

Date: 2026-09-27. Branch head b9d4bdd2, merge base 36580826, 10 commits, 171 files, +36,218 / -13,079.

Subject: the autolayout work (core composition, OGDF bridge, settings resolution, UI) and the replacement of command-based undo/redo by first-write journal capture and change-set replay. Focus: code quality, architecture quality, robustness, performance.

Method: library sources read in full (view_undo_state.cpp/.h, undo.cpp/.h, messaging, auto_layout*.cpp, layout_ogdf.cpp, layout_description.cpp, layout_engine.cpp, view_geometry.cpp) plus the capture hooks in view.cpp, model.cpp, landscape.cpp, folders.cpp, labeled.h. Qt integration: app_manager.cpp (full diff), view_gui.cpp (gesture tokens, maintenance), view_gui_layout.cpp (autolayout execution), auto_layout_ui.cpp; the other Qt files through their diffs (see §8). Timings quoted are the branch’s own measurements from docs/20260924-undoredoanalysis.md. Nothing was built, run or profiled for this review.

Severity: **High** – conflicts with a stated project target or guarantee in normal use. **Medium** – latent defect, unmeasured risk against the scale targets, or structural debt that weakens a guarantee. **Low** – local quality.

1. Summary

The core of the undo/redo redesign is sound. One capture point (the first write through a setter), exact qualified record identities, prepare-then-apply replay with stale-state preflight and post-apply recapture, explicit Applied/Unchanged/Refused outcomes, retained native objects instead of reconstruction, reachable/unreachable save points, and history invalidation by connected component together remove the class of bugs the old per-command undo data had. The branch deletes a large amount of hand-maintained before/after state (e.g. the merge undo structs in import_merge.h). The autolayout core is cleanly layered: backend-neutral descriptions, a validated bridge registry, composition and validation in core, an OGDF bridge behind an approved exception boundary, and atomic application with a preflight equality check.

The significant problems sit around that core:

1. Forward view maintenance moved into the action finalizer and into landscape load, and its scope is derived from record *kinds*. Landscape- and model-level property edits fan out to full geometry maintenance of every view in scope, open or closed (P1, P2).
2. The capture guarantee holds only while a thread-local journal is bound; unbound writes pass silently. Public-field structs and void setters that drop refused writes make “single capture authority” a convention rather than an enforced invariant (A1, A3, A4).
3. The JSON restore helpers survive as a second, lenient apply path, used by autolayout, inside-view creation and one Qt command (A2).
4. Autolayout composition has several superlinear loops and runs the solver synchronously on the UI thread; the performance test covers a flat view only (P5, P6).
5. A residue of stale comments, error texts and history references from the replaced design, plus project style-rule violations in new code (C2-C6).

2. Findings overview

ID	Sev	Area	Finding
P1	High	Perf	Finalizer runs full view maintenance on every view touched by a landscape/model property record, including closed views
P2	High	Perf	Landscape load now maintains every view; formerly only the displayed view at scene rebuild
A1	High	Arch	Unbound writes to published documents succeed silently; divergence surfaces later as refused undo

ID	Sev	Area	Finding
P3	Medium	Perf	Structural edits cost $O(\text{owner list})$ at seal, undo and redo
P4	Medium	Perf	JSON history representation: ~ 17 KB retained per changed record
P5	Medium	Perf	Autolayout composition: repeated full-view builds per scope, $O(n^2)$ helpers, $O(k^3)$ packing
P6	Medium	Perf	Autolayout solver runs synchronously on the UI thread (up to 60 s time limit)
A2	Medium	Arch	Lenient legacy JSON apply/restore helpers remain a second apply path
A3	Medium	Arch	Capture coverage depends on caller discipline; public-field structs; hand-listed byte accounting
A4	Medium	Arch/Rob	void setters silently drop refused writes; the action continues on half-applied state
A5	Medium	Arch	Mandatory derived-state maintenance lives in qtgui, outside mechanical test reach
R1	Medium	Rob	Throwing nlohmann accessors on undo payloads terminate instead of refusing
A6	Low	Arch	Autolayout workspace duplicates live UUIDs; workspace models answer global delete requests
A7	Low	Arch	Composition vocabulary spelled as string literals in five places
A8	Low	Arch	Algorithm identity validated through a dummy bridge catalogue
A9	Low	Arch	Autolayout fields optional on read although always written
A10	Low	Arch	Non-allocating apply discipline has no failure mode to protect under no-exceptions
P7	Low	Perf	Capture-side scans proportional to landscape size
R2	Low	Rob	Error-level logs for ordinary outcomes; log flood after a refusal
R3	Low	Rob	Silent defaults in restore (“blob predates the field”, default Point)
R4	Low	Rob	Paste ignores autolayout-settings refusals

ID	Sev	Area	Finding
C1–C8	Low	Quality	File size, stale texts, history in comments, naming, style rules, repetition, vestiges
T1–T3	–	Tests	Coverage strengths and gaps

3. Performance

P1 — Finalizer maintenance fan-out (High)

`app_manager.cpp:320-327` installs a finalizer that runs `EACMP_QT_maintainViewGeometry` (`view_gui.cpp:467`) on every view returned by `EACMP_QT_affectedViews` (`app_manager.cpp:236`). The affected set follows the record kind, not the changed field:

- `LandscapeProperties`, and `GenericLabel` in the landscape pool → **every view of the landscape**.
- `ModelProperties`, and `GenericLabel` in a model pool → `model->viewsShowingAnyItem()`, which itself iterates every element and relation of the model (`model.cpp:683-702`).
- `ModelElement` / `ModelRelation` → every view depicting the item.

Actions that write landscape or model property records include: landscape rename, landscape grid spacing, landscape default language, Browser > Keep Sorted Alphabetically (`organizationSorted` is a landscape property; its setter captures the landscape), landscape generic-label add/edit/ remove, model rename, model hide/show, model scheme selectors, model autolayout settings.

Per view, maintenance is `deriveModelBoundNesting` + `deriveBoundaryAttachment`, then per element a linear ownership check (`view_gui_hittest.cpp:922`), a shape render (`view.cpp:1492`) and a scan of all view relations (`view_gui_hittest.cpp:933`): $O(n \cdot (n+r))$ comparisons plus n shape renders. At the scale target (several thousand views of ~200 elements / ~300 relations) one “Keep Sorted” toggle or landscape rename performs on the order of 10^5 – 10^6 shape renders and 10^8 identity comparisons synchronously inside the action. Several of the triggering edits (sort flag, grid spacing, landscape name) cannot affect endpoint projection at all.

Setters capture on call, not on change (e.g. `EACMP_ViewElement::setPosition` in `view.cpp`), so every maintenance write – including rewriting an unchanged value – also pays a full JSON record capture into the action’s journal.

On replay, `setReplayNotification` publishes `VIEW_CONTENT_CHANGED` for the same set (`app_manager.cpp:357`): $O(\text{views})$ messages per undo of such an action.

The journal scale test (`undo_test.cpp:4566`) measures the library without the finalizer; this path is unmeasured.

P2 — Load-time maintenance of every view (High)

`app_manager.cpp:1817-1825` passes a load finalizer that runs `EACMP_QT_maintainViewGeometry` on every view of the loaded landscape. At the merge base the same work ran inside `rebuildScene` for the displayed view only (`view_gui.cpp:211-240` at 36580826). The change makes scene rebuild read-only – a real improvement, since rebuild previously wrote to the document outside any action – but moves the cost to open time and makes it proportional to (number of views × view size). For a landscape at the scale target this adds a full-landscape shape-render pass to every open. The trade-off between correctness and open latency is neither measured nor recorded as a decision, as the project rules require for snappiness conflicts.

P3 — Structural edits cost $O(\text{owner list})$ (Medium)

`prepareOrderChange` (`view_undo_state.cpp:1847-1890`) copies the owner’s complete identity/order list and builds an `unordered_set` of all its UUIDs on the first membership change per owner – at seal, undo and redo. `finishOwnershipPreparation` (:1923) and `ownerOrderUnchanged` (:2018) walk it again. The branch’s own measurements (analysis document, journal table):

Operation	Small model	Beside 200,000 records
Create one element and seal	0.049 ms	17.75 ms
Undo that creation	0.030 ms	14.01 ms
Restore a populated view	16.62 ms	32.80 ms

A single structural edit in a large model thus scales with model size, contrary to “view cost must not scale with landscape size”. Only the organization-list variant of this cost is recorded as deferred in `EACMP_Dev-deferred.md`.

P4 — History memory footprint (Medium)

Each record is retained as complete before/after JSON. Branch measurements: changing all 500 records of a 200/300 view retains 8,561,984 bytes (≈ 17 KB per record); deleting them retains 5.2 MB. A single autolayout or make-room over a full view is therefore ≈ 8.5 MB of history; at the 1 GB default budget the history holds on the order of 120 such actions, fewer for larger views. Capture, seal comparison, replay preflight and post-apply validation each allocate and compare whole JSON trees (seal 12.4 ms, undo 25 ms for 500 records).

P5 — Autolayout composition is superlinear (Medium)

- `EACMP_buildViewLayoutItems` builds the whole working view (label areas, text measurement, minimum sizes, stack axes) once per container scope in `EACMP_normalizeLayoutScope` (`auto_layout.cpp:392`) and again per scope in `arrangeScope` (`:666`), where only the container’s own entry is used: $O(\text{containers} \times n)$ label resolutions and text measurements.
- The builder is itself $O(n^2)$: `viewElementByUUID` is a linear scan (`view.cpp:1018`) called twice per item (`view_geometry.cpp:201-202`).
- Each text measurement from the workspace performs a linear lookup in the source view (`auto_layout_geometry.cpp:750`).
- `EACMP_layoutSubtreeFootprint` (`auto_layout.cpp:58`) and `EACMP_translateLayoutSubtree` (`:313`) walk the whole view with a descendant test per element and are called per child (`:569, :609, :718`): $O(n^2)$ footprint computations. Translation also resolves each bridge route’s relation by linear lookup (`:324`).
- `EACMP_packLayoutGroup` (`:332-359`) generates $O(k)$ candidate abscissae and ordinates, $O(k^2)$ positions, each tested against k obstacles: $O(k^3)$.
- The before/after diff and the application preflight resolve every item by linear lookup (`auto_layout_geometry.cpp:766, :773, :787-816`).

At 200/300 in a flat view each is small; with nesting the scope count multiplies the full-view builds. `auto_layout_perf_test.cpp` uses a flat fixture and relative thresholds only (`:168`: $\text{large} \leq 4 \times \text{small} + 100$ ms), so the nested cost is unmeasured.

P6 — Solver on the UI thread (Medium)

`EACMP_QT_executeAutoLayout` calls `EACMP_prepareAutoLayout` on the UI thread (`view_gui_layout.cpp:934`). The orthogonal pipeline’s planarization budget defaults to 5 s and accepts up to 60 s (`layout_ogdf.cpp:138`); scopes are capped at 2,000 nodes / 6,000 edges (`:29-30`). There is no progress indication and no cancellation; the window is unresponsive for the solver’s duration.

P7 — Capture-side scans proportional to landscape size (Low)

- First capture of a `ViewLink` scans every view of every landscape in the context (`view_undo_state.cpp:3499-3504`).
- Every organization-node capture locates its forest by scanning all models and views (`EACMP_organizationRecordKey`, `:131-160`; callers `:3705, :3741, :3795`). Deleting k organized items performs k such scans.
- First capture of a view element iterates the element’s views with a linear `viewElementByUUID` in each (`:3474-3476`).
- A model-level autolayout settings edit publishes `VIEW_PROPS_REFRESH` for every view of the landscape (`undo.cpp:2378-2391`).

Each is small; together they explain why several branch timings roughly double in the large landscape although the changed records are identical.

4. Architecture

Strengths

- One capture authority at the storage boundary instead of per-command before/after data.
- Exact qualified identities (landscape, owner, kind, object) and replay by identity, never by semantic search.
- Prepare-then-apply replay: resolution, stale-state checks and capacity reservation precede the first write; post-apply recapture must equal the expected state.
- Retained native objects (EACMP_RetainedDocumentRecord) preserve addresses and reverse registrations across delete/undo.
- Save points with explicit reachability; history invalidation by connected component instead of wholesale clearing.
- Buffered document notifications delivered after completion; UI container lifecycle (close/open views, clear target model) moved behind committed notifications.
- Modal continuations re-resolve targets by UUID after every dialog.
- Canvas gestures own preview tokens, with stale-callback protection (view_gui.cpp:150-173) and a re-entrancy guard (:267-299).
- Autolayout: neutral description vocabulary, validated registry, strict bridge-result validation, lock closure, atomic application with preflight, core/bridge separation.

A1 — The capture guarantee depends on an implicit binding (High)

EACMP_captureBeforeWrite, EACMP_captureBeforeInsert, EACMP_captureBeforeTransfer and EACMP_prepareDocumentRemoval succeed when no journal is bound (view_undo_state.cpp:4314-4391). A write to a published landscape outside execute/updatePreview – a missed call site, a timer or bus handler running after publishAfter, a dialog continuation – changes the document without history. Replay’s preflight (:3158-3183) later detects the divergence and refuses the undo, so the document stays consistent; but the user loses undo at a point unrelated to the cause, and nothing reports the uncaptured write when it happens.

Initialization already has an explicit scope (EACMP_DocumentInitializationScope), used by the loader, clipboard staging, import staging and the autolayout workspace, so the unbound case is not needed for detached roots. The remaining legitimate unbound writers are not enumerated anywhere, and no development-tier check distinguishes “unbound write to a registered landscape” from them. The comment at :4316 (“outside the migrated action boundary”) describes a transition state rather than a rule.

A2 — Two apply paths for view-item state (Medium)

Replay applies descriptor state with validation. The exported helpers eacmpApplyViewElementState, eacmpApplyViewRelationState, eacmpRestoreViewElementState, eacmpRestoreViewRelationState (view_undo_state.h:366-374; view_undo_state.cpp:5075-5108) remain in use by:

- EACMP_CreateInsideViewCommand (undo.cpp:3005, :3012),
- the autolayout workspace copy (auto_layout_geometry.cpp:112, :128),
- autolayout application (auto_layout_geometry.cpp:809, :813),
- resetSelectedLabelSizeToContent (view_gui_commands.cpp:1698-1717).

They are lenient where replay is strict: a blob without uuid/luid mints a fresh identity (:5079-5083, :5099-5103); relation endpoints are first resolved to the *first* depiction of the semantic endpoint (:5095-5098) and corrected only if the blob carries the conditional endpoint fields; absent keys keep the current value (:4821, :4846); an absent or unknown attachment kind becomes Point (:4759-4768). JSON snapshots thereby act as a general copy/transfer format with weaker semantics than the replay path. The header still documents them as “used by delete/undo” (view_undo_state.h:369) and the file comment as “snapshots + appearance-override command” (:1-7). resetSelectedLabelSizeToContent takes its own snapshot to roll back a tentative edit inside an action whose journal already treats a net-zero record as unchanged.

A3 — Capture coverage depends on caller discipline (Medium)

- EACMP_GenericLabel is a public-field struct (generic_label.h:15). Its key/texts are assigned directly in undo.cpp:3518-3519, :3558-3559, landscape.cpp:418, model.cpp:711, serialization.cpp:526-527; the edit is captured only because undo.cpp:3557 calls EACMP_captureBeforeWrite first. Folder-node parent/children are likewise public.
- Every setter in view.cpp, model.cpp, landscape.cpp, folders.cpp and labeled.h begins with a capture call by convention. The extended field-parity test proves that *captured keys* match the schema (field_parity_test.cpp); it does not prove that every *mutating path* captures. A new setter without the call is silently uncaptured and falls into A1’s late refusal.

- `EACMP_DocumentRecordAccess` is a friend of the core classes (e.g. `landscape.h:35`) and reads and writes private members. `retainedRecordBytes` (`view_undo_state.cpp:434`) enumerates member fields by hand, so a new member escapes byte-budget accounting without any test noticing.

This is the “derived coverage, not listed coverage” invariant applied to capture: field coverage is derived, mutation-path coverage is listed.

A4 — Refused writes are dropped by void setters (Medium)

Setters such as `EACMP_ViewElement::setPosition` return void and skip the assignment when capture refuses. The refusal poisons the journal (`view_undo_state.cpp:3446-3451`) and the owner rolls the action back on completion, so the final state is correct. Until then the rest of the action runs on a mix of applied and dropped writes, Synchronous-topic subscribers can observe that state, and each further write logs another Error (R2). Callers cannot distinguish “written” from “dropped”.

A5 — Mandatory maintenance lives in qtgui (Medium)

Forward maintenance (nesting derivation, boundary attachment, endpoint reprojection) is a required completion step of every action, but it is invoked by the Qt finalizer (`app_manager.cpp:320-327`) and implemented in qtgui (`view_gui.cpp:467-496`, `view_gui_hittest.cpp:918-942`). Its only Qt dependency is shape-name resolution (`resolvedShapeName`, a `QString`); the branch now resolves the same visual in core (`EACMP_resolveViewElementVisual`, `view_geometry.cpp:12`). As placed, the completion contract cannot be exercised by mechanical tests, and a history produced without the Qt finalizer (the library tests) differs from one produced in the application.

A6 — Duplicate live identities in the autolayout workspace (Low)

`EACMP_prepareAutoLayout` builds a temporary `EACMP_Landscape` whose view, models, elements and relations carry the live UUIDs and LUIDs (`auto_layout_geometry.cpp:28-98`, `:741-744`). Every model constructor subscribes to `MODEL_ITEMS_DELETE_REQUEST` keyed by model UUID (`model.cpp:281`, `:846-858`), and the live-membership guard (`:855`) passes for the workspace model because it is a member of its own landscape. A delete request published while a workspace exists would be executed by both copies. Preparation is synchronous and publishes no such request today, so this is latent; correctness relies on no UUID-keyed global state being reached during preparation.

A7 — Composition vocabulary as string literals (Low)

The grouping tokens (`automatic/grouped/intermixed`) and output modes are spelled in `layout_description.cpp:139-140`, `auto_layout_settings.cpp:128`, `auto_layout_resolve.cpp:335`, `layout_ogdf.cpp:84-91` and `:461`, next to an existing constant for one of them. The OGDF bridge dispatches on the algorithm identifier string (`layout_ogdf.cpp:371`) after having found the matching description entry (`:359`).

A8 — Identity validation through a dummy catalogue (Low)

`EACMP_readAutoLayoutGrammar` validates a strategy’s algorithm identifier by constructing a dummy `EACMP_LayoutAlgorithmDescription` and passing it through the full bridge-catalogue validator (`auto_layout_strategy.cpp:100`). Grammar loading thereby depends on every catalogue rule accepting an empty parameter list.

A9 — Optional-on-read persisted fields (Low)

The writers always emit `autoLayout` and `autoLayoutGrouping`, and the format version check is strict, yet the readers treat absence as “use referrals” (`serialization.cpp:1496`, `:1677`, `:1700`; `clipboard.cpp:1117`, `:1214`), and the schema does not require the fields. Absence gets an implicit meaning, contrary to the explicit-values rule; a damaged or hand-edited file loads with different settings instead of being refused.

A10 — Non-allocating apply discipline (Low)

Replay reserves capacity and applies by swaps so the apply phase cannot allocate (`view_undo_state.cpp:976`, `:3274`). Under the no-exceptions policy an allocation failure terminates the process anyway, so the discipline adds code and constraints without a failure mode it protects against.

5. Robustness

R1 — Throwing JSON access on undo payloads (Medium)

Restore helpers use `.at()` and `.get<T>()`: `EACMP_restoreLabelEntries` (`view_undo_state.cpp:2385-2400`), its callers (`:2419, :2469, :2535-2543`), `EACMP_restoreRawMultiplicity` (`:2557-2563, :2588-2589`) and the view appearance-override restore. The payload is produced only by capture, but any asymmetry between a capture and its restore (renamed key, changed type) raises a `nlohmann::json` exception that nothing catches: the process terminates during undo instead of refusing it. Everywhere else, replay turns invalid input into `Refused`.

R2 — Error-level logs for ordinary outcomes (Low)

- compose logs `Error` for ordinary non-mergeable pairs (`view_undo_state.cpp:3308`).
- A paste that lands nothing returns `Refused` (`undo.cpp:3087`), which rolls back and reports a failure.
- After a refusal, every further write in the action logs another `Error` (`:3450`); a refused import can emit thousands.

Errors reach the system log and the Log Window; routine outcomes at Error level bury real faults.

R3 — Silent defaults in restore (Low)

`readUndoAttachmentKind` returns `Point` for an absent key (`view_undo_state.cpp:4763`); the optional-size and multiplicity-placement restorers skip absent keys as “blob predates the field” (`:4821, :4846`). History is never persisted, so no blob can predate a field; these branches can only mask a capture defect.

R4 — Paste ignores settings refusals (Low)

Paste calls `setAutoLayoutSettings` / `setAutoLayoutGrouping` on the new view and model and discards the `bool` result (“validated at capture/load”, `clipboard.cpp`). Validation precedes capture, so a refusal does not poison the journal: the pasted holder silently keeps default referrals.

6. Code quality

C1 — File size and cohesion

`view_undo_state.cpp` (5,108 lines) combines record access, journal, change set, replay preparation, field descriptors, the legacy restore helpers and the thread-local bindings. `undo.cpp` 3,823; `view_window.cpp` 6,173; `view_gui_mouse.cpp` 5,391.

C2 — Texts made stale by the redesign

- 13 error texts “...(undo/redox identity bug)” in commands that now execute once, where a missing target is a stale UI target (e.g. `undo.cpp:3267, :3271, :3296, :3300, :3327, :3384`).
- “GR-3: ... command becomes a no-op then” (`undo.cpp:1590, :1643`) – execution now refuses.
- “idempotent: applying twice is harmless” (`undo.cpp:1010`).
- “the landscape UUID groups this on the undo stack” (6×) – history scope is derived from captured records (`undoLandscapeUUIDs`, `undo.cpp:842`).
- “(recoverable)” on paths that now return `Refused` (`undo.cpp:3658, :3683, :3727, :3758, :3790-3819`).
- `undo.h:793` “restorable from captured data”, `:826` “Populated once, at construction”, `:882` “Undo removes everything the paste created, innermost last”, `:1103` “restoring it on undo”; `:222` an empty “Undo stack” section header.
- `view_undo_state.h:1-7` and `:369` (see A2).

C3 — History in code comments (project rule)

`view_undo_state.cpp:4316` “outside the migrated action boundary”, `:4395` “Unmigrated containers”, `:4996` “(was reconstructed from model connections, which picks the first instance)”, `:4821/:4846` “blob predates the field”.

C4 — Naming

New free functions mix prefixes within one header: `EACMP_modelUndoPropertyKeys`, `EACMP_viewUndoPropertyKeys` next to `eacmpViewElementUndoFieldKeys`, `eacmpCaptureViewElementState` (`view_undo_state.h`); `eacmpRegisterOGDFLayoutEngine` next to `EACMP_registerLayoutBridge`. The convention requires the `EACMP_` prefix.

C5 — Style-rule violations in branch code

- Two statements on one line after a braced if: 22 setters in `view.cpp` of the form `if (eacmpEmptyReferralRefused(...)) { return; } m_field = value;`.
- Assignment plus return inside one brace pair on one line: `view_geometry.cpp` (9×, e.g. `:30, :85, :92`), `layout_ogdf.cpp:292-305`, `auto_layout.cpp`, `auto_layout_geometry.cpp`.
- 80 setters in `view.cpp` place the opening brace on the signature line; functions elsewhere put it on its own line.
- Brace-less if bodies in modified functions (`undo.cpp:1048, :1528, :1761`; `view_gui_commands.cpp:1690, :1695`).
- Dead variable `oldValue` (`app_manager.cpp:3356`); `(void)landscape`; for an unused structured binding (`undo.cpp:3757`); a mis-indented `return`; in `importModelFromLandscape`; blank-line runs (`undo.cpp:3400-3402`, `undo.h:918-920`).

C6 — Repetition instead of abstraction

- The modal-continuation block (`prepare`, `re-resolve` by `UUID`, `log`, `return`) is written out 7 times in `app_manager.cpp` with the same comment.
- The comment “Unavailable targets refuse before dependent work; no ‘else’ branch.” appears 70 times; “no ‘else’ branch” 429 times in the four core files, often where no else would be expected.
- 81 call sites construct a command with `execute(*std::make_unique<Command>(...))`: a heap allocation dereferenced for the duration of one expression, where a local object states intent.

C7 — Vestiges of the symmetric command design

- Per-command `apply(value)` helpers kept although commands execute once (`undo.cpp:3262, :3291, :3321, :3351, :3380, :3552`).
- `EACMP_UndoCommand::m_landscapeUUID` (`undo.h:215-219`) kept although history scope is derived.
- `EACMP_DeleteFromModelCommand` keeps two constructors (context vs landscape pointer).
- Two protocols per command: `execute()`, and `notifyApplied()` for callers that already mutated (`undo.cpp:3403`).

C8 — Record-key hash

`EACMP_DocumentRecordKeyHash` (`view_undo_state.cpp:3120-3127`) combines three `UUID` hashes by XOR with 1- and 2-bit shifts. For container-property records the three `UUIDs` are often identical; the mix is weak, though acceptable with random `UUIDs`.

7. Tests

T1 — Strengths

`undo_test.cpp` adds 31 test functions covering applied recording, coordinator and preview lifecycle, replay, save-point lifecycle, composition and merging, every record family’s journal (landscape, model, view, view link, generic label, folder, organization, ownership, transfer), initialization scopes, duplicate/inside-view history, import, and a scale test. `view_undo_state_test.cpp`, `auto_layout_test.cpp`, `auto_layout_dialog_test.cpp` and `messaging_test.cpp` are extended; field parity now checks container records against the schema.

T2 — Gaps

- The Qt finalizer fan-out (P1) and the load finalizer (P2) are neither tested nor measured.
- No test derives mutation-path capture coverage (A3).
- No test feeds a malformed payload to a restore path to show refusal rather than termination (R1).
- Canvas preview tokens and modal re-resolution are covered only by manual scenarios (consistent with project policy).

T3 — Performance tests

undo_test.cpp:4592-4594 logs timings without thresholds; auto_layout_perf_test.cpp asserts availability < 10 ms and a relative bound ($\text{large} \leq 4 \times \text{small} + 100 \text{ ms}$) on a flat fixture only. No measurement covers nested composition, load time, or finalizer cost at the scale targets.

8. Not covered

- No build, test run or profiling.
 - Read at function level or in full: app_manager.cpp (diff), view_gui.cpp, view_gui_layout.cpp, auto_layout_ui.cpp. Reviewed only through their diffs for the action protocol (prepareForAction, execute, preview tokens): view_window.cpp, view_window_props.cpp, view_gui_mouse.cpp, view_gui_commands.cpp, tree_browser.cpp. Not read: auto_layout_dialog.cpp.
 - import_merge.cpp, born_with_parts.cpp, relation_reroute.cpp, clipboard.cpp: diff level.
 - Reference-document changes were not checked against the code.
-

9. Codex assessment — 2026-09-28

I checked the principal findings against the implementation, the production Qt tests, the recorded measurements and the refactor's written scope. The current head, dc012096, has the same implementation as the reviewed b9d4bdd2; its additional changes are the review, a manual-test fixture, query history and the reminder. This assessment includes no new build, test run or profiling. Quoted timings remain the earlier measurements. I did not independently recount every style occurrence or repeat the original review's complete source coverage.

My overall conclusion is that the architecture should be retained, but the claim that capture is an enforced authority is stronger than the implementation warrants. The review identifies real completion gaps. P1 and A1 are the most consequential: unnecessary maintenance can make an ordinary metadata edit expensive, and an uncaptured write can bypass both history and its safeguards. The existing tests establish substantial correctness within the exercised action paths; they do not establish that every possible persistent write enters those paths or that every action scales.

Maintenance scope and performance evidence

P1: agree, high priority. EACMP_QT_affectedViews selects geometry work from record kinds, without inspecting which properties changed. A landscape sort toggle therefore takes the same route as a change that could affect every depiction. Moving maintenance before sealing is correct; giving unrelated properties the same maintenance scope is not required by that design.

There is an additional distinction: the finalizer receives **captured** records before sealing removes unchanged ones. Merely calling a setter can select maintenance even when the action will ultimately be a no-op. P1's capture-cost wording needs narrowing, however: a record's first write captures its full state; subsequent writes use the journal's cached identity and baseline. It does not reconstruct the JSON on every setter call (EACMP_DocumentJournal::capture, view_undo_state.cpp:3464). Unnecessary first captures across many views remain a real cost.

The architectural requirement is to distinguish what changed from what maintenance it invalidates, and to keep the resulting dependency work inside the original action. A collection of special cases for particular menu commands would leave the same problem for the next caller. Reintroducing persistent writes during scene rebuilding would also break the exact-replay guarantee.

P2: agree with the cost and measurement gap; qualify the severity evidence. Every loaded view is maintained before publication. That is demonstrably broader work, but neither this review nor the existing timings measures its added open latency across thousands of populated views. Opening a landscape necessarily scales with its contents; linear dependence on the number of views is not itself a defect. The issue is how much geometry work opening requires and whether the result meets the responsiveness target. I regard this as an unresolved acceptance risk, rather than a measured High-severity latency regression. Deferring writes until a closed view is displayed is not a free alternative: it would need a defined initialization/repair boundary consistent with saving and undo.

The earlier measurements varied **undisplayed semantic population and mostly empty extra views**. They do not answer P1 or P2. A roughly 5 ms maintenance pass over one populated view cannot certify the same pass across thousands. Likewise, the recorded production gesture's roughly 48 ms release and 100 ms queued refresh show where time is spent; their stability with background population is not proof that interactive latency is satisfactory. My earlier evidence should be read within those limits.

Capture, refusal and restoration

A1 and A3: agree. The unbound case explicitly permits writes, and public mutable fields provide another way around capture. Initialization scopes are useful infrastructure, but do not themselves enforce the distinction between an unpublished workspace and a published document. A published document needs an enforceable mutation boundary; legitimate initialization and replay need their own explicit authority. Simply requiring a journal for every object everywhere would wrongly conflate standalone construction with editing a live document.

The consequence of A1 can be broader than a later refused undo. An unrecorded edit to a record no retained action touches need never encounter replay's stale-state check. It also bypasses the history-based accounting of saved state. This is a consequence of the boundary design, not a reproduced user failure in this assessment.

Field parity proves representation coverage, not mutation coverage. More field-parity assertions alone cannot close that gap. Encapsulated writes and checked mutation authority are the structural issue. A3's memory-accounting point also needs precision: `sizeof(record)` already covers new inline members; newly owned variable-sized allocations can escape the hand-written accounting.

A4: agree with the API weakness, with two qualifications. A capture refusal poisons the action, so the code is designed to roll back rather than commit a half-applied result. Ordinary document notifications are buffered and discarded on refusal; they do not expose that intermediate result to normal UI subscribers. The synchronous change/delete protocol remains a separate concern.

Continuing dependent work after a dropped write is nevertheless unsafe reasoning for a caller. Failure needs to be visible and propagated. Immediate rollback from inside a setter is not an acceptable shortcut: the caller may still hold addresses that rollback would invalidate. The current decision to restore after the outer callback returns has a real lifetime rationale.

A2 and R3: agree about leniency, not that every forward-copy operation is redundant. Autolayout preparation and inside-view creation still need to construct or apply new state. That is a different operation from history replay. The defect is the permissive contract: inferred identity, first-depiction endpoint resolution and absent-field defaults in helpers presented as exact state restoration. Forward application must also remain captured; directly transplanting replay's private swaps into it would bypass that requirement. Shared strict state decoding and explicit creation identity choices are compatible with keeping those two operations distinct.

R1: agree as a latent robustness defect. The throwing accessors are present. These are internal payloads, so this is not evidence that an ordinary malformed landscape directly reaches them or that valid undo currently crashes. A capture/restore mismatch can nevertheless bypass the intended refusal path. Checked preparation should reject incomplete or mistyped state before live mutation, without compatibility defaults or a new exception-catching layer.

A9 and R4: agree. Optional-on-read autolayout selectors conflict with the explicit-value rule, even though the current schema permits omission. Strict version checking alone does not enforce required fields. Paste's ignored `bool` results are also real: those setters validate before capture, so validation refusal need not poison the journal. Earlier validation is a precondition, not a reason to discard a later failure result. I have not reproduced that refusal through a normal valid clipboard operation.

Autolayout and structural costs

P3 and P7: agree about the scaling. Copying and validating an affected owner's ordering is real work, and the measured creation cost of 17.75 ms beside 200,000 records should not be described as independent of model size. It also does not, by itself, prove an unacceptable user-visible pause. The distinction between local geometry edits and model membership edits matters. The recorded deferral for very large organization lists is specific; it does not automatically constitute acceptance of every other owner-list or owner-discovery scan.

P4: agree about the cost; retain complete-state semantics. Approximately 8.6 MB for the synthetic 500-record change is substantial. These figures are the implementation's retained-memory accounting, not measured process RSS or peak memory during preparation. The approximately 17 KB per record is an average for that fixture, not a universal record size. Complete records were a deliberate way to eliminate omitted-dependent-field bugs. The measurement supports scrutiny of representation and allocation cost, not a return to per-command hand-selected undo fields.

P5: agree. The repeated full-view builders, linear identity lookups, subtree scans and candidate packing are visible in the source. Nested composition multiplies work that the flat performance fixture does not exercise. The cubic packing bound describes one placement against its obstacles; it is not a measurement of whole-action duration. This is a credible scale risk requiring evidence, not proof that every ordinary 200-element layout is slow.

P6: agree. Preparation runs synchronously through `EACMP_QT_executeAutoLayout`. The 5-60 s setting is the planarization component's budget, not a deadline for the whole action; multiple scopes can invoke the backend, and composition/conversion add work. It should not be read as an observed 60-second

freeze, or as a guaranteed upper bound. A future asynchronous execution boundary would have to isolate temporary models and backend state, retain valid measurement inputs, and validate the result against the live document. Wrapping the existing call in a worker is not enough.

Corrections and disagreements

A5 and T2 overstate the test gap. Mandatory maintenance is in qtgui, but it is within mechanical test reach. `view_undo_state_test.cpp` links the production Qt implementation and contains:

- `EACMP_QT_testCanvasGeometryMaintenance` (line 261): captured maintenance, deferred refresh, closed-view completion and exact replay.
- `EACMP_QT_testGeometryMaintenanceScale` (line 155): the complete maintenance helper at 200/300 displayed items, with small and large background populations.
- `EACMP_QT_testApplicationActions` (line 733): production viewport gestures, cancellation, another window interrupting a preview, stale release events, undo and save interruption.
- `EACMP_QT_testGestureScale` (line 581): production gesture and completion timings.

Thus canvas previews are not covered only manually, and maintenance itself is neither untested nor unmeasured. T3 omits these Qt measurements. The genuine gaps remain selection of maintenance scope, fan-out across many populated views, load latency and nested-layout performance. This correction does not claim comprehensive modal-continuation coverage or interactive platform verification.

There is a legitimate layering question in A5: a core caller without the application finalizer does not automatically receive the same geometry-completion policy. But the written refactor plan explicitly preserved the location and calculations of existing Qt/core geometry routines (20260924-undoredoanalysis.md, Recommended architecture and phase 4). Relocating them would have expanded the agreed scope. I would record this as a contract/layering issue, not as absence of tests or proof that the refactor violated its plan.

A10: I would retain the prepared, non-failing apply phase. It is correct that reserving memory does not provide recoverable out-of-memory handling under the present policy. That does not make the overall discipline pointless. Replay completes resolution, validation and replacement construction before changing live ownership or fields; commit then has a deliberately restricted set of operations. Particular reservation machinery can be assessed for complexity, but there is no justification here for interleaving fallible preparation with live mutation.

A related wording correction to §§1 and 4: the item-state recapture/equality check runs on the **prepared detached replacement before live application** (`view_undo_state.cpp`, `EACMP_DocumentChangeSet::replay`). It is not post-commit validation that might discover a mismatch after changing the document.

C6: the repeated marker is required; empty repetition is not. Section 7.1 of `EACMP-documentationincode.md` expressly requires the no 'else' marker and a reason. Its occurrence count is therefore not itself a defect. Generic repeated explanations can still fail to explain the local reason. Temporary heap allocation merely to dereference a command for one synchronous call has no comparable justification. Repeated modal code warrants examination of its common contract, rather than abstraction solely because its spelling repeats.

C7: distinguish vestiges from current responsibilities. A single-use `apply(value)` wrapper and obsolete scope metadata can be unnecessary residue. `execute` versus `notifyApplied`, however, expresses a current distinction: perform an initial mutation, or announce one already performed inside the original capture callback. `recordApplied` checks that callback is active (`undo.cpp:450`). Eliminating the latter merely to make commands look uniform risks either applying the mutation twice or recording too late.

Remaining observations and conclusion

A6 identifies a genuine isolation weakness: a workspace model with a copied UUID participates in the global delete-request bus. Copied identity is useful for exact correspondence and is not itself wrong in a detached workspace; participation in live application protocols is the problem. The current synchronous path explains why this is latent. It matters especially to any concurrency proposal for P6.

A7 and A8 are bounded maintainability issues. The dummy catalogue validation is confirmed; sharing identifier validation need not couple grammar parsing to unrelated catalogue requirements. R2 correctly distinguishes normal inability to merge from a corrupt composition request: they should not all produce the same Error diagnostic. Repeated refusal logs add noise without changing the failure. A paste that does nothing still needs its reason classified before deciding whether it is an ordinary no-op or a genuine refusal.

C2–C5 identify misleading residue and project-rule violations that I accept responsibility for; passing tests does not excuse them. C1 needs a cohesion argument, not a line-count threshold. C8 does not establish a practical hash-performance problem and belongs below the demonstrated scans. Neither justifies an unrelated restructuring exercise.

I would retain the journal, exact native replay, notification buffering and backend-neutral layout design. The main unfinished work concerns enforcement of the mutation contract and the scope of maintenance, followed by strict restoration/refusal behavior and the unmeasured layout/load cases. My previous correctness and performance evidence does not close those gaps. Conversely, the review should not discard the existing production Qt coverage or treat intentional preparation and preview protocols as obsolete merely because the command-based inverse implementation was removed.

10. Response to the Codex assessment — 2026-09-28

I checked each correction in §9 against the source before accepting or contesting it. Sections 1–8 are left as written; this section records what changes.

Corrections accepted (errors in §§1–8)

1. **Replay validation runs before commit, not after.** The recapture/equality check runs on a detached replacement that is prepared before live application (`EACMP_DocumentChangeSet::replay`). “Post-apply recapture” in §1 and in the §4 strengths list is wrong.
 2. **P1 capture cost is paid once per record, not once per setter call.** Later writes hit the journal’s cached key and baseline (`view_undo_state.cpp:3464–3473`). The correct statement is that every record maintenance touches pays one full-state capture, even if its value ends unchanged.
 3. **A3 byte accounting is narrower than I stated.** `retainedRecordBytes` starts from `sizeof(record)` (`view_undo_state.cpp:439`), so new inline members are counted. Only new owned variable-size storage escapes the hand-written accounting.
 4. **A5, T2 and T3 are wrong about test reach.** When built with Qt, `view_undo_state_test` links `eacmp_qtgui` under `EACMP_QT_UNDO_INTEGRATION_TEST` and runs offscreen (`mechanical_tests/CMakeLists.txt:464–469`). It exercises:
 - production maintenance, including closed views and exact replay (:261);
 - maintenance at 200/300 items (:155);
 - gestures, cancellation, interruption from another window, stale release events and save interruption (:733);
 - gesture timings (:581).
- I missed this. A5 is reduced to a layering observation: a core caller without the application finalizer gets different completion behaviour. The refactor plan kept these routines where they were, and I did not claim otherwise. I now rate A5 **Low**.
5. **C6: the marker is required.** `EACMP-documentationincode.md` §7.1 requires every `elseless if` to carry a reason ending in no `'else'` branch, and `make audit` enforces it. The count is not a defect. What remains:
 - generic, reason-free wording repeated verbatim (the 70 identical lines);
 - the `*std::make_unique` idiom.
 6. **C7: notifyApplied is a current protocol, not a vestige.** `recordApplied` refuses outside the original mutation callback (`undo.cpp:450–457`). The residue is limited to:
 - the single-use `apply(value)` wrappers;
 - `m_landscapeUUID`;
 - the stale comments about the landscape UUID “grouping” commands on the undo stack.
 7. **A10 overreached.** The prepare-then-commit ordering is the valuable part, and I do not question it. My point concerns only the capacity-reservation and swap-only machinery. Its sole protected failure is allocation, which terminates anyway under the no-exceptions policy. “Adds code without a failure mode it protects against” applies to that machinery, not to the phase separation.

Extensions accepted

- **A1 is a data-loss path.** Unsaved state is derived only from history (`hasCommandsForLandscape`, `undo.cpp:730–745`). An uncaptured write to a clean landscape therefore leaves it clean, so closing it or quitting does not prompt, and the change is lost. The write also escapes replay’s stale check whenever no retained entry touches that record. That makes A1 the most consequential correctness finding. It is a consequence of the design, not a reproduced failure.
- **Merely touching a record selects maintenance.** `capturedRecordTransitions` reports every captured record, including those whose value ends unchanged (`view_undo_state.cpp:3413–3428`). An action that rewrites an unchanged landscape field still fans out (P1).
- **A4:** ordinary document notifications are buffered and discarded on refusal. Only the synchronous protocol can observe the intermediate state, which is what A4 says. I also agree with the lifetime argument against rolling back inside a setter.

- **P4:** the 17 KB per record is the implementation's own accounting for that fixture, not process memory. Complete-state records are the right semantics. The finding concerns their representation cost, not their content.
- **P6:** the 5–60 s figure is one component's budget. It is neither a total nor an upper bound.

Points I hold

- **P1 stays High.** §9 says a single-view timing cannot certify thousands of views. It does give a first-order estimate, though, because fan-out is a sequence of independent per-view passes. At the branch's measured ~5 ms per populated 200/300 view (20260924-undoredoanalysis.md, lines 233, 286, 307):
 - 1,000 full views ≈ 5 s;
 - 3,000 full views ≈ 15 s.

That time would be spent on a sort toggle, which needs no geometry work at all. Even at a tenth of that density it breaks the responsiveness target, and the pass is pure waste for the non-geometric triggers.
- **P2 moves to Medium, by my own severity scale (“unmeasured risk”).** The same ~5 ms is an upper estimate per full-size view. The scale target describes the worst view, not the average one. Maintenance cost grows faster than view size, so the mostly smaller views of a real landscape are cheap. I agree that load time growing linearly with the number of views is not a defect in itself. What remains is new per-view work for views that are never displayed in the session, and a missing measurement. I agree that deferring it needs a defined repair boundary and is not free.
- **A2:** I did not claim that forward copying is redundant. A2 is about the permissive contract, and §9 agrees. The one redundancy I named stands: `resetSelectedLabelSizeToContent` takes a private snapshot and rollback inside an action whose journal already treats a net-zero record as unchanged.
- **P3 stays Medium as a principle violation.** §9 is right that its current magnitude (~18 ms per structural action beside 200,000 records) is not a user-visible pause.

Revised ranking

ID	Was	Now	Reason
A1	High	High	Strengthened: silent loss of unsaved changes
P1	High	High	Measured per-view cost × view count
P2	High	Medium	Unmeasured; the per-view cost is a worst-case estimate
A5	Medium	Low	Production Qt maintenance is mechanically tested
A10	Low	Low	Narrowed to the reservation machinery
C6, C7	Low	Low	Narrowed as above
T2, T3	–	–	Remaining gaps: scope selection, fan-out across many populated views, load latency, nested layout, malformed-payload refusal

Order of consequence: 1. A1, then P1. 2. The strict-contract group: R1, A2, A3, A4, A9 and R4. 3. The measurement gaps: P2, P5 and P6. 4. The scaling observations: P3, P4 and P7. 5. The Low items.

11. Codex agreement assessment — 2026-09-28

We are close to agreement on both diagnosis and priorities. I accept Claude's revised ranking in §10.

Point	My position now
A1: uncaptured writes	Full agreement. Potential silent loss of unsaved changes makes this the first correctness concern. No actual application occurrence has been reproduced.
P1: maintenance fan-out	Full agreement: High. The 5–15 second extrapolation is plausible evidence of risk, although not a measured action time.
Strict-contract findings	Agreement on the problems with capture coverage, permissive restoration and ignored refusals.
P2 and A5 downgrades	Agree: Medium for unmeasured loading cost; Low for the remaining layering concern.
Test-coverage corrections	Resolved. Claude accepts the production Qt coverage.
A2's local label-size rollback	Having checked the setter, I agree that this particular full-record snapshot/restore is redundant.

A10 is the remaining substantive judgement call. I accept the narrowed point: preallocation does not provide recoverable allocation failure under the current policy. My defence of phase separation did not fully answer that narrower criticism. An allocation-free commit phase still makes the implementation easier to reason about, but whether its machinery earns its complexity is a maintainability judgement—not an additional recovery guarantee.

Two timing statements need qualification:

- Approximately **5 ms is a measured sample, not an upper bound.** We have not established the size distribution of real views.
- Approximately **18 ms does not alone establish either acceptable or unacceptable latency;** it contributes to the complete action.

There is no competing architectural direction left: retain the journal and layout separation, address A1 and P1 first, then the strict-contract issues and missing measurements.

12. Remaining differences — Claude response, 2026-09-28

After §11 three points remain open: A10, and the two timing qualifications. I re-examined each against the source and the recorded measurements. None stays in dispute.

A10 — withdrawn

I read the commit phase of `EACMP_DocumentChangeSet::replay (view_undo_state.cpp:3274-3298)` and sorted its machinery by what each part is for.

- **Required by validate-before-commit and address stability.** These would exist in any design that validates a replacement before committing it:
 - detached replacement records, swapped into live targets whose addresses must not change;
 - field-swap callbacks for container properties;
 - prepared view-registration vectors (:969-974).

They carry data that has already been validated into live storage; they are not allocation machinery.

- **Present only to keep the commit allocation-free:**

- `prepareCapacity (:976-998);`
- `prepareIdentityCapacity (:2039-2058);`
- the prepared UUID/LUID index nodes they reserve for.

Together these are about 45 lines plus the prepared nodes: they reserve hash buckets and nodes in advance.

So the complexity that is specific to allocation is small. The reasoning benefit §11 names – a commit that runs no computation, observer or fallible step – comes from the prepare/commit ordering, which both assessments keep. A10's original wording ("adds code and constraints without a failure mode it protects against") overstated a cost of about 45 lines as a design issue. **A10 is withdrawn.**

The ~5 ms per-view figure — agreed: a measured sample, not a bound

§10 called it an “upper estimate” per full-size view. That rested on two assumptions, neither of which is established:

- **View size.** The assumption was that no view exceeds the 200/300 target. That target is approximate, and no distribution of real view sizes is known.
- **Cache state at load.** The assumption was that the cost at load equals the warm cost. The recorded figure is warm (20260924-undoredoanalysis.md:307, “Warm Qt maintenance averaged 5.42 ms”), and load-time maintenance runs cold.

“Upper estimate” in §10’s P2 entry should therefore read “estimate”. P1 does not depend on the exact figure. For the sort flag, grid spacing and landscape name, the correct amount of geometry work is zero, so any positive per-view cost multiplied by thousands of views is pure waste. P1 stays High and P2 stays Medium.

The ~18 ms structural-edit cost — agreed: not decisive on its own

“Not a user-visible pause” (§10, P3) overstated the case. The recorded release, maintenance and sealing of a move gesture takes about 48 ms (20260924-undoredoanalysis.md:279). A structural edit in the large model adds about 18 ms at seal (about 14 ms at undo) to its own complete action, and that complete action’s duration has not been measured. P3 stays Medium, for two reasons: it breaks the principle that local edits must not scale with landscape size, and its total contribution to the action has not been measured.

Agreed final state

- There are no remaining differences of opinion.
- Final severities are the §10 revised ranking, with A10 withdrawn.
- Priority order:
 1. A1, then P1.
 2. The strict-contract group: R1, A2, A3, A4, A9, R4.
 3. The measurement gaps: P2, P5, P6.
 4. The scaling observations: P3, P4, P7.
 5. The Low items.
- A1 remains a design-level finding. Neither assessment has identified or reproduced an actual unbound write in the application.

13. Implementation status — 2026-09-29

Scope: the accepted findings in §12. A10 is withdrawn; prepare-before-commit and the allocation-free commit remain. The separately deferred optimization of very large organization forests remains deferred.

- ☑ A1: enforce mutation authority for published native documents, including inserts, transfers, removals, organization edits and LUID renumbering; preserve detached initialization and private replay.
- ☑ P1: classify actual field changes before maintenance and replay notification; cover metadata, unchanged setters, geometry and closed-view dependencies at application scale.
- ☑ R1/A2/R3: use one checked raw-state decoder for forward copying and prepared replay; require complete state and explicit construction identity, and refuse invalid replacements before live writes.
- ☑ A3/A4: encapsulate native property storage, derive storage visitation and retained accounting, propagate mutation failure, and prevent synchronous observations after action refusal.
- ☑ A9/R4: require persisted and clipboard autolayout policies and propagate paste refusal.
- ☑ A5: move persistent geometry maintenance into core, independently of Qt scenes.
- ☑ P2: populated-view loading is measured; the user chose to retain eager maintenance, accepting the measured load-time cost to publish fully maintained documents.
- ☑ P5: cache full-view geometry, scope membership, subtree footprints and indexed lookups; replace cubic packing checks and measure a nested fixture.
- ☑ P6/A6: isolate the workspace from live message protocols; execute the solver on a worker with progress, cancellation, reentry protection and stale-result rejection.
- ☑ P3/P4/P7: remove complete owner-list copies from ordinary history preparation, encode complete retained states compactly, and use direct native ownership during capture.
- ☑ A7/A8: centralize composition vocabulary and validate qualified algorithm identifiers directly.
- ☑ R2: distinguish ordinary no-op/noncomposable outcomes from errors and avoid refusal log floods.
- ☑ C1–C8: remove applicable command residue, stale text and needless allocations; correct naming, comments and formatting; strengthen record-key hashing without unrelated decomposition.

☒ T1-T3: regressions and documentation are updated. The final full check passed 82/82 tests; comment and documentation-reference audits are clean.

Mutation and replay evidence

The audit found two concrete uncaptured writers, superseding the earlier design-only qualification of A1: import-clash renumbering and the registered Recenter View action. Renumbering now captures native LUIDs and prepares collision-checked index changes for replay. Recenter runs through the application action coordinator. Tests cover all nine LUID-bearing native kinds, reuse of vacated identifiers, and three-cycle exact recenter undo/redo through the registered application action.

Native document fields expose const reads and checked writes. Public whole-record copying, assignment and moving are unavailable. Private storage declarations generate replay visitation and retained-capacity accounting, including owned nested values. Compile-time tests check the public mutation surface and the absence of per-field owner-pointer overhead. Runtime tests cover published/unbound writes, structural changes, invalid setters, rollback and synchronous subscriber suppression. A refusal result is available to callers; commands, descriptor readers and compound mutation paths propagate it before dependent work. The journal also quarantines a refused action when a caller neglects that result.

The strict raw decoder rejects missing keys, wrong types, invalid identities, invalid enums and non-finite values. Replacement batches validate before changing native records. Storage-reader repair behavior remains distinct: the corruption suite caught two regressions in that policy, which were repaired and passed afterward. Inside-view copying remaps the exact relation endpoints and surviving anchor identities; external anchors are omitted without expanding the selection.

Autolayout policy requirements use landscape format 73, with a 72-to-73 migration and migrated fixtures/schema. Invalid stored and clipboard policies refuse explicitly. Clipboard tests also reject unknown element/relation display and attachment/routing tokens.

Performance evidence and limits

Selected measurements from the passing full run on this checkout:

Measurement	Result
Application metadata, 1 populated view	Action 0.124 ms; replay 0.098 ms
Same metadata, 1,000 populated views plus 200,000 undisplayed records and 101 models	Action 0.056 ms; replay 0.041 ms
Append and seal beside 200,000 undisplayed records	0.035 ms; undo 0.012 ms
Transfer and seal in the large owner	0.096 ms; undo 0.046 ms; redo 0.031 ms
Complete history for 500 changed records	1,583,998 estimated retained bytes; undo 70.5 ms; redo 71.8 ms
Load 1,000 views, each with 200 elements and 300 relations	19,135.5 ms total, including 1,979.3 ms maintenance; 651,596,129-byte file
Nested layout: 40 containers, 160 children, 120 relations	Preparation 34.7 ms; solver 17.5 ms; conversion 26.3 ms; application 43.0 ms; 880 text measurements

The metadata regression invokes the production coordinator/finalizer, checks geometry across every populated view, and requires zero content-change notifications. It also covers unchanged setters. Sparse owner-order preparation has a separate 96-operation independent order oracle with repeated replay. Native middle-vector insertion still shifts following pointers; the separately deferred forest costs remain.

The retained-history estimate is about 81.5% below the review's 8,561,984-byte baseline. It counts retained capacity, not process peak memory. The replay times are not a controlled before/after comparison and do not establish a latency improvement.

The load fixture has newly projected geometry but a warm filesystem cache and one grammar. It is not a cold-filesystem benchmark or a bound for real landscapes. The user chose to retain eager maintenance on 2026-09-29, accepting the measured load-time cost to preserve a fully maintained document at publication. **P2 is closed.**

The nested fixture is shallow and does not bound arbitrary nesting depth. An independent 10,000-case floating-point packing oracle caught a comparison reassociation; the corrected predicate matched every generated case.

The Qt solver regressions exercise the production progress dialog's Cancel button, timer responsiveness, refused reentrant actions/direct writes/history/save preparation, successful asynchronous undo/redo, and rejection of a late worker result. An executing OGDF call cannot be interrupted internally;

cancellation returns control promptly and discards its eventual result. The application owns and joins outstanding solver workers during shutdown.

Cohesion, documentation and verification

C1 is addressed by responsibility boundaries rather than a line-count target. Persistent geometry maintenance resides in the core geometry module; solver execution remains behind the provider interface and Qt owns progress/event handling. Raw codecs, native private access and replay preparation remain together because they enforce the same record contract. The implementation file is still large; no arbitrary split or general GUI refactor is part of this task.

Single-use command helpers, the alternate delete-command constructor and temporary heap commands were removed. The remaining landscape UUID supports forward target resolution and notifications; it does not determine history scope. The execute/already-applied distinction remains necessary for capture callbacks. Required no-else markers retain local reasons.

The architecture and user manual describe the storage boundary, strict replay, field-sensitive maintenance and cancellable layout. Manual scenarios MT2-7 and MT2-9 cover cancellation and cross-window recenter replay; import/merge coverage remains relevant to MT2-13 and MT2-20. The automated tests exercise real Qt windows and actions offscreen. No interactive manual GUI run is claimed.

The final source passed the full mechanical check: 82/82 tests and zero comment-audit findings. The documentation-reference audit resolved all 2,151 citations. Existing relation-reroute initializer, OGDF-header and duplicate-library compiler/linker warnings remain outside these fixes. All accepted review findings are addressed. The changes are uncommitted.