer to other designs, error families, or CAD systems remains untested. Finally, we keep the LLM fixed to study causal-model learning and not the impact of different LLMs, but model choice may affect both evolved programs and repair behavior.

We learn executable causal models from changes in CAD dependency graphs, rank causes through counterfactual testing, and verify explanations in live Fusion 360 sessions. These explanations remain traceable to the underlying CAD design and improve mean downstream repair performance across error families. We believe that learning causal models for root-cause analysis of CAD errors is a rich space that is yet to be thoroughly explored. We hope that our work has shown the potential effectiveness of LLM-driven learning of causal models for this task, and that it will inspire future work in this area.

AI use statement

Generative AI tools were used to assist with writing code for this work. LLM-generated code was verified and tested for correctness by the main author. The authors take responsibility for the final content of this work.

REFERENCES

Anthropic. Model context protocol. https://modelcontextprotocol.io/, 2024. Accessed 2026-09-24.

Autodesk. Introducing the fusion mcp: Opening fusion to ai-powered workflows. https://www.autodesk.com/products/fusion-360/blog/introducing-the-fusion-mcp-opening-fusion-to-ai-powered-workflows/, 2026. Autodesk Fusion 360 Blog, accessed 2026-09-24.

Rafael Bidarra and Willem F. Bronsvoort. Semantic feature modelling. *Computer-Aided Design*, 32 (3):201–225, 2000. doi: 10.1016/S0010-4485(99)00090-1.

Rafael Bidarra, Patrick J. Nyirenda, and Willem F. Bronsvoort. A feature-based solution to the persistent naming problem. *Computer-Aided Design and Applications*, 2(1–4):517–526, 2005. doi: 10.1080/16864360.2005.10738399.

Vincent A. Cicirello and William C. Regli. Resolving non-uniqueness in design feature histories. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*, pp. 76–84, 1999. doi: 10.1145/304012.304020.

Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular framework. *Evolutionary Computation*, 26(1):173–210, 2018. doi: 10.1162/EVCO_a_00199.

Joseph Y. Halpern and Judea Pearl. Causes and explanations: A structural-model approach. part i: Causes. *The British Journal for the Philosophy of Science*, 56(4):843–887, 2005. doi: 10.1093/bjps/axi147.

Peter P Ikubanni, Adekunle A Adeleke, Olayinka O Agboola, Chiebuka T Christopher, Boluwatife S Ademola, Joseph Okonkwo, Olanrewaju S Adesina, Peter O Omoniyi, and Esther T Akinlabi. Present and future impacts of computer-aided design/computer-aided manufacturing (cad/cam). *Journal Européen des Systèmes Automatisés*, 55(3):349, 2022.

Rolf Isermann. Model-based fault-detection and diagnosis—status and applications. *Annual Reviews in Control*, 29(1):71–85, 2005. doi: 10.1016/j.arcontrol.2004.12.002.

Panagiotis Kyratsis, Anastasios Tzotzis, and João Paulo Davim. A review on cad-based programming for design and manufacturing. *CAD-based Programming for Design and Manufacturing*, pp. 1–26, 2025.

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. Evolution through large models. *arXiv preprint arXiv:2206.08896*, 2022.

Bryan Lim, Manon Flageat, and Antoine Cully. Large language models as in-context AI generators for quality-diversity. *arXiv preprint arXiv:2404.15794*, 2024. doi: 10.48550/arXiv.2404.15794.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language models. *arXiv preprint arXiv:2401.02051*, 2024. doi: 10.48550/arXiv.2401.02051.

Christopher G. Lucas and Charles Kemp. An improved probabilistic account of counterfactual reasoning. *Psychological Review*, 122(4):700–734, 2015. doi: 10.1037/a0039655.

Yongsheng Ma, Gang Chen, and Georg Thimm. Change propagation algorithm in a unified feature modeling scheme. *Computers in Industry*, 59(2–3):110–118, 2008. doi: 10.1016/j.compind.2007.06.006.

Sean Memery and Kartic Subr. Discovering high level patterns from simulation traces. *arXiv preprint arXiv:2602.10009*, 2026. doi: 10.48550/arXiv.2602.10009.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019. doi: 10.1016/j.artint.2018.07.007.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Harald E. Otto and Ferruccio Mandorli. Dormant deficiency: A novel concept to direct cause–effect CAD model analysis. *Research in Engineering Design*, 35:43–71, 2024. doi: 10.1007/s00163-023-00423-5.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2 edition, 2009.

Tadeg Quillien. When do we think that X caused Y? *Cognition*, 205:104410, 2020. doi: 10.1016/j.cognition.2020.104410.

Tadeg Quillien and Christopher G. Lucas. Counterfactuals and the logic of causal selection. *Psychological Review*, 130(6):1548–1580, 2023. doi: 10.1037/rev0000428.

BF Robertson and DF Radcliffe. Impact of cad tools on creative problem solving in engineering design. *Computer-aided design*, 41(3):136–146, 2009.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024. doi: 10.1038/s41586-023-06924-6.

Morteza Soleimani, Felician Campean, and Daniel Neagu. Diagnostics and prognostics for complex systems: A review of methods and challenges. *Quality and Reliability Engineering International*, 37(8):3746–3778, 2021. doi: 10.1002/qre.2947.

Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Transactions on Evolutionary Computation*, 22(4):623–630, 2018. doi: 10.1109/TEVC.2017.2735550.

Venkat Venkatasubramanian, Raghunathan Rengaswamy, Kewen Yin, and Surya N. Kavuri. A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. *Computers & Chemical Engineering*, 27(3):293–311, 2003. doi: 10.1016/S0098-1354(02)00160-6.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31(2):841–887, 2018.

Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Jihong Wang, Fengbin Yin, Lunting Fan, Lingfei Wu, and Qingsong Wen. RCAgent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 4966–4974, 2024. doi: 10.1145/3627673.3680016.

Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic CAD construction from human design sequences. *ACM Transactions on Graphics*, 40(4):1–24, 2021. doi: 10.1145/3450626.3459818.

Rundi Wu, Chang Xiao, and Changxi Zheng. DeepCAD: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6772–6782, 2021.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.03409.

Jeongsam Yang and Soonhung Han. Repairing CAD model errors based on the design history. *Computer-Aided Design*, 38(6):627–640, 2006. doi: 10.1016/j.cad.2006.02.007.

A Experiment Details

Experiment 1: Repair performance.

Explanation conditions. All conditions use the same repair agent, LLM provider, Fusion 360 tool interface, episode, turn budget, and scoring procedure. They differ only in the root-cause context supplied before live investigation.

Condition	Diagnostic context
No explanation	No generated root-cause analysis. The agent must inspect the broken design using Fusion tools.
LLM-only explanation	An explanation generated directly from the episode metadata and dependency-graph diff, without executing a learned causal model or applying CESM.
Causal explanation	An explanation generated from selected learned causal models, including active variables, paths through their causal graphs, and factors ranked through CESM counterfactual sampling.

Table 3: Inputs provided to the repair agent in Experiment 1.

The causal condition may use up to five high-fitness causal models selected for semantic and structural diversity. This avoids reducing the archive to one hypothesis when several programs have similar predictive performance but attribute the failure to different mechanisms. The explanation generator checks candidate claims against the episode's dependency-graph data before producing the repair context.

Repair metrics. Let s_{err} indicate error resolution, s_{sim} measure similarity to the clean pre-edit design, and s_{root} measure whether the erroring feature was repaired rather than removed. The episode score is

$$s_{\text{overall}} = s_{\text{err}}\, s_{\text{sim}}\, s_{\text{root}}. \tag{13}$$

Error resolution is binary. When Fusion reports unhealthy features, success requires those feature-health errors to be cleared. For failures that do not register an unhealthy feature, restoration of body and timeline state toward the clean baseline is used as a proxy. An unchanged broken state is explicitly rejected as a no-op.

The continuous score averages relative similarity over available body, sketch, timeline, active-feature, feature, and component counts. Root-cause quality distinguishes an in-place repair from deleting or suppressing the feature that exhibited the error. Removal receives zero for families where an in-place repair is expected. It receives partial credit, currently 0.7, for `BOOLEAN_FAILURE` and `FILLET_CHAMFER_FAILED`, where removing a correctly identified geometrically impossible operation can be a legitimate work-around.

Evaluated error families.

`MISSING_REF_GEOMETRY`	An upstream edit removes or changes the identity of a referenced face, edge, or body, so a downstream feature can no longer resolve its geometric input.
`MISSING_REF_SKETCH`	A sketch, its support plane or face, or a stored profile reference is deleted or suppressed, invalidating a downstream sketch-based operation.
`NO_PROFILES`	A sketch remains available but no longer contains a valid closed region, often because curves were removed, opened, or made self-intersecting.
`BOOLEAN_FAILURE`	A join, cut, or intersect operation lacks a valid tool or target body, or the participating bodies have no volumetric overlap.
`FILLET_CHAMFER_FAILED`	Referenced edges are missing or the requested radius or distance is incompatible with the local edge length, wall thickness, or face configuration.

PATTERN_FAILED	A pattern loses its seed or construction reference, creates invalid instances, or cannot combine the generated instances with the target geometry.
SELF_INTERSECTION	A sweep, loft, revolve, or blend folds through itself after an edit, producing intersecting faces or other invalid geometry.
TOPOLOGY_FAILURE	Regeneration produces a non-manifold, zero-thickness, inside-out, or otherwise inconsistent face–edge–vertex structure that the geometry kernel rejects.

	Learning checkpoint				
Statistic	1	2	3	4	5
Mean overall score	0.1728	0.1923	0.2141	0.2216	0.2246
Standard deviation	0.0213	0.0191	0.0174	0.0142	0.0084

Table 4: Repair performance across learning checkpoints. Checkpoints are ordered from earliest (1) to latest (5).

Experiment 2: Learning–acting performance over time.

Experiment 3: Semantic representation analysis.

Causal-model sample and semantic annotation. The analysis samples up to 100 causal models per family after restricting each family to the strongest quartile of its learning runs. It includes the eight families in Experiment 1 and the broader GEOMETRY_DEGENERATE archive. Superseded candidates are retained to characterise representations discovered throughout search.

Each variable is represented by its name, textual description, source error family, and causal depth. Three independent LLM calls assign a semantic class and mechanism stage. The majority label is retained. Variables without a valid majority are excluded; no residual "other" semantic class is used. The frozen taxonomy is listed in Table 7.

Occurrences versus definitions. The population contains 14,829 variable occurrences across 5,535 distinct variable definitions and 678 models. A variable that appears on a root-to-error path in many related models contributes once per model, so occurrences are not independent observations and the effective sample size is closer to the number of models and learning runs. All distributional statistics are computed over occurrences; all resampling is at the run and model level.

Coverage. Classification covers 11,351 of 14,829 occurrences (76.6%) and 3,280 of 5,535 definitions (59.3%), but coverage is uneven across families, depths, and variable roles (Tables 5–6). Coverage is lowest for FILLET_CHAMFER_FAILED, where only 18.7% of occurrences are classified, and for the deepest layers. Because of this imbalance, the aggregate matched statistic remains the only inferential result, and we report a sensitivity check that excludes models containing any unclassified variable.

Semantic category	Definition
body_availability	A required solid body still exists and is usable as input.
sketch_availability	A required sketch still exists and is usable as input.
profile_availability	A required closed sketch region still exists.
edge_availability	A required edge still exists and is selectable.
feature_availability	A required upstream feature still exists in the timeline.
reference_target_availability	The specific entity targeted by a reference still exists.
reference_resolvability	A reference resolves to a valid entity, even if its identity differs from the original.
dependency_edge_preservation	A dependency edge between two features survives the edit.
downstream_input_availability	A downstream feature's required input remains available.

Continued on next page.

Table 7 continued.

Semantic category	Definition
`dependency_chain_preservation`	A multi-hop dependency chain remains intact from source to consumer.
`downstream_feature_orphaning`	A downstream feature loses its producer and becomes orphaned or zombie.
`profile_validity`	A sketch profile remains closed and non-self-intersecting.
`geometric_degeneracy`	Geometry has zero or near-zero thickness, area, or volume.
`self_intersection`	Geometry intersects itself.
`local_geometry_feasibility`	Local geometry can support the requested operation, such as a fillet radius relative to an edge.
`tool_target_overlap`	Tool and target bodies overlap sufficiently for a Boolean operation.
`topology_validity`	The resulting face–edge–vertex structure is valid and consistent.
`operation_target_availability`	The feature or body on which an operation acts is available.
`operation_precondition_satisfaction`	Preconditions beyond entity availability are satisfied.
`operation_parameter_validity`	An operation parameter lies within a valid range.
`operation_extent_validity`	An extent, distance, or direction is valid for the current geometry.
`feature_output_availability`	A feature successfully produces its expected body, face, or other output.
`body_count_change`	The edit changes the number of bodies in the design.
`result_geometry_validity`	The final geometry produced by an operation is valid overall.
`downstream_feature_invalidation`	A downstream feature becomes unhealthy as a consequence of the edit.

Table 7: The 25 semantic categories used in Experiment 3.

Mechanism stages. The stage label is coarser than the semantic category and describes where a variable lies in the failure process. The seven allowed stages are listed in Table 8. The residual `other` value is available for stage annotation only; unlike the semantic taxonomy, it does not enter as a catch-all semantic concept.

Statistical controls. Semantic similarity is computed independently at each causal depth from the error node in each model's causal graph. The matched statistic compares the average similarity at depths 1 and 2 with similarity at depth 3 for the same family pair. Confidence intervals use a hierarchical bootstrap: learning runs are resampled first, then models within each resampled run, so variables within a model and models within a run remain grouped. A flat occurrence-level or model-level bootstrap would understate the uncertainty created by shared mutation lineage. The bootstrap does not correct for dependence induced by families sharing one LLM provider and one prompting framework; that dependence remains a limitation.

The shuffled-label null reassigns semantic labels among variables within each model while preserving model size, the number of variables at every causal depth, and the model's global semantic-class frequencies. This tests whether causal depth explains the observed similarity beyond class frequency. It is the feasible shuffled-label control; an untrained-model null would require re-running search. The generic-class ablation removes the most frequent downstream category and then the three most frequent downstream categories before recomputing the matched statistic.

Sensitivity to excluded variables. Recomputing the matched statistic over only the 370 of 678 models with no unclassified variables gives $\Delta = 0.163$, against 0.114 when every selected model is used. The aggregate effect is therefore not driven by models containing excluded variables; if anything, it is larger among fully classified models.

Descriptive per-pair, path, and stage results. The following results are descriptive and are not used as confirmatory evidence. Nineteen of 36 family pairs are individually significant under the shuffled-label test; because the 36 pairs reuse nine families, these tests are dependent and are not

Error family	Occurrences	Classified	Coverage
MISSING_REF_GEOMETRY	1,630	1,552	95.2%
SELF_INTERSECTION	1,541	1,501	97.4%
TOPOLOGY_FAILURE	2,592	2,509	96.8%
MISSING_REF_SKETCH	1,101	985	89.5%
NO_PROFILES	624	535	85.7%
BOOLEAN_FAILURE	1,944	1,644	84.6%
PATTERN_FAILED	1,125	822	73.1%
GEOMETRY_DEGENERATE	2,482	1,468	59.2%
FILLET_CHAMFER_FAILED	1,790	335	18.7%
All	**14,829**	**11,351**	**76.6%**

Table 5: Classification coverage by error family, measured over variable occurrences.

Stratum	Occurrences	Classified	Coverage
Depth 1	5,424	4,066	75.0%
Depth 2	6,998	5,488	78.4%
Depth 3	2,064	1,579	76.5%
Depth 4	283	189	66.8%
Depth 5	56	28	50.0%
Root variables	7,790	6,275	80.6%
Intermediate variables	7,039	5,076	72.1%

Table 6: Classification coverage by causal depth and variable role. Depths 4–5 are too sparse for inference.

multiple-comparison corrected. Representative root-to-error paths are selected from the top 20 models per family by the path with the highest average semantic-class overlap with the other candidates. Mechanism-stage distributions report the share of a family's classified occurrences in each stage. Full values for all three are in `outputs/semantic_causal_analysis.json`.

B LLM PROMPTS

The prompts below correspond to the LLM-dependent stages in Section 4. Dynamically inserted episode data, causal-model code, dependency-graph schemas, tool documentation, and prior feedback are shown as placeholders. Repeated API examples and formatting details are omitted for space; the task instructions are retained.

Causal-model evolution.

Causal-model mutation prompt

```
You are writing a Python causal model that predicts whether a
Fusion 360 edit caused a {error_family} error.

Error family: {error_family} -- {error_family_description}

You receive fusion_data containing only:
  graph_before: the dependency graph before the edit
  graph_after:  the dependency graph after the edit
  graph_diff:   added/removed nodes and edges, newly suppressed
                features, and newly zombie features

Derive every signal from these graphs. Do not use action or outcome
metadata from the episode. Every variable must be a concrete,
measurable property of the CAD design.

Define:
```

Stage	Interpretation
input_state	Raw entities, parameters, and geometry before feasibility is assessed.
reference_resolution	Whether a stored reference still resolves to a usable entity.
dependency_propagation	Transmission of an edit through dependency edges and downstream consumers.
operation_feasibility	Whether the requested feature operation can execute under the current conditions.
geometry_result	Validity of the geometry or topology produced by the operation.
downstream_consequence	Late effects such as unhealthy features, orphaning, or changed body counts.
other	A stage that cannot be assigned to the six process-specific categories.

Table 8: Mechanism-stage labels used in Experiment 3.

```
  ERROR_FAMILY = "{error_family}"
  NODES = [ordered variable declarations]
  NODE_FUNCTIONS = {
      variable_name:
          fn(fusion_data, episode, parents) -> value
  }

Root variables read fusion_data directly. Intermediate and outcome
variables read previously computed values from parents. List every
variable after all variables it depends on. Categorical variables
    must
declare every value their function can return.

The error function must discriminate positive error episodes from
comparison episodes. Prefer mechanisms grounded in removed
    references,
feature availability, operation preconditions, and resulting
    geometry
over direct copies of feature names.

{acting_feedback}
{shared_variable_vocabulary}

Return one complete, executable causal-model program.
```

Explanation synthesis.

Causal explanation synthesis prompt

```
You are a CAD engineering expert investigating a confirmed Fusion
    360
{error_family} error.

Dependency-graph evidence:
{graph_before_after_and_diff}

Candidate causal analyses:
{causal_paths_ranked_factors_and_counterfactuals}

Write one engineering explanation that:
1. Names the edit and the specific features, bodies, sketches, faces
    ,
   or edges involved.
2. Describes the causal path best supported by the raw graph
    evidence.
```

```
3. States the strongest counterfactual: what change would have
   prevented the error.
4. Gives the root cause and a concrete recommendation.

Treat candidate analyses as hypotheses. Drop claims unsupported by
    the
dependency graph, resolve disagreements using the raw evidence, and
    do
not invent thresholds or measurements. Use direct past-tense
    language.
Do not mention model IDs, fitness, internal variable names, or CESM.
Maximum 100 words.
```

Live Fusion verification.

Acting-agent investigation prompt

```
You are investigating a live Fusion 360 design to determine the real
cause of a {error_family} error following this edit:
{triggering_edit}

You have tools for executing Fusion Python inspection scripts and
reading document information.

Candidate causal models claim the following explanations:
{model_ids_active_nodes_and_top_counterfactual_factors}

Investigate which explanation, if any, is supported by the design.
Inspect the feature history, health states, sketches, bodies,
references, and geometry. You may make a small disposable edit to
    test
a hypothesis. Do not assume any candidate is correct.

When sufficient evidence has been collected, call submit_verdict
    once
with:
  best_supported_model_id
  contradicted_nodes
  concrete evidence from Fusion
  the actual cause if it differs from every candidate
```

Feedback incorporation.

Mechanism-feedback refinement prompt

```
You maintain feedback for the LLM that writes Python causal models
    for
{error_family}. Refine the current document using evidence from live
Fusion investigations.

Current feedback:
{current_feedback}

Investigation verdicts:
{
    supported_models_contradicted_variables_evidence_and_alternative_causes
    }

Variables used by the selected models:
{observed_model_variables}
```

```
Remove stale or contradicted claims. Add specific advice about what
    a
good model should measure, which intermediate variables were missing
    ,
and which structural functions were too simple. Every recommendation
must be computable from dependency-graph data. Preserve the existing
section structure and output only the refined feedback.
```

Shared-variable vocabulary refinement prompt

```
Refine the shared variable vocabulary for {error_family} causal
    models.

Current vocabulary:
{current_variable_vocabulary}

Weaknesses found by live investigation:
{investigation_summary}

Variables used by selected models:
{observed_model_variables}

Remove abstract, redundant, or unused variables. Add at most three
variables that directly address an observed weakness. Every variable
must have a snake_case name, one-sentence definition, and a boolean,
numeric, or categorical type, and must be computable from the
    dependency
graphs or their diff. Return only a JSON array.
```

Repair evaluation.

Repair-agent prompt

```
You are fixing a live Fusion 360 design with a {error_family} error.
The triggering edit has already been applied:
{triggering_edit_and_parameters}

Current broken-state summary:
{broken_design_state}

Diagnostic context for this condition:
{none_or_llm_explanation_or_causal_explanation}

1. Inspect the live design before changing it.
2. Identify the root cause, not only the reported symptom.
3. Apply a targeted repair without undoing the triggering edit.
4. Re-inspect the design and verify that the error is gone.

Do not suppress several features as a first move. Prefer repairing a
recoverable reference or parameter. If the operation is
    geometrically
impossible and has no recoverable target, modify or suppress only
    the
specific offending feature. Derive numeric values from the actual
geometry rather than guessing.

Finish by calling finish_fix with the specific change made and why
    it
```

```
addresses the root cause.
```

Semantic representation analysis.

Semantic-variable classification prompt

```
You are labeling causal variables from Fusion 360 failure models
    with a
fixed taxonomy, so variables with the same underlying causal meaning
     can
be compared across independently learned models.

Two variables share a semantic_class only if replacing one with the
other preserves the meaning of the causal step. For each variable,
assign:
  semantic_class: exactly one of {semantic_classes}
  stage: exactly one of {
      input_state,
      reference_resolution,
      dependency_propagation,
      operation_feasibility,
      geometry_result,
      downstream_consequence,
      other
  }

Return only a JSON array in the input order:
[{"i": index, "semantic_class": "...", "stage": "..."}, ...]

Variables:
{variable_names_descriptions_and_error_families}
```

C Expert Feedback Examples

Three Fusion CAD professionals independently reviewed five designs each. For every design, one LLM-only explanation and one causal-model explanation from our method were presented side by side. The Discussion quotes the following complete comments. In the review materials, Method/Explanation A is the LLM-only baseline and Method/Explanation B is our causal-model explanation. Spelling and emphasis are lightly normalized for typesetting, but the substance is unchanged.

Reviewer 1: Overall comparison

Essentially, both explanations reflect the underlying dependency graph.
Method A. Method A generally provides raw information about the dependency graph and gives simple explanations that directly reflect the status of the graph. However, the suggested fixes, such as suppressing features, are not particularly useful for addressing the root causes of the failures. It also focuses primarily on presenting raw dependency graph information rather than suggesting good design principles or useful rules to follow.
Method B. Method B provides a more detailed analysis of the dependency graph and offers more reasonable root-cause analysis of the failures. The explanations are also presented in a more structured way.
I am not sure whether the editing operation was used as an input, but Method B seems to identify the root cause more accurately. It also provides better analysis of the consequences of the operation and explains more clearly how the operation leads to errors.

Reviewer 2: Episode 1 presentation feedback

Explanation B is clearer and visually easier to follow than Explanation A, which is described in a paragraph. In this case the explanations identified different resulting error families.
Explanation A uses inline code formatting for the errors and edits, which is helpful, but sentences with a bold header should have their own paragraph or section.
Explanation B has a formatting error: "Recommendation" should start as a fifth list item. The resulting error family and edit vocabulary should use inline code formatting.

Reviewer 3: Episode 1 causal accuracy

Both explanations are bad. The feature modified to cause the error (`Extrude1`) was not given. Neither explanation called out the failing `Extrude10`. This is obvious from looking at the model or using the MCP tools.
Single-stepping through the timelines before and after the edit shows catastrophic loss of design intent following damage to `Extrude1`. The geometry no longer looks like a model of a room.
The style of Explanation B is nicer, with less enum-like jargon.
Using the out-of-the-box MCP tools and Composer 2.5, two prompt iterations were sufficient to perform the root-cause analysis. This is the kind of information users will be looking for.

Reviewer 1: Episode 6 repair validity

Although Method B provides a more readable explanation, the recommended fix is completely incorrect and cannot produce a valid design.