TikZRefinery

Senior Design (423-IDE401 / 424-IDE402) Core Report

Version 0.1.4-alpha

Alpha Project Specification — Submission Copy

by

Vincent Sun

Stevens Institute of Technology

November 24, 2026

© Stevens Institute of Technology 2026
ALL RIGHTS RESERVED

TikZRefinery

Senior Design (423-IDE401 / 424-IDE402) Core Report

Version 0.1.4-alpha
Alpha Project Specification — Submission Copy

Vincent Sun
Stevens Institute of Technology
November 24, 2026

TikZRefinery is a C++/Qt desktop project for graphically creating and editing TikZ diagrams while preserving editable source, including drawings that begin as artificial intelligence (AI)-generated code. This report records requirements, use cases, architecture, planned evaluation and dated prototype evidence.

The `0.1.4-alpha` editor displays explicit-coordinate rectangles, straight lines and default arrows. A separate rectangle module supplies presentation, selection and constrained movement/resizing; the host controls source rewrites and document acceptance. Its sidebar shell provides Shapes/Files tabs, collapse/reopen and session page/width retention. Shapes shows provider status with creation disabled; Files offers guarded Open. Qt parenting owns the UI widgets; the scene still owns its plugin store.

Session undo/redo restores exact before/after source snapshots for completed coordinate edits. The document owns history and publishes prepared source, records and history together. Save retains history; successful Open clears it. Vincent approved no automatic eviction for this prototype; a future limit awaits memory measurements. Generic transactions and file-store extraction remain planned.

Corner crossing, thin minima, bounds, cancellation, fitted canvas and save/reopen retain bounded evidence. The November 24 increment adds session-history tests and a matching package. These checks do not establish unrestricted TikZ support or product acceptance; older reports and packages remain preserved.

Catalogue/creation and shared-service ownership remain planned. New-document behavior, insertion contexts and folder browsing need further review. Shape discovery/creation target the later `0.2.0-alpha` milestone; updates, migrations and integrated compiler/PDF viewing are also unfinished. Demo 3 now presents the matching `0.1.4-alpha` baseline; earlier slides remain preserved.

Document Version

Current baseline: 0.1.4-alpha — November 24, 2026. The Alpha includes the rectangle plugin, collapsible sidebar shell and session undo/redo. The complete revision table and numbering policy are in Appendix A: Document History.

Table of Contents

Document Version iv

1 Introduction 1
- 1.1 Project Overview . . . 1
- 1.2 Project Member . . . 1

2 Team Declaration 2
- 2.1 Team Name and Membership . . . 2
- 2.2 Project Proposal . . . 2
- 2.3 Mission Statement . . . 3
- 2.4 Key Drivers . . . 3
- 2.5 Key Constraints . . . 3

3 Development Plan 4
- 3.1 Introduction . . . 4
- 3.2 Roles and Responsibilities . . . 5
  - 3.2.1 Stakeholder List . . . 5
- 3.3 Method . . . 6
  - 3.3.1 Selected Implementation Design . . . 7
  - 3.3.2 Review Gates and Change Control . . . 9
  - 3.3.3 Initial Work Breakdown . . . 9
- 3.4 Workspace . . . 9
- 3.5 Communication Plan . . . 11
- 3.6 Timeline and Milestones . . . 12
  - 3.6.1 Provisional Phase Gantt Chart . . . 13
  - 3.6.2 Possible Future UML Capability . . . 13
- 3.7 Testing Plan . . . 14
- 3.8 Risks . . . 15
- 3.9 Assumptions . . . 17
- 3.10 Milestone 2 Constraints and Design Iterations . . . 18
  - 3.10.1 Milestone 2 Submission Coverage . . . 19
  - 3.10.2 Course Pattern Adaptation . . . 20

4 User Stories **21**
4.1 User Story Register . . . 21
4.2 Validation Links . . . 25
4.3 Questions for Stakeholder Review . . . 26

5 Requirements **27**
5.1 Approach and Priority . . . 27
5.2 Category-Specific Requirement Registers . . . 28
5.2.1 Business Requirements . . . 28
5.2.2 Functional Requirements . . . 29
5.2.3 Data Requirements . . . 32
5.2.4 Interface Requirements . . . 36
5.2.5 Security Requirements . . . 38
5.2.6 Architectural Requirements . . . 39
5.2.7 Developer Requirements . . . 43
5.2.8 Performance Requirements . . . 45
5.2.9 Constraint Requirements . . . 45
5.3 Class Naming Interpretation . . . 46
5.4 Initial Explicit-Coordinate Subset . . . 46
5.5 Evidence and Review Boundaries . . . 47

6 Use Cases **48**
6.1 High-Level Use-Case Diagram . . . 48
6.2 Open and Preview a Diagram . . . 50
6.2.1 UC-001 Diagram . . . 51
6.3 Move or Resize a Supported Object . . . 52
6.3.1 UC-002 Diagram . . . 53
6.4 Save Revised Source . . . 55
6.4.1 UC-003 Diagram . . . 56
6.5 Undo or Redo an Edit . . . 57
6.5.1 UC-004 Diagram . . . 58
6.6 Reposition an Attached Label . . . 59
6.6.1 UC-005 Diagram . . . 60
6.7 Select an Obscured Element . . . 61
6.7.1 UC-006 Diagram . . . 62
6.8 Choose a Plugin or Shape Provider . . . 63
6.9 Handle an Available Plugin Update . . . 65
6.10 Create Using a Plugin-Provided Shape . . . 67

7 Software Architecture **69**
7.1 Scope and Evidence . . . 69
7.1.1 Feature Baselines and Package Identity . . . 70
7.2 Current Alpha Deployment . . . 71
7.3 Initial Persona and Interface Design . . . 72

7.3.1 PERS-001: TikZ-Literate Report Author . . . 72
7.3.2 Needs Mapped to the Current Interface . . . 72
7.3.3 Current Application Screenshot . . . 73
7.3.4 Persona Needs Mapped to Planned Creation . . . 74
7.3.5 Persona and Interface Review Still Required . . . 74
7.4 Model-View-Controller Allocation . . . 75
7.4.1 MVC Diagram . . . 76
7.5 Current Class Diagram . . . 77
7.6 Individual Class Diagrams . . . 79
7.6.1 tkrEditHistory . . . 79
7.6.2 tkrTikzDocument . . . 80
7.6.3 tkrMainWindow . . . 81
7.6.4 tkrDiagramView . . . 82
7.6.5 tkrDiagramScene . . . 83
7.6.6 tkrEditorController . . . 85
7.6.7 tkrTikzParserExplicit . . . 86
7.6.8 tkrCoordinateMapper . . . 87
7.6.9 tkrEditService . . . 88
7.7 Implemented Rectangle Plugin . . . 89
7.7.1 Current Provider Class Relationships . . . 90
7.7.2 Current Plugin Deployment . . . 92
7.7.3 Implemented Evidence and Limits . . . 92
7.8 Planned Transaction Design . . . 94
7.8.1 Proposed Data Records . . . 94
7.8.2 Planned Class Collaboration . . . 95
7.8.3 Responsibility and Design Review . . . 97
7.8.4 Reviewed Operation Contracts . . . 98
7.8.5 Planned Collaborator Class Details . . . 99
7.8.6 Elements, Source Bindings and Attachments . . . 101
7.8.7 Reviewed Label and Selection Interaction . . . 103
7.8.8 Reviewed Provider Boundary . . . 103
7.8.9 Rectangle Resize and Shape Plugin . . . 104
7.9 Reviewed Shape Plugin Contract . . . 105
7.9.1 UML and Diagram Convention . . . 105
7.9.2 Responsibilities and Module Boundary . . . 105
7.9.3 Versioned Data and Operation Records . . . 107
7.9.4 Provider Interface and Host Operations . . . 109
7.9.5 Planned MVC Allocation . . . 112
7.9.6 Registration, Compatibility and Lifetime . . . 113
7.9.7 Later Palette, Creation and Compiled Fallback . . . 115
7.9.8 First Extraction and Evidence Gate . . . 115
7.10 Reviewed Sidebar and Creation Design . . . 116
7.10.1 Implemented Sidebar Shell . . . 116
7.10.2 Planned Ownership and Collaborators . . . 119

7.10.3 Proposed Responsibilities and Method Roles . . . 120
7.10.4 Catalogue Types and Instance Boundary . . . 122
7.10.5 Planned Creation Activity . . . 123
7.10.6 Planned Sidebar Wireframes . . . 124

8 Open Design Decisions **125**
8.1 Temporary Review Workspace . . . 125
8.2 Next Bounded Review . . . 126

9 Codes, Standards and Conventions **127**
9.1 Applicability and Derived Design Obligations . . . 127
9.2 Applicability Limits . . . 128

10 Tests **130**
10.1 Proposed Test Cases . . . 130
10.2 Milestone 2 Test Execution Plan . . . 141
10.2.1 Levels, Data and Entry Conditions . . . 141
10.2.2 Oracles, Results and Coverage . . . 142
10.3 Coordinate and Box Reference Checks . . . 143
10.3.1 Independent Coordinate Expectations . . . 143
10.3.2 Separated Boxes Moved into Contact . . . 143
10.4 Explicit Reference Files . . . 145
10.5 Initial Native Build and GUI Check . . . 146
10.6 Rectangle Parsing and Canvas Display . . . 147
10.7 Line and Arrow Display . . . 148
10.8 Whole-Element Editing and Saved Output . . . 149
10.9 Canvas Contrast Check . . . 150
10.10 Windows Prototype Package Check . . . 150
10.11 Transaction Contract Verification Plan . . . 150
10.12 Rectangle Resize Check . . . 151
10.13 Rectangle Plugin Extraction Check . . . 151
10.14 Deployment Label Fixture . . . 151
10.15 Sidebar Shell Check . . . 152
10.16 Session History Check . . . 152

11 Traceability **153**
11.1 Milestone 2 Representative Traceability Paths . . . 153
11.2 Traceability Matrix . . . 154
11.3 Reverse Coverage . . . 159

12 Verification **160**
12.1 Purpose and Evidence Boundary . . . 160
12.2 Full Requirement Verification Map . . . 160
12.3 Evidence Already Obtained . . . 176

13 Validation **177**
13.1 Stakeholder Method and Acceptance . . . 177
13.2 Full User-Story Validation Map . . . 177

A Document History **181**

B Weekly Reports **196**
B.1 Week Report 12 (November 20, 2026) . . . 196
B.2 Week Report 11 (November 13, 2026) . . . 197
B.3 Week Report 10 (November 6, 2026) . . . 198
B.4 Week Report 9 (October 30, 2026) . . . 199
B.5 Week Report 8 (October 23, 2026) . . . 200
B.6 Week Report 7 (October 16, 2026) . . . 201
B.7 Week Report 6 (October 9, 2026) . . . 202
B.8 Week Report 5 (October 2, 2026) . . . 203
B.9 Week Report 4 (September 25, 2026) . . . 204
B.10 Week Report 3 (September 18, 2026) . . . 205
B.11 Week Report 2 (September 11, 2026) . . . 206
B.12 Week Report 1 (September 4, 2026) . . . 207

C Report Roadmap **208**
C.1 Purpose and Authority . . . 208
C.2 Adaptation of the Course Pattern . . . 208
C.3 Priority and Status Model . . . 209
C.4 Report Requirements Register . . . 210
C.5 Milestone 3 Submission Coverage . . . 225
C.6 Inclusion and Review Rules . . . 225

D Design Documents **226**
D.1 Scope and Evidence . . . 226
D.2 Open Source Activity . . . 228
D.3 Move/Resize and Source-Update Activity . . . 229
D.4 Prototype Move/Resize Sequence . . . 230
D.5 Current Undo/Redo Sequence . . . 231
D.6 Save Source Activity . . . 232
D.7 Close and Lifetime Activity . . . 233
D.8 Planned Common Transaction Activity . . . 234
D.9 Planned Transaction Interaction Sequence . . . 236
D.10 Planned Label and Selection Activity . . . 238

Bibliography **239**

Glossary **241**

Acronyms **244**

List of Tables

3.1 Development Roles and Responsibilities 5
3.2 Initial Class Responsibilities 7
3.3 Provisional Phase-Level Work Breakdown 9
3.4 Project Communication Plan 11
3.5 Development Phases and Milestones 12
3.6 Planned Test Levels 14
3.7 Initial Development Risk Register 15
3.8 Initial Planning Assumptions 17
3.9 Constraints, Alternatives and Consequences 18
3.10 Documented Feedback-to-Design Iterations 19
3.11 Milestone 2 Assignment-to-Report Map 20

4.1 Provisional User Stories 21

5.1 Provisional Business Requirements 28
5.2 Provisional Functional Requirements 29
5.3 Provisional Data Requirements 33
5.4 Provisional Interface Requirements 36
5.5 Provisional Security Requirements 38
5.6 Provisional Architectural Requirements 39
5.7 Provisional Developer Requirements 43
5.8 Provisional Performance Requirements 45
5.9 Provisional Constraint Requirements 45
5.10 Initial Editable Syntax Boundary 46

6.1 Proposed UC-001 Specification 50
6.2 Proposed UC-002 Specification 52
6.3 Proposed UC-003 Specification 55
6.4 Proposed UC-004 Specification 57
6.5 Proposed UC-005 Specification 59
6.6 Proposed UC-006 Specification 61
6.7 Proposed UC-007 Specification 63
6.8 Proposed UC-008 Specification 65
6.9 Proposed UC-009 Specification 67

7.1 Initial Persona Hypothesis . . . 72
7.2 Persona-to-Interface Mapping . . . 72
7.3 Persona-to-Planned-Interface Mapping . . . 74
7.4 Ownership Is Separate from MVC Role . . . 75
7.5 Current Plugin Responsibilities . . . 89
7.6 Planned Transaction Records . . . 94
7.7 Responsibility Review Directions . . . 97
7.8 Reviewed Planned Operations . . . 98
7.9 Reviewed Shared Element Information . . . 101
7.10 First Shape Plugin Design Decisions . . . 104
7.11 Plugin Delivery Boundaries . . . 105
7.12 Reviewed Plugin Record Sketches . . . 107
7.13 Sidebar and Creation Decisions . . . 118
7.14 Planned Sidebar Collaborator Roles . . . 120

8.1 Remaining Design Review . . . 125

9.1 Applicable Specifications, Guidance and Checks . . . 127

10.1 Proposed Product Test Cases . . . 130
10.2 Test Levels and Representative Data . . . 142
10.3 Initial Explicit-Coordinate Reference Corpus . . . 145
10.4 First Scaffold Check: Observations and Limits . . . 146

11.1 Representative Need-to-Design-to-Evaluation Paths . . . 153
11.2 Provisional Product Traceability . . . 154

12.1 Requirement-to-Verification Map . . . 161
12.2 Existing Bounded Prototype Evidence . . . 176

13.1 User-Story-to-Validation Map . . . 178

A.1 Document Update History . . . 182

C.1 Report Requirement Status Definitions . . . 209
C.2 Report Requirements Register . . . 210
C.3 Milestone 3 Topic Coverage . . . 225

D.1 Design Diagram Evidence and Requirement Links . . . 226

List of Figures

3.1 Provisional development schedule 13

6.1 Use-case goals overview 49
6.2 UC-001: Open and preview 51
6.3 UC-002: Move or resize 54
6.4 UC-003: Save source 56
6.5 UC-004: Undo or redo 58
6.6 UC-005: Reposition a label 60
6.7 UC-006: Select obscured element 62
6.8 UC-007: Choose a provider 64
6.9 UC-008: Handle provider update 66
6.10 UC-009: Create from plugin shape 68

7.1 Current Alpha physical deployment 71
7.2 Actual 0.1.4-alpha interface 73
7.3 Current MVC allocation 76
7.4 Current core host classes 78
7.5 Current session history 79
7.6 Current document 80
7.7 Current main window 81
7.8 Current diagram view 82
7.9 Current diagram scene 84
7.10 Current controller 85
7.11 Current explicit parser 86
7.12 Current coordinate mapper 87
7.13 Current edit service 88
7.14 Current provider classes 90
7.15 Current native module deployment 93
7.16 Planned generic transaction collaborators 96
7.17 Planned generic collaborator contracts 100
7.18 Planned shared element contracts 102
7.19 Reviewed wider plugin component boundaries 106
7.20 Reviewed wider plugin records 108
7.21 Planned provider interface extensions 110

7.22 Planned generic provider interaction . . . 111
7.23 Reviewed provider MVC allocation . . . 112
7.24 Planned provider lifecycle . . . 114
7.25 Current sidebar shell classes . . . 117
7.26 Planned shared-service ownership . . . 119
7.27 Planned catalogue and creation MVC . . . 121
7.28 Planned catalogue records . . . 122
7.29 Planned creation activity . . . 123
7.30 Planned sidebar interface sketches . . . 124

10.1 LaTeX rectangle contact reference . . . 144

D.1 Current Open activity . . . 228
D.2 Current move and resize activity . . . 229
D.3 Current move and resize sequence . . . 230
D.4 Current Undo and Redo sequence . . . 231
D.5 Current Save activity . . . 232
D.6 Current close and lifetime activity . . . 233
D.7 Planned generic edit acceptance . . . 235
D.8 Planned generic transaction interactions . . . 237
D.9 Planned label selection and movement . . . 238

Chapter 1

Introduction

1.1 Project Overview

TikZRefinery is a proposed graphical editor for diagrams written in LaTeX using TikZ, including diagrams initially generated by artificial intelligence. The project is motivated by the difficulty of making precise visual corrections directly in TikZ source code. The application is intended to let a user adjust diagram elements through a graphical interface while preserving editable TikZ source and supporting export through the LaTeX toolchain.

1.2 Project Member

Vincent Sun is independently designing and developing TikZRefinery as a desktop application. He is interested in learning how desktop applications are structured and in investigating how direct graphical edits can be synchronized with editable TikZ source. His current responsibilities include requirements analysis, architecture, user-interface design, implementation, testing, documentation, and project management.

Chapter 2

Team Declaration

2.1 Team Name and Membership

Team name: TikZ Refinery.
Team member: Vincent Sun, independent developer.

Vincent is responsible for project planning, requirements, architecture, interface design, implementation, testing, documentation, and review. The project will continue as an individual desktop-software effort.

2.2 Project Proposal

TikZRefinery is a proposed desktop graphical editor for creating and refining diagrams expressed in TikZ. TikZ produces precise technical graphics from text that works well with version control, but small visual corrections often require users to find and edit coordinates, dimensions, styles, and relationships indirectly in source code. This is especially inconvenient when an AI-generated diagram is useful but needs careful human adjustment.

The application would combine direct manipulation with source-level control. A user would enter TikZ code, compile and preview it through a local LaTeX toolchain, and adjust supported objects graphically. TikZRefinery would reflect supported edits in source that remains readable and usable outside the application. The initial minimum viable prototype will emphasize a deliberately limited TikZ subset, compilation and preview, basic object manipulation, and reliable round-trip editing rather than attempting to interpret every TikZ command.

The proposed product is a local desktop application rather than a hosted web service. This supports private documents and integration with an installed LaTeX environment. Vincent selected C++ with Qt 6 for implementation on October 1, 2026, replacing the earlier PySide6 candidate. The initial class design is recorded in Chapter 3; end-to-end feasibility and product verification remain pending.

2.3 Mission Statement

TikZRefinery's mission is to reduce the effort required to turn rough or AI-generated TikZ diagrams into precise, publication-ready, and still-editable source. Its business intent is to improve the productivity and accessibility of technical diagram authoring by joining graphical interaction with the reproducibility, portability, and version-control advantages of text-based LaTeX documents.

2.4 Key Drivers

The first driver is the gap between visual intent and source-level editing. TikZ is precise, but users must translate visible spacing or alignment problems into the responsible commands or coordinates. Direct manipulation could shorten that feedback loop while preserving the source.

The second driver is increasing use of AI-generated starting material. It can save time but often needs human judgment to correct layout, scale, labels, and relationships. TikZRefinery lets AI accelerate a first draft while a person retains precise, inspectable control instead of accepting an opaque image or repeatedly rewriting prompts.

The third driver is reproducible local work. TikZ source fits established Git and LaTeX workflows, while a desktop tool can process private documents without a remote service. Editable source also prevents the application from becoming the only place where a diagram can be maintained.

2.5 Key Constraints

The project is being completed by one developer within the senior-design schedule, so scope must be controlled aggressively. The first release cannot support all TikZ syntax, packages, and custom macros; it must prioritize a testable subset and coherent workflow over many partially reliable features.

The central technical constraint is source synchronization. TikZ is a programming language rather than a simple drawing-file format, so visually moving an object can affect coordinates, relative positioning, styles, and user-written expressions. The system must avoid silently damaging unsupported source. Early designs will need explicit parsing boundaries, predictable edit rules, and a preservation strategy for content the graphical layer cannot safely understand.

Compilation depends on an external LaTeX toolchain whose installation, packages, errors, and execution time vary between computers. The application must detect missing tools, present actionable failures, avoid freezing during compilation, and treat untrusted input cautiously. Preview and export must remain consistent with the saved source.

Finally, usability and verification must fit the schedule. Interaction and source feedback must remain clear. The previous group's public SDK has now been inspected as a reference, without importing code or assuming it satisfies this project's source-preservation contract; reuse decisions remain open.

Chapter 3

Development Plan

This chapter defines how TikZRefinery will be planned, developed, reviewed, and tested as an individual senior-design project. It converts the mission and constraints in Chapter 2 into a working process while leaving unbaselined product requirements and implementation feasibility open for review. C++ with Qt is the selected development direction. The plan will be revised when stakeholder feedback, feasibility evidence, or the course schedule changes.

3.1 Introduction

TikZRefinery is intended to become a local desktop application for graphically refining TikZ diagrams without giving up readable, editable source. The initial development objective is a verifiable end-to-end workflow: accept a supported TikZ document, compile and preview it through a local LaTeX toolchain, expose supported objects for graphical manipulation, and reflect an approved edit in source without damaging unsupported content.

This plan governs the work needed to reach that objective. It addresses responsibility, method, workspace, communication, scheduling, testing, risks, and assumptions. Product behavior is governed separately by Chapter 5; that chapter will contain the reviewed requirements and mappings needed for later use cases and tests.

3.2 Roles and Responsibilities

TikZRefinery is currently an individual project. One person may perform several roles, but the responsibilities remain distinct so that planning, implementation, and review do not become indistinguishable.

Table 3.1: Development Roles and Responsibilities

Participant or Role	**Current Assignment**	**Responsibilities**
Project owner and developer	Vincent Sun	Own scope and priorities; elicit and maintain requirements; design, implement, test, and document the product; retain project records; and approve submitted work.
Course instructor and project originator	Professor Darian Muresan	Provide the originating project intent, course expectations, early stakeholder feedback, milestone review, and evaluation. Instructor input is evidence and direction for review; implementation remains Vincent's responsibility.
AI-assisted tooling	OpenAI Codex, under Vincent's review	Assist with research, drafting, coding, consistency checks, builds, and test preparation within the controlled rules in internal AI records. It does not approve requirements, invent stakeholder evidence, or replace human review.
Candidate users and reviewers	TikZ and LaTeX users to be identified	Review workflows, terminology, usability, and prototype behavior when access is available. Until then, their presumed needs remain provisional.

Vincent is the final decision authority for project execution and academic submissions. The instructor remains the authority for course requirements and grading. If another student, advisor, or external reviewer formally joins the project, this table and the communication plan will be revised before assigning responsibility.

3.2.1 Stakeholder List

Professor Darian Muresan is the currently identified stakeholder beyond the project team. He originated the project idea and has an interest in whether the resulting editor supports the intended graphical TikZ workflow. As course instructor, he also reviews the project's academic milestones. His Qt recommendation is recorded in the method below; specific editing needs still require his review and should not be inferred from that recommendation.

Vincent is the project owner and developer, as listed above. No additional TikZ or LaTeX users have been confirmed as stakeholders. Their possible needs are represented only by provisional user stories in Chapter 4 until direct feedback is available.

3.3 Method

The project will use a lightweight iterative method rather than claim conformance to a full named process. Weekly work will be selected from reviewed course obligations, product risks, and the current requirements backlog. Early iterations will emphasize feasibility and risk reduction; later iterations will deliver narrow vertical slices that connect source input, compilation, preview, interaction, source update, and verification.

Each iteration will follow the same working cycle:

1. Select a small outcome supported by a reviewed requirement or explicit feasibility question.
2. Define the acceptance evidence before implementation begins.
3. Design and implement the smallest coherent change.
4. Run relevant automated and manual tests, record failures, and retain useful evidence.
5. Update requirements, risks, decisions, documentation, and the next iteration plan.

Feasibility prototypes answer technical questions but do not by themselves establish completed requirements. Vincent selected C++ with Qt on October 1, 2026, following the instructor recommendations as he recalls them. This replaces the earlier PySide6 candidate; no performance comparison was executed to establish that one language is faster for this application. The scaffold uses C++17, Qt 6 Core and Widgets, and CMake. A focused prototype must still open a representative TikZ sample, select and move one supported element, update its source, compile it with a local toolchain, and display the result. Measure responsiveness, implementation complexity, source preservation, and error recovery within the selected implementation. Two complete editor implementations are not planned solely for language comparison.

3.3.1 Selected Implementation Design

Vincent agreed the following initial responsibilities and ownership during the October 1–6 design discussion. This adapts the course pattern of recording components and coding conventions to the current development plan; fuller architectural views remain future work. Each primary class has matching `.h` and `.cpp` files, grouped under `code/src/` by concern.

Table 3.2: Initial Class Responsibilities

Class	Folder	Responsibility
`tkrTikzDocument`	`model/`	Preserve original source bytes, document revisions, parsed records, and validated minimal edits.
`tkrTikzParserExplicit`	`parsing/`	Interpret the initial supported syntax and identify editable coordinates and source ranges.
`tkrCoordinateMapper`	`geometry/`	Convert centimetre coordinates to and from scene positions using documented origin, scale, and axis rules.
`tkrEditorController`	`controller/`	Coordinate the document, scene, parser, mapper, file loading, and saving.
`tkrDiagramScene`	`ui/`	Own graphics items and report selection and drag gestures.
`tkrMainWindow`	`ui/`	Own the application components and provide window controls and unsaved-change handling.

The current organization also includes `tkrDiagramView` (viewport fit and paper rendering) and `tkrEditService` (geometry/replacement preparation). The source-based host-class view is in Section 7.5; the planned plugin boundary is in Section 7.9.

Ownership. The window owns the document, controller, and scene. The controller has non-owning references to the document and scene, and owns the parser, mapper and bounded edit service by value. The canvas view is owned through the window's widget hierarchy. The scene owns its graphics items. An accepted final-window close ends the event loop; window destruction then releases its owned components. Save, Discard, or Cancel handles unsaved work.

Conventions. Project class names use a lowercase `tkr` prefix, PascalCase main name, and optional specialization suffix; methods use camelCase. Variables combine type or category prefixes with descriptive meaning; private members add `m_`, meaning member. Allow at most three project-defined levels (main/base, child, grandchild), excluding Qt's existing ancestry. The current concrete classes remain `final`; the module realizes one provider interface for the present prototype. Prefer composition; document non-owning aggregation. Shared records are in `model/EditorTypes.h`; details are in `code/CODING_GUIDE.md` and Section 5.2.7. The naming and depth clarification was recorded through Vincent's account of instructor feedback on October 15.

Editing and evidence. Preserve source as bytes with guarded byte ranges. The whole-element workflow previews geometry during a drag, commits source once on release, and restores unchanged source geometry on cancellation. Saving uses `QSaveFile` with direct-write fallback disabled. The scaffold built successfully on October 20 with Qt 6.12.0 and MinGW 13.1.0; the foundation test passed. Vincent confirmed source display in the running window (Section 10.5). Rectangle, line and arrow display have bounded checks in Section 10.7. Whole-element editing and saved-output checks follow in Section 10.8; endpoint editing, broader parsing, compiler and PDF display components remain unfinished. This supports REQ-A-004–REQ-A-007 and REQ-D-001 without marking them complete.

The MVC allocation in Section 7.4 distinguishes the document model, window/scene views, controller, and its parser/mapper services. The responsibility overview in Figure 7.4 links each class to its attributes and operations in Section 7.6 in Chapter 7. Opening, whole-element dragging, saving, and closing activities remain in Appendix D. Existing control flow and future interaction are distinguished in each diagram.

3.3.2 Review Gates and Change Control

The near-term gates are problem and mission reviewed, development plan reviewed, initial stakeholder stories reviewed, initial requirements and use cases mapped, framework feasibility demonstrated, prototype scope baselined, and end-to-end workflow verified. Passing a gate requires the relevant report content and evidence; a successful demonstration without updated records is not sufficient.

A material change to mission, scope, supported TikZ syntax, framework, requirement wording, or milestone timing will be assessed for effects on requirements, use cases, architecture, tests, risks, and schedule. Stable identifiers will be preserved, superseded items will be marked Retired rather than deleted, and approved report changes will be entered in Document History.

3.3.3 Initial Work Breakdown

This breakdown groups work by deliverable and review stage. It does not assume that the current candidate requirements or the supported TikZ subset have been approved. Vincent currently owns each work package; tasks can be refined after stakeholder and feasibility review.

Table 3.3: Provisional Phase-Level Work Breakdown

ID	Work package	Principal output	Depends on
WBS-01	Needs and scope review	Reviewed stakeholder goals, reference examples when available, candidate requirements, and initial editable boundary	Current proposal
WBS-02	Concept and feasibility	Compared approaches, a measured Qt and source-editing prototype, and a recorded language and framework decision	WBS-01
WBS-03	Preliminary design	Architecture, source model, interaction flows, compiler boundary, and test design for the selected concept	WBS-02
WBS-04	Prototype and alpha evaluation	A working positional-editing path, integrated components, and requirement-linked test evidence	WBS-03
WBS-05	Beta refinement and final evaluation	Corrective changes, reviewed evaluation evidence, and the final report and release decision	WBS-04

3.4 Workspace

The private Git repository has separate `documents/` and `code/` folders. The report source is in `documents/latex/`, and the reviewed PDF is in `documents/`. Course originals remain as

references in `documents/template/`. Application source, tests, and examples belong under `code/`.

Development uses Windows, Git, and Visual Studio Code; the report uses the shared TinyTeX installation. The application will invoke a local LaTeX toolchain as an external process. Application development uses the selected C++/Qt direction, with CMake 3.21 or later and a compiler compatible with the installed Qt kit. Qt Core and Widgets are the current dependencies. The October 20 first build used Qt 6.12.0, MinGW 13.1.0, CMake 3.30.5, and Ninja 1.12.1. The build and foundation test passed; initial source display was observed. VS Code remains Vincent's preferred editor; Qt Creator is optional. The feasibility gate will assess the workflow described in the Method and Testing Plan.

The repository root will be the primary folder of a local project in the ChatGPT desktop app so related tasks share files and project instructions. Successive increments may continue in the same chat; a new chat is optional when requested or useful, following Vincent's October 21 preference. The repository's `HANDOFF.md` records the current checkpoint and next task; approved requirements and design remain authoritative in this report.

Git will track authored source and controlled records, while reproducible build artifacts remain ignored. Small edits use Git history for rollback; temporary staging copies are reserved for risky restructuring.

3.5 Communication Plan

Because the project currently has one developer, communication emphasizes traceable review rather than internal team meetings.

Table 3.4: Project Communication Plan

Communication	**Timing**	**Record or Outcome**
Weekly status report	Each assigned reporting week	Completed work, next work, action items, and current issues or risks.
Instructor discussion and feedback	Classes, milestone reviews, demos, and questions as needed	Clarifications, stakeholder observations, corrections, and decisions reflected in the appropriate controlled chapter.
Demonstration	At course demo dates	Concise presentation of the most important plan and product evidence, followed by recorded feedback and action items.
Repository and report review	Before each submission or controlled baseline	Source changes, document version, build result, cross-reference review, and Document History entry.
AI-assisted work review	Whenever assistance materially affects the project	Prompt-log record, unresolved assumptions, and Vincent's review or correction.

Instructor feedback will be recorded with its source and date before it is used to revise stories or requirements. Feedback from TikZ or LaTeX users would improve validation if available; absent interviews will not be replaced with invented statements. Initial stories remain open to later correction.

3.6 Timeline and Milestones

Work is organized around the milestone sequence shown in the available course slides and the technical dependency from stakeholder intent to verified implementation. The confirmed near-term date is Demo 1 on September 24, 2026. Vincent supplied tentative targets for the later phases: concept development in October–November 2026, preliminary design by November 2026, implementation and alpha evaluation by January 2027, and beta refinement and final evaluation in February–March 2027. These are planning targets, not confirmed course deadlines; they may change with course announcements, instructor feedback, or feasibility results. The chart below is a provisional phase view. A controlled Gantt chart with confirmed dates, task dependencies, critical path, and status will follow when the schedule is established, as required by RPT-025.

Table 3.5: Development Phases and Milestones

Phase	**Timing**	**Principal Outcome**	**Current State**
Project definition	Through September 18, 2026	Team Declaration, mission, drivers, constraints, and initial prototype boundary	Completed and subject to continuing refinement
Development-plan and inception review	September 22–24, 2026	Development plan plus initial stakeholder stories, mapped requirements, mapped use cases, glossary and index examples, and supporting citations for Demo 1	In progress
Concept development and specification	October–November 2026 (tentative)	Reviewed user needs, requirement baseline, use-case model, concept alternatives, supported TikZ subset, and selected development approach	Planned
Preliminary design	By November 2026 (tentative)	Architecture, interaction design, source model, compiler boundary, test design, and implementation plan	Planned
Implementation and alpha evaluation	By January 2027 (tentative)	Working vertical slices, subsystem evidence, integrated alpha scope, and updated cost and schedule data	Planned
Beta refinement and final evaluation	February–March 2027 (tentative)	Integrated desktop application, performance and usability evidence, controlled release, and final report	Planned

3.6.1 Provisional Phase Gantt Chart

The concept and beta phase ranges reflect Vincent's tentative months. November for preliminary design and December–January for implementation and alpha evaluation are illustrative planning windows inferred from the phase sequence and target months; their start dates and any overlap remain unconfirmed. All bars are provisional, and none indicates completed work.

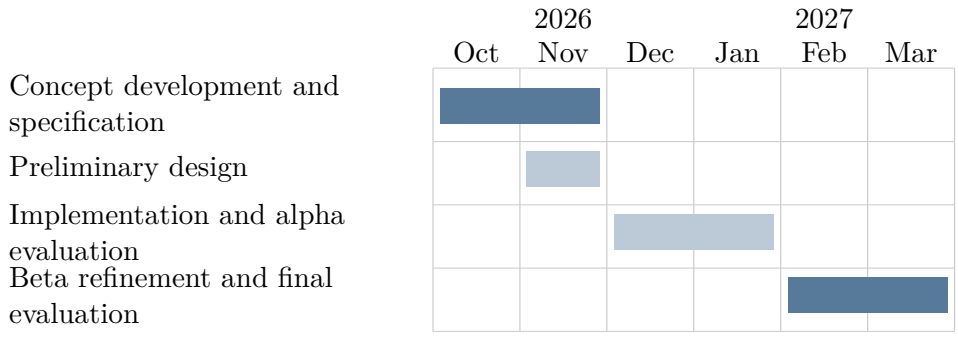


Figure 3.1: Provisional broad-phase schedule. Dark bars use Vincent's stated ranges; light bars show inferred working windows.

The immediate critical path is to review stakeholder evidence, define a small supported editing workflow, map initial stories to requirements and use cases, and perform the framework feasibility work. A basic working application must demonstrate that minimum workflow before development expands to broader features.

3.6.2 Possible Future UML Capability

Vincent reports that he and Professor Muresan discussed creating UML diagrams with the editor as a possible later capability. It is outside the minimum positional-editing demonstration. Supported UML notation, generation behavior, and source representation require separate review before becoming requirements.

3.7 Testing Plan

Testing will grow with reviewed requirements and implementation. Map each approved product requirement to a use case and repeatable verification method. Keep planned tests, executed tests, and passing evidence distinct.

Table 3.6: Planned Test Levels

Level	Purpose	Representative Evidence
Feasibility tests	Resolve architectural uncertainty before committing to a framework or source-transformation approach	Small executable prototypes, observations, limitations, and decision records
Unit tests	Verify parsing, model, coordinate, transformation, and source-generation behavior in isolation	Automated inputs, expected outputs, edge cases, and regression results
Integration tests	Verify interactions among source handling, the external compiler, preview generation, graphical state, and saved output	Controlled sample documents, subprocess results, generated previews, and source diffs
User-interface tests	Verify principal workflows, feedback, error handling, keyboard or pointer interaction, and recovery	Repeatable procedures, screen evidence when useful, and observed results
System verification	Demonstrate that the integrated prototype satisfies its approved requirements under the stated environment	Requirement-linked test records and pass/fail evidence
Stakeholder validation	Determine whether the resulting workflow addresses the actual diagram-refinement problem	Instructor and available user review, demonstration feedback, and resulting actions

The regression corpus will cover supported examples, malformed input, unsupported constructs, compiler failures, and source preservation during load/save. Maintain detailed cases and results in the testing, traceability, verification, and validation chapters.

The first geometry checks will independently verify coordinate mapping and move separated boxes into contact (TC-028 and TC-029). Comparing the editor with a PDF compiled from saved source at a common origin, scale, and fixed extent will guide the conversion design. Start with the explicit-coordinate subset and reference corpus in Chapter 5 and Chapter 10; instructor examples will inform later syntax coverage when available.

On a documented computer and toolchain, time opening and parsing, graphical movement, source updates, compilation, and preview display separately. Use representative diagrams, including a complex one when available, and record memory use, pauses, failures, and implementation effort. Stakeholder feedback will set quantitative response and preview targets. Investigate measured bottlenecks in the selected C++/Qt implementation and record trade-offs; language choice alone does not establish acceptable performance.

3.8 Risks

Likelihood and impact are preliminary qualitative assessments. Vincent owns the current responses and will revise them when feasibility or stakeholder evidence changes the assessment.

Table 3.7: Initial Development Risk Register

ID	Risk	Likelihood	Impact	Response
RSK-001	Arbitrary TikZ syntax exceeds the capacity of an individual project.	High	High	Baseline a deliberately limited subset and prioritize one verifiable end-to-end workflow.
RSK-002	A graphical edit changes or damages source the application does not fully understand.	High	High	Keep source authoritative, define explicit editing boundaries, preserve unsupported content, show proposed changes, and test round trips against a regression corpus.
RSK-003	External LaTeX installations, packages, errors, or execution time produce inconsistent behavior.	Medium	High	Detect tools and packages, run compilation without blocking the interface, capture diagnostics, define a reference environment, and test common failure paths.
RSK-004	Untrusted TeX input causes unsafe external-process behavior.	Medium	High	Treat documents as untrusted, use restrictive compilation settings, avoid enabling shell escape by default, and document residual limits.
RSK-005	Solo-developer capacity or schedule pressure prevents broad feature coverage.	High	High	Rank work by priority and risk, keep increments small, reserve schedule margin, and defer breadth before weakening the core workflow.
RSK-006	The preferred desktop framework or graphics model proves unsuitable.	Medium	Medium	Complete a focused C++/Qt feasibility test before the architecture is baselined, measure the core workflow, and preserve platform-independent requirements.

Continued on next page

Initial Development Risk Register (continued)

ID	Risk	Likelihood	Impact	Response
RSK-007	Limited access to additional users leaves usability assumptions weakly validated.	Medium	Medium	Use the instructor as the initial stakeholder, label provisional claims, seek feedback during demos, and add broader user review when feasible.
RSK-008	Differences from the previous project's graph/export approach may hide relevant lessons or unsupported assumptions.	Medium	Medium	Its public source and SDK were inspected; compare reusable concepts against our source-preserving contract rather than assuming equivalent behavior.

3.9 Assumptions

The following assumptions support planning but are not product requirements. Each will be reviewed as evidence becomes available.

Table 3.8: Initial Planning Assumptions

ID	Assumption	If the Assumption Is Invalid
ASM-001	Explicit-coordinate lines, arrows, and drawn rectangles are adequate for initial feasibility tests. Vincent selected this starting boundary; broader syntax coverage remains a product goal.	Expand the reference corpus using instructor examples and revise the editing approach as needed.
ASM-002	Users who need final TikZ output can provide or install a compatible local LaTeX toolchain.	Define clearer setup support or reconsider how compilation and preview are supplied.
ASM-003	TikZ source remains the authoritative portable artifact, and the graphical model may represent only supported portions.	Revisit the data model and round-trip objective before implementation continues.
ASM-004	Professor Muresan can provide the principal initial stakeholder perspective and early feedback.	Treat the initial stories as incomplete and seek another qualified TikZ or LaTeX user before baselining affected needs.
ASM-005	The project will continue with one developer for the current planning period.	Reassign responsibilities and revise communication, schedule, and governance if membership changes.
ASM-006	Work can proceed without the previous group's repository.	Continue with independent evidence; if the material arrives, review it for constraints, lessons, and reusable ideas before changing the plan.
ASM-007	The Windows development environment can support the selected desktop framework and local compiler integration.	Reassess the framework, supported platform baseline, and schedule using feasibility evidence.

These assumptions will be converted into requirements, dependencies, or risks when review shows that they impose a binding obligation or measurable product condition.

3.10 Milestone 2 Constraints and Design Iterations

These constraints come from the project context and reviewed decisions, not from assumed user-study results. Rechecks below are explicitly planned or bounded historical observations.

Table 3.9: Constraints, Alternatives and Consequences

Constraint / driver	**Decision and alternative considered**	**Tradeoff / check**
Solo developer and course deadline	Use a conservative supported subset and focused vertical increments; defer universal syntax and broad refactoring.	Lower coverage in exchange for source safety and reviewable progress; retain corpus regressions and record unsupported forms.
Editable source must remain authoritative	Choose byte-bound source transactions over regenerating the entire diagram; REQ-D-001, REQ-A-008.	Requires bindings/context and conservative rejection; exact source diffs, shared-definition and history checks detect unintended effects.
External LaTeX dependencies and variable run time	Preview compilation is separate from live interaction; REQ-A-001, REQ-P-001.	Additional asynchronous state and diagnostics; test slow/failing/missing dependencies and stale preview handling before acceptance.
Local and offline workflow	Keep core local, subject to stakeholder validation; REQ-C-001, VAL-006.	Local installation/configuration burden; disconnected workflow and dependency diagnostics must be evaluated.
Overlapping labels and varied source forms	Separate geometry/bindings/capabilities and use explicit selection menus; REQ-D-009, REQ-I-008.	More data contracts and menu interaction; preserve attachments/shared definitions and validate selection with authors.
Exact undo within finite memory	Start with exact session snapshots; document owns history.	Simple exact restoration costs memory; measure before setting retention/optimization policy. No unreviewed eviction limit is imposed.

Table 3.10: Documented Feedback-to-Design Iterations

Iteration / finding	Design change	Recheck and evidence	Disposition
ITR-001: October 21 display priority	Vincent reported that lines/arrows did not compile in the editor and prioritized display. Inspection distinguished unsupported editor interpretation from valid original TikZ; the subset was extended.	Eight fixed-extent independent TinyTeX comparisons and six startup fixtures checked display; retained evidence in Section 10.7.	Completed bounded recheck; broader syntax, native acceptance and integrated compilation remain pending.
ITR-002: October 27–30 ownership review	Concern about controller responsibility led to document-owned history and controller-owned edit/file services with atomic source/history publication; REQ-A-012–REQ-A-014.	TC-042, TC-056, TC-057; inspect ownership and inject publication/I/O failures.	Design reviewed with Vincent; implementation and rechecks planned.
ITR-003: October 29 label/selection review	Attached labels and obscured hosts exposed two-point record and hit-selection limits. Separate bindings, attachments and presentation and add point/area candidate menus; REQ-D-009, REQ-I-008–REQ-I-010.	TC-044, TC-047, TC-053, TC-054, TC-055 and VAL-009, VAL-010; measure observed user problems before adjusting the 6-pixel starting setting.	Owner interaction decisions recorded; code and participant rechecks planned.

3.10.1 Milestone 2 Submission Coverage

Assignment inspected in Vincent's Chrome tab on October 30, 2026: Milestone 2 – Project Specification, due October 30 at 11:59 p.m. The required single updated team report is the submission edition; internal AI records remain in the developer edition.

Table 3.11: Milestone 2 Assignment-to-Report Map

Assignment item	Report location / evidence boundary
Bidirectional stories to validation with acceptance criteria	Chapter 4, Chapter 13; full ten-story map with reciprocal links and planned participant evidence.
Bidirectional requirements to verification with criteria/method/evidence	Chapter 5, Chapter 10, Chapter 12; all requirement/check rows are mapped; product acceptance remains pending.
Use cases, activities, proposed classes and interaction sequences	Chapter 6, Chapter 7, Appendix D; current and planned diagrams are labelled separately.
Testing plan with levels, cases, data, outcomes and traceability	Section 10.2, full case register and Chapter 11; historical results retain dates/limits.
Dedicated codes/standards applicability and design/check links	Chapter 9; external specifications, adopted guidance and local conventions are distinguished.
Multiple constraints, tradeoffs and feedback-driven iteration	Tables 3.9 and 3.10; one bounded completed recheck plus planned design rechecks.

3.10.2 Course Pattern Adaptation

Adopt stable VER/VAL identifiers and reciprocal maps from the course verification and validation components; adapt their examples into planned local TikZ tasks and measurable oracles, retaining no example Pass claims. Adapt the codes/standards and constraints headings from the course Part II into project-specific content; integrate constraints in this existing plan. Use TikZ class/activity notation and a sequence where atomic interaction order needs explanation. Defer broad 4+1/hardware views without evidence. The temporary open-decisions chapter is explicitly requested by Vincent and will be removed when its decisions have entered the controlled artifacts.

Chapter 4

User Stories

These user stories were drafted with Vincent for Demo 1 review. They are not interview findings or a fully approved backlog. The transaction/history behavior and two plugin capabilities were subsequently agreed with Vincent on October 22, 2026; implementation remains pending. Professor Muresan originated the graphical TikZ editor idea and recommended Qt. The specific editing behavior below reflects Vincent's recollection of the intended workflow; it is not presented as a quotation or confirmed instruction from the professor. The remaining details are design proposals to validate with him and, if available, TikZ users.

4.1 User Story Register

The following acceptance criterions are examples for review.

Table 4.1: Provisional User Stories

ID	Goal and reviewable acceptance example	Priority	Requirements / validation
US-001	As a TikZ author, I want to open a local document and see its compiled diagram, so I can identify what to refine. I need the preview to represent the source I selected while leaving that input file unchanged. Example: a supported sample opens and displays its current rendered result.	Must	REQ-A-001, REQ-A-002, REQ-A-003, REQ-A-004, REQ-F-001, REQ-F-002, REQ-F-008, REQ-I-017, REQ-I-018, REQ-I-019, REQ-I-020 VAL-001

ID	Goal and reviewable acceptance example	Priority	Requirements / validation
US-002	As a TikZ author, I want to move supported lines, arrows, and boxes visually, adjust a line or arrow endpoint, and resize a supported box, so I can refine their layout without manually finding coordinates. I want a clearly bounded, fitted canvas with clean drag feedback, shapes that stay clear when I zoom, and compiled output that matches the edit. Examples: dragging a segment body moves both endpoints, dragging one handle changes only its endpoint, dragging a box body preserves its size, and a resize handle changes its dimensions. These gestures reflect Vincent's account of the professor discussion and remain provisional.	Must	REQ-A-001, REQ-A-004, REQ-A-006, REQ-A-007, REQ-A-023, REQ-B-001, REQ-D-013, REQ-D-014, REQ-D-015, REQ-F-003, REQ-F-004, REQ-F-009, REQ-F-010, REQ-F-011, REQ-F-025, REQ-I-001, REQ-I-011, REQ-I-012, REQ-I-013, REQ-I-014, REQ-P-001 VAL-002
US-003	As a TikZ author, I want edits to preserve source outside the supported object, so I can continue editing my document normally. A diagram may contain nearby commands the graphical editor does not understand, and I do not want those commands rewritten silently. Example: a source diff shows only the intended coordinate edits when a segment or box moves.	Must	REQ-A-006, REQ-A-007, REQ-D-001, REQ-D-010, REQ-D-017, REQ-F-004, REQ-F-024, REQ-F-034 VAL-003
US-004	As a TikZ author, I want clear feedback when a document fails to compile or contains unsupported syntax, so I can recover without losing my work. I need to know whether the preview failed or an edit was blocked while keeping the opened source available. Example: the document stays available and a diagnostic identifies the problem.	Should	REQ-D-015, REQ-F-005, REQ-F-007, REQ-F-031, REQ-F-032, REQ-I-006, REQ-I-007, REQ-I-016, REQ-I-017, REQ-S-001 VAL-004
US-005	As a TikZ author, I want to save the revised source locally, so I can compile or edit it again with standard LaTeX tools. The saved file should remain usable outside TikZRefinery rather than depend on an internal-only format. Example: reopening and compiling the saved file outside TikZRefinery retains the approved edit.	Must	REQ-A-014, REQ-B-002, REQ-D-001, REQ-F-006 VAL-005

ID	Goal and reviewable acceptance example	Priority	Requirements / validation
US-006	As a TikZ author working with private or offline documents, I want the core editing workflow to work without a network service, so I can refine local diagrams wherever the local LaTeX toolchain is available. Example: I can open, preview, move a supported object, and save while disconnected. This is a proposal derived from the local-desktop project direction, not a stakeholder interview finding.	Must	REQ-C-001, REQ-F-028 VAL-006
US-007	As a TikZ author, I want to undo and redo completed edits while retaining exact source and clear saved/unsaved state, so I can explore layouts and recover earlier results. A completed drag is one step; cancellation creates none. The save/open history rules were agreed with Vincent on October 22.	Must	REQ-A-008, REQ-A-010, REQ-A-011, REQ-A-012, REQ-A-013, REQ-A-014, REQ-D-002, REQ-D-003, REQ-D-004, REQ-D-005, REQ-D-006, REQ-D-007, REQ-D-008, REQ-D-011, REQ-F-012, REQ-F-013, REQ-F-014, REQ-F-015, REQ-F-016, REQ-F-017, REQ-F-018, REQ-F-019, REQ-F-020, REQ-I-002, REQ-I-003, REQ-I-004, REQ-I-005, REQ-I-006 VAL-007

ID	Goal and reviewable acceptance example	Priority	Requirements / validation
US-008	As a TikZ author, I want compatible shape and syntax providers with distinguishable capabilities, explicit provider choices and non-disruptive updates, so extension support grows without losing source or changing an active element's provider unexpectedly. Vincent reviewed registration, operation and lifecycle policy in November; the bounded rectangle extraction is implemented; wider mechanisms and user validation remain pending. The planned palette and canvas share host catalogue/provider access with explicit ownership.	Must	REQ-A-008, REQ-A-009, REQ-A-015, REQ-A-016, REQ-A-017, REQ-A-018, REQ-A-019, REQ-A-020, REQ-A-021, REQ-A-022, REQ-A-023, REQ-A-024, REQ-D-009, REQ-D-010, REQ-D-012, REQ-D-013, REQ-D-014, REQ-D-015, REQ-D-016, REQ-D-017, REQ-D-018, REQ-F-026, REQ-F-027, REQ-F-028, REQ-F-029, REQ-F-030, REQ-F-031, REQ-F-032, REQ-F-033, REQ-I-015, REQ-I-016, REQ-I-017 VAL-008
US-009	As a TikZ author, I want to reposition a supported attached label along and away from its host without moving the host or losing attachment, so I can improve readability while preserving the diagram relationships. Example: move a label in both axes, retain its unrequested appearance, and undo the action exactly. Vincent reviewed this behavior in October; stakeholder validation remains planned.	Must	REQ-A-012, REQ-A-013, REQ-A-015, REQ-A-016, REQ-A-017, REQ-D-009, REQ-D-010, REQ-D-012, REQ-F-021, REQ-F-022, REQ-F-023, REQ-F-024, REQ-I-006, REQ-I-007 VAL-009

ID	Goal and reviewable acceptance example	Priority	Requirements / validation
US-010	As a TikZ author working with overlapping objects, I want to identify and select an obscured element without moving other objects first, so I can refine crowded diagrams. Example: choose a hidden host from a highlighted point/area candidate menu. Vincent reviewed the interaction; broader user validation remains planned.	Must	REQ-A-015, REQ-D-009, REQ-D-012, REQ-I-008, REQ-I-009, REQ-I-010 VAL-010
US-011	As a TikZ author, I want to browse distinguishable shapes in a collapsible sidebar, click or drag to place a supported element, preview it and recover from cancellation or rejection, so I can create diagrams while retaining editable source. Accepted creation selects the element and returns to selection. Vincent reports Professor Darian requested creation; New/Save As and insertion contexts remain open, and this is not an interview finding.	Must	REQ-A-024, REQ-D-018, REQ-F-033, REQ-F-034, REQ-F-035, REQ-I-006, REQ-I-007, REQ-I-013, REQ-I-015, REQ-I-018, REQ-I-019, REQ-I-020, REQ-I-021 VAL-011

4.2 Validation Links

The initial PERS-001 profile and current interface mapping connect the editing, source and saving stories to a proposed author context. The persona remains a hypothesis; it adds no participant evidence or accepted backlog item.

Each story row links to its planned validation scenario. The validation register states the realistic task, acceptance criterion, method and expected evidence separately from requirement verification. Owner design agreement does not establish that a user has completed the scenario successfully.

4.3 Questions for Stakeholder Review

The Demo 1 discussion can supply the stakeholder detail missing from these drafts. In particular, ask Professor Muresan to confirm these points:

1. Which visual edits and source-level feedback must the first prototype support?
2. Which TikZ constructs should be in the initial editable subset?
3. Is offline operation a required constraint?
4. What would demonstrate that source preservation and saving are acceptable to a TikZ author?

Record his answers as stakeholder evidence before treating the affected stories or requirements as approved.

Chapter 5

Requirements

This candidate product register supports ongoing project review. No entry has completed implementation and verification or entered a product baseline. The initial explicit-coordinate test subset is documented below; quantitative thresholds and final acceptance procedures remain to be agreed. On October 22, 2026, Vincent agreed the transaction/history behavior and plugin scope recorded directly in the registers below; those requirements remain Open because implementation and verification are pending. The October 27–30 review also settled collaborator ownership, result feedback, label attachment/local edits, overlap selection and provider boundaries. Plugin registration, operation and lifecycle policies were reviewed in November; Section 7.9 records their planned interfaces. The November 22 sidebar/creation review is recorded as planned design in Section 7.10; it does not change the 0.1.2-alpha editor. The November 22 sidebar/creation review is recorded as planned design in Section 7.10; it does not change the 0.1.2-alpha editor. Detailed mechanisms remain in Chapter 8. The first objective is one working path from local source to preview, edit, and saved source.

5.1 Approach and Priority

Stories in Chapter 4 express user value; the statements below are candidate obligations. Stable IDs support later change tracking. Must marks the minimum workflow and protection of source changes; Should marks important recovery behavior. Status remains Open until implementation and verification evidence exists. Requirements will be reviewed against a documented TikZ subset and mapped to use cases and proposed tests in this report. User-story links for each applicable requirement appear in Table 11.2 in Chapter 11. This separation follows requirements engineering guidance [3]. The IDs identify categories: B for business outcomes, F for functional behavior, D for data integrity, I for interface behavior, S for security, A for architecture, DEV for developer conventions, P for performance, and C for an operating constraint. Business outcomes state the value sought from the prototype; their linked functional and data requirements specify the behaviors that make those outcomes possible.

5.2 Category-Specific Requirement Registers

The nine registers below group the current candidate requirements by their existing IDs. Each retains the template's priority, status, implementation version, use-case, and test-case fields. Categories without a supported candidate requirement are not filled with placeholder entries.

5.2.1 Business Requirements

The two candidate outcomes are an end-to-end positional correction and a revised diagram that remains usable in a standard LaTeX workflow. The minimum demonstration will use a supported reference diagram; Vincent reports that Professor Muresan has samples that may inform the final test selection. No participant study or time-saving threshold is required for this initial capability check.

Table 5.1: Provisional Business Requirements

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-B-001	The minimum prototype shall enable a LaTeX documentation author to reposition at least one supported TikZ diagram element through the graphical editor.	Must	Open	∅	UC-002	TC-011 VER-001
REQ-B-002	The minimum prototype shall let a LaTeX documentation author continue editing and compiling a graphically revised diagram with standard LaTeX tools outside TikZRefinery.	Must	Open	∅	UC-003	TC-011 VER-002

© Stevens Institute of Technology

5.2.2 Functional Requirements

The transaction and history rules agreed on October 22 are recorded as individual requirements here and in the Data, Interface and Architectural registers. They remain Open pending implementation and verification.

Table 5.2: Provisional Functional Requirements

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-F-001	The application shall open a user-selected, readable local TikZ source file without changing it.	Must	Open	∅	UC-001	TC-001 VER-003
REQ-F-002	The application shall compile an opened supported document with a configured local LaTeX toolchain and display its rendered result.	Must	Open	∅	UC-001	TC-002 VER-004
REQ-F-003	The application shall let the user select and move an object within the documented editable TikZ subset.	Must	Open	∅	UC-002	TC-003 VER-005
REQ-F-004	For a supported move, the application shall update the corresponding source coordinates and show a preview consistent with that change.	Must	Open	∅	UC-002	TC-004, TC-027, TC-029 VER-006
REQ-F-005	The application shall report compiler failure and unsupported editable constructs without discarding the opened source.	Should	Open	∅	UC-001, UC-002	TC-005, TC-006 VER-007
REQ-F-006	The application shall save a user-approved revision as local TikZ source and report whether the write succeeded.	Must	Open	∅	UC-003	TC-007, TC-008 VER-008
REQ-F-007	If a preview compilation fails, the application shall keep the last successfully generated preview visible and identify it as a prior successful result rather than the result of the failed compilation.	Should	Open	∅	UC-001, UC-002, UC-004	TC-012, TC-039 VER-009
REQ-F-008	When generating a supported document's preview, the application shall include its available required packages, styles, and definitions and report any required dependency that is unavailable.	Must	Open	∅	UC-001	TC-013 VER-010
REQ-F-009	For a supported straight line or arrow, dragging its body shall translate the entire segment so both endpoints move by the same displacement.	Must	Open	∅	UC-002	TC-021, TC-022 VER-011

Continued on next page

© Stevens Institute of Technology

Provisional Functional Requirements (continued)

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-F-010	For a supported straight line or arrow, dragging an endpoint handle shall move only the selected endpoint while the other endpoint stays fixed.	Must	Open	∅	UC-002	TC-023, TC-024 VER-012
REQ-F-011	The application shall let the user select and move a supported rectangular box as a whole through the graphical editor.	Must	Open	∅	UC-002	TC-025 VER-013
REQ-F-012	The application shall let the author undo and redo each completed editing action as one step, restoring exact source bytes including comments, whitespace, line endings and numeric spelling.	Must	Open	∅	UC-004	TC-035, TC-037 VER-014
REQ-F-013	Edit history shall last only for the current editing session; it shall not be embedded in the TeX file or restored after application restart.	Must	Open	∅	UC-004	TC-038 VER-015
REQ-F-014	Invoking Undo during an active drag shall cancel that preview before undoing the last completed edit.	Must	Open	∅	UC-004	TC-036 VER-016
REQ-F-015	A new accepted changed edit after Undo shall discard the abandoned redo branch; selection, preview cancellation, rejected edits and no-op actions shall preserve that branch.	Must	Open	∅	UC-002, UC-004	TC-037 VER-017
REQ-F-016	A successful Save shall preserve the current undo and redo history.	Must	Open	∅	UC-003, UC-004	TC-038 VER-018
REQ-F-017	Successfully opening another document or reloading the current document shall clear its edit history.	Must	Open	∅	UC-001, UC-004	TC-038 VER-019
REQ-F-018	A failed Save or cancelled or failed Open shall preserve the current document and history; a successful Save explicitly chosen before a failed or cancelled Open shall remain effective.	Must	Open	∅	UC-001, UC-003, UC-004	TC-038 VER-020
REQ-F-019	Undo and Redo shall change only in-memory document state without writing the source file to disk.	Must	Open	∅	UC-004	TC-038 VER-021
REQ-F-020	Before replacing a document with unsaved changes, Open or Reload shall offer Save, Discard and Cancel; Cancel or a failed requested Save shall prevent replacement.	Must	Open	∅	UC-001, UC-004	TC-038 VER-022

Continued on next page

Provisional Functional Requirements (continued)

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-F-021	For a supported attached label, dragging in both canvas axes shall adjust its relative placement without changing host geometry, text, rotation, anchor, alignment or styling unless explicitly requested.	Must	Open	∅	UC-005	TC-044 VER-023
REQ-F-022	Moving a supported host shall move its attached labels according to their existing attachment rules.	Must	Open	∅	UC-002, UC-005	TC-045 VER-024
REQ-F-023	Dragging an attached label shall retain attachment regardless of drag distance; detachment shall require an explicit user action and a supported source rewrite.	Must	Open	∅	UC-005	TC-046 VER-025
REQ-F-024	An edit of one selected label shall preserve shared definitions and unrelated elements, using a supported local adjustment; if no supported independent adjustment exists, the operation shall be unavailable with a reason.	Must	Open	∅	UC-005	TC-047 VER-026
REQ-F-025	The editor shall resize a supported explicit-coordinate rectangle with screen-sized corner handles, keeping the opposite corner fixed, allowing crossing, and constraining width and height independently to at least 0.01 cm. Preview shall preserve source; release shall commit only necessary coordinates as one guarded edit.	Must	Open	∅	UC-002	TC-060 VER-073
REQ-F-026	The application shall present duplicate declared plugin IDs with candidate name, version and file location, and let the author choose which candidate to activate; conflicting candidates shall not be selected by discovery order.	Must	Open	∅	UC-007	TC-069 VER-081
REQ-F-027	The application shall allow compatible providers for the same shape family to coexist, bind each element to its chosen provider instance/version, and preserve that binding during an operation.	Must	Open	∅	UC-002, UC-007	TC-070 VER-082
REQ-F-028	Available compatible plugin updates shall normally be applied at application startup before diagrams are opened or restored; an unavailable update service shall not prevent the local editing workflow.	Must	Open	∅	UC-008	TC-075 VER-083

Continued on next page

Provisional Functional Requirements (continued)

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-F-029	During a session the application shall prepare an update in the background and activate it only after validation and a fresh check that its provider is unused.	Must	Open	∅	UC-008	TC-076 VER-084
REQ-F-030	An update discovered for an in-use provider shall leave that version active and let the author defer the update or restart through the unsaved-work guard; deferred updates shall be considered at the next startup.	Must	Open	∅	UC-008	TC-077 VER-085
REQ-F-031	Plugin installation and activation shall retain the previous working version until replacement succeeds; failure shall preserve document source and provider choices and report the reason.	Must	Open	∅	UC-008	TC-078 VER-086
REQ-F-032	A missing or incompatible provider shall preserve source, identify unavailable support and its reason, and shall not silently substitute another provider.	Must	Open	∅	UC-001, UC-002, UC-007	TC-074 VER-087
REQ-F-033	The application shall enumerate catalogue entries offered by active compatible providers, with shape family/type, readable name, preview and creation support, while distinguishing provider alternatives.	Must	Open	∅	UC-009	TC-079 VER-088
REQ-F-034	The application shall create supported elements using a selected catalogue/provider entry and a supported syntax-insertion route; host validation and document acceptance shall preserve unrelated source and assign accepted element identity.	Must	Open	∅	UC-009	TC-080 VER-089
REQ-F-035	The editor shall support click placement using provider defaults and rectangle drag placement using opposite corners; clicked rectangles shall be centred where valid and shifted inward without changing default size to fit a declared editing extent.	Must	Open	∅	UC-009	TC-080 VER-106

5.2.3 Data Requirements

Table 5.3: Provisional Data Requirements

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-D-001	The application shall preserve source outside the explicitly edited supported construct during graphical editing and saving.	Must	Open	∅	UC-002, UC-003, UC-009	TC-004, TC-007, TC-021, TC-022, TC-023, TC-024, TC-025, TC-080 VER-027
REQ-D-002	The application shall prepare candidate source and history before publishing them as one atomic change; a rejected or failed edit or history restoration shall leave authoritative source and history unchanged.	Must	Open	∅	UC-002, UC-004	TC-034, TC-042 VER-028
REQ-D-003	Each accepted new edit that changes source shall create exactly one history entry; previews, cancellation, rejected edits and empty or byte-equivalent proposals shall leave source and history unchanged.	Must	Open	∅	UC-002, UC-004	TC-034, TC-036 VER-029
REQ-D-004	Each accepted changed edit or history restoration shall advance the document revision; Undo and Redo shall never restore an earlier revision number.	Must	Open	∅	UC-002, UC-004	TC-035 VER-030
REQ-D-005	A successful Save shall set the saved-source baseline to the bytes written; a failed Save shall retain the previous baseline.	Must	Open	∅	UC-003, UC-004	TC-038 VER-031
REQ-D-006	The document shall be marked unsaved exactly when its current source bytes differ from its saved-source baseline, including after Undo or Redo.	Must	Open	∅	UC-003, UC-004	TC-038 VER-032
REQ-D-007	Undo and Redo shall move the existing history position without appending a new edit entry.	Must	Open	∅	UC-004	TC-035, TC-037, TC-042 VER-033

Continued on next page

Provisional Data Requirements (continued)

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-D-008	A transaction proposal shall identify its document, base revision and operation description, and give each edit's original byte range, expected bytes and replacement bytes. All ranges shall use the same pre-edit source; zero-length ranges insert and empty replacements delete.	Must	Open	∅	UC-002, UC-004	TC-034, TC-040 VER-034
REQ-D-009	The element representation shall separate a revision-scoped ElementDescription (identity, shape family/type, schema version, interpreted geometry, appearance and relevant content) from SourceBinding, presentation/selection data, attachments and per-element available operations.	Must	Open	∅	UC-002, UC-005, UC-006	TC-048 VER-035
REQ-D-010	Source bindings, provider bindings and attachment references shall identify their document and revision; accepted edits shall invalidate old records and rebuild relationships from accepted source.	Must	Open	∅	UC-002, UC-005	TC-049 VER-036
REQ-D-011	Transaction proposals and exact before/after history snapshots shall be held in memory for the open document; saving shall not embed history in the TikZ file or restore it after application restart. The initial prototype shall retain session entries without automatic eviction; a future limit requires review informed by memory measurements.	Must	Open	∅	UC-003, UC-004	TC-050 VER-037
REQ-D-012	Each selectable element shall provide selection geometry and a readable description independent of source spelling and graphics-item pointers; presentation coordinates shall be distinct from viewport pixels.	Must	Open	∅	UC-005, UC-006	TC-048, TC-053 VER-038
REQ-D-013	Operation requests shall identify host-defined common operations such as `tkr.move` and `tkr.resize`, or provider-qualified custom operations; parameters and their schema version shall be separate fields.	Must	Open	∅	UC-002	TC-072 VER-090
REQ-D-014	An existing-element edit request shall contain target document/element identity and revision, the initial element description, operation-specific parameters and host constraints in document centimetres, without mouse events, widget pointers or source byte ranges.	Must	Open	∅	UC-002	TC-072 VER-091

Continued on next page

© Stevens Institute of Technology

Provisional Data Requirements (continued)

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-D-015	Shape preparation shall return Prepared change with proposed interpreted changes, Unchanged, or Cannot prepare with a reason; it shall not claim that source or history has been applied.	Must	Open	∅	UC-002	TC-073 VER-092
REQ-D-016	Declared stable plugin/provider identity shall be distinct from the host-assigned runtime instance and loaded-version identity used by element bindings and outstanding operations.	Must	Open	∅	UC-007, UC-008	TC-068, TC-070 VER-093
REQ-D-017	A replacement provider shall accept existing element data or use an explicitly defined validated conversion preserving relevant information and used capabilities; otherwise the previous provider shall remain available and no implicit lossy conversion shall occur.	Must	Open	∅	UC-007, UC-008	TC-071 VER-094
REQ-D-018	Catalogue entries shall describe offered shape types, readable names, preview data and creation support separately from document elements; the host shall associate provider identity and version.	Must	Open	∅	UC-009	TC-079 VER-105

5.2.4 Interface Requirements

This register addresses what an author sees during graphical editing. The canvas may rasterize its final screen pixels, but zooming a supported geometric item should redraw its shape rather than enlarge a fixed-resolution picture.

Table 5.4: Provisional Interface Requirements

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-I-001	At supported zoom levels, the editor shall redraw supported lines, arrows, and rectangular boxes from geometric data without enlarging a fixed-resolution canvas image.	Should	Open	∅	UC-002	TC-026 VER-039
REQ-I-002	The interface shall disable Undo or Redo when the corresponding history entry is unavailable.	Must	Open	∅	UC-004	TC-043 VER-040
REQ-I-003	The application shall emit exactly one successful document-change notification per accepted changed edit or restoration, and none for previews, cancellation, rejected or failed operations, or no-op results.	Must	Open	∅	UC-002, UC-004	TC-034, TC-036, TC-042, TC-043 VER-041
REQ-I-004	After an accepted edit or history restoration, the application shall reparse the accepted source and refresh the canvas from that state.	Must	Open	∅	UC-002, UC-004	TC-043 VER-042
REQ-I-005	If history restoration fails, the interface shall report the failure to the author.	Must	Open	∅	UC-004	TC-043 VER-043
REQ-I-006	Editing results shall distinguish Applied, Unchanged and Rejected; a Rejected result shall contain a user-readable reason, while cancellation and unchanged actions shall not be reported as editing errors.	Must	Open	∅	UC-002, UC-004, UC-005, UC-009	TC-051, TC-080 VER-044
REQ-I-007	A rejected completed edit shall show one nonmodal bottom message strip with a bright red icon, red accent and readable reason; it shall remain until dismissed or replaced by the next completed operation, with no repeated messages for pointer-movement previews.	Must	Open	∅	UC-002, UC-005, UC-009	TC-052, TC-080 VER-045

Continued on next page

Provisional Interface Requirements (continued)

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-I-008	Normal hit selection shall prioritize a label over its overlapping host; a direct right-click menu shall list all intersecting selectable elements, including obscured hosts, using readable descriptions and candidate highlighting.	Must	Open	∅	UC-006	TC-053 VER-046
REQ-I-009	A left drag on empty canvas, or anywhere with the Select area tool active, shall draw a retained highlighted rectangle; right-clicking inside it shall list elements whose selection geometry intersects it. Choosing an entry shall select that element; drawing the region shall not move objects or automatically select the group, and Escape shall clear it.	Must	Open	∅	UC-006	TC-054 VER-047
REQ-I-010	Direct right-click selection shall initially search a circle of radius 6 logical screen pixels, mapped to document space so its screen size stays constant under zoom; this provisional setting shall be reviewed with user feedback.	Must	Open	∅	UC-006	TC-055 VER-048
REQ-I-011	The editor shall visually distinguish the Canvas pane from the source pane and keep supported diagram outlines legible in light and dark themes.	Must	Open	∅	UC-002	TC-062 VER-075
REQ-I-012	By default, the editor shall fit the full canvas to the available pane on opening a diagram and resizing the pane, preserving its aspect ratio and source coordinates.	Must	Open	∅	UC-002	TC-063 VER-076
REQ-I-013	When a document declares a fixed editing extent, the editor shall distinguish it visually and constrain move, resize and creation preview and acceptance to valid geometry inside it, including stroke; reaching a limit shall not be reported as rejection.	Must	Open	∅	UC-002, UC-009	TC-064, TC-080 VER-077
REQ-I-014	During a drag, the editor shall redraw the canvas without leaving visible marks at previous preview positions.	Must	Open	∅	UC-002	TC-065 VER-078
REQ-I-015	The Shapes page shall show provider-distinguishable previews and names within the collapsible workspace sidebar, and expose creation only when both the shape and syntax-insertion route support it.	Must	Open	∅	UC-009	TC-079 VER-100

Continued on next page

Provisional Interface Requirements (continued)

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-I-016	An update notification for an in-use provider shall be nonmodal and offer Later or Restart and update; restart shall respect Save, Discard and Cancel without interrupting an active edit by replacing its provider.	Must	Open	∅	UC-008	TC-077 VER-101
REQ-I-017	When compiler/PDF integration exists, a diagram missing graphical provider support shall be viewable through a successful read-only compilation when dependencies are available; the displayed result shall identify its source revision and any stale status.	Must	Open	∅	UC-001	TC-081 VER-102
REQ-I-018	The workspace sidebar shall collapse to a visible tab strip through which its pages can be reopened.	Must	Open	∅	UC-001, UC-009	TC-079 VER-107
REQ-I-019	Shapes and Files tabs shall have distinct icons, readable identification and clear active-state feedback.	Must	Open	∅	UC-001, UC-009	TC-079 VER-108
REQ-I-020	The sidebar shall retain its selected page and previous expanded width during the window session.	Must	Open	∅	UC-001, UC-009	TC-079 VER-109
REQ-I-021	After accepted creation, the editor shall select the new element and return to selection; Escape shall cancel, while rejected placement shall retain the creation tool for retry and report its reason once per attempt.	Must	Open	∅	UC-009	TC-080 VER-110

5.2.5 Security Requirements

Table 5.5: Provisional Security Requirements

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-S-001	The application shall invoke the configured compiler without enabling TeX shell escape by default.	Should	Open	∅	UC-001	TC-009 VER-049

© Stevens Institute of Technology

5.2.6 Architectural Requirements

These candidates specify component responsibilities and runtime process boundaries for a readable and testable editor. Vincent selected C++ with Qt 6 and agreed the initial class boundaries, naming conventions, shallow inheritance, and ownership relationships. The selected design is recorded in Section 3.3.1 and implemented as an initial scaffold under `code/src/`. These design decisions do not establish completed product requirements. The native build and foundation checks passed on October 20 (Section 10.5); initial subset display checks followed on October 21 (Section 10.7). Whole-element interaction has bounded evidence in Section 10.8; endpoint editing, broader parsing and LaTeX/PDF integration remain unfinished.

Table 5.6: Provisional Architectural Requirements

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-A-001	The application shall run preview compilation outside the UI thread so the editor remains responsive while the compiler runs.	Must	Open	∅	UC-001, UC-002	TC-014 VER-050
REQ-A-002	The application shall generate preview files in a separate temporary workspace without placing compilation intermediates in the opened document's directory.	Should	Open	∅	UC-001	TC-015 VER-051
REQ-A-003	The application shall separate compilation control and diagnostics from PDF loading and display into distinct components.	Should	Open	∅	UC-001	TC-016 VER-052
REQ-A-004	The editor shall use object-oriented components with documented, distinct responsibilities for graphical interaction, TikZ source interpretation and editing, preview compilation, and PDF display.	Should	Open	∅	UC-001, UC-002	TC-018 VER-053
REQ-A-006	TikZ source interpretation and edit logic shall be callable and testable without launching the graphical interface or a LaTeX compiler.	Must	Open	∅	UC-002	TC-020 VER-054
REQ-A-007	The application shall use a documented, consistent mapping between supported TikZ coordinates and canvas scene coordinates for both display and source edits.	Must	Open	∅	UC-002	TC-028, TC-029 VER-055
REQ-A-008	Built-in and plugin editing operations shall submit revision-guarded source transactions through one common validation/application mechanism that supports insertion, deletion and replacement without requiring a GUI or LaTeX run.	Must	Open	∅	UC-002, UC-004	TC-034, TC-040, TC-042 VER-056

Continued on next page

Provisional Architectural Requirements (continued)

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-A-009	The plugin API shall distinguish element support for additional object kinds from syntax support for additional source forms; a provider shall be able to implement either or both through the common editing contract.	Must	Open	∅	UC-001, UC-002	TC-041 VER-057
REQ-A-010	Preview compilation shall verify an accepted source revision subsequently; a compilation failure shall report diagnostics without automatically reversing that transaction or its history entry.	Must	Open	∅	UC-002, UC-004	TC-039 VER-058
REQ-A-011	Before publication, the core shall validate document identity, base revision, range bounds, expected bytes, range conflicts and supported-operation rules. Ambiguous same-position insertions shall be combined unambiguously by the planner or rejected.	Must	Open	∅	UC-002, UC-004	TC-034, TC-040, TC-042 VER-059
REQ-A-012	The document shall control transaction application and own history; it shall call history methods to record accepted changes and retrieve restoration targets, coordinating source and history-position updates as one atomic operation.	Must	Open	∅	UC-002, UC-004, UC-005	TC-056 VER-061
REQ-A-013	The controller shall coordinate requests, results and interface notifications, owning the edit service and file store; the edit service shall prepare supported proposals from a read-only snapshot without mutating live source, history, saved state or graphics items.	Must	Open	∅	UC-002, UC-003, UC-004, UC-005	TC-056 VER-062
REQ-A-014	The file store shall read bytes and perform guarded writes without managing undo history; after successful I/O the controller shall request document load or saved-baseline updates using the bytes actually read or written. Failed or cancelled operations shall preserve state subject to any prior explicit successful Save.	Must	Open	∅	UC-001, UC-003, UC-004	TC-057 VER-063

Continued on next page

© Stevens Institute of Technology

Provisional Architectural Requirements (continued)

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-A-015	Syntax providers shall interpret source and prepare bound rewrites. Shape providers shall consume shared descriptions and return presentation, selection and interpreted operation proposals, without directly mutating document, history, scene or files; the scene shall create and own graphics items.	Must	Open	∅	UC-001, UC-002, UC-005, UC-006	TC-058 VER-064
REQ-A-016	Established element and source bindings shall identify their responsible provider instance/version; subsequent edit preparation shall route to those providers without reselecting an interpreter or replacing a provider during the gesture.	Must	Open	∅	UC-002, UC-005	TC-059 VER-065
REQ-A-017	An element operation shall be available only when the shape provider, syntax/source binding and current document context support it; the host shall check both when enabling the interaction and when processing its request, and explain unavailable operations.	Must	Open	∅	UC-002, UC-005	TC-058 VER-066
REQ-A-018	The editor shall support the prototype rectangle through the common shape API from a separately loadable plugin, preserving its supported display, move and resize behavior.	Must	Open	∅	UC-002	TC-061 VER-074
REQ-A-019	Provider registration shall declare stable ID, display name, release version, element/syntax role, supported types and schemas, API compatibility and supported operations; incompatible registrations shall remain inactive with a reason.	Must	Open	∅	UC-007	TC-068, TC-071 VER-095
REQ-A-020	Provider compatibility shall check the API contract, element-data schema and required capabilities; the first rectangle extraction shall require the explicitly supported API version and matching documented build environment.	Must	Open	∅	UC-007, UC-008	TC-071 VER-096
REQ-A-021	A provider shall remain available while any open document, active gesture or pending operation depends on it; activation or unloading shall recheck those dependencies at a safe point.	Must	Open	∅	UC-002, UC-008	TC-076, TC-077 VER-097

Continued on next page

Provisional Architectural Requirements (continued)

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-A-022	Plugin updates shall be associated with an explicit update origin and installed and validated separately from the active version before activation; detailed distribution, authentication and recovery mechanisms require later design.	Must	Open	∅	UC-008	TC-078 VER-098
REQ-A-023	Preview and release shall calculate shape proposals from initial gesture geometry and total document-space displacement under the same constraints, rather than accumulate changes from earlier previews.	Must	Open	∅	UC-002	TC-060, TC-072 VER-099
REQ-A-024	The main window shall own shared plugin services and the catalogue; consumers shall borrow access and be released before the services on which they depend.	Must	Open	∅	UC-002, UC-009	TC-082 VER-111

5.2.7 Developer Requirements

Developer requirements govern how contributors organize and name source code so that people can understand and maintain it. This is a project-specific category adapted from the template's provision for additional registers; it uses the same fields and traceability rules. Architectural requirements retain component boundaries, compilation separation, and runtime coordinate mapping. The source-module rule moves here with its stable ID REQ-A-005; the historical A prefix is retained to avoid breaking existing references. New entries use DEV, distinct from D for data.

Vincent reports that the instructor suggested matching the documentation's `tkr` prefix and using a main class name followed by a specialization. Vincent confirmed `tkr` and clarified the permitted inheritance depth on October 15, 2026. The naming and depth rules below record that discussion through his account, not as a direct instructor quotation.

Table 5.7: Provisional Developer Requirements

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-A-005	Each primary application class shall be defined in its own source module; small helper types may share the module when they support that class's responsibility.	Should	Open	∅	*Cross-cutting*	TC-019 VER-060
REQ-DEV-001	Project-owned class names shall follow `tkr<MainClassName><Specification>`, with PascalCase name components and an optional suffix identifying a specialization; matching header and implementation filenames shall use the complete class name.	Must	Open	∅	*Cross-cutting*	TC-030 VER-067
REQ-DEV-002	Application identifiers shall follow the documented coding convention: camelCase methods, descriptive type/category-prefixed variables, and an additional `m_` marker for private member variables.	Should	Open	∅	*Cross-cutting*	TC-031 VER-068
REQ-DEV-003	Any inheritance path shall contain at most three project-defined class levels: a main/base class, its direct subclass, and one further subclass; third-party Qt ancestry shall be excluded from this count.	Must	Open	∅	*Cross-cutting*	TC-032 VER-069
REQ-DEV-004	Each class relationship shall document whether it owns or borrows the related object and the lifetime mechanism that enforces that relationship.	Must	Open	∅	*Cross-cutting*	TC-033 VER-070
REQ-DEV-005	Project-owned application logic shall use meaningfully named constants, enum values, or members for numeric values. Numeric literals shall appear in named definitions rather than directly in executable logic.	Must	Open	∅	*Cross-cutting*	TC-066 VER-079

Continued on next page

Provisional Developer Requirements (continued)

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-DEV-006	The project shall use `0.feature.revision-alpha`: completed document/code revisions advance revision, and a completed feature release advances feature and resets revision. A build shall identify the exact architectural baseline it implements even if the report subsequently advances.	Must	**Open**	∅	*Cross-cutting*	TC-067 VER-080
REQ-DEV-007	Normal application titles and package-folder labels shall show the implemented baseline as `0.feature.revision-alpha` without a commit hash; later report revisions shall not relabel an existing package.	Must	**Open**	∅	*Cross-cutting*	TC-067 VER-103
REQ-DEV-008	About shall show the implemented baseline followed by a Git-generated shortened checkpoint, retain the full checkpoint in details and build metadata, and identify clean, modified or unavailable source state truthfully.	Must	**Open**	∅	*Cross-cutting*	TC-067 VER-104

Verification uses source/design review and build checks; they need no artificial user-story or use-case mapping. Prefer composition and explicit borrowed references where they express the actual relationship. The depth rule is a ceiling, not a target: concrete host/module classes remain `final`; the provider implementation realizes its interface at two levels.

© Stevens Institute of Technology

5.2.8 Performance Requirements

The interaction rule below avoids coupling each pointer movement to a full LaTeX run. A measurable response-time threshold awaits representative samples and stakeholder review.

Table 5.8: Provisional Performance Requirements

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-P-001	During a supported drag the application shall show geometric feedback without waiting for LaTeX. After a successful Save, integrated preview shall compile the saved revision in the background rather than launch compilation for pointer movement, pauses or release.	Must	Open	∅	UC-002	TC-017 VER-071

5.2.9 Constraint Requirements

A constraint requirement limits an acceptable solution or operating environment rather than describing a user action. The template gives mandated technologies, dependencies, and solution boundaries as examples. The current explicit operating constraint is REQ-C-001: the supported core workflow must work without a network service.

The configured local LaTeX toolchain is already required by REQ-F-002, while REQ-F-003 and REQ-F-005 limit graphical editing to a documented TikZ subset and require feedback for unsupported constructs. These are recorded once under the behavior they constrain rather than duplicated as C requirements. The single-developer schedule is a project constraint in Chapter 3, not a product behavior to test. Other product constraints will be added only when their scope and acceptance checks are clear enough for review.

Table 5.9: Provisional Constraint Requirements

ID	Candidate requirement	Priority	Status	In Version	Use case(s)	Test / verification
REQ-C-001	The core open, preview, graphical-edit, and save workflow for a supported local document shall function without access to a network service.	Must	Open	∅	UC-001, UC-002, UC-003	TC-010 VER-072

© Stevens Institute of Technology

Version convention. As in the course template, **In Version** records the first controlled implementation version that contains a completed requirement. All current entries are Open and therefore show ∅. Test-case links point to proposed procedures in Chapter 10, not completed evidence.

5.3 Class Naming Interpretation

The literal lowercase prefix is `tkr`. The main name describes the class's role; an optional PascalCase specification identifies a concrete specialization. For example, `tkrMainWindow` has no specification suffix, and `tkrTikzParserExplicit` identifies the explicit-coordinate parser. Third-party classes such as `QObject` retain their vendor names; shared structs and enums retain the coding guide's PascalCase convention.

A future hypothetical pair `tkrDiagramItem` and `tkrDiagramItemRectangle` illustrates the main-name/suffix convention. Those classes do not exist in this scaffold. A suffix describes specialization; it does not by itself establish C++ inheritance. The current parser has no project-defined base class, and this naming change introduces none.

Where inheritance is justified, the maximum is main/base, child, and grandchild: two levels after the main class. A fourth project-defined level is outside the agreed limit. New levels require an actual substitutable specialization, rather than being added to reach the maximum. Current code, build lists, and diagrams use the prefixed names. Native compilation passed on October 20; the complete proposed developer reviews remain unverified.

5.4 Initial Explicit-Coordinate Subset

Vincent selected explicit coordinates and individual drawing commands for the initial feasibility tests on October 1, 2026. The boundary below makes that choice reproducible. It specifies initial test inputs; subset parsing and display have limited evidence in Section 10.7, while product verification remains pending. Broader coverage of equivalent TikZ syntax remains a product goal, and this subset is not a required generation format for all future input diagrams.

Table 5.10: Initial Editable Syntax Boundary

Feature	Initial test boundary
Coordinates	Two-dimensional absolute Cartesian pairs; each component is an integer or decimal literal with an explicit `cm` unit. Negative and zero values are included. No expressions or variable references.
Straight segments	One `\draw` command containing two coordinates joined by `-`; a body drag translates both coordinates, while an endpoint drag changes only the selected coordinate.
Arrows	The same straight segment with `->`, `<-`, or `<->`; preserve the arrow option and direction during editing.

Feature	Initial test boundary
Rectangular boxes	One `\draw` command using `rectangle` between two opposite corners. Translate both corners equally; resizing is outside the first workflow.
Appearance and layout	Solid black outlines with default stroke width or the explicit option `line width=0.6pt`; no drawing transformations, custom styles, fills, or text. Each movable element has its own command.
Source and wrapper	Preserve comments, whitespace, command order, and unrelated content. Reference files use a `tikzpicture` environment and a fixed bounding-box declaration for comparison; that declaration is calibration data, not a movable element.

Named coordinates, nodes and anchors, relative or polar coordinates, transformations, calculated expressions, custom macros, and loops are later coverage candidates. They may produce the same visible geometry but need additional source mapping and edit semantics [12]. Until an interpretation is supported, REQ-F-005 governs feedback and source preservation. Successful LaTeX rendering alone does not establish that a construct can be edited safely. Vincent reports that Professor Muresan has diagram examples that can inform later syntax coverage; these examples have not yet been supplied for this test corpus.

5.5 Evidence and Review Boundaries

The graphical-editor concept motivates the workflow. Vincent's recollection motivates candidate selection and movement behavior; the professor's Qt recommendation does not itself approve these requirements. Qt Graphics View documents scene items and pointer interaction [7]; QProcess documents launching external programs [9]. These sources inform feasibility, not stakeholder acceptance. External compiler invocation must be reviewed against REQ-S-001. The LaTeX Project documents the underlying authoring ecosystem [13]. The object-oriented and class-organization candidates reflect Vincent's readability preference and design discussion, not an instructor-approved coding mandate. The offline operating constraint is a proposal derived from the local-desktop direction and private-document driver in Chapter 2; it also awaits stakeholder confirmation.

Before baseline, review supported constructs and manipulation gestures, preview timing, error recovery, dependency handling, save behavior, security, priority, and observable test criteria with Vincent and Professor Muresan. Acceptance examples are proposed checks, not test results. TikZ's documented coordinates and paths provide a basis for selecting the first subset [12]. The initial test syntax is specified in Section 5.4. Supported zoom range, comparison tolerances, quantitative targets, and later syntax coverage remain open. Additional quality and constraint requirements can be added when evidence supports them.

Chapter 6

Use Cases

The following use cases translate the provisional stories and requirements into an initial workflow. Full product acceptance remains pending; bounded whole-element editing evidence is recorded in Section 10.8. The October review added planned label and overlap-selection goals. Rectangle provider API v1 is implemented; wider provider choice, updates and creation are planned. The transaction/history behavior and architecture were agreed with Vincent beginning October 22, 2026. Bounded session history is implemented for coordinate edits (Section 10.16); generic transactions and integrated compilation remain future. Broader design diagrams are in Section 7.8 and Figure D.8. The selection-and-move gesture in `UC-002` reflects Vincent's recollection and awaits stakeholder confirmation.

The course pattern of stable IDs, a boundary overview, and one specification table per use case is retained. Following Milestone 1 feedback reported by Vincent, it is adapted with a focused TikZ diagram for each goal and reciprocal links between diagrams and specifications. These are proposed actor-goal views, distinct from the source-based activity diagrams in Appendix D.

6.1 High-Level Use-Case Diagram

The nine goals share one primary actor, the TikZ author. A compact boundary diagram gives an overview of the workflow; the tables below define each interaction and its alternatives. Select a goal in the diagram to jump to its specification table. Each table also links to this overview and its own focused diagram. Provider choice and update handling are planned goals; creation is a later workflow. The actor remains the TikZ author. These additions do not claim an installer, updater, palette or creation implementation.

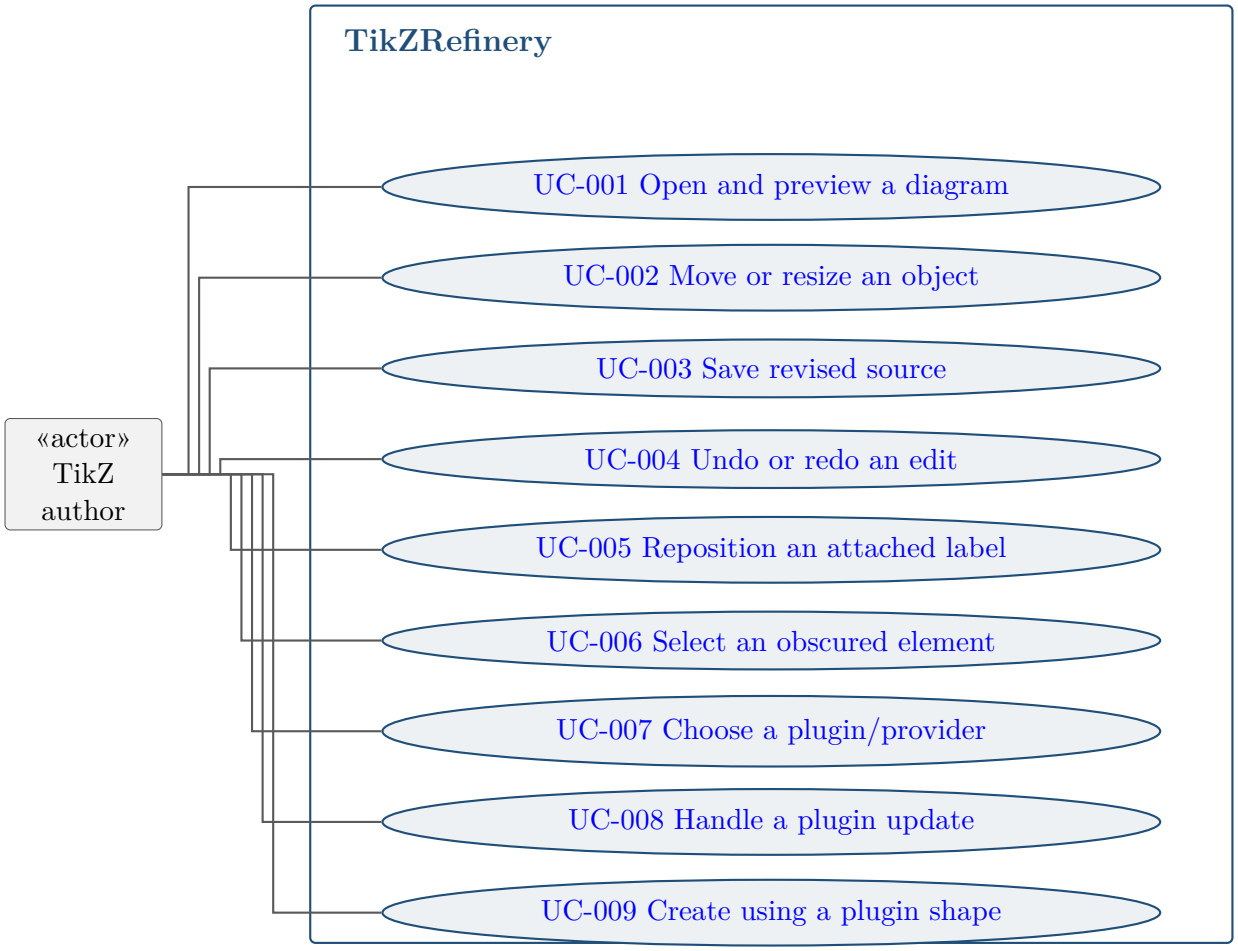


Figure 6.1: Provisional actor and use-case overview.

6.2 Open and Preview a Diagram

Table 6.1: Proposed UC-001 Specification

Identifier	UC-001
Diagrams	Overview (Figure 6.1); focused diagram (Figure 6.2)
Mapped stories	US-001, US-002, US-004, US-005, US-006, US-007, US-008, US-009, US-010, US-011
Mapped requirements	REQ-F-001, REQ-F-002, REQ-F-005, REQ-F-007, REQ-F-008, REQ-F-017, REQ-F-018, REQ-F-020, REQ-F-032, REQ-I-017, REQ-I-018, REQ-I-019, REQ-I-020, REQ-S-001, REQ-A-001, REQ-A-002, REQ-A-003, REQ-A-004, REQ-A-009, REQ-A-014, REQ-A-015, REQ-C-001
Goal	Open a local TikZ document and see its rendered diagram.
Primary actor	TikZ author
Preconditions	A readable local source file and a configured local LaTeX toolchain are available.
Trigger	The author chooses a source file through Open or the planned Files page. Folder selection and activation details remain open.
Main flow	Every opening entry point passes through the same Save, Discard and Cancel guard before replacing the current document. The application reads the file, prepares its supported dependencies in a temporary workspace, invokes the configured compiler outside the UI thread, and displays the generated preview.
Alternate flow	If compilation fails, the application presents diagnostics and retains the source for correction. Any last successful preview remains visible and is marked as a prior successful result. Missing or incompatible providers preserve source and identify unavailable support without substitution. A read-only compiled fallback is conditional on future compiler integration and available LaTeX dependencies.
Postconditions	A preview or explicit failure state is displayed; the input file is unchanged.

6.2.1 UC-001 Diagram

The association connects the TikZ author to the goal. Dotted connectors attach explanatory notes; they are not control-flow or `include`/`extend` relationships. The goal and caption link back to the specification.

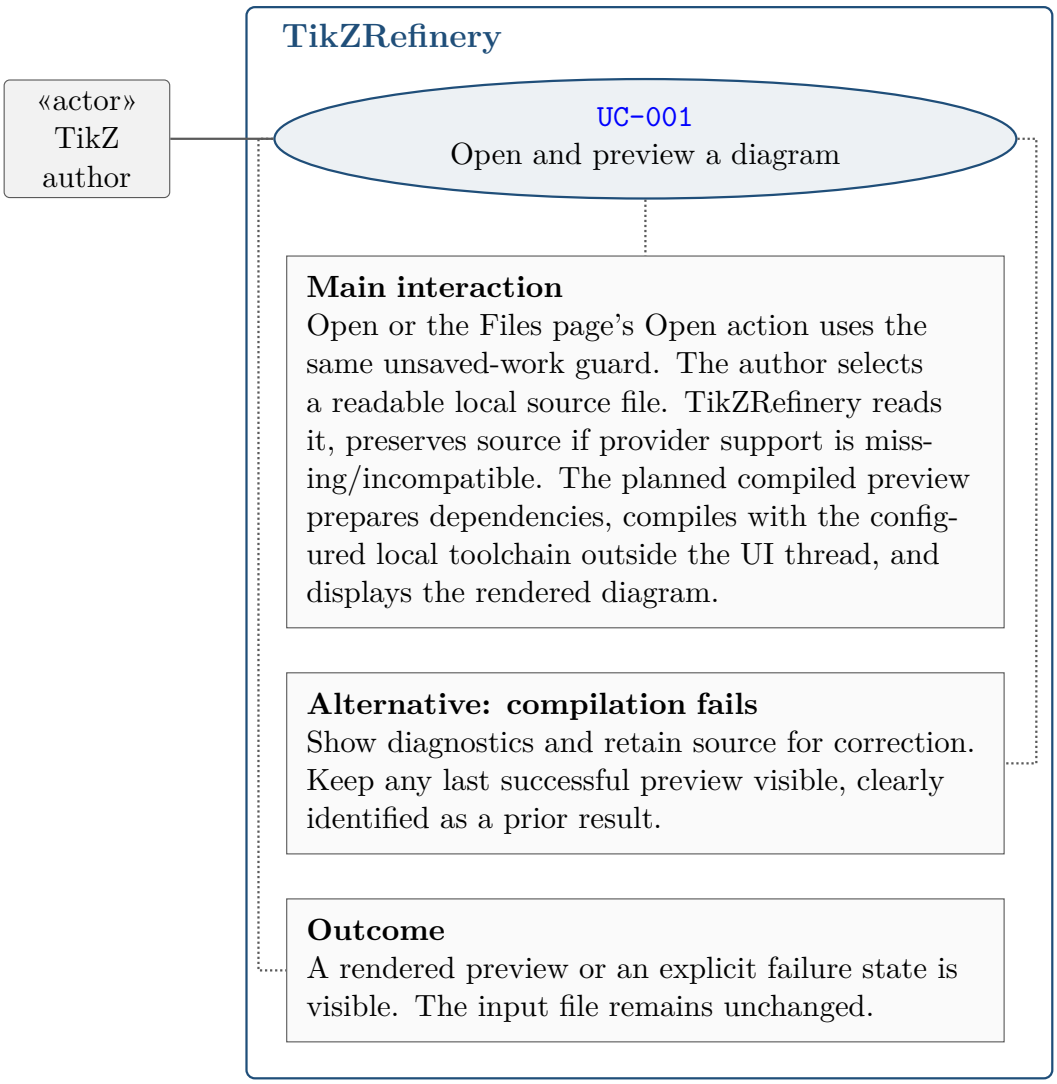


Figure 6.2: Proposed actor-goal view for UC-001: opening and previewing. See the overview for all nine goals.

6.3 Move or Resize a Supported Object

Table 6.2: Proposed UC-002 Specification

Identifier	UC-002
Diagrams	Overview (Figure 6.1); focused diagram (Figure 6.3); prototype drag sequence
Mapped stories	US-001, US-002, US-003, US-004, US-005, US-006, US-007, US-008, US-009, US-010, US-011
Mapped requirements	REQ-B-001, REQ-F-003, REQ-F-004, REQ-F-005, REQ-F-007, REQ-F-009, REQ-F-010, REQ-F-011, REQ-F-015, REQ-F-022, REQ-F-025, REQ-F-027, REQ-F-032, REQ-D-001, REQ-D-002, REQ-D-003, REQ-D-004, REQ-D-008, REQ-D-009, REQ-D-010, REQ-D-013, REQ-D-014, REQ-D-015, REQ-I-001, REQ-I-003, REQ-I-004, REQ-I-006, REQ-I-007, REQ-I-011, REQ-I-012, REQ-I-013, REQ-I-014, REQ-A-001, REQ-A-004, REQ-A-006, REQ-A-007, REQ-A-008, REQ-A-009, REQ-A-010, REQ-A-011, REQ-A-012, REQ-A-013, REQ-A-015, REQ-A-016, REQ-A-017, REQ-A-018, REQ-A-021, REQ-A-023, REQ-A-024, REQ-P-001, REQ-C-001
Goal	Adjust a supported object's position or rectangle dimensions through the graphical view while retaining editable source.
Primary actor	TikZ author
Preconditions	A document is open and the target object belongs to the documented editable TikZ subset.
Trigger	The author selects a supported object directly or through UC-006, then drags its body or an available edit handle.
Main flow	The editor maps the object to source, previews the gesture, then updates only the needed coordinates. A body drag translates the whole object. An endpoint drag moves one endpoint. A box handle changes its dimensions. Attached labels follow their host. Independent label movement follows UC-005. The fitted canvas shows any fixed edge and repaints cleanly. Zoom preserves coordinates. The host routes preparation to the bound providers and rechecks available operations before acceptance. Future background compilation follows successful Save, not drag pauses or release.
Alternate flow	Preview and release use the same constrained result: stop at a fixed edge, retain the minimum rectangle size, and allow corner crossing (REQ-I-013). Ambiguous mapping blocks the edit with a reason. A later compilation failure reports diagnostics.
Postconditions	The canvas follows accepted source. Rejection preserves source. A later compilation failure preserves accepted source and history and marks the preview status.

6.3.1 UC-002 Diagram

The notes summarize the existing specification's supported gestures and unmappable-source alternative. They annotate one goal rather than introduce additional use cases. This view describes the full proposed behavior; bounded whole-element editing is implemented; this increment adds corner resizing. Endpoint gestures and integrated preview remain pending.

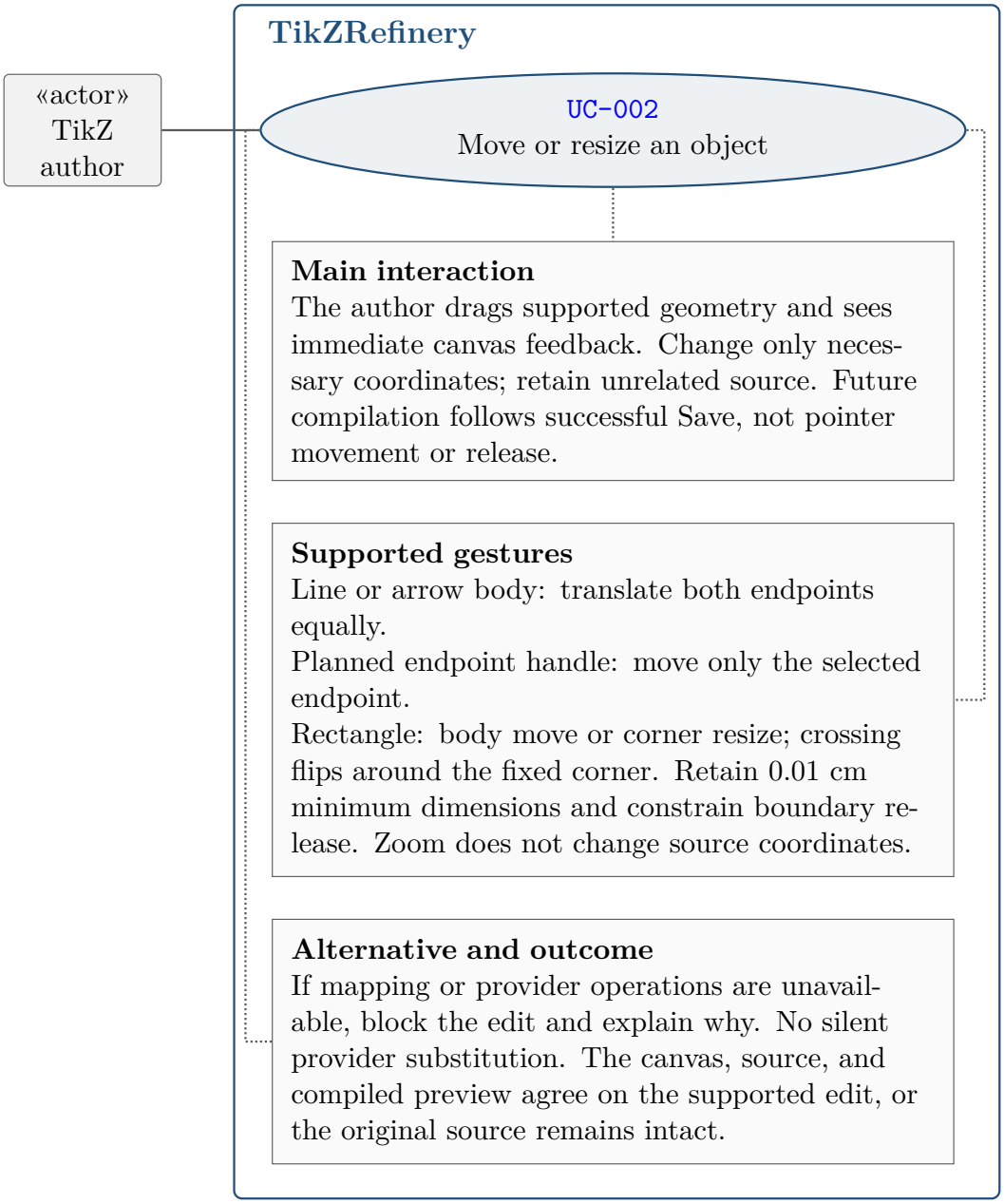


Figure 6.3: Proposed actor-goal view for UC-002: editing supported geometry while preserving source. See the overview for all nine goals.

6.4 Save Revised Source

Table 6.3: Proposed UC-003 Specification

Identifier	UC-003
Diagrams	Overview (Figure 6.1); focused diagram (Figure 6.4)
Mapped stories	US-003, US-005, US-006, US-007, US-009
Mapped requirements	REQ-B-002, REQ-F-006, REQ-F-016, REQ-F-018, REQ-D-001, REQ-D-005, REQ-D-006, REQ-D-011, REQ-A-013, REQ-A-014, REQ-C-001
Goal	Retain an approved graphical edit as local TikZ source.
Primary actor	TikZ author
Preconditions	A revised source document is open for review.
Trigger	The author chooses to save the revision.
Main flow	The application writes the reviewed source locally and reports that the save succeeded. When integrated compilation exists, queue compilation of those saved bytes; show readiness/failure without undoing the save.
Alternate flow	If the write fails, the application reports the failure and keeps the unsaved revision available.
Postconditions	A successful save can be reopened and compiled with the local toolchain; after failure, the unsaved revision remains available.

6.4.1 UC-003 Diagram

The notes summarize the save goal and its failure outcome. Reopening and external compilation describe the successful source file's intended usability. Compilation is manual today; planned background integration follows successful Save under REQ-P-001 and remains outside the current save path.

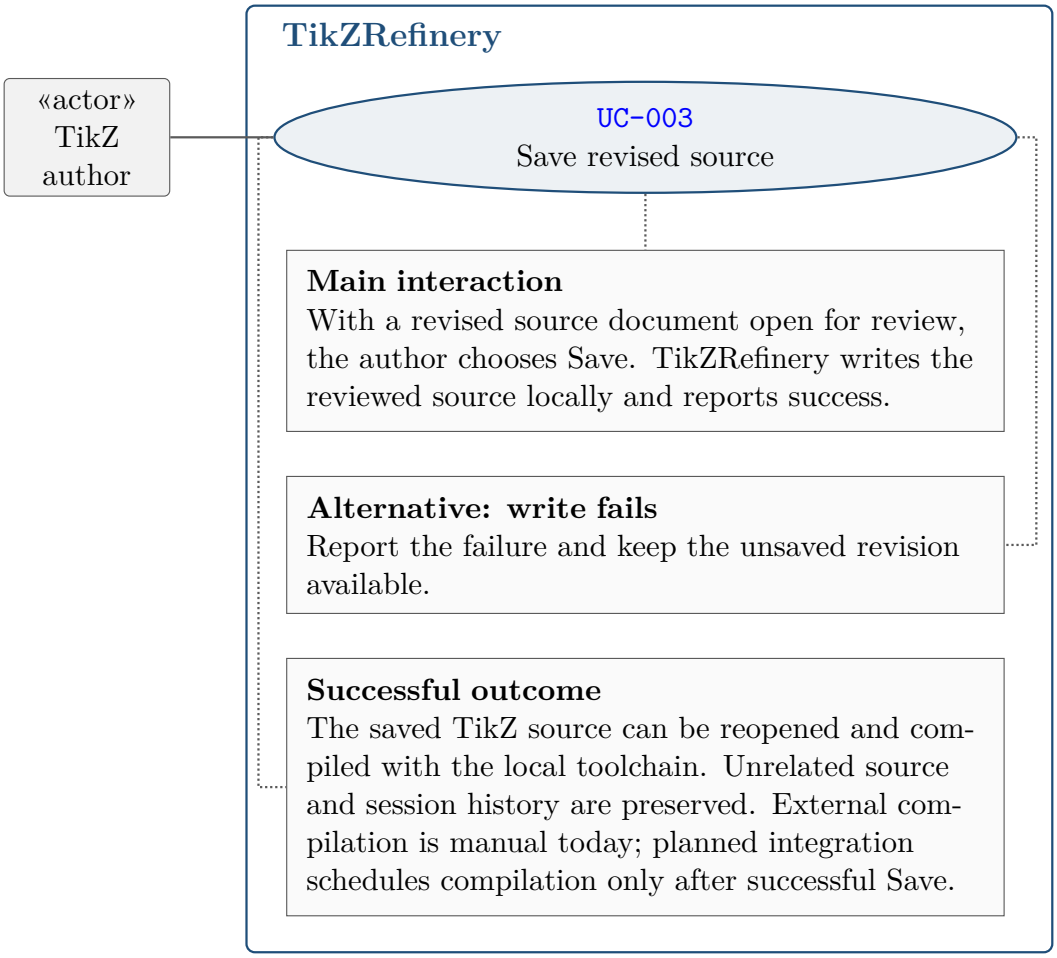


Figure 6.4: Proposed actor-goal view for UC-003: saving revised source. See the overview for all nine goals.

6.5 Undo or Redo an Edit

Table 6.4: Proposed UC-004 Specification

Identifier	UC-004
Diagrams	Overview (Figure 6.1); focused diagram (Figure 6.5); current restoration sequence
Mapped stories	US-004, US-005, US-007, US-008, US-009, US-011
Mapped requirements	REQ-F-007, REQ-F-012, REQ-F-013, REQ-F-014, REQ-F-015, REQ-F-016, REQ-F-017, REQ-F-018, REQ-F-019, REQ-F-020, REQ-D-002, REQ-D-003, REQ-D-004, REQ-D-005, REQ-D-006, REQ-D-007, REQ-D-008, REQ-D-011, REQ-I-002, REQ-I-003, REQ-I-004, REQ-I-005, REQ-I-006, REQ-A-008, REQ-A-010, REQ-A-011, REQ-A-012, REQ-A-013, REQ-A-014
Goal	Restore an earlier or later completed source edit.
Primary actor	TikZ author
Preconditions	A document is open and an undo or redo entry is available.
Trigger	The author invokes Undo or Redo.
Main flow	Cancel any active drag preview. Prepare the exact source state recorded by the selected entry, reparse and validate its display before acceptance. The document publishes the restored source/records and history position together, advances its revision and notifies the interface once. Refresh dirty state and controls. Future compiler integration marks any compiled view stale until successful Save requests compilation.
Alternate flow	If history is unavailable, disable the action. If restoration cannot complete, preserve both source and history position and report the failure. A later compilation failure does not reverse the accepted restoration.
Postconditions	Source reflects the selected history state, disk remains unchanged until Save, and the interface indicates whether source differs from the saved baseline.

6.5.1 UC-004 Diagram

This actor-goal view records the agreed behavior; document-owned history and restoration are shown in Figures 7.16 and D.8. Bounded implementation evidence appears in Section 10.16; future compiler behavior is not claimed.

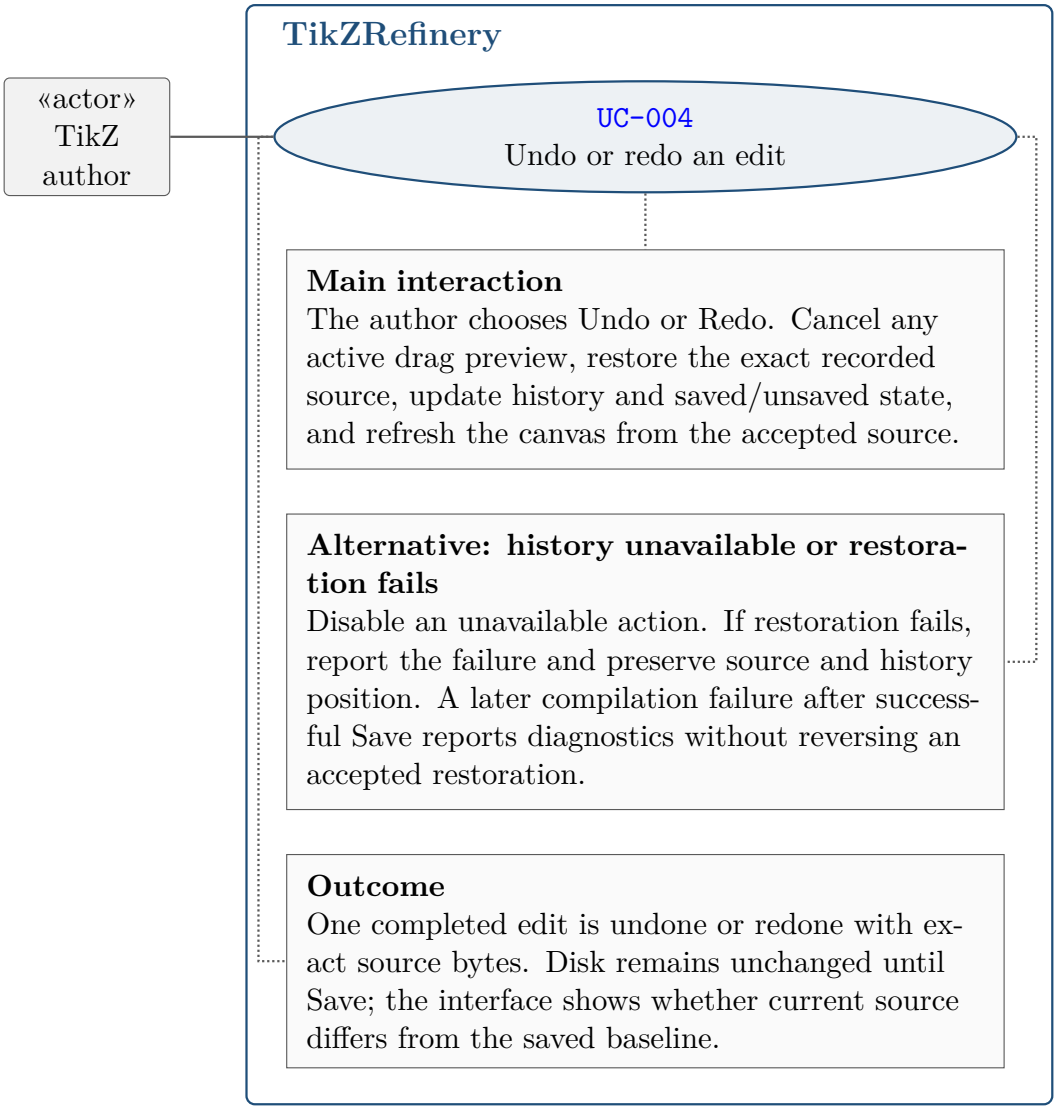


Figure 6.5: Actor-goal view for UC-004: undoing or redoing an edit. See the overview and Chapter 5 for the transaction and history rules.

6.6 Reposition an Attached Label

Table 6.5: Proposed UC-005 Specification

Field	Specification
Identifier	UC-005
Diagrams	Overview; focused diagram
Mapped stories	US-003, US-004, US-007, US-008, US-009, US-010, US-011
Mapped requirements	REQ-F-021, REQ-F-022, REQ-F-023, REQ-F-024, REQ-D-009, REQ-D-010, REQ-D-012, REQ-I-006, REQ-I-007, REQ-A-012, REQ-A-013, REQ-A-015, REQ-A-016, REQ-A-017
Goal	Improve label placement while preserving its attachment and host geometry.
Primary actor	TikZ author
Preconditions	A supported label/host relationship and current binding are available; movement capabilities permit the requested adjustment.
Trigger	The author selects a label and drags it in either or both canvas axes.
Main flow	The label receives priority over its host. Preview adjusts relative placement without source changes. On release, the responsible syntax provider proposes local source changes; common validation and document/history acceptance precede reparse and display refresh. Text, rotation, anchor, alignment and style are preserved unless requested.
Alternate flow	Cancellation restores the preview. Unsupported movement is unavailable with a reason; a rejected completed attempt shows the red-accented bottom strip once. Shared definitions and host coordinates remain unchanged; drag distance never detaches the label.
Postconditions	An accepted move is one undoable source change with attachment retained; noncommits preserve source/history.

6.6.1 UC-005 Diagram

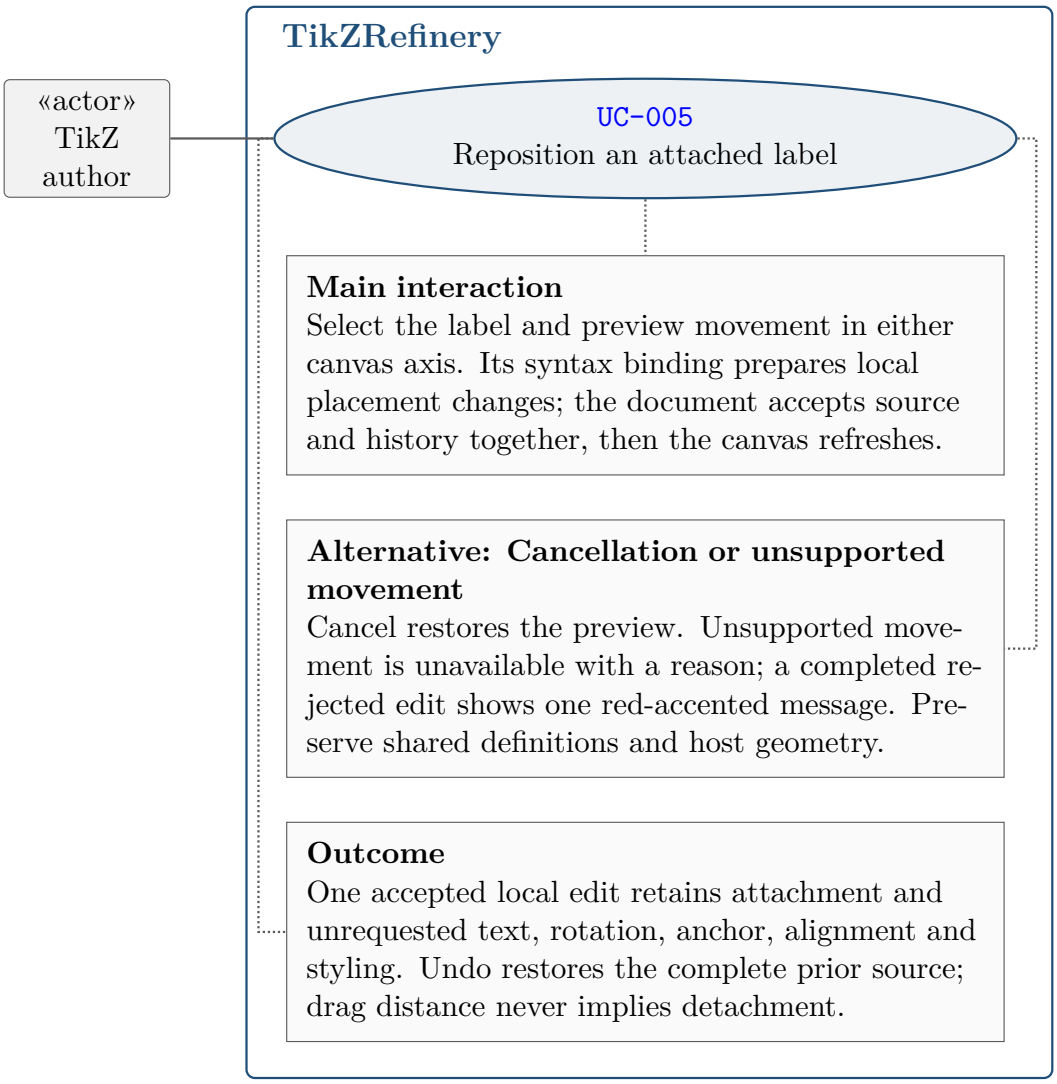


Figure 6.6: Planned actor-goal view for UC-005.

6.7 Select an Obscured Element

Table 6.6: Proposed UC-006 Specification

Field	Specification
Identifier	UC-006
Diagrams	Overview; focused diagram
Mapped stories	US-008, US-009, US-010
Mapped requirements	REQ-D-009, REQ-D-012, REQ-I-008, REQ-I-009, REQ-I-010, REQ-A-015
Goal	Select an overlapping or obscured element without moving blocking objects.
Primary actor	TikZ author
Preconditions	Supported selectable elements provide hit geometry and readable descriptions.
Trigger	The author right-clicks near objects or draws a region and right-clicks inside it.
Main flow	Direct right-click lists all elements intersecting the provisional 6-logical-pixel radius circle. Alternatively, an empty-canvas drag or Select area tool draws a retained rectangle; the menu lists intersecting elements, including partial intersections. Hover highlights each candidate; choosing one selects it. Labels precede overlapping hosts in ordinary hit selection.
Alternate flow	The Select area tool permits a region to start over objects; ordinary object drags retain movement behavior. Escape clears the region. No matches gives an empty/no-candidate indication without source changes.
Postconditions	The chosen element is selected; region creation and candidate browsing create no source edits or history entries.

6.7.1 UC-006 Diagram

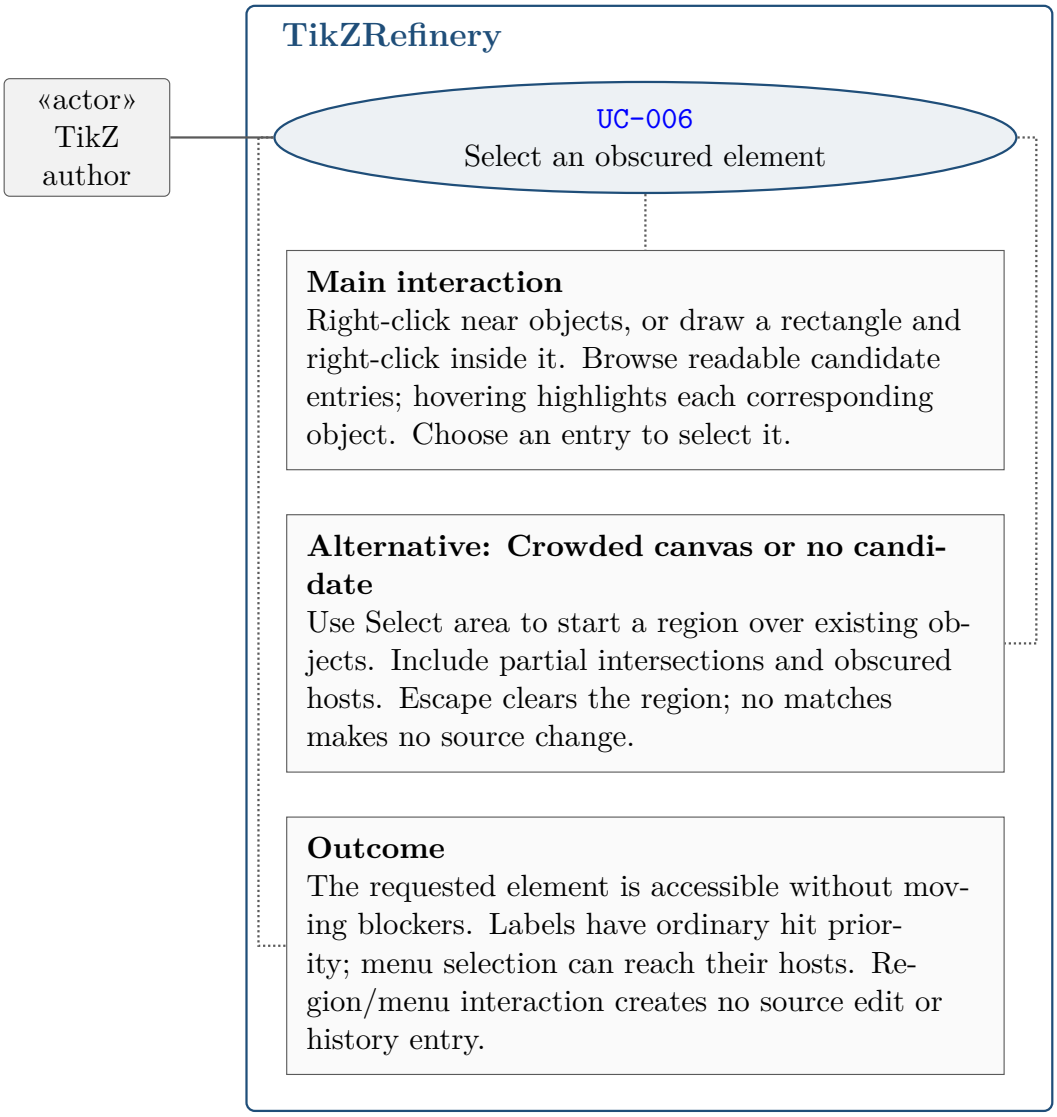


Figure 6.7: Planned actor-goal view for UC-006.

6.8 Choose a Plugin or Shape Provider

Table 6.7: Proposed UC-007 Specification

Identifier	UC-007
Diagrams	Overview; focused diagram
Mapped stories	US-003, US-004, US-008
Mapped requirements	REQ-F-026, REQ-F-027, REQ-F-032, REQ-D-016, REQ-D-017, REQ-A-019, REQ-A-020
Goal	Choose a Plugin or Shape Provider.
Primary actor	TikZ author
Preconditions	compatible provider candidates are discovered.
Trigger	a duplicate ID or ambiguous provider choice requires a decision.
Main flow	Inspect candidate name, version, location and capabilities; explicitly choose a compatible candidate. Different shape variants may coexist and bind different elements.
Alternate flow	Cancel leaves conflicts unresolved. Incompatible candidates remain inactive with a reason. Existing bindings are not silently replaced.
Postconditions	The explicit choice is recorded for the applicable scope; source and history are unchanged.

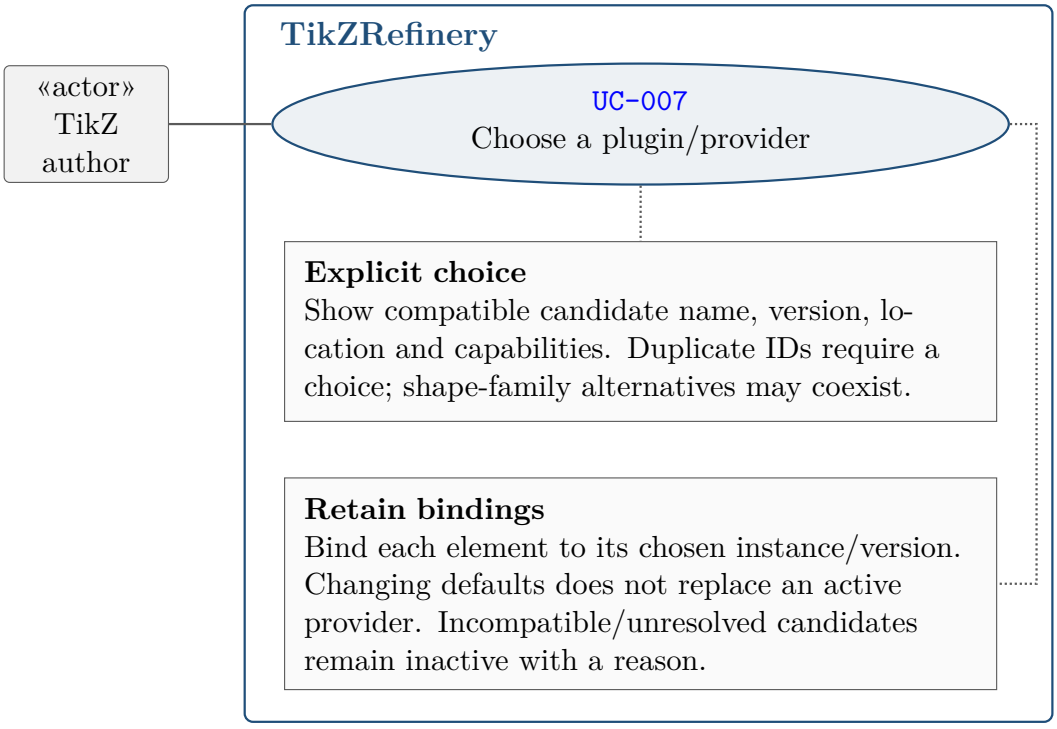


Figure 6.8: Planned UML actor-goal view for UC-007.

6.9 Handle an Available Plugin Update

Table 6.8: Proposed UC-008 Specification

Identifier	UC-008
Diagrams	Overview; focused diagram
Mapped stories	US-003, US-004, US-006, US-008
Mapped requirements	REQ-F-028, REQ-F-029, REQ-F-030, REQ-F-031, REQ-D-016, REQ-D-017, REQ-I-016, REQ-A-020, REQ-A-021, REQ-A-022
Goal	Handle an Available Plugin Update.
Primary actor	TikZ author
Preconditions	an update origin and compatible candidate can be checked.
Trigger	startup or session discovery finds an update.
Main flow	At startup validate updates before restoring diagrams. During a session stage updates separately; activate only after rechecking that the provider is unused.
Alternate flow	For an in-use provider show a nonmodal Later or Restart and update choice. Retain the current version. Restart respects unsaved work. Failed staging/activation retains working support and reports the reason.
Postconditions	Accepted activation preserves supported data and choices; deferral or failure does not alter authoritative source.

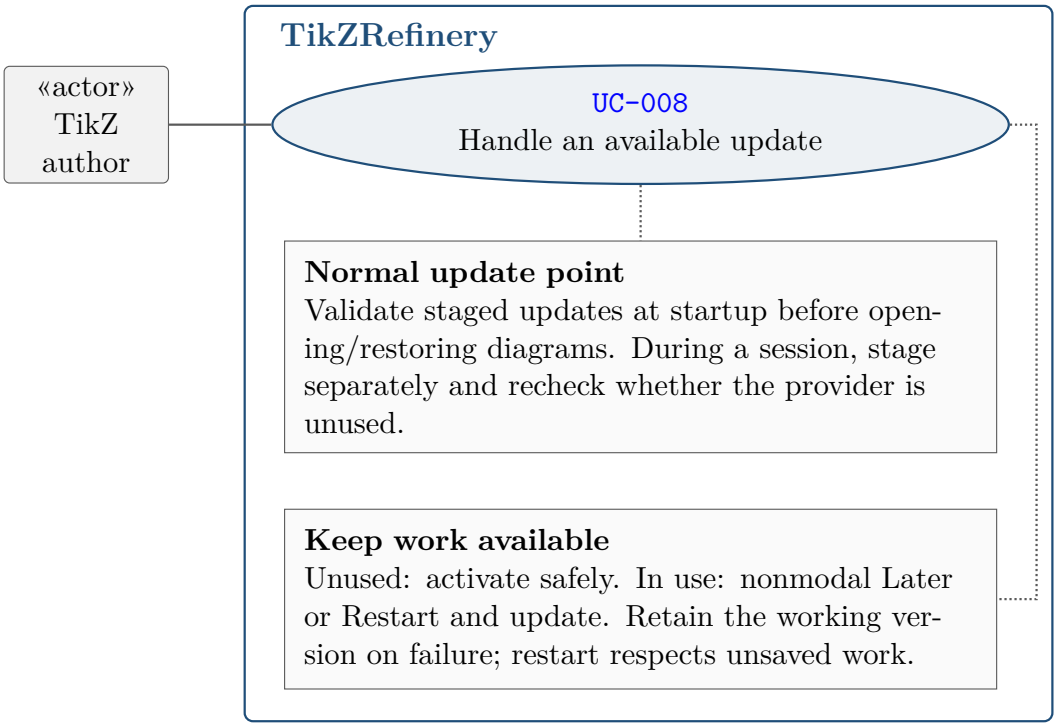


Figure 6.9: Planned UML actor-goal view for UC-008.

6.10 Create Using a Plugin-Provided Shape

Table 6.9: Proposed UC-009 Specification

Identifier	UC-009
Diagrams	Overview; focused diagram
Mapped stories	US-001, US-002, US-003, US-004, US-005, US-007, US-008, US-009, US-011
Mapped requirements	REQ-F-033, REQ-F-034, REQ-F-035, REQ-D-001, REQ-D-018, REQ-I-006, REQ-I-007, REQ-I-013, REQ-I-015, REQ-I-018, REQ-I-019, REQ-I-020, REQ-I-021, REQ-A-024
Goal	Create Using a Plugin-Provided Shape.
Primary actor	TikZ author
Preconditions	A compatible shape provider, reviewed placement parameters and a supported document/syntax insertion context are available. Detailed New/Save As and insertion rules remain open.
Trigger	the author chooses a shape to create a diagram or element.
Main flow	Open Shapes from the persistent sidebar strip and choose a distinguishable catalogue/provider entry. Click for provider-default placement or drag to size a rectangle. Preview obeys minima and stroke-aware bounds. The provider prepares interpreted geometry; the host prepares source insertion and the document validates acceptance. Acceptance assigns identity, selects the new element and returns to selection.
Alternate flow	Missing support or an unfittable default explains unavailable creation. Escape cancels without source changes. Rejected placement reports one reason per completed attempt and retains the tool for retry. New/Save As, insertion locations and exact gesture threshold remain for review.
Postconditions	Accepted creation retains editable source and unrelated bytes. Saving follows UC-003; no creation implementation is claimed.

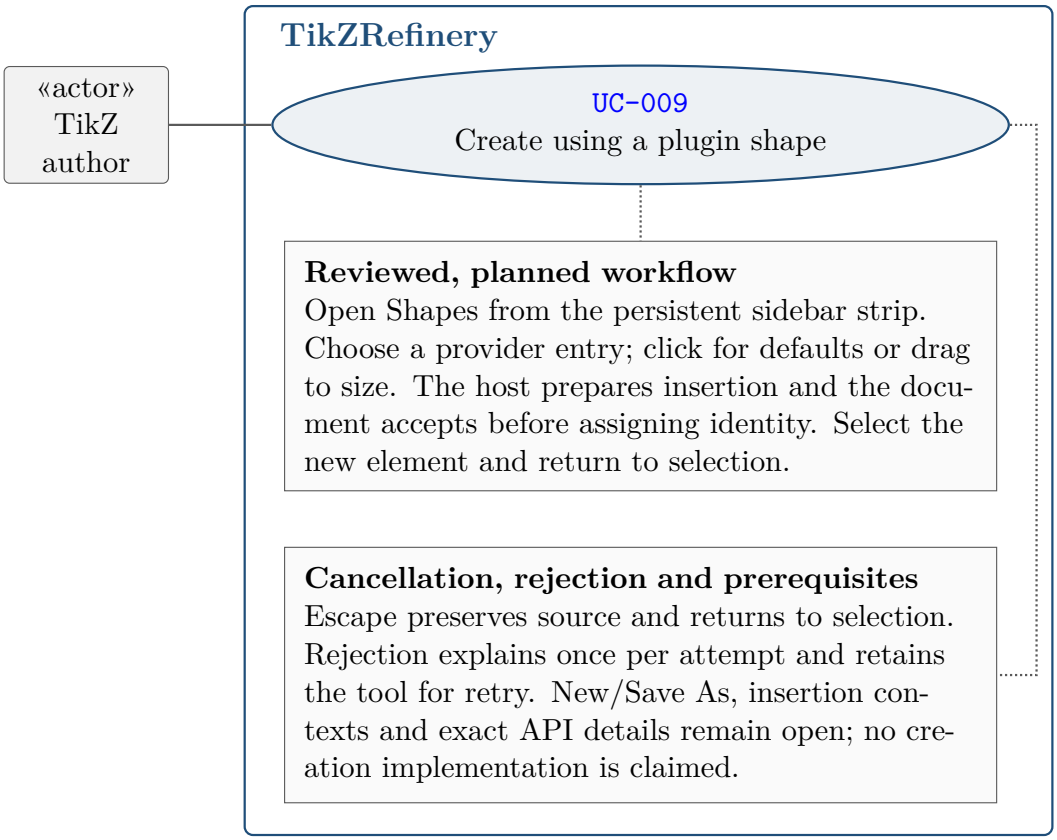


Figure 6.10: Planned UML actor-goal view for UC-009.

Chapter 7

Software Architecture

7.1 Scope and Evidence

The List of Figures links all active figures. The repository's generated `documents/DIAGRAM_INDEX.md` and machine-readable `documents/diagram-index.json` additionally locate editable sources, current/planned status and physical pages in both editions. They support targeted diagram review; material design changes still require whole-document impact and layout review. This chapter records C++/Qt MVC roles, ownership, class contracts and the Windows Alpha deployment (Section 7.2), adapting the course architecture and component patterns. Broader logical, process and development views remain deferred. Supporting activities appear in Appendix D; the implementation rationale is in Section 3.3.1.

The current diagrams follow source. Section 7.10.1 records the implemented 0.1.3-alpha sidebar shell; broader catalogue/creation and shared-service ownership views remain separately planned. The loaded rectangle module and development deployment are in Section 7.7. Historical packages and reports remain frozen; active diagrams describe 0.1.4-alpha. On October 20, 2026, the scaffold built successfully and its foundation test passed; Vincent then confirmed that opening the box sample displayed its source. Section 10.5 records this limited evidence. Rectangle, line and arrow display now have limited evidence in Section 10.7. Whole-element dragging, cancellation, save/reopen and saved-output comparisons follow in Section 10.8; endpoint editing, broader parsing and LaTeX/PDF integration remain unfinished. Neither the build nor these diagrams establish product acceptance. `REQ-DEV-006` aligns the Alpha application's version with the architecture metadata. The existing CMake path reads the numeric baseline from `DocumentVersion`; a build-time step generates the version, Git checkpoint and source state for the title/About UI and package metadata. An unavailable Git context is explicitly labelled. The source archive therefore remains buildable without inventing a checkpoint. The feature-based title/About policy is implemented for the separate 0.1.3-alpha sidebar-shell package; the frozen executable is unchanged. The reviewed transaction/history design in Section 7.8 now has a bounded coordinate-edit implementation in the matching `0.1.4-alpha` package. Generic transactions and file-store extraction remain planned. Demo 3 now presents `0.1.4-alpha`; the older package and slides remain preserved at `0.1.3-alpha`. See Section 10.16.

The MVC and class views support architectural rules `REQ-A-004` through `REQ-A-007`,

together with developer rules REQ-DEV-001 through REQ-DEV-004. Figures 7.4 and 7.3 link to the class details; each detail provides links back to the overview and MVC diagram.
Notation. The class overview uses associations, dashed dependencies, composition diamonds at owners and hollow generalization triangles toward Qt bases. MVC packages group roles; their arrows show dependencies rather than execution order. Ownership is also recorded in Table 7.4. All routes are orthogonal; UML messages and aligned links remain straight. Class details use `+`, `-`, and `#` for public, private, and protected visibility. Signatures use UML-style `name: type` parameters and return types; C++ `const`, reference, and pointer markers are retained. Inherited members, compiler-generated operations, and macro expansions are omitted. Signals and slots have their own headings. Endpoint editing and compiler/PDF integration remain future work.

7.1.1 Feature Baselines and Package Identity

Numbering is controlled by Document History and REQ-DEV-006–REQ-DEV-008. A report revision may advance without a new application package. The active deployment below describes the current 0.1.4-alpha package; archived 0.10.3 evidence retains its original identity. The separate 0.1.2-alpha package implements rectangle extraction and About identity. The retained 0.1.3-alpha package adds the sidebar shell; 0.1.4-alpha adds bounded session undo/redo. Shape discovery/creation define the later 0.2.0-alpha milestone.

7.2 Current Alpha Deployment

Figure 7.1 shows the current Windows `0.1.4-alpha` package on the author's computer. Perspective UML device and execution-environment nodes contain the deployed artifacts: executable, rectangle module, bundled runtime and build records. Figure 7.15 details the native module boundary. Local file I/O follows REQ-C-001; identity follows REQ-DEV-006. TinyTeX and PDF viewing remain separately operated tools. Dashed arrows are usage dependencies; solid arrows annotate file I/O and the author's manual workflow, rather than structural ownership. Frozen 0.10.3 packages, reports and the prior editable figure are retained as historical evidence.

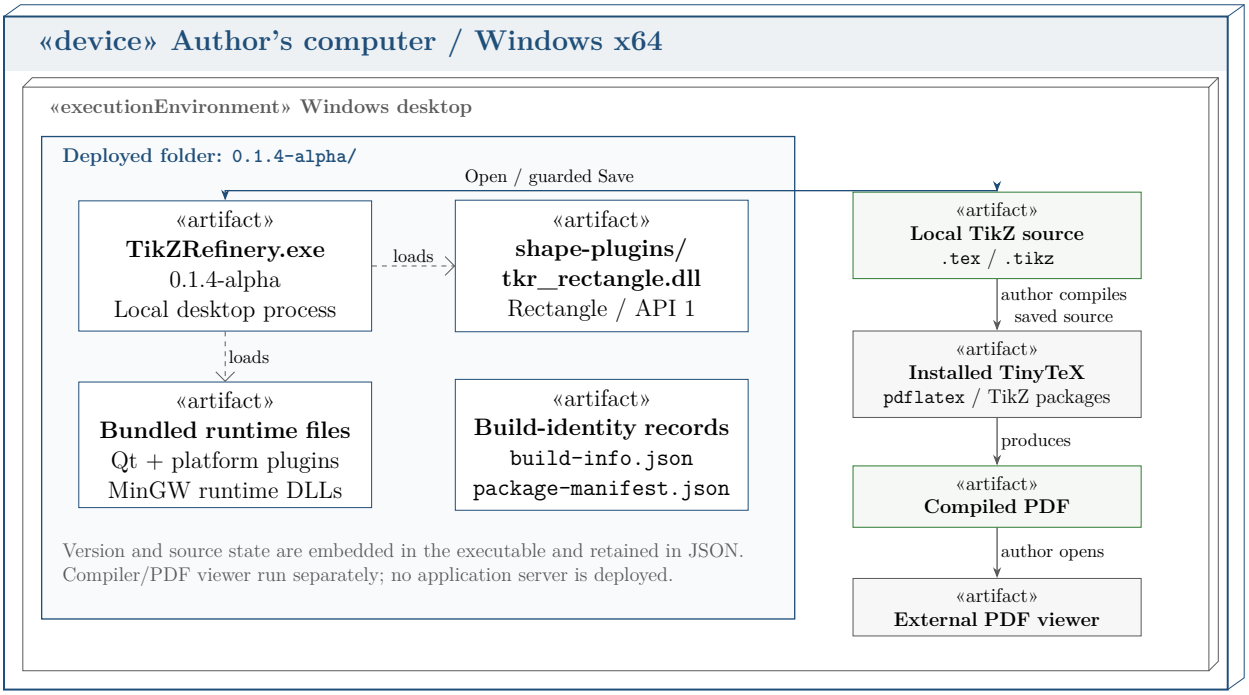


Figure 7.1: Current Alpha deployment and external compilation boundary. Compilation and PDF viewing are manual; in-app compiler/PDF integration remains unimplemented.

7.3 Initial Persona and Interface Design

PERS-001 is a draft role-based persona based on Vincent's project motivation and US-001–US-006. It is a hypothesis for user review, not an interview finding. The course's persona/task/wireframe pattern is adapted as linked architecture views and an actual application screenshot.

7.3.1 PERS-001: TikZ-Literate Report Author

Table 7.1: Initial Persona Hypothesis

Aspect	Working assumption to review
Role and context	A technical-report author uses existing or AI-generated TikZ diagrams, a local desktop and an external LaTeX toolchain.
Knowledge	Recognizes basic TikZ and Open/Save; prefers visual edits to repeated coordinate searches.
Goal	Move/resize a supported shape, inspect and save its source, then compile in standard LaTeX tools.
Frustrations	Coordinate guessing, unexpected source changes, unclear editing bounds and unsupported syntax.
Priority needs	Legible fitted canvas, selection/resize feedback, side-by-side source inspection and guarded saving.
Evidence status	Draft hypothesis; priorities and usability require user review. No participant results are claimed.

7.3.2 Needs Mapped to the Current Interface

Table 7.2: Persona-to-Interface Mapping

Persona need	Current interface response	Existing trace
Understand available drawing space	Labelled Canvas pane; contrasting paper, fixed bounds and automatic fit.	US-002; REQ-I-011–REQ-I-013
Refine a supported layout	Select/drag a body or rectangle corner; constrained preview and Escape/focus-loss cancellation.	UC-002; REQ-F-025
Inspect and retain editable source	Read-only TikZ source pane beside the canvas; changes appear after accepted graphical edits.	US-003; REQ-D-001
Return to the LaTeX workflow	Open/Save controls, saved/unsaved identity and Save/Discard/Cancel guard; compilation remains external.	US-005; UC-003

7.3.3 Current Application Screenshot

Figure 7.2 maps PERS-001 to the actual 0.1.4-alpha Windows package with the sidebar shell and Edit menu. The native capture shows the Shapes page, unavailable creation, provider-backed geometry and the read-only source pane. The version in the normal title matches the package; About contains Git and provider details. The earlier wireframe remains editable source but is no longer used as the current UI figure.

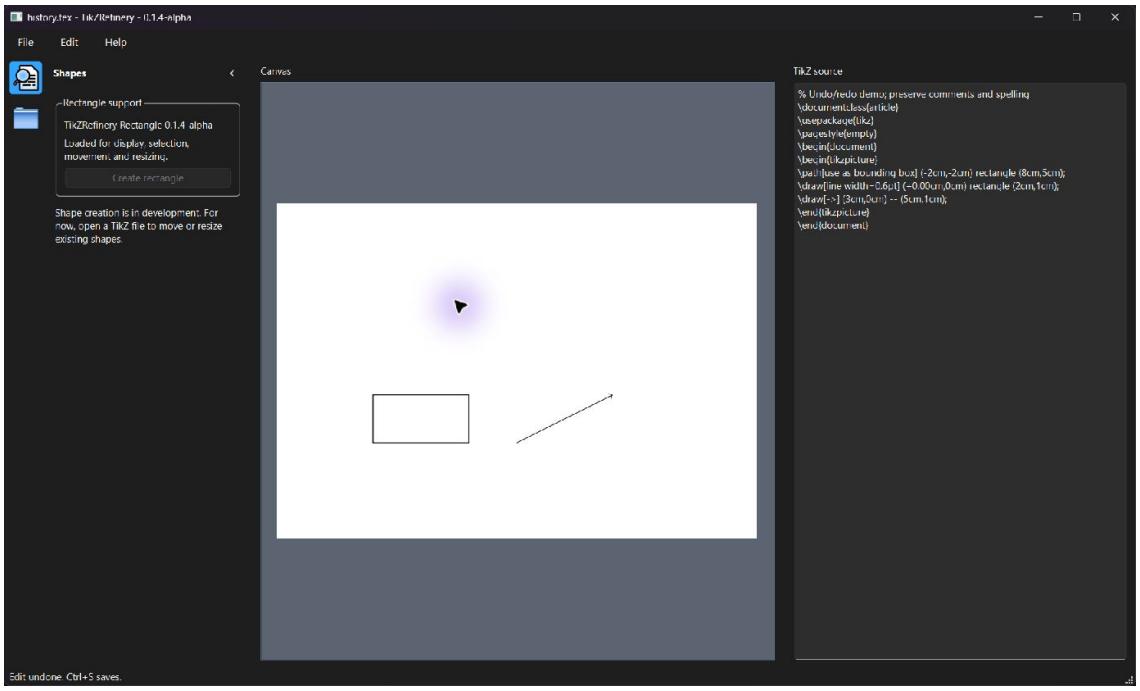


Figure 7.2: Actual 0.1.4-alpha packaged interface matched to the initial persona. Source inspection is read-only; external compilation remains separate.

The author opens a supported fixture, selects and moves/resizes a rectangle, inspects accepted source changes, then saves and reopens it. The split view supports visual and source inspection together. Save is disabled until a committed edit exists; preview leaves source unchanged. Open/Save shortcuts and labelled panes exist; complete keyboard-only and assistive-technology support remain unverified.

Unsupported source retains its bytes with a diagnostic. Proposed rejection feedback follows `REQ-I-006` and `REQ-I-007`. The Shapes/Files shell is implemented, but catalogue and creation in `UC-009`, overlap selection and in-app PDF viewing remain planned. Edit-menu Undo/Redo now supplies bounded session restoration with standard shortcuts. Earlier sidebar screenshots remain retained as dated evidence. Sidebar browsing does not alter source. Files Open reuses the existing guard; no folder tree, catalogue enumeration or creation is implied.

7.3.4 Persona Needs Mapped to Planned Creation

The existing persona remains a hypothesis. Planned creation/sidebar responses are separate from the current interface table and screenshot.

Table 7.3: Persona-to-Planned-Interface Mapping

Need to review	Planned response	Trace
Find available shapes	Provider-distinguishable catalogue previews/names; explicit creation availability.	REQ-F-033; REQ-D-018
Keep canvas space usable	Sidebar shell now supplies persistent Shapes/Files tabs and session width; user validation of the interaction remains planned.	REQ-I-018–REQ-I-020
Create a layout visually	Click provider defaults or drag rectangle corners; preview constraints, cancel/retry and select accepted creation.	UC-009; REQ-F-035; REQ-I-021
Continue file work	Files requests reuse the existing unsaved-work guard; folder-selection/opening details remain unresolved.	UC-001

Editable planned views appear in Figure 7.30; their icons are illustrative. Default Qt styling is retained; external styling and icon-library adoption are deferred. VAL-011 remains Planned.

7.3.5 Persona and Interface Review Still Required

Use VAL-002, VAL-003 and VAL-005 to observe moving/resizing, unrelated-source checks and saving for external compilation. Ask whether bounds, selection, source updates and saved state are clear; record confusion and recovery before revising/rechecking the interface. Persona-specific usability and acceptance remain unexecuted; this persona review adds no new product obligations.

7.4 Model-View-Controller Allocation

Vincent reported that the instructor revisited MVC and preferred class responsibilities in the diagrams. This section records that discussion through Vincent's account. MVC separates document state, presentation, and response to user input [1]. Figure 7.3 applies those roles to the current editor; it is a project-specific allocation, not a claim of strict conformance to every MVC variant.

Document state belongs to `tkrTikzDocument`; the window, view and scene supply presentation, including the sidebar and its Shapes/Files pages. `tkrEditorController` coordinates actions. Parsing and coordinate conversion and edit preparation are GUI-independent support services owned by the controller. They interpret or transform data; they are not additional controllers or persistent document state.

The window also assembles components, reads document state for display, and handles dialogs and the close guard. The document is an ordinary C++ class with no Qt signals: the controller emits `documentChanged()`, and the window then reads source, path, and dirty state. The scene does not mutate source. This application-level allocation is distinct from Qt's item model/view framework, which combines view/controller responsibilities and uses item models and delegates [8]. The scaffold does not use `QAbstractItemModel`.

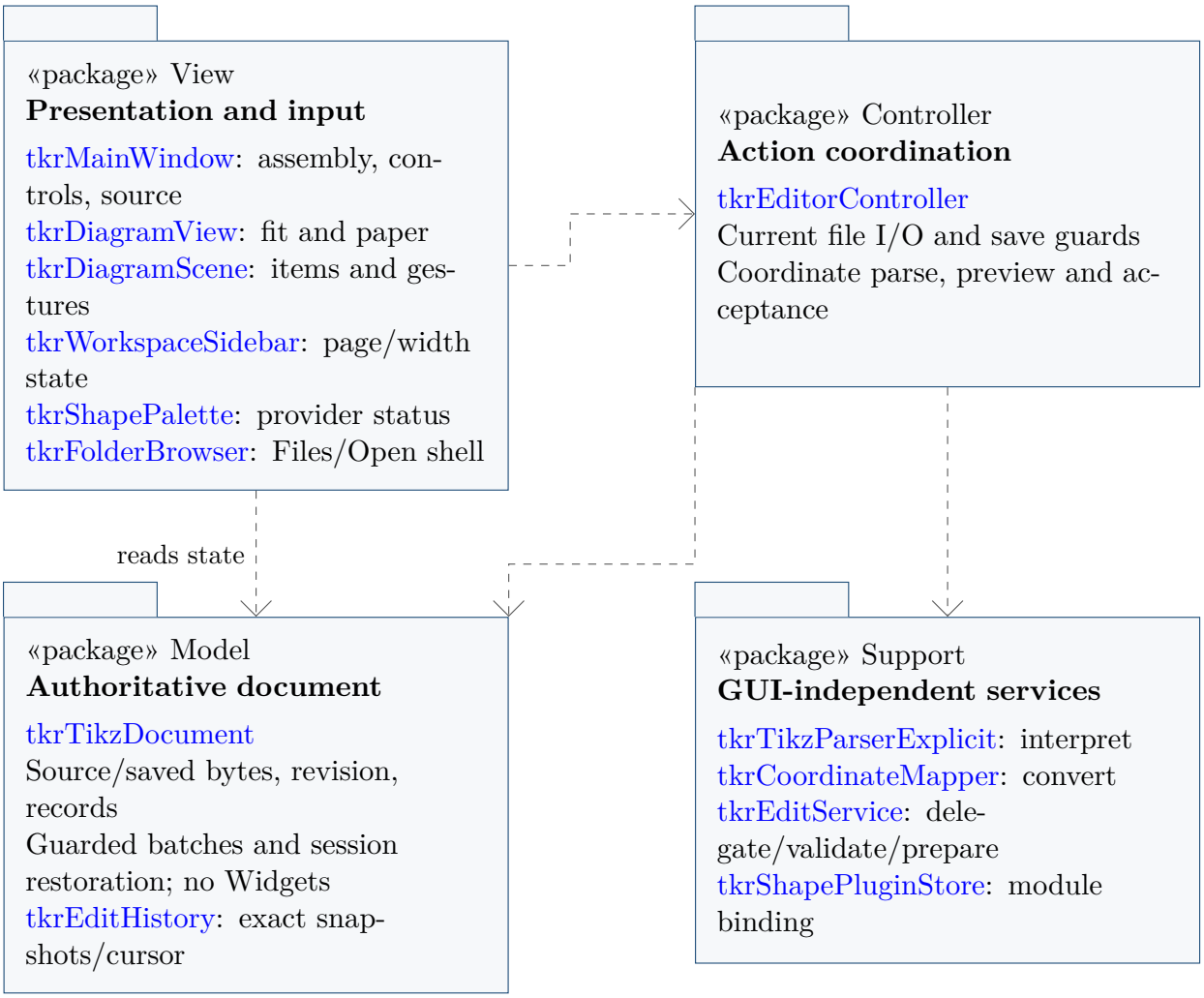
Table 7.4: Ownership Is Separate from MVC Role

Owner	Owned object	Mechanism and lifetime
`main.cpp`	`tkrMainWindow`	Stack object; destroyed after the final window close ends the event loop.
`tkrMainWindow`	Document; controller	Document by value; controller by `unique_ptr`, destroyed before the document.
`tkrMainWindow`	Scene and UI objects	QObject parent ownership, including the canvas view and sidebar through central splitters. The sidebar owns its stacked page widgets; `QGraphicsScene` owns its graphics items.
`tkrTikzDocument`	Edit history	Value member; exact session snapshots, no automatic eviction. The document alone publishes/traverses history.
`tkrDiagramScene`	Shape module store; mapper calibration	Value members; store owns loader and borrows its pinned provider root.
`tkrEditorController`	Parser; mapper; edit service	Value members. Document and scene references are non-owning; both must outlive the controller.

Ownership and MVC roles are distinct. The window owns the model without becoming the model. File-store extraction remains a separate planned refactor.

7.4.1 MVC Diagram

Select a class name to open its detailed diagram. UML packages group current role allocations. Package grouping does not imply C++ inheritance or object ownership.



Packages express MVC role allocation, not ownership or a C++ namespace change. Lifetime mechanisms remain in the ownership table and class overview.

Figure 7.3: Current MVC roles and communication. Whole-element gestures are implemented. Ownership is given in Table 7.4; class relationships appear in Figure 7.4.

7.5 Current Class Diagram

There are thirteen concrete host classes, all `final`: ten core classes in this overview and three sidebar UI classes in Figure 7.25. The provider interface and module realization are in Figure 7.14. Qt's existing ancestry is outside the three-level project inheritance-depth limit (main/base, child, grandchild). The host has no project-defined inheritance chain; the module realizes its provider interface. This overview emphasizes responsibilities and structural dependencies/ownership. Select a class name to open its attributes, operations, and contracts. See Figure 7.3 for MVC roles.

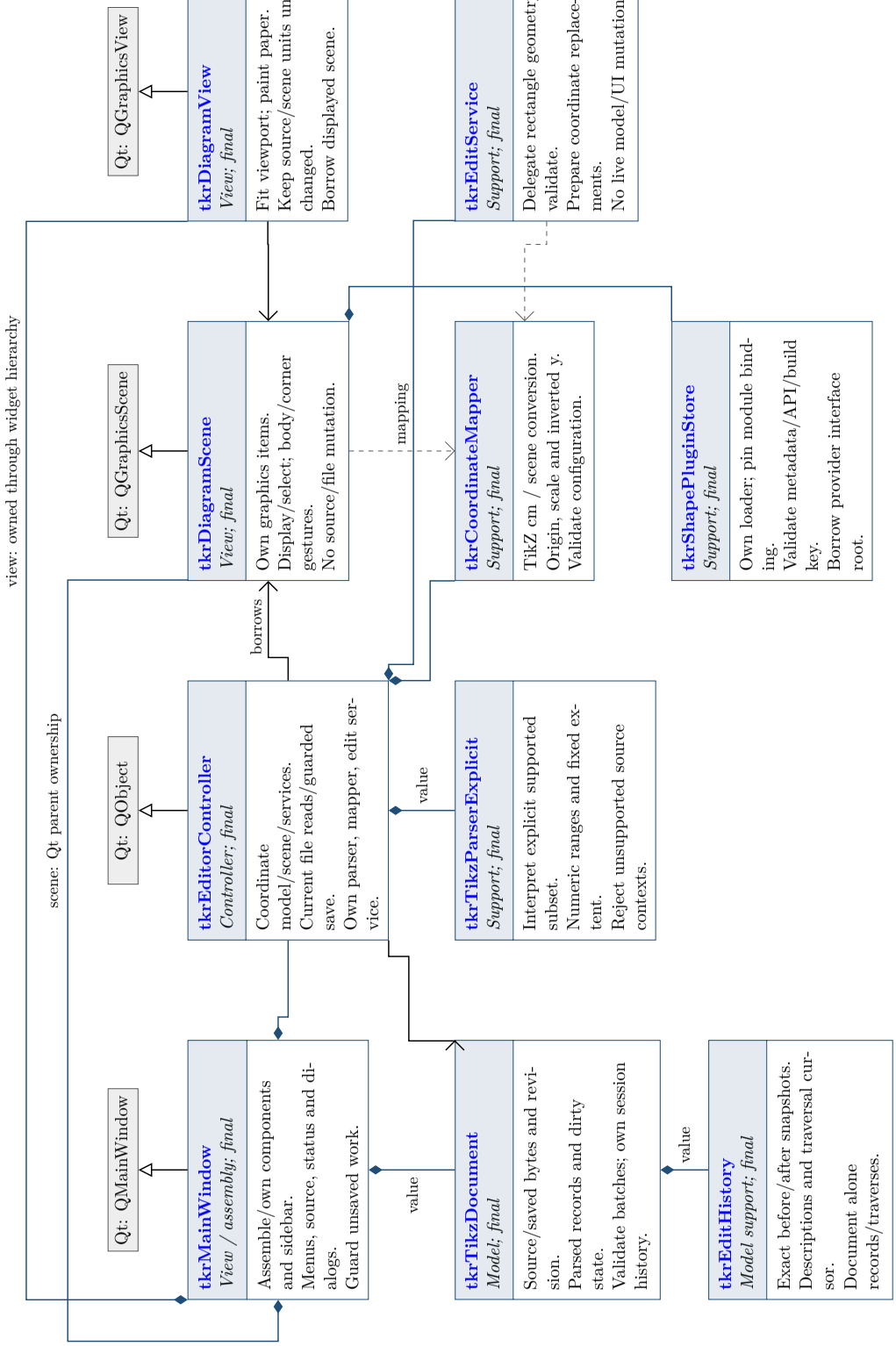


Figure 7.4: Current UML class responsibilities, ownership, borrowed dependencies and direct Qt bases. Sequence diagrams describe temporal communication separately.

7.6 Individual Class Diagrams

7.6.1 tkrEditHistory

Source: `code/src/editing/tkrEditHistory.h` and its implementation. Only the owning document can record or traverse entries. Candidate storage is prepared before publication; allocation failure preserves the redo branch.

tkrEditHistory *final; Model support; owned by document*

Responsibilities
Retain exact session snapshots and operation descriptions.
Prepare a new entry without losing the existing redo branch on failure.
Supply restoration targets; move cursor only through the owning document.

```
- m_vecEntries: QVector<Entry>
- m_nCursor: qsizetype = 0
```
Entry contains before/after QByteArray and QString description.

```
+ canUndo(), canRedo() const: bool
+ undoDescription(), redoDescription() const: QString
```
Private; friend tkrTikzDocument
```
- record(baBefore, baAfter: const QByteArray&,
  strDescription: const QString&): void
- restorationSource(bRedo: bool) const: QByteArray
- advance(bRedo: bool): void
- clear(): void
- swap(historyOther: tkrEditHistory&): void [noexcept]
```

Figure 7.5: Implemented session snapshot storage; no automatic eviction.

7.6.2 tkrTikzDocument

Source: `code/src/model/tkrTikzDocument.h` and its implementation. The saved byte copy determines dirty state. Loading increments the revision; accepted source changes increment it and invalidate parsed records. Edit batches check revision, bounds, overlaps, and expected bytes before mutation.

tkrTikzDocument *final; Model; no base class*

Responsibilities
Keep source bytes authoritative; expose document state.
Store parsed records for the current revision.
Validate a complete coordinate-edit batch before changing source.

Private attributes
- `m_baSource: QByteArray` (current bytes)
- `m_baSavedSource: QByteArray` (dirty comparison)
- `m_strFilePath: QString`
- `m_vecElements: QVector<ElementRecord>`
- `m_uRevision: quint64 = 0`
- `m_bIsLoaded: bool = false`
- `m_documentId: QUuid`
- `m_history: tkrEditHistory`

Public operations
```
+ loadSource(baSource: const QByteArray&,
  strFilePath: const QString&): void
+ sourceBytes() const: const QByteArray&
+ filePath() const: const QString&
+ elements() const: const QVector<ElementRecord>&
+ revision() const: quint64
+ identity() const: const QUuid&
+ isLoaded() const: bool
+ hasUnsavedChanges() const: bool
+ markSaved(): void
+ setParsedElements(vecElements: const QVector<ElementRecord>&,
  uExpectedRevision: quint64): OperationResult
+ canUndo(), canRedo() const: bool
+ undoDescription(), redoDescription() const: QString
+ restoreHistory(bRedo: bool): OperationResult
+ applyCoordinateEdits(vecEdits: const QVector<CoordinateEdit>&,
  strDescription: const QString&)
  : OperationResult
```

Private; controller is a friend
```
- acceptPreparedState(documentCandidate: tkrTikzDocument&,
  uExpectedRevision: quint64): OperationResult
```

Figure 7.6: Model state and guarded edits. Return to the class overview or MVC diagram.

7.6.3 tkrMainWindow

Source: `code/src/ui/tkrMainWindow.h` and its implementation. An optional startup-file argument loads a source through the controller. The source widget decodes bytes only for viewing. Open and close ask whether to save, discard, or cancel when dirty. Save errors keep the window open. Widget/action parent chains ultimately belong to the window. The window labels the Canvas and TikZ source panes. Its child `tkrDiagramView` extends `QGraphicsView`; it fits the scene to the pane uniformly and draws white paper only within an explicit fixed editing extent, with a grey surround. Without a fixed extent it keeps a white viewport. This view transform leaves scene coordinates and source unchanged. See Section 7.6.4 for the view's responsibilities.

tkrMainWindow *final; View / assembly; extends QMainWindow*

Responsibilities
Assemble and own components; connect controls and controller signals.
Display read-only source, status, file title, and unsaved-state prompts.
Ask the controller to open/save; accept or reject a close request.

Private attributes
- `m_document: tkrTikzDocument` (owned value)
- `m_pScene: tkrDiagramScene* = nullptr`
- `m_pView: tkrDiagramView* = nullptr`
- `m_pSidebar: tkrWorkspaceSidebar* = nullptr`
- `m_pSourceView: QPlainTextEdit* = nullptr`
- `m_pSaveAction: QAction* = nullptr`
- `m_pUndoAction, m_pRedoAction: QAction* = nullptr`
- `m_upController: std::unique_ptr<tkrEditorController>`

Scene/widgets/actions use QObject parent ownership.

Public lifetime operations
```
+ tkrMainWindow(pParent: QWidget* = nullptr,
  strInitialFilePath: const QString& = {}) [explicit]
+ ~tkrMainWindow() [override]
```

Protected operation
```
# closeEvent(pEvent: QCloseEvent*): void [override]
```

Private operations
```
- openDocument(): void
- saveDocument(): bool
- confirmDiscardOrSave(): bool
- updateDocumentView(): void
```

Figure 7.7: View controls, assembly, and close protection. Return to the class overview or MVC diagram.

7.6.4 tkrDiagramView

The window supplies scene bounds and whether they are fixed. The scene owns the diagram items and keeps the preview within a fixed extent. The view owns the viewport transform and background: its uniform fit preserves the aspect ratio, and white paper ends at a declared editing boundary. During partial repaints, background fills temporarily disable antialiasing. The view paints the whole paper against the viewport's existing repaint clip so the paper edge stays fixed. Painter state is restored before drawing items, which keep smooth outlines. The view makes no source edits. This supports REQ-I-011–REQ-I-014.

tkrDiagramView *final; View; extends QGraphicsView*
Responsibilities Fit the scene to the viewport and show fixed editing limits. Paint the background without disturbing item rendering.

Private attributes
`- m_rectCanvas: QRectF`
`- m_bFixedBounds: bool = false`
The base view references the scene; it does not own it.

Public operations
`+ tkrDiagramView(pScene: QGraphicsScene*, pParent: QWidget* = nullptr)`
`+ setCanvasBounds(rectCanvas: const QRectF&, bFixedBounds: bool): void`
Protected overrides
`# drawBackground(pPainter: QPainter*, rectExposed: const QRectF&): void`
`# resizeEvent(pEvent: QResizeEvent*): void`
Private operation
`- fitCanvas(): void`

Figure 7.8: Fitted canvas view, owned through the window's widget hierarchy. Return to the class overview or MVC diagram.

7.6.5 tkrDiagramScene

Source: `code/src/ui/tkrDiagramScene.h` and its implementation. `resetDiagram()` clears owned items and resets the scene extent. `displayElements()` validates and draws rectangles and straight segments with TeX-point widths. Default curved arrow tips use the installed PGF geometry, round tip strokes and shortened butt-cap shafts. Each segment is one filled outline item, including its tips. A separate comparison extent sets scene bounds without creating an item and limits the drag preview. Release still validates the constrained edit. Selectable items use scene-controlled movement. Selected rectangles show screen-sized corner handles drawn in the foreground, without adding diagram elements. Mouse events emit move or corner-resize gestures with revision, element ID and scene positions; no item pointer survives a rebuild. Escape or focus loss cancels the preview. Endpoint gestures remain unsupported.

tkrDiagramScene *final; View; extends QGraphicsScene*

Responsibilities
Own graphics items through Qt and clear obsolete geometry.
Display rectangles and line/arrow outlines using mapper and extent.
Emit move/corner gestures; draw screen-sized selection handles.
Restore source geometry on cancellation.

Attributes and state
Private: `m_bFixedBounds: bool`, `m_bDragActive: bool`, `m_dragUpdate: DragUpdate`.
Inherited scene state owns graphics items; no retained item pointers.
Owns `m_shapePlugins: tkrShapePluginStore` and copied `m_mapper: tkrCoordinateMapper`.
Source bytes, parsing, and filesystem access remain outside this class.

Public constructor and operations

```
+ tkrDiagramScene(pParent: QObject* = nullptr,
  strRectangleModule: const QString& = {}) [explicit]
+ displayElements(vecElements: const QVector<ElementRecord>&,
  coordinateMapper: const tkrCoordinateMapper&,
  comparisonExtent: const ComparisonExtent& = {})
  : OperationResult
+ resetDiagram(): void
+ hasFixedBounds(): bool [const]
+ previewGeometry(uElementId: quint64, uDocumentRevision: quint64,
  ptFirstScene: const QPointF&, ptSecondScene: const QPointF&,
  bResize: bool): OperationResult
+ selectElement(uElementId: quint64): void
+ cancelInteraction(): void
+ shapeProvider() const: const tkrShapeProvider*
+ pluginDescription() const: QString
```

Protected event overrides

```
# drawForeground(pPainter: QPainter*, rectExposed: const QRectF&): void
# mousePressEvent(pEvent: QGraphicsSceneMouseEvent*): void
# mouseMoveEvent(pEvent: QGraphicsSceneMouseEvent*): void
# mouseReleaseEvent(pEvent: QGraphicsSceneMouseEvent*): void
# keyPressEvent(pEvent: QKeyEvent*): void
# focusOutEvent(pEvent: QFocusEvent*): void
```

Signals

```
+ dragPreviewRequested(dragUpdate: const DragUpdate&): void
+ dragCommitRequested(dragUpdate: const DragUpdate&): void
+ dragCancelled(): void
```

Figure 7.9: Scene presentation and gesture interface. Return to the class overview or MVC diagram.

7.6.6 tkrEditorController

Source: `code/src/controller/tkrEditorController.h` and its implementation. Open success means bytes were read, even if parsing is unavailable. Saving compares the disk file with its baseline, then uses `QSaveFile`; it marks saved only after a successful commit. This comparison does not eliminate a concurrent-write race. Refresh clears obsolete records/items on failure; success stores supported elements. Preview validates the gesture and moves the scene geometry without source mutation. The owned edit service prepares constraints and coordinate replacements; release validates a candidate parse/display before applying one guarded batch. Unchanged axes retain their spelling. Cancellation rebuilds from source; invalid requests preserve the document. Ordinary boundary gestures apply the furthest valid result.

tkrEditorController *final; Controller; extends QObject*

Responsibilities
Coordinate model, scene, parser, and mapper; perform file I/O.
Guard saving against observed external changes; notify the window.
Delegate move/resize preparation; coordinate guarded source edits.

Private attributes
- `m_refDocument: tkrTikzDocument&` (non-owning)
- `m_refScene: tkrDiagramScene&` (non-owning)
- `m_parser: tkrTikzParserExplicit` (owned value)
- `m_mapper: tkrCoordinateMapper` (owned value)
- `m_editService: tkrEditService` (owned value)
- `m_comparisonExtent: ComparisonExtent` (value)
- `m_baFileBaseline: QByteArray` (last open/save bytes)

Public constructor and operations
```
+ tkrEditorController(document: tkrTikzDocument&,
  diagramScene: tkrDiagramScene&, pParent: QObject* = nullptr)
+ openDocument(strFilePath: const QString&): OperationResult
+ saveDocument(): OperationResult
+ undo(), redo(): OperationResult
+ refreshParsedDocument(): OperationResult
```
Public slots
```
+ handleDragPreview(dragUpdate: const DragUpdate&): void
+ handleDragCommit(dragUpdate: const DragUpdate&): void
+ cancelDrag(): void
```
Signals
```
+ documentChanged(): void
+ statusChanged(strMessage: const QString&): void
```
Private operations
```
- validateDrag(dragUpdate: const DragUpdate&) const: OperationResult
- prepareGesture(dragUpdate: const DragUpdate&) const: GestureGeometry
- rejectDrag(strReason: const QString&): void
- restoreHistory(bRedo: bool): OperationResult
```

Figure 7.10: Controller coordination, borrowed references, and owned services. Return to the class overview or MVC diagram.

7.6.7 tkrTikzParserExplicit

Source: `code/src/parsing/tkrTikzParserExplicit.h`.

The implementation uses the matching `.cpp` file. The parser accepts explicit centimetre rectangles and straight segments with optional forward, backward or double arrowheads. It masks comments without changing byte offsets, and returns revision-tagged numeric ranges. It recognizes only bare command fragments, one unconfigured picture, or the corpus wrapper. Unknown syntax rejects the entire interpretation. A bounding path is calibration data, separate from element records. Broader syntax remains unsupported. Shared records, including `ComparisonExtent`, are declared in `code/src/model/EditorTypes.h`.

tkrTikzParserExplicit *final; Support service; no base class*
Responsibilities *explicit subset parsing* Recognize cm rectangles, lines and default-direction arrows. Identify element geometry and coordinate byte ranges for minimal edits. Reject unsafe contexts; return diagnostics without changing source.
Attributes and state No declared data members. Inputs are borrowed for one parse call. No document, scene, widget, file, or compiler ownership.

Public operation
`+ parse(baSource: const QByteArray&,`
`  uDocumentRevision: quint64) const: ParseResult`
Current result
Safety flag, revision-tagged numeric byte ranges, comparison extent.
Unknown syntax: false flag, no records, explanatory diagnostic.
Host scene/controller use supported records for guarded gestures.

Figure 7.11: Stateless subset parser and conservative source contract. Return to the class overview or MVC diagram.

7.6.8 tkrCoordinateMapper

Source: `code/src/geometry/tkrCoordinateMapper.h` and its implementation. The mapper stores scene origin and scene units per centimetre. Configuration rejects nonfinite values and nonpositive scale before mutation. Mapping methods assume finite input points; they do not validate them. The defaults are scaffold values, not stakeholder performance or accuracy targets.

tkrCoordinateMapper *final; Support service; no base class*

Responsibilities
Map TikZ centimetres to scene units and provide the inverse mapping.
Apply an explicit origin, positive scale, and opposite y-axis direction.
Reject invalid configuration before changing the previous mapping.

Private attributes
```
- m_ptSceneOrigin: QPointF = QPointF(200.0, 300.0)
- m_rSceneUnitsPerCentimeter: qreal = 100.0
```

Public operations
```
+ configure(ptSceneOrigin: const QPointF&,
  rSceneUnitsPerCentimeter: qreal): OperationResult
+ mapToScene(ptTikzPosition: const QPointF&) const: QPointF
+ mapToTikz(ptScenePosition: const QPointF&) const: QPointF
```

Coordinate contract
For origin (o_x, o_y), scale s, and TikZ point (x_t, y_t):
$x_s = o_x + sx_t, \quad y_s = o_y - sy_t.$
$x_t = (x_s - o_x)/s, \quad y_t = (o_y - y_s)/s.$
Viewport pixels, zoom, and pan are outside this mapping.

Figure 7.12: Coordinate conversion state, operations, and inverse equations. Return to the class overview or MVC diagram.

7.6.9 tkrEditService

Source: `code/src/editing/tkrEditService.h`.

The implementation uses the matching `.cpp` file. This GUI-independent value member prepares move/resize geometry and coordinate replacements. It shares the stroke envelope with the parser through `ShapeGeometryLimits`. Offsets use 0.000001 cm precision, rounded inward at boundaries; decimal addition retains original high-precision coordinates. No source bytes or history are mutated here. The current document validates revision, ranges, expected bytes and overlaps before applying a batch. The general transaction API remains proposed; this service is its bounded edit-preparation step. Rectangle calculations now use the borrowed provider interface; host validation and exact source rewriting remain here (current extraction).

tkrEditService *final; GUI-independent support service*

Responsibilities
Delegate rectangle geometry to its provider; constrain built-in segments.
Validate provider identity, operation invariants and host constraints.
Prepare only changed-coordinate replacements with exact decimal addition.
Share the parser's stroke envelope; never mutate source, history or widgets.

State and dependencies
Stateless; owned by value in `tkrEditorController`.
Borrowed inputs: element, gesture, mapper, extent and source bytes.
`ShapeGeometryLimits`: 0.01 cm minimum; six-decimal offsets.
`GestureGeometry`: operation result and two TikZ positions.

Public operations

```
+ prepareGeometry(element: const ElementRecord&, drag: const DragUpdate&,
  mapper: const tkrCoordinateMapper&, extent: const ComparisonExtent&,
  pShapeProvider: const tkrShapeProvider* = nullptr)
  const: GestureGeometry
+ prepareEdits(element: const ElementRecord&, baSource: const QByteArray&,
  geometry: const GestureGeometry&, vecEdits: QVector<CoordinateEdit>&)
  const: OperationResult
```

Figure 7.13: Implemented edit preparation service. Return to the class overview or MVC diagram.

7.7 Implemented Rectangle Plugin

Vincent authorized the bounded extraction after the Week 12 draft. The rectangle extraction originated in `0.1.2-alpha`; the current `0.1.4-alpha` package also implements the bounded sidebar shell and session undo/redo for coordinate edits. Catalogue/creation and shared-service ownership remain planned. The broader `0.2.0-alpha` milestone is shape discovery and creation. The original `0.10.3` demo remains frozen. Earlier report/submission snapshots retain their version labels; the pre-extraction `0.1.2-alpha` PDFs are archived separately rather than relabelled as evidence of this implementation.

Table 7.5: Current Plugin Responsibilities

Class / boundary	Implemented responsibility
`tkrShapeProvider`	Qt Core-only Strategy interface: available operations, presentation and preparation. No source ranges, widgets or graphics-item pointers.
`tkrRectangle ShapeProvider`	Separate native module supplies outline and selection rectangles, readable description, corner IDs/positions and constrained move/resize calculations. Uses initial geometry and total centimetre displacement.
`tkrShape PluginStore`	Scene-owned value object owns its loader and borrows the root interface. Validates fixed candidate metadata/build key; pins a runtime binding and exposes module diagnostics/identity. No discovery-order selection.
Host scene and service	Scene creates/owns items, renders provider values and handles input. Service checks operations/proposals and prepares exact coordinate rewrites. Parser and source bindings stay in the built-in host route.
Document	Assigns document UUIDs to revision-scoped elements; validates expected revision, source ranges and bytes before applying a batch; owns bounded session history. Generic transactions remain separately planned.

SDK version 1 uses opposite-corner geometry, black unfilled/empty-content rectangles and presentation records in `code/plugin-api/`. Requests carry document/element identity and revision, parameter schema, initial description, total displacement, semantic corner and constraints. PreparedChange, Unchanged and CannotPrepare remain distinct from source application. The host rechecks identity, finite geometry, bounds, minima, unchanged appearance/content and move/fixed-corner invariants before preparing source edits.

7.7.1 Current Provider Class Relationships

The host now has thirteen concrete classes, including the three sidebar widgets. The module adds one concrete class and realizes one project interface: fifteen class/interface declarations in total. The focused provider view below follows the current scene-owned module store. The implemented class is still `tkrRectangleShapeProvider`. Vincent reported the instructor's requested specialization suffix, `tkrShapeProviderRectangle`; that coordinated code/files/build rename remains pending under REQ-DEV-001. The DLL identity is separate. All concrete classes are `final`; the interface realization is a two-level project inheritance path. Qt ancestry is excluded from the project's limit.

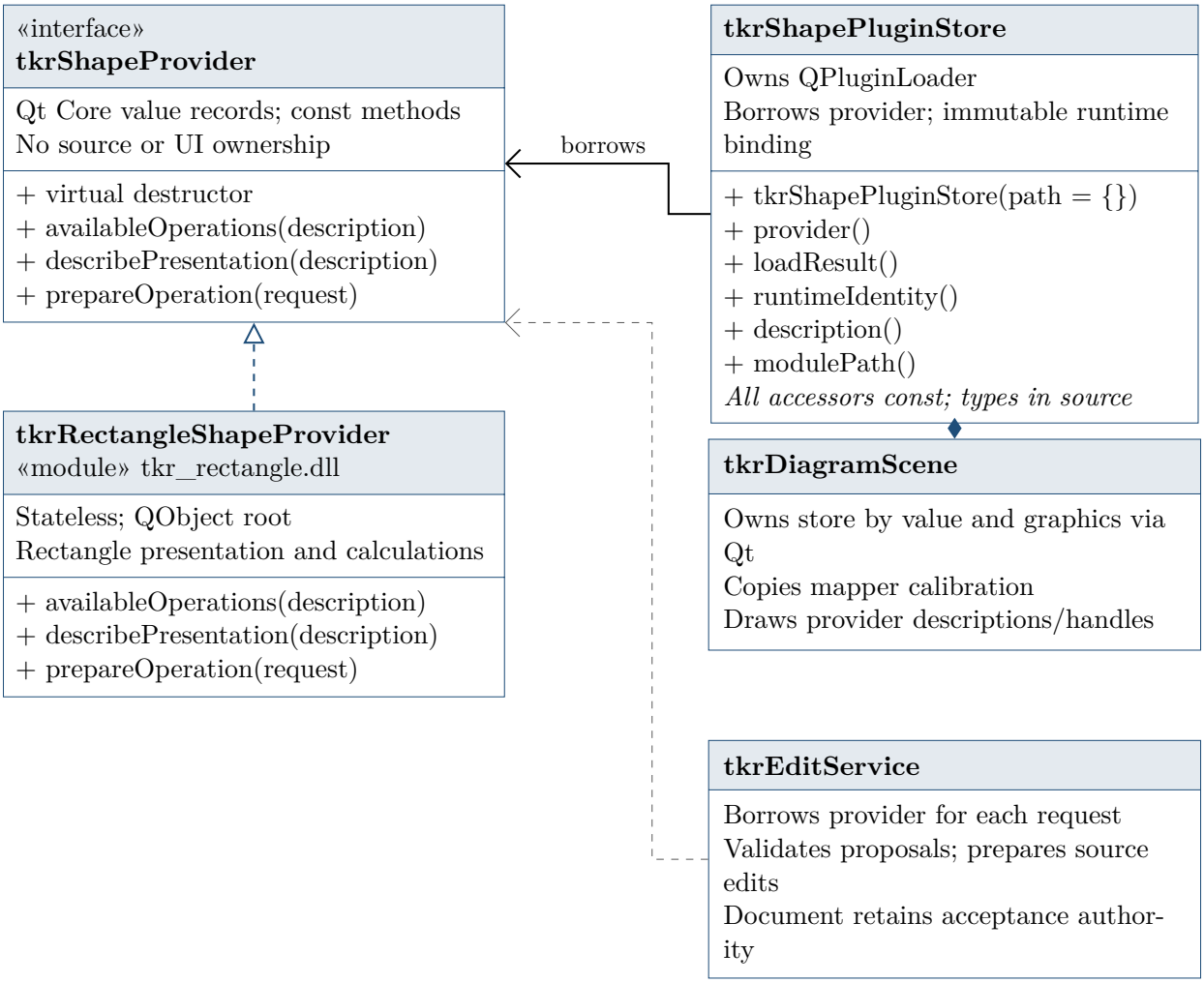


Figure 7.14: Implemented UML provider interface, module and host collaboration. All relationships are structural; the scene owns Qt items and its module store.

Source contracts:
`code/plugin-api/tkrShapeProvider.h`
`code/src/plugins/tkrShapePluginStore.h`

The matching rectangle header is under `code/plugins/rectangle/`. Store private state comprises its loader, borrowed provider, load result, runtime ID, name and version. The module is stateless; metadata declares identity/version, role/type, API/data/parameter versions, operations and exact build key. Interface operations return an operation list, presentation record and preparation result respectively.

7.7.2 Current Plugin Deployment

The separate development folder is `code/packages/0.1.4-alpha/`. The loader uses its own executable directory and the explicit `shape-plugins/tkr_rectangle` module, not a filesystem scan. Native C++ compatibility requires the exact API, geometry/parameter versions, Qt/compiler/platform/pointer-size/configuration key. Qt also checks module compatibility. This is a trusted same-toolchain boundary, not a stable cross-vendor ABI or plugin sandbox.

Missing or incompatible modules preserve authoritative source and report unavailable rectangle support without a hidden implementation. Existing conservative display policy clears graphical records/items for that interpretation; a segment-only document still uses the built-in route. The scene/store binding never changes during an open session; Qt keeps the module resident. Multiple-provider choice, hot updating, conversions/adapters and shape creation remain planned. Normal title and current package folder show `0.1.4-alpha` without a Git hash; About includes the shortened checkpoint, full checkpoint/state and provider path.

7.7.3 Implemented Evidence and Limits

TC-061 evidence is in Section 10.13. The extraction originally passed six suites; the sidebar-shell build passed all seven in Debug and Release, adding the UI state/lifetime suite in Section 10.15. Existing display/drag/resize and real-module contract checks remain included. The history build passes eight suites per configuration (Section 10.16). These are bounded execution results, not acceptance of every product requirement or the full later plugin design. The actual UI appears in Figure 7.2.

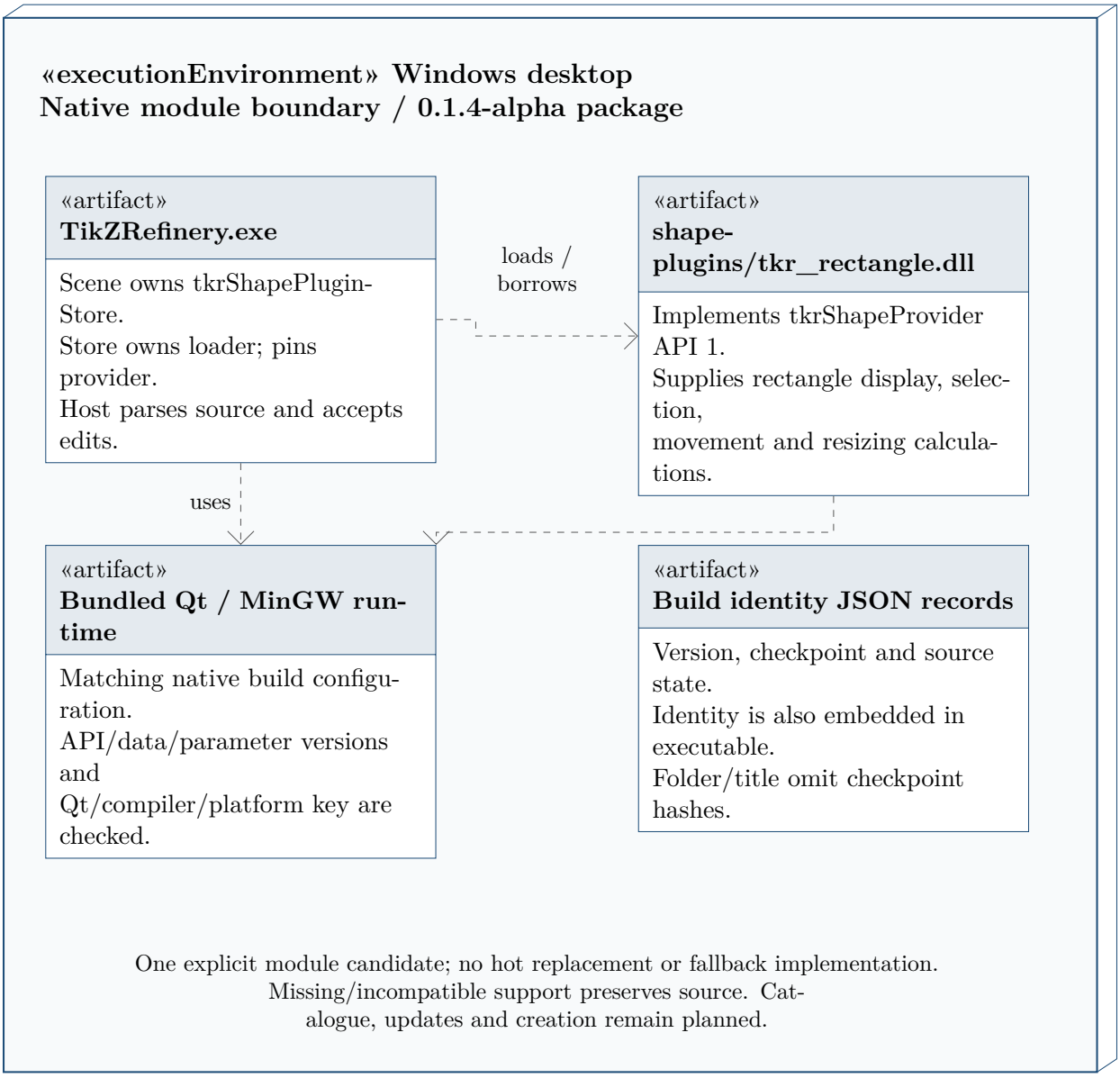


Figure 7.15: Current 0.1.4-alpha in-process module deployment detail. Dashed arrows are usage dependencies; compilation/PDF viewing remain external. The physical node and external tools are shown in Figure 7.1.

7.8 Planned Transaction Design

The agreed behavior is defined by the category registers in Chapter 5: Functional (Table 5.2), Data (Table 5.3), Interface (Table 5.4), and Architectural (Table 5.6). The undo/redo behavior and architectural responsibilities are agreed; the representations and operation names below are implementation sketches. For the prototype, Vincent approved unbounded session snapshots on November 24; a measured memory budget, eviction and provider-dependent retained work remain future decisions in Chapter 8. Agreed design does not mean implemented code. The current code implements coordinate batches, whole-element moves and rectangle resizing; session history now supports these coordinate edits; generic transactions remain unimplemented. Rectangle plugin loading is recorded in Section 7.7. The current class diagrams remain source-based. The bounded implemented `tkrEditService` is described in Section 7.6.9; its future generic transaction contract below remains proposed. Figure D.7 is a separately labelled planned activity. Collaborator ownership and provider role boundaries were reviewed in October; November provider decisions are controlled in Section 7.9; detailed mechanisms and preview scheduling remain open.

7.8.1 Proposed Data Records

These are proposed representations of the requirements, not C++ declarations or a frozen plugin ABI. `REQ-D-008` defines proposal contents; `REQ-F-012` and `REQ-D-004`–`REQ-D-007` define restoration, revision and saved-state behavior. Exact before/after snapshots with an operation description are the proposed initial history storage, now used by the bounded implementation without automatic eviction. Memory limits and optimization await measurements. The current `CoordinateEdit` remains unchanged in code.

Table 7.6: Planned Transaction Records

Record	Planned contents and contract
`SourceEdit`	Original byte range, expected original bytes, replacement bytes. Supports insertion/deletion/replacement; carries no graphics-item pointer.
`EditTransaction`	Document identity, base revision, operation description and a collection of source edits, all using the same original source revision.
History entry	Exact before/after source bytes and operation description. The entry's history position is distinct from the monotonically advancing document revision and the saved-source baseline.

The previous team's TikZiT-based MGB-UML source provides a core parser plus plugins for palette metadata, Qt node drawing and TikZ output, with an SDK and starter template [5]. Its graph/export approach is a reference, not evidence that the source-preserving contract is implemented. The reviewed interface has no syntax-provider parsing hook. Source inspection did not include building or running that application; no reference code was imported into TikZRefinery.

7.8.2 Planned Class Collaboration

The October 27–30 review settled the planned ownership: the main window continues to own document/controller; the document owns history; the controller owns the edit service and file store. Owning services supports coordination, not coordinate arithmetic, history bookkeeping or filesystem logic in the controller. The bounded edit service and document-owned history exist; file-store extraction and the generic transaction extension remain unimplemented. Current source-based diagrams distinguish thirteen host classes (ten core plus three sidebar UI classes) from the provider interface/module; see Figures 7.4 and 7.25.

The document calls history methods to record changes and retrieve restoration targets. Candidate source and history storage must be prepared before either is published; all required state changes succeed together. History supplies restoration bytes and advances its position only with successful restoration. The file store returns read/write results to the controller, which requests document load or saved-baseline updates only after successful I/O. Requirements REQ-A-012–REQ-A-014 and REQ-D-002 control these boundaries.

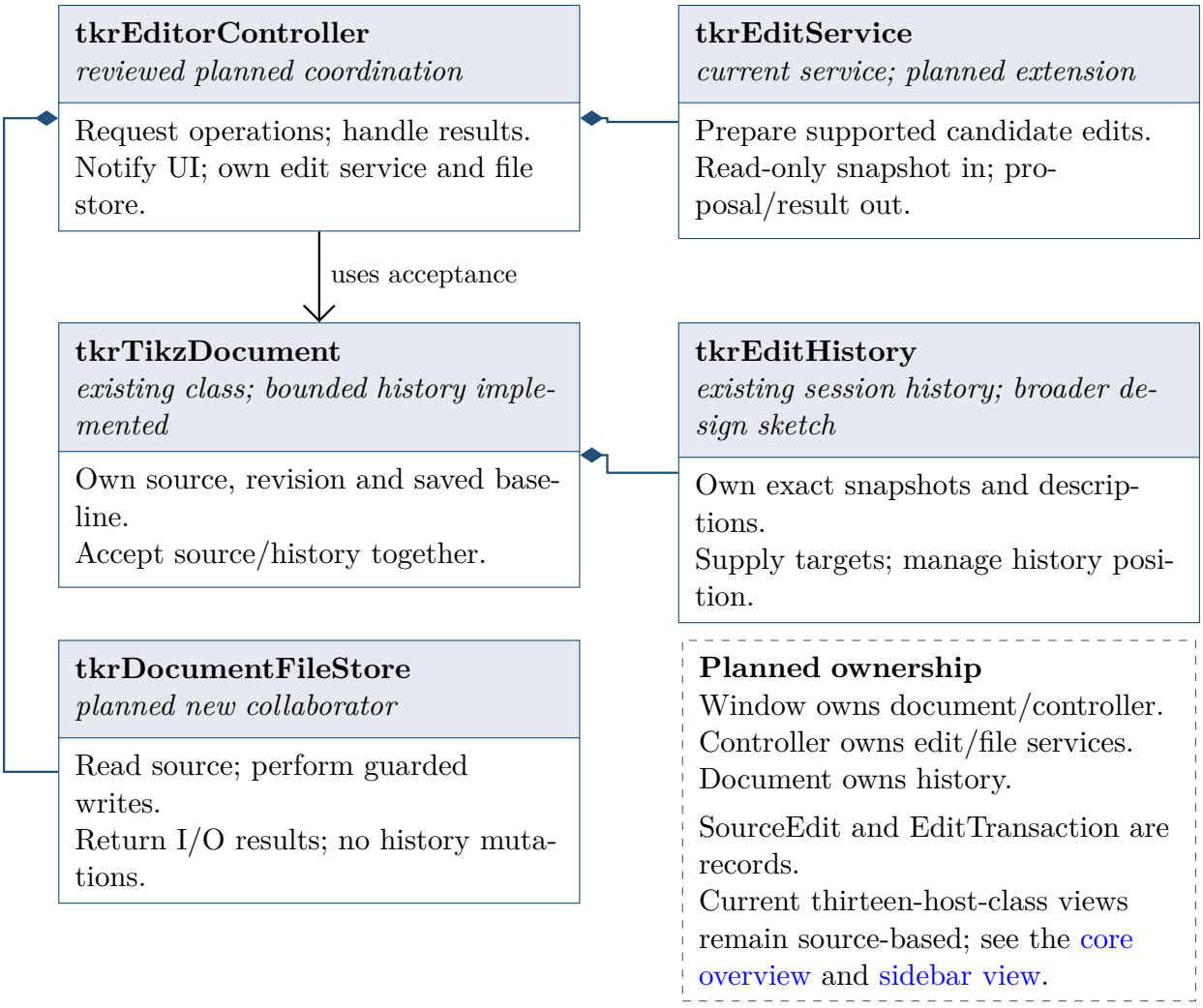


Figure 7.16: Planned transaction collaborators. Explicit labels identify planned roles; diamonds show reviewed planned ownership. Current source classes remain as shown in Figure 7.4.

7.8.3 Responsibility and Design Review

Vincent requested periodic whole-design review and asked that findings from the separate chat *Research class size guidelines* inform it. Prefer one coherent responsibility; accept closely related work when justified and review independent growth before adding another concern. A diagram's bullet count is not a responsibility count: origin/scale validation and forward/inverse conversion all support coordinate mapping. Martin's explanation groups work by reasons for change [4].

That investigation proposed a practical target below 300 code lines, review at 500, and refactoring or an explicit exception at 1,000. These remain proposed project review thresholds, not universal limits or approved hard coding rules. Count nonblank, non-comment-only lines across the header/implementation for a class and also inspect supporting helpers; monitor physical file length separately. The earlier observed controller/scene pairs contained 369/308 physical lines including comments, blanks and helpers; these dated totals are not current code-line counts after the history increment. The controller merits review because persistence, edit planning and UI coordination have different reasons for changing, independent of its length.

Table 7.7: Responsibility Review Directions

Component	Review direction
Controller	Coordinate requests and notifications; delegate editing/history and persistence through explicit collaborators.
Document and history	Keep authoritative source/revision/saved state separate from history bookkeeping; publish accepted changes consistently.
Scene and mapper	Retain scene interaction and one coordinate mapping. Review extraction of rendering as plugin geometry grows.
Plugin infrastructure	Separate module loading/lifetime from provider registration/selection as needed; do not grow one manager into all editor roles.

Completed review now covers transaction collaborators, label/source-binding separation, attachment/local edits, overlap selection and provider boundaries. Reviewed registration, operation and lifecycle policy is now in Section 7.9. Detailed syntax ambiguity, loader wiring and preview scheduling remain subsequent implementation review. Update requirements, tests, diagrams and traceability as decisions settle. Existing source-based diagrams remain current until refactoring is implemented. `TC-034`–`TC-043` are planned checks, not new results.

7.8.4 Reviewed Operation Contracts

Names below are interface sketches, not current C++ declarations or a frozen ABI. All proposal ranges refer to one original revision. REQ-I-006 distinguishes Applied, Unchanged and Rejected; rejection carries a reason. Cancellation is intentional and does not create an error/history entry.

Table 7.8: Reviewed Planned Operations

Component	Operation sketch	Contract
`tkrEditService`	`prepareEdit (snapshot, request)`	Read-only snapshot and semantic intent in document coordinates; prepare candidate and supported-operation checks; return transaction or rejection. No live model/UI mutation.
`tkrTikzDocument`	`applyTransaction (transaction)`	Final identity/revision/range/byte checks; atomically publish accepted source and history. No-op preserves history and redo.
`tkrTikzDocument`	`undo(), redo()`	Retrieve history target, prepare restoration and coordinate source/position update. Advance revision; do not append another edit entry.
`tkrTikzDocument`	`loadSource(), markSaved(snapshot)`	Successful replacement resets history. Successful save records the actual written bytes as baseline and retains history. A preceding explicit successful Save remains effective if Open later fails.
`tkrEditHistory`	Record change; retrieve target; complete restoration; reset	Own exact before/after bytes, descriptions and cursor in memory. Calls that mutate history are coordinated through the document; no scene/file access.
`tkrDocumentFileStore`	`read(path), write(path, bytes, baseline)`	Read source or perform guarded write; return bytes/results to controller. External-change comparison retains its concurrent-write race limitation. No history logic.

7.8.5 Planned Collaborator Class Details

The diagram shows planned extensions of the existing document and edit service, plus two new collaborators: history and the extracted file store. Arrows in Figure 7.16 give their dependencies; these are reviewed future contracts, not claims of refactoring already performed.

tkrTikzDocument
existing class; bounded history implemented

Own source, revision, saved baseline and history.
Prepare acceptance before publishing source/history together.

+ applyTransaction(transaction): result
+ undo(), redo(): result
+ loadSource(...); markSaved(writtenSnapshot)
Restore exact bytes; advance revision; append no restoration entry.

tkrEditService
existing class; planned generic extension

Read-only snapshot and requested intent.
Provider supplies geometry; host/syntax route prepares byte edits.
No live source, history, file or widget mutation.

+ prepareEdit(snapshot, request): proposalResult
Prepare candidate and supported-operation checks.

tkrEditHistory
existing session history; broader design sketch

- entries: exact before/after bytes and descriptions
- position: current history cursor

+ prepare entry; retrieve undo/redo target
+ complete accepted change/restoration; reset
+ query availability/description
Document coordinates publication; failure moves no cursor.

tkrDocumentFileStore
planned new class; owned by controller

Path, source bytes and expected disk baseline.
No edit planning or history state.

+ read(path): readResult
+ write(path, bytes, baseline): writeResult
Successful I/O permits document load/baseline update.
Save keeps history; successful document replacement clears it.

Reviewed design; bounded history exists; file store/generic transactions remain planned.
No automatic eviction; future limits and provider-dependent retained work remain open.
Exact C++ signatures and construction mechanisms require implementation review.

Figure 7.17: Reviewed planned document, service, history and file-store contracts. Visibility is UML-style; exact C++ signatures and storage mechanisms remain implementation details.

7.8.6 Elements, Source Bindings and Attachments

The current two-point `ElementRecord` remains unchanged in code. The reviewed design separates information needed for display from source rewriting and per-binding operations (REQ-D-009–REQ-D-012). Record names below are sketches. Interpreted positions use document coordinates; screen/zoom coordinates remain a presentation concern. IDs and host references are valid only in their document revision and are rebuilt after acceptance.

Table 7.9: Reviewed Shared Element Information

Record	Reviewed information boundary
`ElementDescription`	Document/revision/element identity, element type, interpreted geometry, relevant text and appearance. Type/schema versions must be documented; exact schema remains open.
`SourceBinding`	Responsible syntax-provider identity, original byte ranges and provider information needed for edits; no graphics-item pointer. Equivalent display may have different bindings.
`AttachmentRecord`	Host reference and relative placement rule, including supported along-host position and offset. Calculated canvas position is derived; attachment is not C++ ownership.
`PresentationDescription`	Drawing description, selection geometry and readable element description; scene creates/owns graphics items. Selection bounds are not editable coordinate pairs.
`AvailableOperations`	Per-element/binding supported intents and constraints; display support does not imply edit support. Label movement requires a supported syntax route.

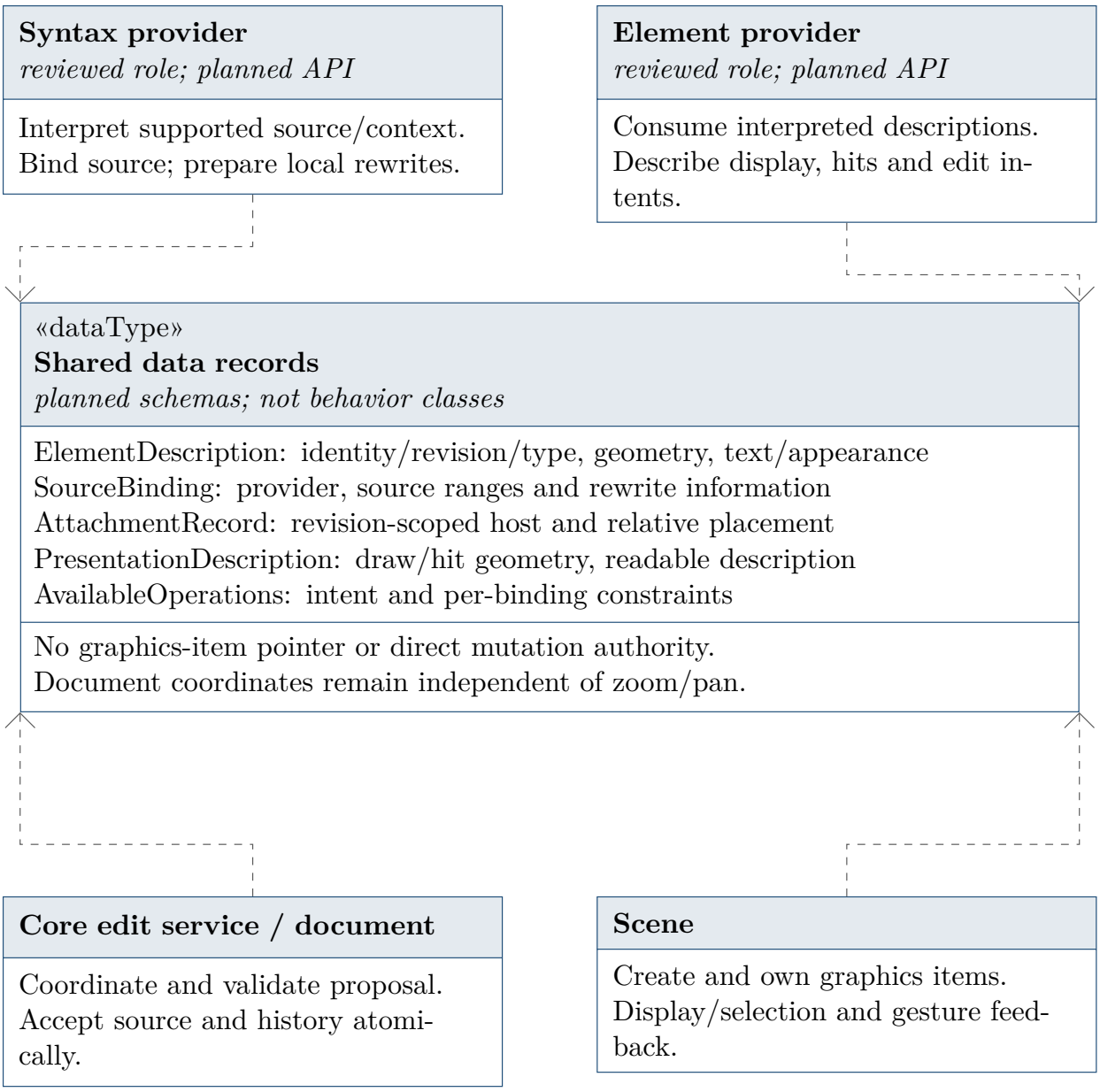


Figure 7.18: Planned record and provider boundaries. Provider boxes describe reviewed roles rather than additional implemented classes. Records contain data; document acceptance remains authoritative.

7.8.7 Reviewed Label and Selection Interaction

UC-005 and UC-006 define the author goals; their requirements and planned checks are authoritative. A label can move along and away from its host while preserving attachment and unrequested text, style, rotation and anchor/alignment. The host geometry is unchanged during label movement; moving the host carries the label. A local edit preserves shared coordinates/styles/macros; if no supported independent adjustment exists, movement is unavailable with a reason. Detachment is explicit later work, requiring a supported structural rewrite in one transaction.

Normal hit selection prioritizes the label. A direct right-click lists all selection geometries intersecting a 6-logical-screen-pixel radius circle, including obscured hosts. This is an initial user-review setting, not a measured usability threshold. The radius stays constant on screen under zoom. Readable menu entries highlight their candidates. An empty-canvas left drag draws a retained rectangle; the Select area tool permits starting over objects. Right-clicking inside lists partial/full intersections; choosing an entry selects one object. Region creation never edits source or automatically selects a group; Escape clears it. Selection of an obscured host must survive the subsequent drag-start hit test rather than being replaced by the top label.

One rejected completed edit shows a nonmodal bottom strip with bright red icon, red accent and readable reason, until dismissed or the next completed operation. Pointer previews produce no repeated rejection messages. Applied edits are visible; unchanged/cancelled gestures create no error or history entry.

7.8.8 Reviewed Provider Boundary

REQ-A-015–REQ-A-017 control the shape/syntax distinction, stable routing and capability intersection. Shape providers return interpreted proposals; syntax providers retain original source bindings and prepare rewrites. The host validates and the document accepts. The scene creates/owns graphics items. Section 7.9 defines the reviewed data/operation contracts and lifecycle policy; detailed mechanisms still appear in Chapter 8.

7.8.9 Rectangle Resize and Shape Plugin

On November 3, Vincent reported the instructor's direction: implement scaling for the existing box, then turn the box into a shape plugin. For this increment, scaling means changing rectangle dimensions in the source. Canvas zoom changes only the view. REQ-F-025 and REQ-A-018 record the planned behavior; TC-060 and TC-061 define its baseline and plugin checks.

Vincent approved corner resizing on November 5 and refined crossing, minima and boundary handling on November 12. The implemented baseline preserves the opposite corner, allows crossing, retains independent 0.01 cm dimensions and clamps to the furthest stroke-aware valid geometry. Exact coincidence and insufficient crossing space retain the nearest feasible side. Preview is temporary; release applies one guarded batch. See REQ-F-025, REQ-I-013 and TC-060 for the controlled rules. Aspect-ratio locking and edge handles remain deferred.

Table 7.10: First Shape Plugin Design Decisions

Concern	Decision
Order of work	Establish resize behavior on the current box fixture. Extract the rectangle provider and verify the same display, move, resize and save/reopen behavior with the separately loaded plugin. Keep the explicit-coordinate parser for this bounded increment.
Shape API	One versioned interface for shape geometry, presentation and supported editing operations. A provider can compute resize geometry and propose a change; source bindings still determine valid rewrites, and the document remains the final authority over source and history.
Strategy	The host delegates shape-specific behavior through the common interface. The rectangle provider can supply those operations directly; different shapes or resize behaviors can supply other implementations [11].
Abstract Factory	Optional inside a provider that needs to construct matching components. A provider can create and reuse a renderer, hit tester and resize handler internally without requiring factory calls in the editor API [2].
Interaction cost	Reuse providers and their components across gestures. Pointer updates change temporary geometry; source edit preparation and validation occur on release. Measure responsiveness in the working prototype.
Loading	Qt's application-plugin interface and `QPluginLoader` are the candidate loading mechanism [10]. The reviewed compatibility policy is in Section 7.9.6; exact ABI, build wiring and any TikZ-only import format remain open.

Resize evidence is scoped in Section 10.12. The bounded rectangle plugin is loaded in the current package; broader provider management and product acceptance remain pending.

7.9 Reviewed Shape Plugin Contract

The November review defines the wider contract. The first native rectangle API is now implemented in Section 7.7; candidate-choice, updating and creation obligations remain planned. Current host classes remain in Section 7.5. This section adapts the course's linked architecture/component pattern to UML [6]; it does not replace current-code evidence with future classes.

Table 7.11: Plugin Delivery Boundaries

Delivery boundary	Meaning
Implemented	Explicit rectangles/lines/default arrows; move, rectangle resize, constraints, cancellation, fitted canvas and guarded source batches; separate rectangle provider for display/selection/move/resize.
Next milestone	Shape discovery and host-controlled creation, targeted at 0.2.0-alpha after detailed interaction review.
Later implementation	Automatic updates, migrations, adapters, generic transaction/history extensions and integrated compiler/PDF viewing.

7.9.1 UML and Diagram Convention

The author uses applicable UML structure and behavior notation with editable TikZ sources. Orthogonal routing is the project layout convention, not a new UML relationship: horizontal/vertical segments and elbows replace diagonals. Aligned links and sequence messages remain straight. Association, dependency, realization, generalization and ownership retain their standard meanings. Explicit planned labels distinguish future elements; dashed lines retain their relationship meaning. This convention applies to current and planned active views, not historical screenshots or submitted decks.

7.9.2 Responsibilities and Module Boundary

REQ-A-015 separates shape providers from syntax support. The host owns registration, binding/routing and acceptance. Module lifetime management and provider registration are distinct responsibilities; the diagram does not require a single manager to perform both or freeze new C++ classes. The application must retain loaded implementations while dependencies exist.

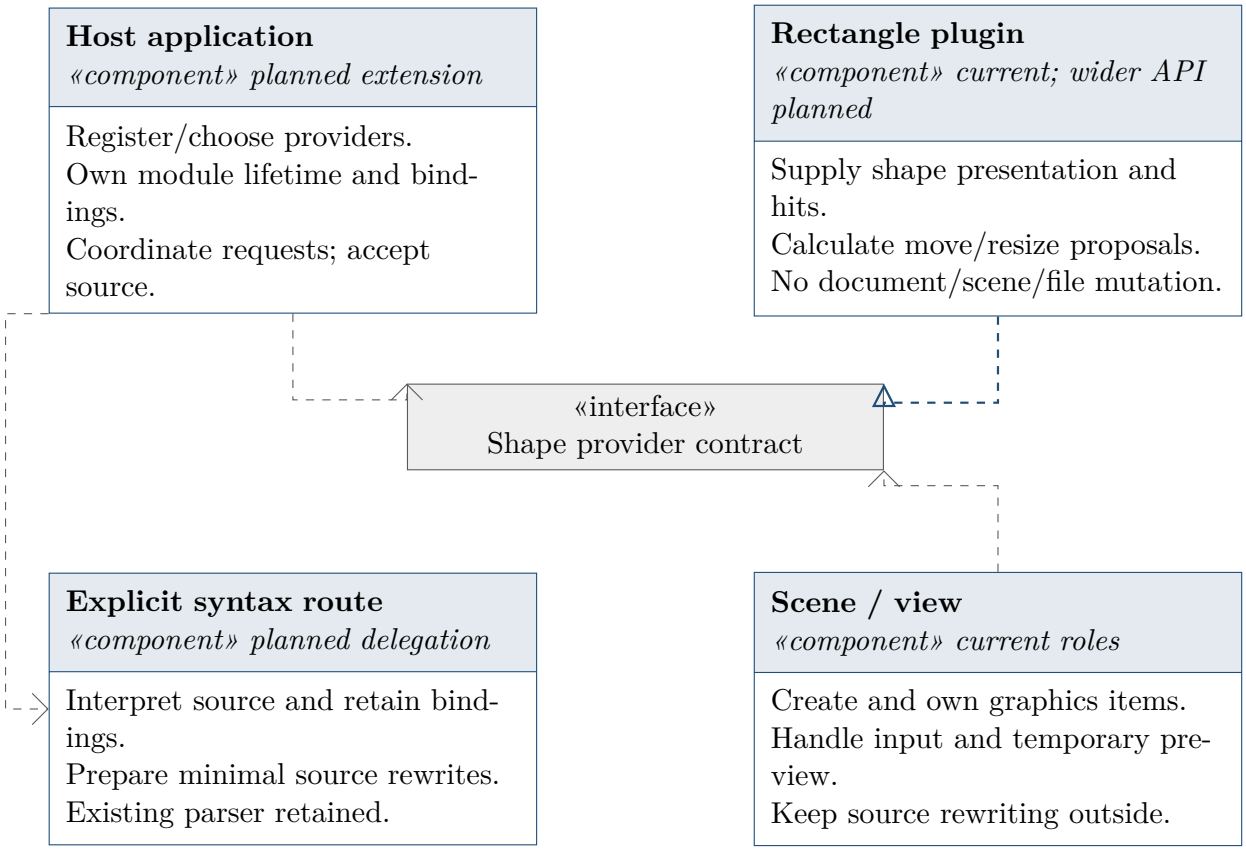


Figure 7.19: Planned UML component/interface boundary. The rectangle module already exists; wider registration and syntax contracts remain planned. Current deployment is Figure 7.1.

7.9.3 Versioned Data and Operation Records

REQ-D-009–REQ-D-017 separate interpreted meaning from source spelling. Records are in memory; saved TikZ remains authoritative. The first rectangle uses two opposite corners in document centimetres. Stroke width is appearance, not another coordinate pair. Other types may document different geometry schemas. Providers borrow inputs for calculation rather than own the document's records.

Table 7.12: Reviewed Plugin Record Sketches

Record sketch	Reviewed contents
ElementDescription	Document/element identity and revision; family, specific type and schema; typed geometry, appearance and relevant content.
ShapeOperationRequest	Operation ID; parameter-schema version; target and initial description; typed parameters and document constraints. Move supplies total displacement; resize supplies corner identity and total displacement.
PreparationResult	Prepared change with interpreted changes; Unchanged; or Cannot prepare with reason. This result never means source was applied.
SourceBinding	Host/syntax-owned revision-scoped source ranges and rewrite information. The shape API receives no original byte ranges or widgets.
PresentationDescription	Drawing/selection geometry, readable description and handle identity/positions. Rectangle API v1 is implemented; additional primitive/handle schemas remain open.

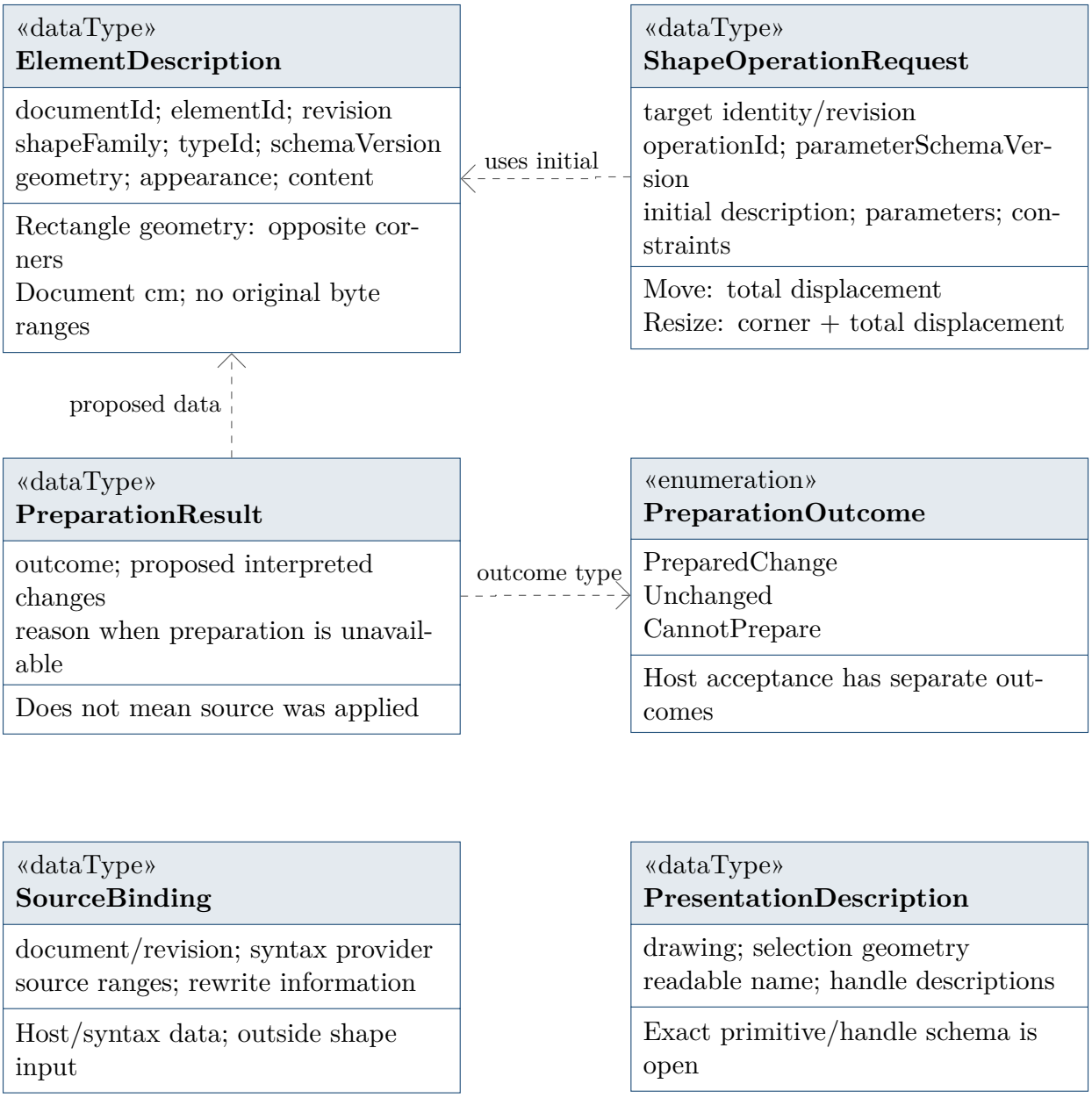


Figure 7.20: Planned UML data types and preparation outcome enumeration. Attributes describe contracts, not a frozen binary layout.

7.9.4 Provider Interface and Host Operations

Common operation IDs `tkr.move` and `tkr.resize` belong to the application. Custom operation IDs are qualified by the provider's stable identity; parameter data and parameter schema versions are separate. Unknown/invalid schemas cannot be guessed. Availability is the intersection of shape support, syntax/binding support and document context, checked both for controls and at execution (REQ-A-017).

The shape calculates proposed geometry under the supplied bounds and its own valid shape rules. Each preview and release uses initial geometry plus total displacement, preserving grab offset and avoiding accumulated drift (REQ-A-023). The host prepares source rewrites, validates the candidate and asks the document to accept. Generic source/history atomicity remains the separately planned transaction contract; the current implementation publishes prepared coordinate batches and session history atomically.

Requested rename: `tkrShapeProviderRectangle`.

Actual declaration: `tkrRectangleShapeProvider`.

Vincent reported Professor Darian's suffix convention. The coordinated rename remains pending; Figure 7.14 follows source. No class, DLL or plugin identity is renamed here.

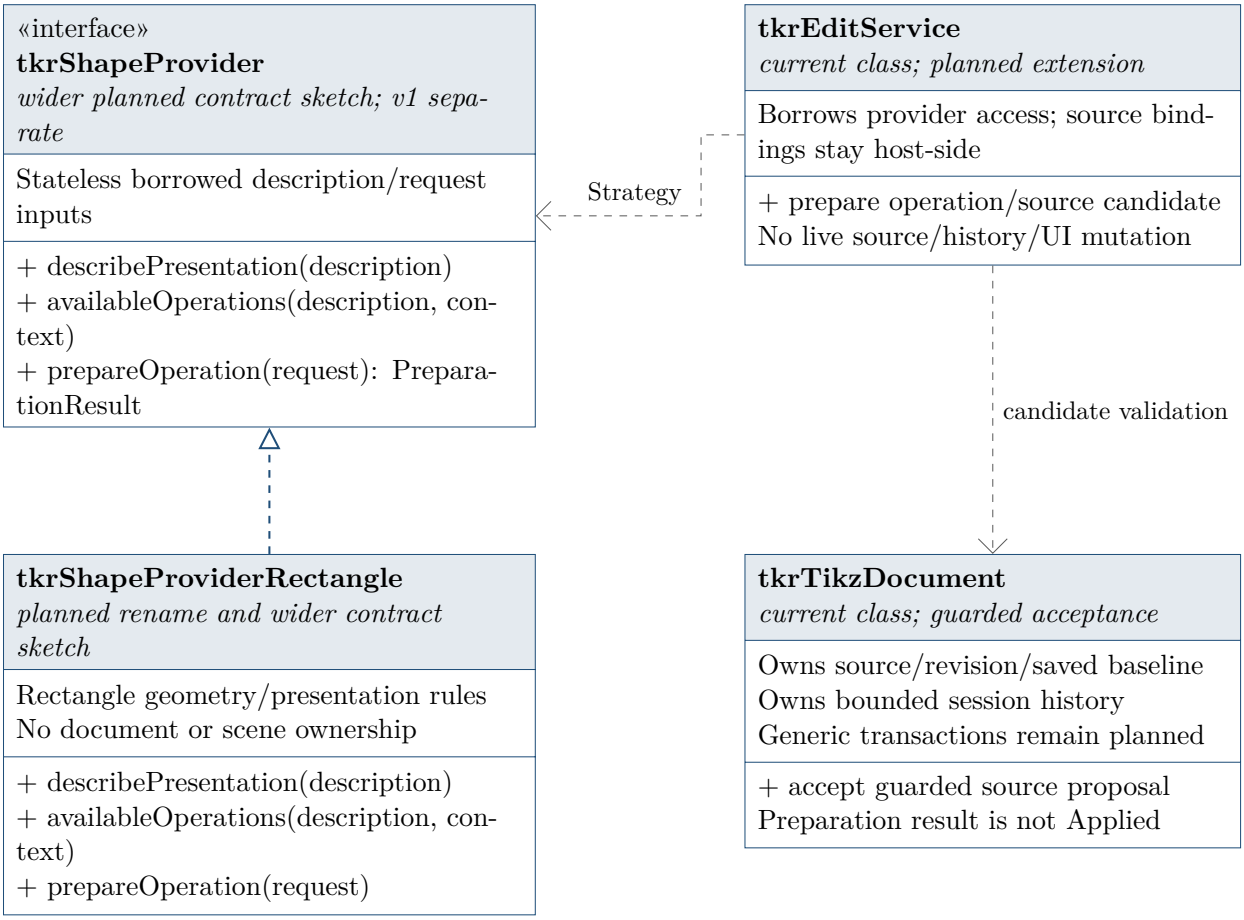


Figure 7.21: Wider planned UML interface realization and host collaboration; implemented API v1 is in Figure 7.14. Operation names and class names are sketches; current classes are not silently replaced. Strategy delegates shape behavior; an internal Abstract Factory is optional.

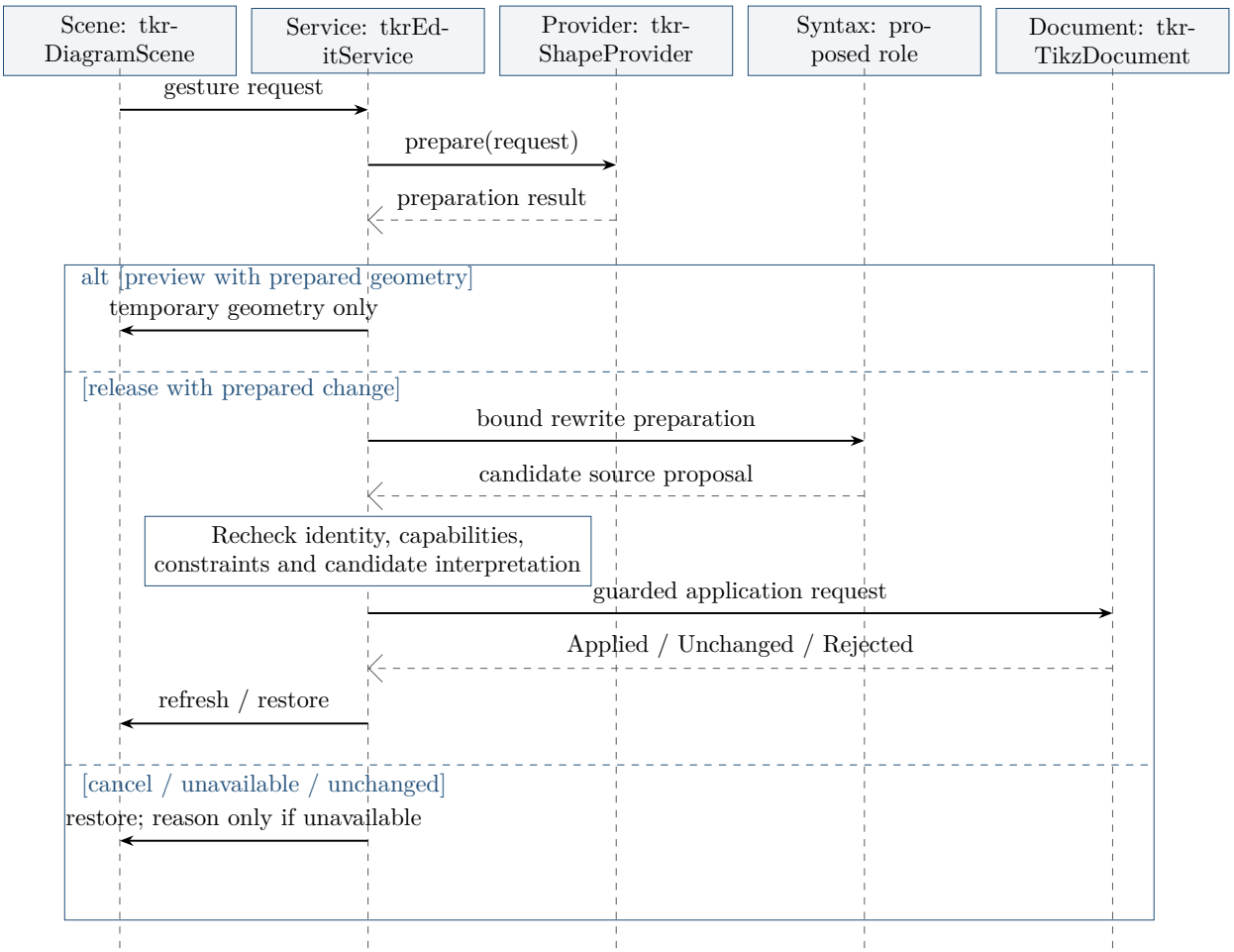


Figure 7.22: Planned shape preparation and host acceptance sequence. Applied is a host outcome; it is never a provider's preparation result.

Cancellation discards temporary geometry; selection and Unchanged create no accepted edit. Cannot prepare and later host rejection preserve authoritative source. Prepared changes may still fail stale-revision, source-binding or candidate checks. Later history behavior follows Figure D.8, not an implemented undo stack.

7.9.5 Planned MVC Allocation

Current allocation remains Figure 7.3. The planned extension keeps presentation and input in View, coordination in Controller, accepted source in Model, and shape/syntax calculation in services/providers. Owning a service does not turn the controller into its implementation. The current fixed-candidate store is scene-owned. The reviewed window-owned shared store/catalogue and sidebar are planned in Section 7.10.2; wider registry and update wiring also remain planned. No additional persistent document owner is introduced.

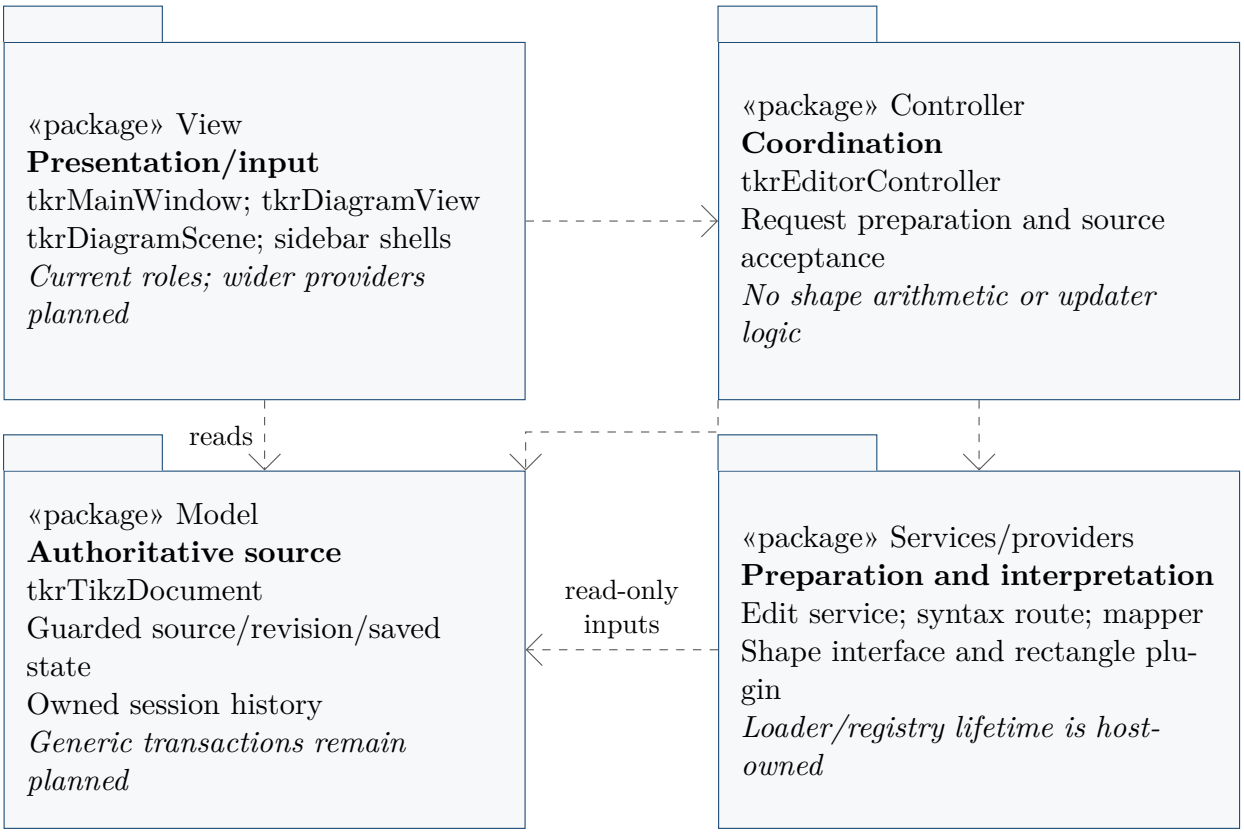


Figure 7.23: Wider planned UML provider allocation; the implemented fixed-candidate rectangle route appears in Figure 7.3. Palette, creation and updater interfaces remain later work.

7.9.6 Registration, Compatibility and Lifetime

UC-007 covers explicit provider choice; UC-008 covers updates. Registration metadata and runtime identity follow REQ-A-019 and REQ-D-016. Different providers for one family may coexist. Duplicate stable IDs require candidate choice; neither discovery order nor a changed default replaces an element's established source binding or shape-provider binding.

Compatibility checks API, interpreted data and used capabilities. A new release must support old data or a defined validated conversion without dropping content; otherwise keep the prior version. Reopening may reconstruct compatible records from source, but is not evidence that an arbitrary conversion is possible. Adapters and migration algorithms remain future work.

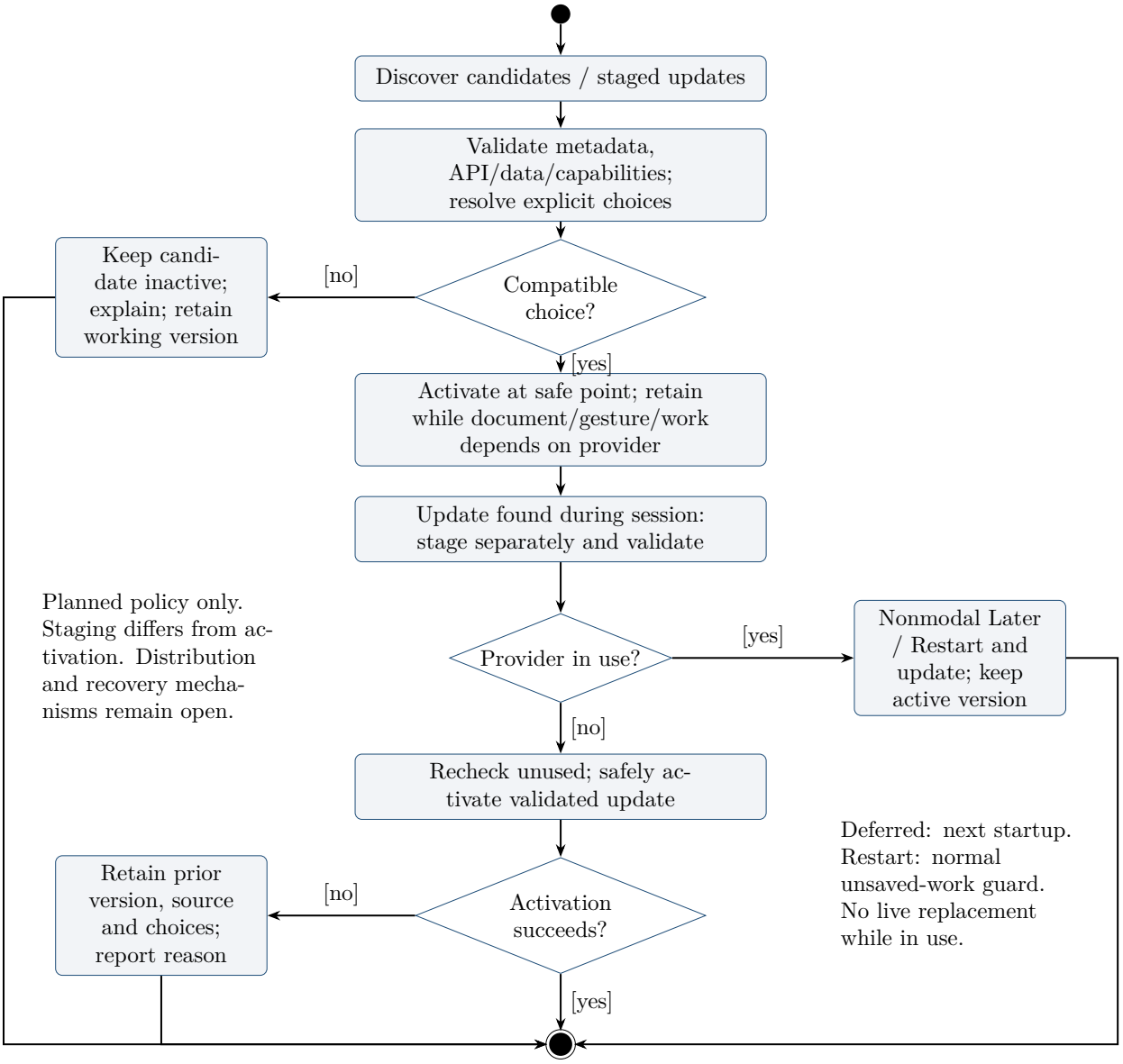


Figure 7.24: Planned UML lifecycle activity with guarded activation and separate update staging. Distribution/recovery mechanisms remain open.

7.9.7 Later Palette, Creation and Compiled Fallback

REQ-F-033, REQ-F-034 and REQ-I-015 record discovery and creation as later product scope, not prerequisites of extraction. A collapsible Shapes panel must keep its reopen control available and distinguish provider alternatives. Browsing changes no source and selecting an entry does not replace an existing element's provider. Creation needs a shape-supported proposal and a syntax-supported insertion; host validation/document acceptance precede new element identity. Click/drag placement, one-placement completion and sidebar roles are reviewed in Section 7.10; New/Save As, insertion contexts and exact API mechanisms remain unresolved.

Missing provider support preserves source and explains unavailable edits. REQ-I-017 permits a future successful read-only compiled view, provided LaTeX dependencies exist and its source revision/stale status is clear. PDF visibility does not restore graphical editing. Unknown syntax still rejects the current interpretation as a whole; no partial-editing implementation is claimed by these future requirements.

7.9.8 First Extraction and Evidence Gate

REQ-A-018, TC-061 and VER-074 define the implemented bounded delivery in Section 7.7. Retain the explicit parser, coordinate mapping, exact source replacement and document acceptance. Move rectangle drawing, selection/handle descriptions and move/resize calculation behind the shape API; keep lines/arrows working through their existing route. Verify actual module delegation, no hidden built-in rectangle fallback, missing and incompatible module diagnostics, geometry parity, crossing/minima/bounds, cancellation, source bytes and save/reopen. Executed rectangle checks are in Section 10.13; broader provider scenarios remain planned.

Use a separate development package. The frozen `0.10.3` package and edited examples remain untouched. The rectangle extraction baseline is `0.1.2-alpha`, with hash-free folder/title and About identity per REQ-DEV-006–REQ-DEV-008. The later `0.2.0-alpha` milestone is shape discovery and creation. Future ABI, registry and presentation-extension mechanisms remain open; reviewed placement and sidebar ownership appear in Section 7.10.

7.10 Reviewed Sidebar and Creation Design

The November 21–22 discussion supplied the reviewed design. On November 23, Vincent authorized the bounded `0.1.3-alpha` sidebar shell below. A shape catalogue supplies offered types; the workspace sidebar hosts their presentation. A placement anchor positions default geometry. Catalogue/creation and shared-service ownership remain planned; the shell does not implement them. The older `0.1.2-alpha` package is preserved and the scene still owns its plugin store. This section adapts the report's linked UML and interface views; obligations remain in the registers rather than a second narrative contract.

7.10.1 Implemented Sidebar Shell

`tkrWorkspaceSidebar` provides a persistent Shapes/Files icon strip, tab selection, an explicit collapse button and click-selected-tab collapse/reopen. It retains the selected page and expanded splitter width during the window session. Icons come from Qt's standard style; no external icon/theme package is adopted. `tkrShapePalette` displays the currently loaded rectangle provider summary and a disabled Create rectangle button with an explanation. This is fixed-provider status, not a new provider-enumeration or creation API. `tkrFolderBrowser` is a placeholder page with Open document; its request reuses the same Save/Discard/Cancel guard and dialog as File > Open.

The main window owns the sidebar/pages through the central widget's Qt parent tree. The sidebar owns its content and page widgets; it copies provider summary text. It holds no source bindings and cannot edit a document. Current scene-owned plugin services remain separate from the planned shared ownership below. TC-079 and TC-082 have bounded shell evidence in Section 10.15; catalogue alternatives, shared services, creation and stakeholder validation remain unexecuted.

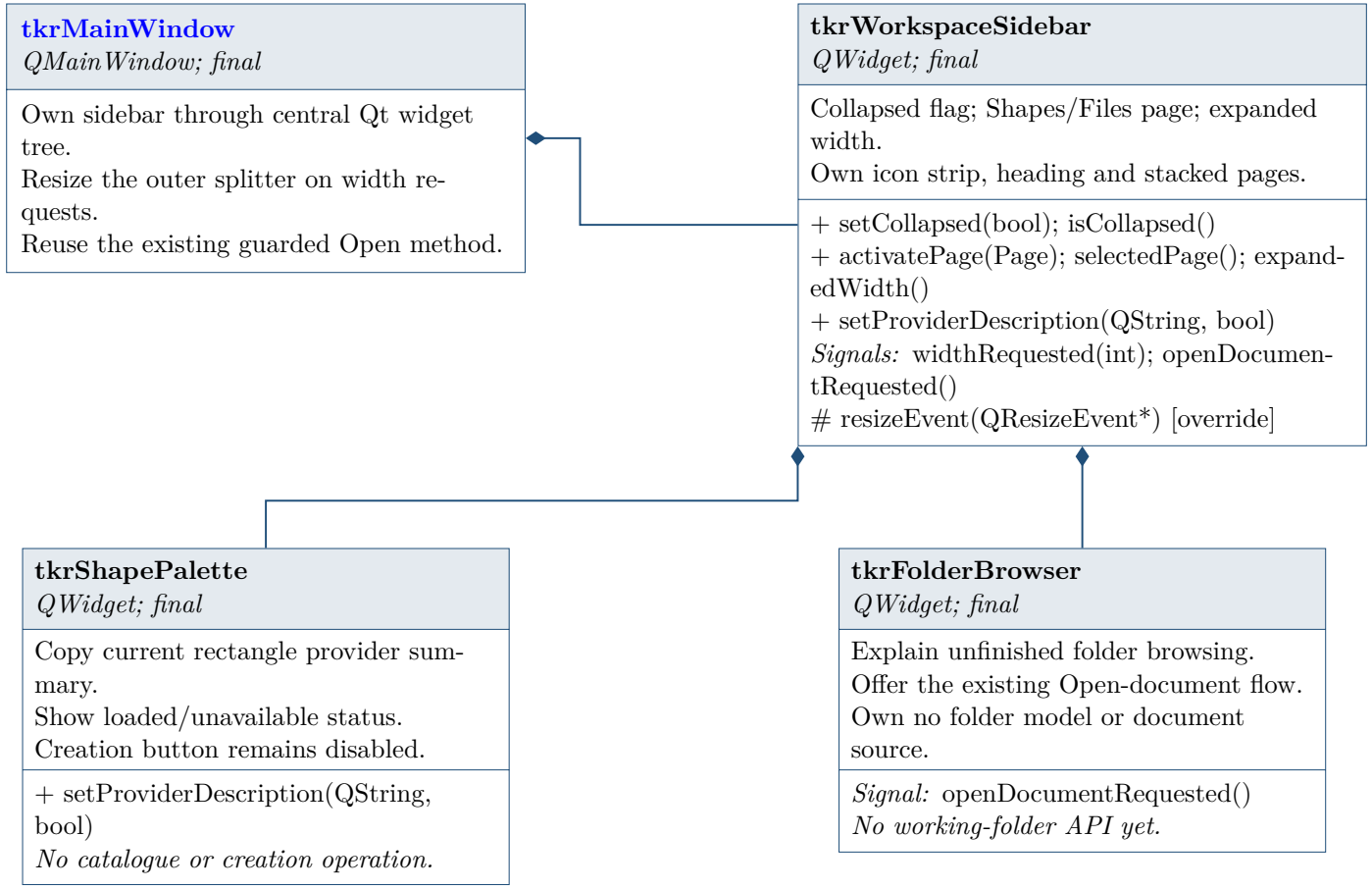


Figure 7.25: Implemented sidebar shell: current responsibilities, public operation names and Qt ownership. Names summarize source signatures; constructor defaults and types remain in the headers.

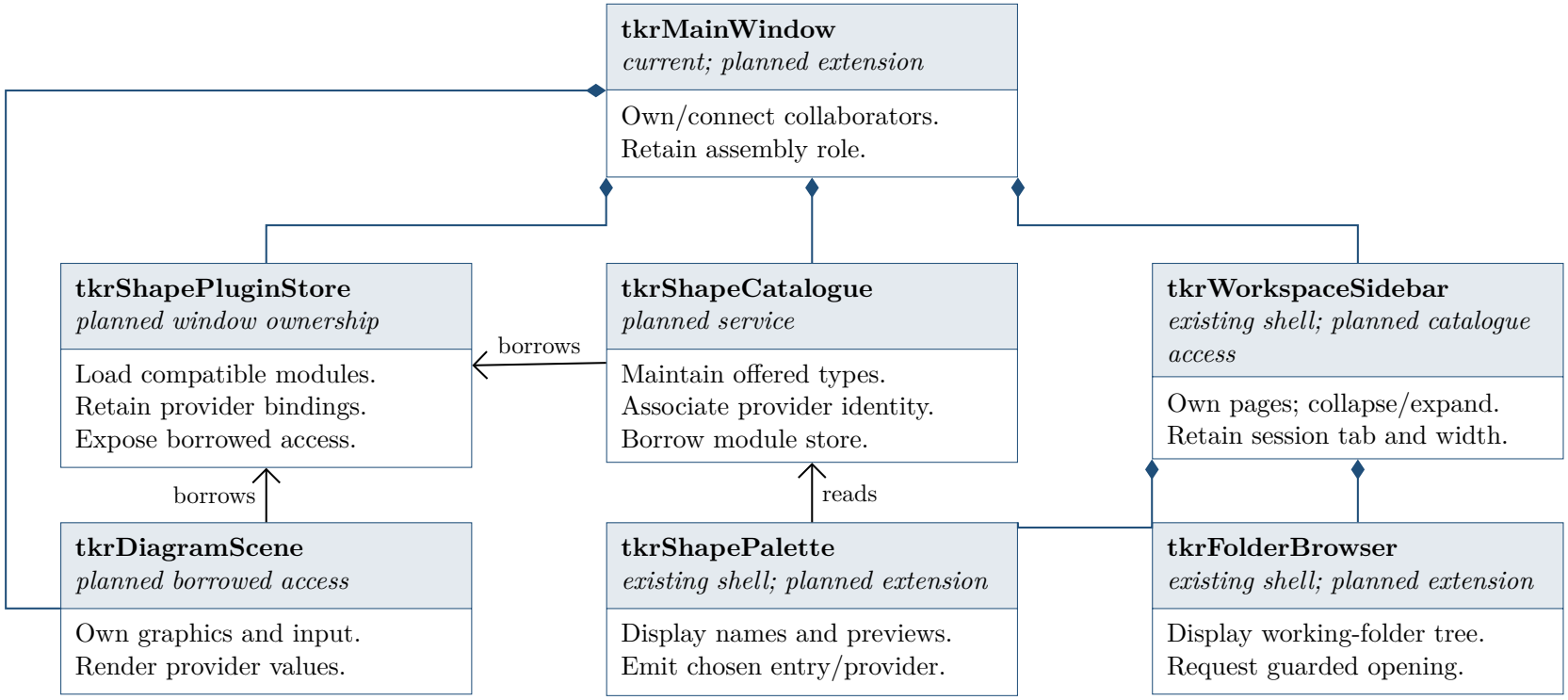
Table 7.13: Sidebar and Creation Decisions

Concern	Reviewed direction / boundary	Controlled trace
Catalogue	Describe offered shape types separately from existing instances; associate provider identity/version in the host.	REQ-F-033; REQ-D-018
Sidebar	Shapes and planned Files pages; distinct icons and readable labels, persistent collapsed strip and retained session page/width.	REQ-I-015; REQ-I-018–REQ-I-020
Placement	Click for provider defaults; rectangle drag defines opposite corners. Centre a clicked rectangle where valid; shift inward to fit without changing size. If the default cannot fit, explain unavailability and preserve source.	REQ-F-035; REQ-I-013
Tool completion	One accepted placement selects the new element and returns to selection. Escape cancels; a rejected attempt reports once and retains the tool.	REQ-I-021; REQ-I-006; REQ-I-007
Ownership	Main window assembles/owns shared host services and sidebar; bounded collaborators perform their work. Consumers borrow and finish first.	REQ-A-024; REQ-A-021
Styling	Retain default Qt styling; Qlementine is deferred. No external theme or icon library is adopted. The shell uses Qt standard-style icons; final icon selection remains open.	planned UI views

Provider defaults and shape minima remain named, unit-bearing values. A 2 cm × 1 cm rectangle is a proposed starting default, not a permanently accepted size or implemented creation feature. Existing independent 0.01 cm rectangle minima are the resize baseline; creation must honour the provider's reviewed valid geometry and document constraints. UC-009 and TC-080 retain insertion/New prerequisites.

7.10.2 Planned Ownership and Collaborators

The shared-service/catalogue extension and future APIs below remain planned. The three UI class names already have limited shell implementations in Figure 7.25; these broader collaborators are not their current declarations. Current code has a scene-owned store (Figure 7.14). Composition indicates ownership; associations/dependencies indicate borrowed access or use.



Lifetime constraint: release dependent work and views, then catalogue, then store.
Existing document/edit-service collaboration is omitted here; exact wiring remains open.

Figure 7.26: Planned UML sidebar/shared-service composition and borrowed access. Window ownership does not transfer catalogue, module or view work into the window.

7.10.3 Proposed Responsibilities and Method Roles

This table describes the broader planned roles and APIs, not a frozen ABI. Only the sidebar collapse/page operations already exist; current palette status and Files Open-request operations are documented in the implemented shell view.

Table 7.14: Planned Sidebar Collaborator Roles

Class	Bounded responsibility/state	Operation / signal roles
`tkrWorkspace Sidebar`	Own page widgets; track collapsed state, selected page and expanded width for this session.	`setCollapsed(...)`; `activatePage(...)`.
`tkrShapePalette`	Present catalogue values, selection and availability; emit chosen shape/provider identity.	`setEntries(...)`; `shapeChosen(...)` signal.
`tkrFolderBrowser`	Present the working-folder tree using a Qt model/view; request opening through the shared guard. Root/filter/activation rules are open.	`setRootFolder(...)`; `openFileRequested(...)` signal.
`tkrShapeCatalogue`	Maintain offered-type entries and host provider associations; borrow the module store. No document-instance ownership.	`entries()`; `refresh(...)` roles; provider enumeration extension remains open.
`tkrMainWindow`	Assemble/own and connect requests. Existing document and controller responsibilities remain bounded.	Proposed shared `requestOpenDocument(...)` routes dialog and Files requests through Save/Discard/Cancel before controller opening.

Pending dependent work and consuming views must be released before their host services; the catalogue must finish using the store before the store is released. Document/provider dependencies in REQ-A-021 still apply. Shell Qt-parent construction and teardown are implemented and tested; concrete shared-service/catalogue wiring still requires implementation review. Shared window ownership does not authorize hot unload or an updater.

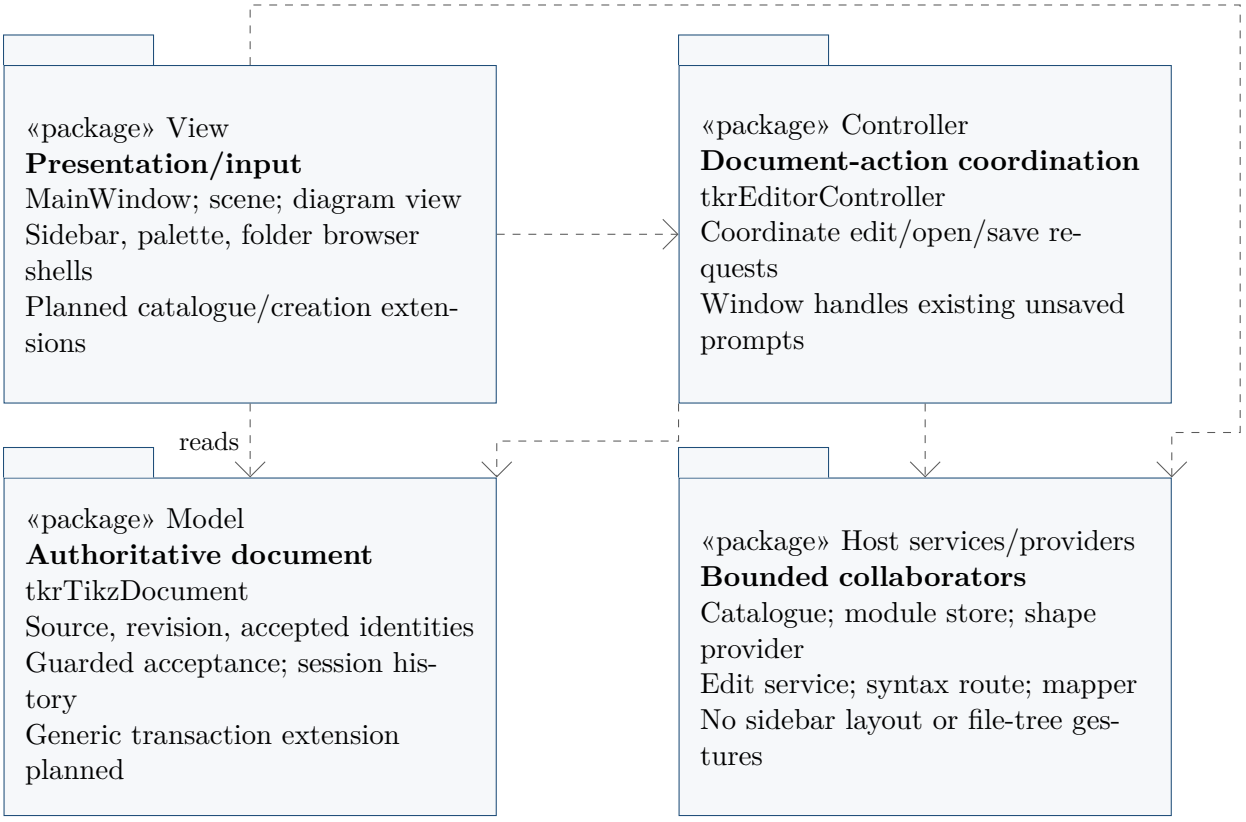


Figure 7.27: Planned UML package allocation for sidebar and shared services. MVC role and object ownership are separate views.

7.10.4 Catalogue Types and Instance Boundary

REQ-D-018 separates a type offered for creation from an accepted document element. Host provider identity includes declared ID/version and a runtime binding under the existing identity contract. A catalogue record has no accepted document-element identity or authoritative source ranges.

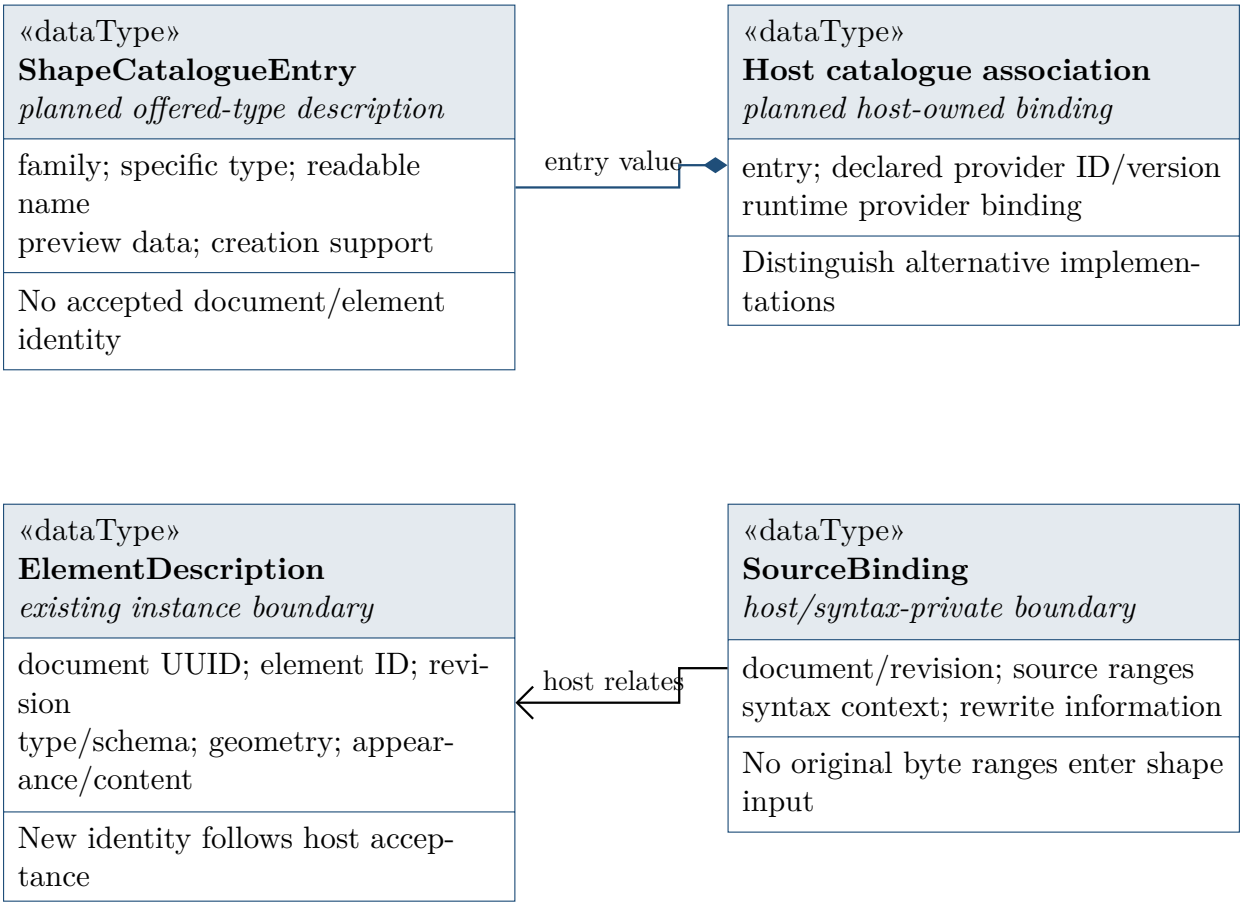


Figure 7.28: Planned UML data-type boundary for catalogue and document instances. Fields describe roles; exact schema, API signatures and compatibility version remain open.

The palette browses entries without changing source or an existing element's provider. Availability remains the intersection of provider, syntax-insertion and document-context support, checked again when executing. Creation supplies candidate interpreted geometry to the host; accepted identity follows document acceptance. The current API v1 has no catalogue/creation entry point.

7.10.5 Planned Creation Activity

UC-009 is an actor goal; the activity below describes its reviewed interaction and acceptance boundary. Provider-default placement, preview and source insertion remain separate responsibilities. Generic history is still planned independently. TC-080 cannot execute until insertion and new-document prerequisites are reviewed and implemented.

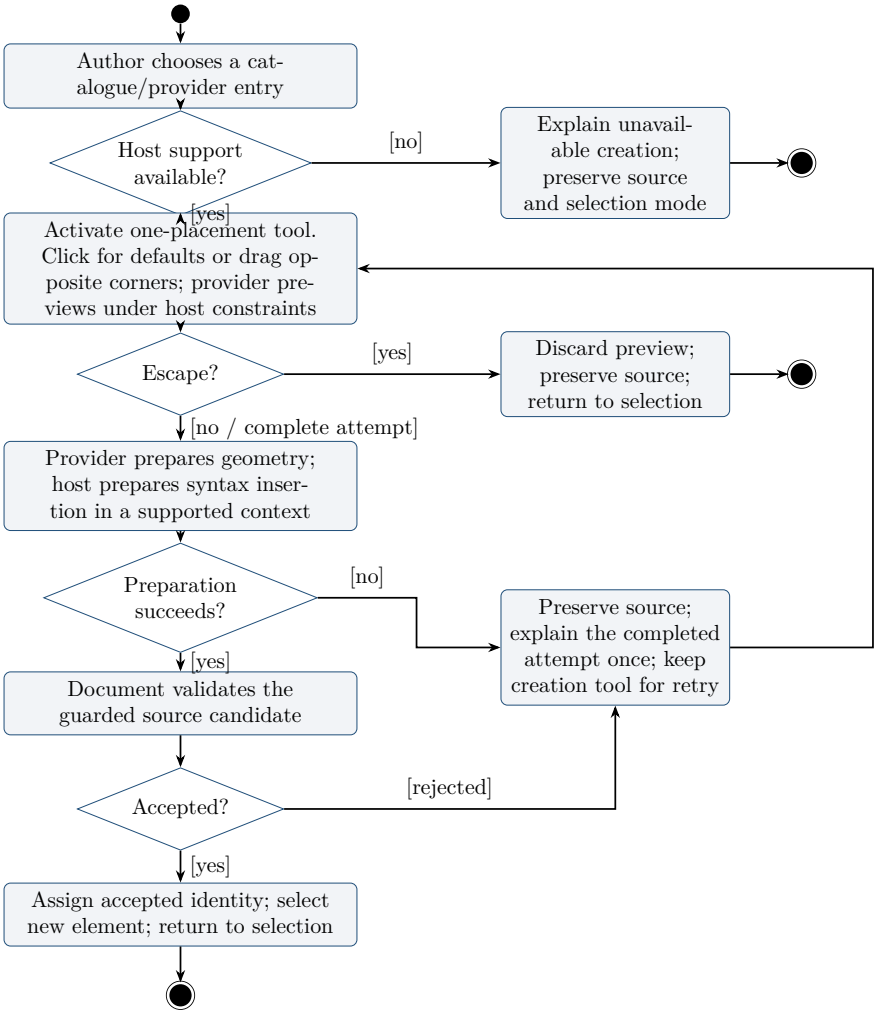


Figure 7.29: Planned UML creation flow with guarded acceptance, cancellation and retry. Failed preparation bypasses document acceptance. Preview changes no source; failure feedback occurs once per completed attempt.

7.10.6 Planned Sidebar Wireframes

These editable UI sketches show expanded Shapes and the persistent collapsed strip. They are not application screenshots or UML relationship diagrams. Geometric tab icons and provider names are illustrative; icons receive readable tooltips and accessible names. No external artwork or icon library is selected. The Files page is a planned entry point whose folder-selection, filters and opening gestures require further review.

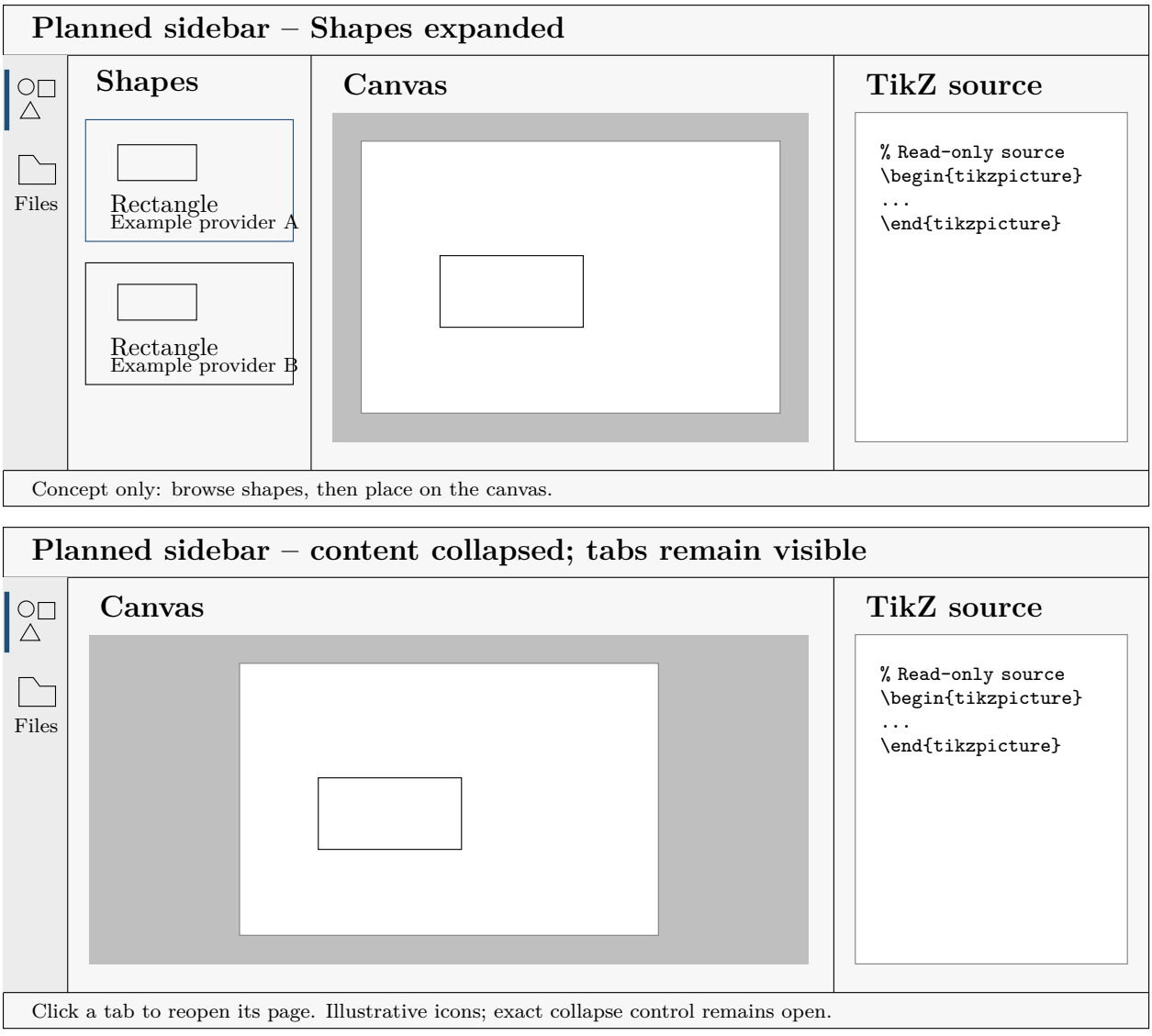


Figure 7.30: Planned expanded and collapsed sidebar using the existing canvas/source split. Default Qt styling is retained; the selected page and width persist only within the window session.

Planned trace: US-011, TC-079 and VAL-011. Catalogue/creation implementation and usability validation remain pending; the bounded shell is shown in Figure 7.25.

Chapter 8

Open Design Decisions

8.1 Temporary Review Workspace

Settled obligations remain in the registers and linked architecture views. This chapter retains unfinished mechanisms, not a second contract. The native rectangle API and bounded sidebar shell are implemented; catalogue/creation and shared-service decisions in Section 7.10 remain planned. Remove this chapter only when its remaining decisions are resolved.

Table 8.1: Remaining Design Review

Topic	**Settled boundary**	**Still to decide**
API/catalogue	API v1 remains current. Catalogue entries describe offered types separately from document instances.	Enumeration/creation signatures, schema and compatibility version; additional providers and registry wiring.
Ownership	Current sidebar/pages use the main window's Qt parent tree; shared services/catalogue are planned window-owned dependencies.	Concrete shared-service construction, consumer teardown and pending-operation coordination.
Sidebar	Persistent Shapes/Files icon strip, collapse/reopen and session tab/width retention; default Qt styling.	The shell uses standard Qt icons, an explicit collapse button and active-tab toggle. Shortcuts, long-term icons and persistence between sessions remain open. Qlementine remains deferred.
Placement	Click provider defaults; rectangle drag uses opposite corners; one-placement completion, boundary fitting, cancellation and retry.	Gesture threshold and provider-default parameters; 2×1 cm is a proposed rectangle starting value, not permanently accepted.
Document creation	Shape proposals and host-controlled insertion precede document acceptance and assigned identity.	New diagram, initial canvas limits, New/Save As and supported syntax-insertion contexts.

Topic	Settled boundary	Still to decide
Working folder	A Files page may request opening through the shared unsaved-work guard.	Working-folder choice, filters and file-opening gestures; folder/document relationship and other file actions require review.
Presentation and syntax	Rectangle presentation/handles exist; source binding stays host-side.	Further primitives/handles, recognition/shared-context protocol, ambiguous syntax and source families beyond the explicit subset.
Updates	Startup normally; unused background activation; notify while in use; retain the working version.	Distribution/authentication, staging/recovery and shutdown coordination.
Migration and adapters	Explicit compatible support or lossless validated conversion; no silent data loss.	Conversion algorithms, prior-project adapters and TikZ-only shape-import format.
History	Exact session snapshots and atomic acceptance are implemented for coordinate edits. Vincent approved no automatic eviction for the prototype.	Measure memory before selecting a budget/eviction policy; provider-dependent retained work and the generic transaction API remain open.
Compiler/PDF	Successful Save triggers future background compilation and explicit revision-aware viewing.	Job scheduling/cancellation, readiness/throbber/failure UI, dependencies and sandbox policy.
Resize/labels	Current crossing/minima/bounds; reviewed labels retain attachment.	Aspect locking/edge handles, label detachment and host deletion.

8.2 Next Bounded Review

The current application is `0.1.4-alpha` with the bounded sidebar shell and coordinate-edit session history.

Shape discovery and creation remain planned for `0.2.0-alpha`. Review document creation, insertion, API signatures and lifetime wiring before implementing that milestone. Shared service ownership also remains planned.

Preserve older packages, edited examples and earlier reports. Creation and shared service refactoring require their own implementation review.

Chapter 9

Codes, Standards and Conventions

9.1 Applicability and Derived Design Obligations

This desktop-software project uses selected technical specifications and engineering guidance. The entries below identify their project-specific application and checks; they do not claim certification, full standards conformance or a completed legal review. Official public descriptions and documentation were consulted on October 30. The engineering obligations below derive from project decisions and the milestone, not invented quotations from unavailable standard clauses.

Table 9.1: Applicable Specifications, Guidance and Checks

Reference / applicability	**Resulting project obligation**	**How application is checked**
ISO/IEC/IEEE 29148:2018 requirements-engineering guidance [3]. Selected guidance; no full conformance claim.	Stable IDs, singular reviewable obligations, acceptance criteria and reciprocal links. RPT-019, RPT-037, RPT-053.	Inspect registers and check every requirement/test/verification and story/validation link in both directions. Review missing criteria and distinguish plans from completed evidence.

Reference / applicability	Resulting project obligation	How application is checked
OMG UML 2.5.1 [6]. Selected UML class/data-type, package, component/interface, actor-goal, activity, state and sequence notation; orthogonal routing is the project authoring rule.	Document distinct roles and ownership: REQ-A-004, REQ-DEV-004; show planned/current views separately.	TC-018, TC-033, TC-056; inspect arrow meanings, lifetimes, guards, ordering and consistency with source or labelled planned contracts.
PGF/TikZ coordinate and source specification [12]. Technical interchange/rendering reference, not a regulatory standard.	Preserve editable source and a consistent coordinate mapping: REQ-D-001, REQ-A-007; preserve local label dependencies: REQ-F-024.	TC-027–TC-029, TC-044, TC-047; compile saved fixtures in the recorded PGF environment and compare fixed extents/bytes.
UTF-8 (RFC 3629) [14]. Encoding reference for byte-preservation fixtures.	REQ-D-001, REQ-F-012: byte offsets and exact snapshots preserve multibyte comments; source-view decoding never becomes saved source.	TC-004, TC-035; exact diffs/restoration with multibyte comments, CRLF and numeric spellings. Broad Unicode text interpretation is not claimed.
Project coding/ownership conventions. Locally adopted rules, not an external standard.	REQ-A-005, REQ-DEV-001–REQ-DEV-005; tkr names, separate modules, descriptive prefixes, shallow inheritance, explicit lifetimes and named numeric values.	TC-019, TC-030–TC-033, TC-066; inspect source/diagrams and configure/build with the recorded Qt kit.
Course milestone and academic-assistance record. Course obligation.	RPT-011, RPT-053: truthful attribution, controlled history and planned/completed evidence distinction.	Inspect Document History, source citations, prompt records and Milestone 2 coverage table. Vincent reviews the exact submitted PDF.

9.2 Applicability Limits

No hardware, building, medical-device or safety certification requirement has been established for this local editor. Do not infer an exemption for future deployment: reassess

obligations if use or distribution changes. Qt, PGF and other dependencies retain their own licensing conditions; redistribution review remains open under RPT-046. The retained Alpha packages are local prototypes, not evidence of licensing or release readiness. The sidebar review adopts no external theme or icon library; later asset choices require separate licensing review.

Chapter 10

Tests

These proposed test cases make the current requirements reviewable. They are plans, not executed results or verification evidence. The initial syntax and reference sources are now defined; before product-test execution, the team must fix platform and toolchain versions, zoom range, visual comparison tolerances, and exact expected diagnostics. Each test should record its run date, environment, result, and evidence when performed. Developer checks TC-019 and TC-030–TC-033 include source/design inspections; they are distinct from end-user interaction tests and remain proposed checks.

10.1 Proposed Test Cases

Table 10.1: Proposed Product Test Cases

ID	Requirements	Procedure	Expected result
TC-001	REQ-F-001	Open a readable, supported local sample; compare the file bytes before and after opening.	The source loads and the selected file remains byte-for-byte unchanged. VER-003
TC-002	REQ-F-002	Open the reference sample and compile it using the configured local toolchain.	The application displays the rendered diagram corresponding to that source without reporting a compiler failure. VER-004
TC-003	REQ-F-003	In a supported sample, select one documented editable object and move it to a specified position.	The intended object can be selected and visibly reaches the requested position. VER-005
TC-004	REQ-D-001, REQ-F-004	Move one supported object, inspect the changed source coordinates and preview, and diff the source against the original.	Coordinates and preview match the requested move; source outside the edited construct is unchanged. VER-006, VER-027
TC-005	REQ-F-005	Open a syntactically invalid sample and request a preview.	A compiler failure is reported, and the original source remains available without changes. VER-007

Proposed Product Test Cases (continued)

ID	Requirements	Procedure	Expected result
TC-006	REQ-F-005	Attempt graphical editing of a construct outside the documented editable subset.	The edit is refused with an explanation; the opened source remains intact. VER-007
TC-007	REQ-D-001, REQ-F-006	Move a supported object, approve and save the revision, reopen the saved file, and diff it against the original.	The save succeeds, the revision reopens, and only the approved supported construct differs. VER-008, VER-027
TC-008	REQ-F-006	Attempt to save the approved revision to a location that denies writing.	The application reports failure and keeps the unsaved revision available for retry. VER-008
TC-009	REQ-S-001	Configure the local compiler, request a preview, and inspect the exact spawned command and arguments.	The invocation does not enable TeX shell escape unless the user has explicitly configured that option. VER-049
TC-010	REQ-C-001	With a configured local toolchain and network access disabled, open a supported local sample, preview it, move one supported object, and save the revision.	The entire core workflow succeeds without contacting or depending on a network service. VER-072
TC-011	REQ-B-001, REQ-B-002	Once a professor sample within the documented editable subset is available, choose one element and target position; open and preview the diagram, move the element graphically, save the source, then open it in a standard text editor and compile it with a local LaTeX toolchain outside TikZRefinery.	The element reaches the target position; the saved source remains editable and compiles outside the application with the approved change visible. VER-001, VER-002
TC-012	REQ-F-007	Generate a valid preview, then introduce a LaTeX error in the working source and request another preview.	The last successful preview stays visible, is marked as older than the current source, and the compile failure is reported. VER-009
TC-013	REQ-F-008	Preview a supported sample that uses a local style or definition file; repeat with that dependency unavailable.	The first preview includes the dependency; the second attempt identifies the missing dependency rather than displaying an apparently current result. VER-010
TC-014	REQ-A-001	Run a deliberately slow preview compilation while attempting to use an unrelated editor control.	The editor accepts the interaction before compilation completes; process or thread inspection confirms compilation is outside the UI thread. VER-050

Proposed Product Test Cases (continued)

ID	Requirements	Procedure	Expected result
TC-015	REQ-A-002	Record the opened document directory, compile a preview, and compare the directory before and after the run.	Preview intermediates are confined to a separate temporary workspace; the opened document directory gains no compilation intermediates. VER-051
TC-016	REQ-A-003	Inspect the component interfaces and replace the PDF display component with a test double during a compilation test.	Compilation results and diagnostics reach the display boundary without compilation control depending on a concrete PDF widget. VER-052
TC-017	REQ-P-001	Drag a supported object through several pointer positions while recording visible updates and preview compiler launches.	Geometry updates without compilation; pointer movement, pause and release launch none. A successful Save requests compilation of the saved revision; failed Save requests none. VER-071
TC-018	REQ-A-004	Review the component design and code for the open, move, and preview workflows; identify the class responsible for each named concern.	Each concern has a documented owning component, and no single class performs graphical interaction, source editing, compilation, and PDF display together. VER-053
TC-019	REQ-A-005	Inspect the application source tree and compare primary class definitions with source modules.	Each primary class has its own module; any shared helper type is subordinate to that module's stated responsibility. VER-060
TC-020	REQ-A-006	Run a source interpretation and supported-move test using a representative TikZ sample without starting the GUI or LaTeX.	The test produces the expected revised source and can run without a window or compiler process. VER-054
TC-021	REQ-D-001, REQ-F-009	Drag the body of a supported straight line by a known offset; inspect both endpoint coordinates and diff the source.	Both endpoints move by the same offset; unrelated source is unchanged. VER-011, VER-027
TC-022	REQ-D-001, REQ-F-009	Drag the body of a supported straight arrow by a known offset; inspect its endpoint coordinates, arrow direction, and source diff.	Both endpoints move by the same offset, the arrowhead retains its intended direction, and unrelated source is unchanged. VER-011, VER-027
TC-023	REQ-D-001, REQ-F-010	Drag one endpoint handle of a supported straight line; inspect the endpoint coordinates and source diff.	Only the selected endpoint changes; the opposite endpoint and unrelated source remain unchanged. VER-012, VER-027
TC-024	REQ-D-001, REQ-F-010	Drag one endpoint handle of a supported straight arrow; inspect the endpoints, arrowhead, and source diff.	Only the selected endpoint changes; the arrowhead follows the revised segment and unrelated source remains unchanged. VER-012, VER-027

Proposed Product Test Cases (continued)

ID	Requirements	Procedure	Expected result
TC-025	REQ-D-001, REQ-F-011	Select and drag a supported rectangular box by a known offset; inspect its source and compiled result.	The whole box moves without changing its size; only the necessary source coordinates change and the compiled box appears at the intended position. VER-013, VER-027
TC-026	REQ-I-001	View supported lines, arrows, and boxes at the documented minimum, normal, and maximum zoom levels; inspect edges and record screenshots.	Shapes redraw with clear edges at each zoom level rather than showing enlarged bitmap blocks or blurred low-resolution strokes. VER-039
TC-027	REQ-F-004	For one line, arrow, and box edit, compare canvas and independently compiled output using the same fixed diagram extent, origin, and scale; inspect position, size, endpoints, and arrow direction.	Each supported edited construct agrees within the documented comparison tolerance; independent cropping does not conceal a position error. VER-006
TC-028	REQ-A-007	Check known TikZ points against independently calculated scene positions, then check inverse conversion and round trips. Repeat pointer moves at supported zoom levels and after panning.	Units, origin, and axis directions agree with the reference mapping; equivalent gestures yield the same source displacement. Zoom or pan alone leaves source coordinates unchanged. VER-055
TC-029	REQ-A-007, REQ-F-004	Open the separated-box reference; drag the right box left until its edge touches the stationary box. Inspect coordinates, save, and compile outside the editor. Compare both views at a common origin and scale.	The moved box reaches the expected coordinates; the stationary box and both sizes remain unchanged. The shared edge agrees in the editor and compiled PDF within the documented tolerance. VER-006, VER-055
TC-030	REQ-DEV-001	Inventory project-owned class declarations, matching headers/implementations, includes, CMake lists, and diagrams; check the tkr/main-name/optional-specification convention, then configure and build with the selected Qt kit.	Every project class and file uses the complete agreed name; references resolve and the native build succeeds. Vendor types and shared records retain their documented naming. VER-067
TC-031	REQ-DEV-002	Inspect methods, parameters, local variables, and private member declarations against the coding guide; check a sample of uses against their declared types and coordinate spaces.	Methods use camelCase; variables use descriptive type/category prefixes; private members add m_; names agree with their current meaning and type. VER-068

Proposed Product Test Cases (continued)

ID	Requirements	Procedure	Expected result
TC-032	REQ-DEV-003	Trace every application inheritance path from its first project-defined class, excluding preceding Qt ancestry; compare source declarations with diagrams.	No path exceeds main/base, child, grandchild. The concrete classes remain final; the rectangle provider realizes a two-level interface chain, with no artificial extra level. VER-069
TC-033	REQ-DEV-004	Compare owned and borrowed class relationships in declarations and constructors with the ownership table and class details; inspect destruction order and QObject parents.	Each relationship has an identified owner or borrower and a consistent lifetime mechanism; borrowed objects outlive users, and parent/unique_ptr ownership is not assigned twice to one object. VER-070
TC-034	REQ-A-008, REQ-A-011, REQ-D-002, REQ-D-003, REQ-D-008, REQ-I-003	Submit multi-range proposals with wrong document identity, stale revisions, mismatched bytes, invalid or conflicting ranges, ambiguous same-position insertions and unsupported operation rules; repeat with a valid batch.	Invalid proposals preserve source/history and emit no successful change notification; one valid changed action creates one entry and one notification. VER-028, VER-029, VER-034, VER-041, VER-056, VER-059
TC-035	REQ-D-004, REQ-D-007, REQ-F-012	Commit a multi-range action with UTF-8 comments, CRLF and precise numeric spellings; undo and redo repeatedly. Inspect history entries, revision values and stale proposal rejection.	Each Undo/Redo restores one complete action with exact bytes, moves only the existing history position and advances the revision; stale proposals remain invalid. VER-014, VER-030, VER-033
TC-036	REQ-D-003, REQ-F-014, REQ-I-003	Preview, cancel, reject and perform empty, byte-equivalent and rounded-zero actions; invoke Undo during a later drag.	Noncommits preserve source/history and emit no successful change notification; Undo cancels preview before restoring the last completed edit. VER-016, VER-029, VER-041
TC-037	REQ-D-007, REQ-F-012, REQ-F-015	Undo an edit; select or cancel a gesture; redo. Undo again and accept a different changed edit.	Noncommits preserve redo; the new accepted edit clears the former redo branch. VER-014, VER-017, VER-033
TC-038	REQ-D-005, REQ-D-006, REQ-F-013, REQ-F-016, REQ-F-017, REQ-F-018, REQ-F-019, REQ-F-020	Save, undo and redo while comparing file bytes and the saved baseline. Exercise Save/Discard/Cancel before Open/Reload, failed writes and Opens, successful replacement, and explicit Save followed by failed Open. Restart and inspect the TeX file.	Save retains undo/redo; only successful writes replace the baseline and exact byte inequality controls dirty state. Undo/Redo never writes disk. Failed/cancelled operations retain state, including a prior explicit Save. Cancel or failed requested Save prevents replacement. Successful Open/Reload clears history; no history is embedded or restored after restart. VER-015, VER-018, VER-019, VER-020, VER-021, VER-022, VER-031, VER-032

Proposed Product Test Cases (continued)

ID	Requirements	Procedure	Expected result
TC-039	REQ-A-010, REQ-F-007	With future compiler integration, accept an edit or undo/redo, successfully save, then simulate failure of the compilation requested for those saved bytes.	Source and history retain the accepted state; diagnostics identify failure and any older successful preview is labelled accordingly. VER-009, VER-058
TC-040	REQ-A-008, REQ-A-011, REQ-D-008	Insert at a zero-length range, delete using an empty replacement and replace several ranges relative to one original source; include unambiguous combined insertions and route through the common entry point.	Proposal identity, revision, description and byte edits are retained; each supported changed operation is atomic and reversible, with unrelated bytes unchanged. VER-034, VER-056, VER-059
TC-041	REQ-A-009	Exercise separate element and syntax providers, then a provider implementing both; route their supported operations through the common contract.	Independent roles retain their declared contract and the same source acceptance guarantees; metadata/lifecycle checks are separately mapped. VER-057
TC-042	REQ-A-008, REQ-A-011, REQ-D-002, REQ-D-007, REQ-I-003	Run planning, validation and history restoration without a GUI/compiler; inject failure before source/history publication.	Successful results match expected bytes; failure preserves source, history position and entries, and emits no successful document-change notification. VER-028, VER-033, VER-041, VER-056, VER-059
TC-043	REQ-I-002, REQ-I-003, REQ-I-004, REQ-I-005	Inspect Undo/Redo with empty history, after an edit, after Undo/Redo and after opening a new document. Compare the refreshed canvas with accepted source, count notifications, and inject a failed restoration.	Unavailable actions are disabled; each accepted edit/restoration refreshes the canvas and emits exactly one successful change notification. Failed restoration is reported without source/history changes or a successful change notification. VER-040, VER-041, VER-042, VER-043
TC-044	REQ-F-021	Use a direct-position label and an attached label on a straight arrow; request known nonzero horizontal and vertical moves, then undo/redo.	The label reaches the target through supported relative-placement edits; host coordinates and unrequested text, anchor, rotation, alignment and style bytes are unchanged. VER-023
TC-045	REQ-F-022	Move a host line/arrow with an attached label by a known vector; reparse and compare both geometries.	The host translates once and the label follows its retained attachment rule; no stale reference is used. VER-024

Proposed Product Test Cases (continued)

ID	Requirements	Procedure	Expected result
TC-046	REQ-F-023	Drag an attached label near and far from its host; inspect the relationship and source. Inspect an unavailable detachment action.	Every drag retains attachment and host geometry. Detachment requires a distinct explicit request; unsupported detachment performs no rewrite. Implementation of detachment remains later work. VER-025
TC-047	REQ-F-024	Use a shared named coordinate referenced by a host, selected label and another object. Try a supported local offset and a form with no supported independent adjustment.	The supported move changes only local placement; shared definitions and unrelated elements are byte-identical. The unsupported move is unavailable with a reason and no state change. VER-026
TC-048	REQ-D-009, REQ-D-012	Inspect equivalent label descriptions from two source forms; compare records and presentation outputs.	Typed, versioned interpreted records are distinct from original byte ranges and Qt scene items; presentation and selectable descriptions remain available as separate data. VER-035, VER-038
TC-049	REQ-D-010	Prepare label bindings/host references at revision N, accept another edit, and submit an old request.	Stale document/revision references are rejected without source/history mutation; reparsing produces references for the accepted revision. VER-036
TC-050	REQ-D-011	Inspect proposals/history in memory; edit, save, undo, reopen another file and restart. Diff the saved source and inspect history. Exercise a longer session and inspect the retention policy.	Exact snapshots support restoration in the session without automatic eviction in the initial prototype. Save retains history without serializing it; successful replacement/restart starts empty history. Failure paths retain current state. VER-037
TC-051	REQ-I-006	Complete a changed edit, a byte-equivalent edit, a rejected edit and an Escape cancellation; capture returned outcomes, messages and history counts.	Changed returns Applied; byte-equivalent returns Unchanged; failure returns Rejected with a reason. Cancellation/no-op shows no error; only the changed action records history. VER-044
TC-052	REQ-I-007	Send many preview updates, then one rejected release; dismiss its message and repeat before a successful operation.	No preview update produces a rejection banner. One rejected attempt produces one red-accented bottom strip with bright red icon and reason; it stays until dismissal or the next completed operation. Canvas remains usable. VER-045

Proposed Product Test Cases (continued)

ID	Requirements	Procedure	Expected result
TC-053	REQ-D-012, REQ-I-008	Overlap a label, host and third element. Click the label, then use direct right-click selection and hover each entry before choosing the obscured host.	Label wins ordinary hit selection. Every intersecting element appears once with a readable description; hovering identifies it and the chosen obscured host can be moved without moving blockers first. VER-038, VER-046
TC-054	REQ-I-009	Drag a region from empty canvas, then start over an object with Select area active. Right-click inside, choose a partly intersecting element, and press Escape in a later region.	Both gestures show a retained rectangle; candidates include partial intersections and obscured objects. Region creation makes no source/history change or automatic group selection; chosen entry selects one element and Escape clears the region. VER-047
TC-055	REQ-I-010	Use independently calculated hit geometries intersecting a 6-logical-pixel circle and geometries separated beyond it. Repeat at 0.5x, 1x and 2x test zoom and two available display scales.	Candidate inclusion follows geometry intersection with the 6-pixel-radius screen circle. Zoom does not change its logical screen size. Record user feedback before changing the provisional value; these zooms are test inputs, not a product range claim. VER-048
TC-056	REQ-A-012, REQ-A-013	Inspect collaborator ownership and run headless preparation, atomic apply and history retrieval tests; inject failure before publication.	Controller owns edit service/file store and only coordinates; document owns history. Preparation cannot mutate live state. A successful edit/restoration updates source and history together; injected failure changes neither. VER-061, VER-062
TC-057	REQ-A-014	Use a file-store test double for successful/failed read and guarded write. Capture the written snapshot; exercise Save before failed Open.	File store touches no history. Only successful I/O permits corresponding load/baseline updates; baseline equals written bytes. Failed operations preserve state, including any preceding explicit successful Save. VER-063
TC-058	REQ-A-015, REQ-A-017	Use headless fake element/syntax providers with equivalent label descriptions and differing edit capabilities; inspect scene ownership.	Only the intersection of shape, syntax/binding and context permits the operation, rechecked at execution. Providers do not mutate model/UI/files; the scene owns created graphics items. VER-064, VER-066

Proposed Product Test Cases (continued)

ID	Requirements	Procedure	Expected result
TC-059	REQ-A-016	Establish a binding with provider A, register another matching provider B, and request a supported label edit.	Existing bindings route to the same responsible instances throughout the gesture; stale requests fail without mutation. Missing-provider behavior follows TC-074. VER-065
TC-060	REQ-A-023, REQ-F-025	Resize each corner of the two-box fixture, including reversed source corners, horizontal/vertical/both-axis crossing, thin fillers, 0.01 cm minima and extent limits. Repeat under view transforms, cancellation and stale requests; save and reopen.	The opposite corner, stationary box and unrelated bytes are preserved. Preview and release agree, including minimum and boundary clamping. One guarded batch is applied per changed release; no-op, cancelled or rejected requests preserve the revision. VER-073, VER-099
TC-061	REQ-A-018	Load the separate rectangle plugin through the common shape API and repeat the box display, move, resize and save/reopen checks on the same fixture. Inspect the scene/controller call sites.	The actual loaded provider supplies rectangle geometry/presentation and resize behavior through the common API. Baseline behavior is retained; missing/incompatible modules fail explicitly without a hidden rectangle implementation. VER-074
TC-062	REQ-I-011	Open the two-box fixture in light and dark palettes.	Pane labels and separation are visible, and the black outlines remain legible. VER-075
TC-063	REQ-I-012	Open the fixed-extent fixture and resize the window and its canvas pane.	The entire canvas remains visible at both sizes, with unchanged proportions, source bytes and edit limits. VER-076
TC-064	REQ-I-013	Compare the visible paper with the declared extent in the two-box fixture. Drag beyond its edge and release.	The paper matches the declared extent, the surrounding view is distinct, the preview stays inside the paper, and release accepts the same furthest valid geometry without a rejection message. VER-077
TC-065	REQ-I-014	Drag the same box repeatedly at a fractional fit scale. Inspect while dragging and immediately after release, before moving focus.	Only the current outline and any selection marks remain visible; the drag path contains no residual marks. VER-078
TC-066	REQ-DEV-005	Inspect project-owned application logic for numeric literals. Start with scene dimensions, hit tolerances, item-data keys and arrow geometry.	Each numeric value comes from a meaningfully named definition with the relevant units or role. No bare numeric literals remain in executable logic. VER-079

Proposed Product Test Cases (continued)

ID	Requirements	Procedure	Expected result
TC-067	REQ-DEV-006, REQ-DEV-007, REQ-DEV-008	Build from a known Git checkpoint with recorded source state. Compare report metadata, generated build identity, window title, About dialog and package manifest. Inspect the no-Git fallback.	Feature/revision changes and `-alpha` labels follow policy. The built baseline is truthful even when the document advances; folder/title omit hashes, About includes short/full identity and source state, and historical artifacts are unchanged. VER-080, VER-103, VER-104
TC-068	REQ-A-019, REQ-D-016	Register well-formed and incomplete metadata, incompatible API declarations and two candidate instances of one plugin. Inspect roles, type/schema declarations and the registry identities.	Only validated registrations activate; stable IDs remain separate from runtime instance/version handles. VER-093, VER-095
TC-069	REQ-F-026	Discover two candidates with the same stable ID in reversed directory orders. Present names, versions and paths; choose one, cancel another choice and restart.	Discovery order never resolves the conflict; the explicit compatible choice is respected and unresolved candidates remain inactive. VER-081
TC-070	REQ-D-016, REQ-F-027	Load basic and text-capable rectangle providers together; bind separate elements, alter defaults and register another provider during a gesture.	Both variants coexist; each element retains its chosen provider and capabilities; no gesture silently changes binding. VER-082, VER-093
TC-071	REQ-A-019, REQ-A-020, REQ-D-017	Try API mismatch, matching API with incompatible geometry schema, loss of used text capability, supported old data and an explicitly validated lossless conversion.	Only declared compatible combinations are usable; incompatible transfers preserve the old version and all relevant data. No implicit conversion is attempted. VER-094, VER-095, VER-096
TC-072	REQ-A-023, REQ-D-013, REQ-D-014	Send common and provider-qualified operations with versioned parameters, centimetre displacement, corner identity and fixed bounds. Compare repeated previews with direct calculation from initial geometry; include stale IDs and unknown schemas.	Routing and parameter interpretation are explicit; preview/release agree without accumulated drift. Invalid or stale requests cannot mutate source. VER-090, VER-091, VER-099
TC-073	REQ-D-015	Prepare changed, byte/geometry-equivalent, unsupported and infeasible operations. Reject one otherwise prepared proposal at the document boundary.	Preparation returns Prepared change, Unchanged or Cannot prepare with reason; only later host acceptance can return Applied. All noncommits preserve source/history. VER-092

Proposed Product Test Cases (continued)

ID	Requirements	Procedure	Expected result
TC-074	REQ-F-032	Open source with a required provider missing or incompatible, and register an alternative provider without choosing it. Attempt an unavailable edit.	Source remains intact, the missing support is explained and no provider is silently substituted. Current whole-interpretation limits are reported rather than a partial-rendering guarantee. VER-087
TC-075	REQ-F-028	At startup provide a compatible staged update, an incompatible one and an unavailable update service before document restoration.	Valid updates are considered before opening/restoring diagrams; incompatible updates retain working support and network failure does not block local editing. VER-083
TC-076	REQ-A-021, REQ-F-029	Stage a replacement for an unused provider while the application runs. Acquire a document/gesture dependency before activation, then release it.	Installation does not disrupt editing; activation rechecks use and waits while a dependency exists. No active version is overwritten in place. VER-084, VER-097
TC-077	REQ-A-021, REQ-F-030, REQ-I-016	Hold provider dependencies through an open document, active gesture and pending operation separately. Discover an update; choose Later, then exercise restart with Save/Discard/Cancel and a failed Save.	The old provider stays active, notification is nonmodal, deferral survives the session and restart respects unsaved-work decisions. Provider replacement never occurs inside an edit. VER-085, VER-097, VER-101
TC-078	REQ-A-022, REQ-F-031	Use an explicit update-origin test double. Fail download/staging, compatibility validation and activation in turn, then perform a valid replacement.	The working version, authoritative source and provider choices survive each failure; activation follows separate validation. Detailed authentication is a later prerequisite, not a claimed implemented mechanism. VER-086, VER-098
TC-079	REQ-F-033, REQ-D-018, REQ-I-015, REQ-I-018, REQ-I-019, REQ-I-020	Enumerate alternate rectangle providers and an edit-only shape. Inspect catalogue records versus document elements, provider identity, previews and capabilities; switch Shapes/Files pages, collapse/reopen and resize the sidebar in both palettes.	Entries are distinguishable and contain no document-element identity. Icons, readable labels and active state identify tabs; the strip remains available and session tab/width state survives collapse. Browsing changes no source; creation needs shape and syntax support. VER-088, VER-100, VER-105, VER-107, VER-108, VER-109

Proposed Product Test Cases (continued)

ID	Requirements	Procedure	Expected result
TC-080	REQ-F-034, REQ-F-035, REQ-D-001, REQ-I-006, REQ-I-007, REQ-I-013, REQ-I-021	In a later supported insertion context, click-place and drag-size rectangles, including thin/crossed geometry and boundary positions; try a default that cannot fit, cancel a preview, reject/retry, then accept/save/reopen. Review new-document and insertion prerequisites before execution.	Provider defaults and documented minima apply; click fitting preserves default size and stroke stays inside declared bounds. Escape/rejection preserve source; one completed rejection gives one reason and keeps the tool. Acceptance assigns identity, selects the new element and returns to selection; unrelated bytes remain exact. VER-027, VER-044, VER-045, VER-077, VER-089, VER-106, VER-110
TC-081	REQ-I-017	With future compiler/PDF integration, compile source missing a graphical provider; then remove a LaTeX dependency, fail compilation and edit source after a successful result.	Successful PDF is read-only and tied to its source revision. Missing dependencies and stale/failed results are explicit; compilation never restores graphical edit support or changes source. VER-102
TC-082	REQ-A-024	Inspect the planned ownership/borrow graph and, after refactoring, exercise window setup, palette/canvas access and closure with pending dependent work.	The window owns shared services/catalogue; views borrow access and perform presentation roles. Consumers and pending work are released before their services. Neither the controller nor views take over module management; ownership agrees with UML and source. VER-111

10.2 Milestone 2 Test Execution Plan

All TC entries are planned product checks. The separately dated prototype checks below are bounded evidence, not complete product verification or user validation. Vincent executes/reviews the checks; Professor Muresan or an available TikZ author evaluates user value through Chapter 13.

10.2.1 Levels, Data and Entry Conditions

Freeze the source/package revision, OS, Qt/compiler/PGF versions, dependencies, input bytes, expected outputs and test settings before each run. Use isolated copies and temporary workspaces, preserving editable demos. Current reference tooling is Windows x64, Qt 6.12.0, GCC 13.1.0, CMake 3.30.5 and shared TinyTeX; record actual versions when testing rather than assuming this reference. Labels, generic transactions, wider provider management and compiler integration require implementation before their product checks can run. Rectangle

API v1 and the sidebar shell retain separately dated bounded evidence. Session history now has bounded evidence in Section 10.16. Reviewed generic transaction views are not executed tests: see Figures 7.17 and D.8. Declare unsupported forms explicitly.

Table 10.2: Test Levels and Representative Data

Level	Cases / data	Method	Pass criterion
Unit	TC-028, TC-034, TC-040, TC-049	Known points; UTF-8 comments, CRLF, decimal spellings, stale identities, insert/delete/conflicting ranges; headless fixtures.	Every expected byte/coordinate and rejection invariant agrees; zero unintended mutations.
Integration	TC-035, TC-038, TC-056, TC-057, TC-058	Document/history failure injection, file-store doubles and fake providers; count notifications and compare saved bytes.	Atomic state transitions, exact restoration and collaborator boundaries satisfy each linked oracle.
System	TC-007, TC-010, TC-027, TC-044, TC-053, TC-054	Native/offscreen gestures, fixed-extent corpus, local dependency/error fixtures, labels and at least three overlapping objects.	Saved source, selection, feedback and compiled output satisfy the case; all required interactions complete.
User validation	VAL-001–VAL-010	Stakeholder-selected realistic diagrams plus controlled failure and overlap tasks; record assistance and feedback.	Participant meets the story-specific criterion and confirms the workflow addresses the need; failures become design actions.

10.2.2 Oracles, Results and Coverage

Use exact byte equality outside approved edit ranges; exact stored bytes for undo/redo and save; independently calculated positions for coordinate tests; event counters for one accepted-action notification and zero noncommit notifications. For the existing fixed-extent prototype comparison only, use 100 pixels/cm and a one-pixel edge allowance with zero unmatched dark edge pixels. That allowance is not a user-acceptance or broader text-rendering tolerance. Fix text/label comparison tolerances before those executions.

Each run records case and requirement IDs, date, executor, source revision, environment, input/expected/actual outputs, pass/fail/blocked result, evidence paths, defects and recheck. A case passes only when every expected condition holds; a missing prerequisite is Blocked, not Pass. Requirement verification requires all mapped checks and Vincent's review. No performance timing target is invented: TC-017 checks live feedback and compiler scheduling; latency/zoom acceptance thresholds remain to be baselined.

The full forward/reverse maps are in Chapter 12, Chapter 13 and Chapter 11. The following historical sections identify evidence already obtained; their dates and limits remain

intact.

10.3 Coordinate and Box Reference Checks

The first visual comparison will start with two separated rectangular boxes, then move one until they touch. Vincent selected this sequence to investigate whether canvas-to-source conversion reproduces the intended placement. He reports that the previous group encountered a difference between adjacent boxes in its editor and externally compiled output, and later fixed it. This check targets the observable failure; the previous group's specific cause is not established here. The cases remain proposed product tests, not evidence that TikZRefinery has been implemented or verified.

10.3.1 Independent Coordinate Expectations

For the initial subset in Section 5.4, document origin, axis directions, and source precision before running TC-028. TikZ's usual canvas coordinates measure rightward and upward; Qt Graphics View uses scene and view mappings with the usual display axis increasing downward [12, 7]. A proposed mapping for explicit centimetre coordinates into a scene with downward-positive y is

$$x_{\text{scene}} = o_x + s x_{\text{TikZ}}, \qquad y_{\text{scene}} = o_y - s y_{\text{TikZ}},$$

where (o_x, o_y) is the scene position of the TikZ origin and $s > 0$ is the number of scene units per centimetre. This is a reference for the first prototype, not a claim to support arbitrary TikZ transformations.

Check the origin, $(1\,\text{cm}, 0)$, $(0, 1\,\text{cm})$, and a point with negative coordinates against independently calculated expected positions. Then test inverse conversion and round trips. A round trip alone can miss two incorrect conversions that cancel each other. Map pointer positions from view to scene before computing a source displacement; zoom and pan must affect the view without rewriting source coordinates. Qt's mapping methods and a small conversion boundary are candidates; whether that boundary is a helper function or a class remains a design decision.

10.3.2 Separated Boxes Moved into Contact

For TC-029, use explicit centimetre coordinates, equal stroke widths, and no rotation, scaling, or node padding. Define the initial left box from $(0, 0)$ to $(1, 1)$ and the initial right box from $(2, 0)$ to $(3, 1)$. Move only the right box by $(-1\,\text{cm}, 0)$, giving final corners $(1, 0)$ and $(2, 1)$. Both boxes remain 1 cm square; the left box stays fixed and the final edge centre lines coincide at $x = 1\,\text{cm}$. This definition accounts for stroke thickness when judging visible contact.

Record the editor placement, revised source, and independently compiled output. Compare with a shared fixed diagram extent, for example $(-0.5, -0.5)$ to $(3.5, 1.5)$ centimetres, so automatic cropping cannot hide a translation. An aligned overlay can expose a gap,



Figure 10.1: Compiled TikZ reference for the separated-box contact test. The right box moves left by 1 cm. These illustrations show expected geometry, not an executed editor test.

overlap, size change, or axis error. Set the geometric and rendered-image tolerances before execution; exact pixel equality is not required because rendering and antialiasing can differ. Repeat after zoom and pan to check that contact survives view changes.

10.4 Explicit Reference Files

The self-contained LaTeX examples under `code/examples/explicit-coordinates/` provide initial inputs and expected revisions. Each file uses explicit centimetre coordinates and one drawing command per movable element. The shared bounding-box declaration fixes the comparison extent. Expected files show source changes for specific gestures; they are comparison references, not output from an implemented editor.

Table 10.3: Initial Explicit-Coordinate Reference Corpus

Test case	Initial file	Expected revision
TC-021	`line-initial.tex`	Translate both endpoints by $(0.5, 0.5)$ cm; compare with `line-translated.tex`.
TC-023	`line-initial.tex`	Move the second endpoint from $(1, 0.5)$ to $(1, 1.5)$ cm; compare with `line-endpoint.tex`.
TC-022	`arrow-initial.tex`	Translate the first arrow by $(0.5, 0.5)$ cm; compare with `arrow-translated.tex`.
TC-024	`arrow-initial.tex`	Move the first arrow's second endpoint from $(2, 1)$ to $(2, 1.5)$ cm; compare with `arrow-endpoint.tex`.
TC-025, TC-029	`boxes-initial.tex`	Move only the right box by $(-1, 0)$ cm; compare with `boxes-touching.tex`.

The line sample includes negative coordinates; the arrow sample includes `->`, `<-`, and `<->` variants. The unchanged arrows also provide checks against unintended edits. Use these samples for TC-026 and TC-027 after documenting zoom and comparison tolerances. The corpus README records coordinates, expected source differences, and independently calculated scene positions for TC-028. Compile the reference files to establish their LaTeX baseline; application verification requires actual editor executions and retained evidence.

10.5 Initial Native Build and GUI Check

On October 20, 2026, the initial scaffold configured and built successfully on Vincent's Windows computer using Qt 6.12.0 (MinGW 64-bit), GCC/MinGW 13.1.0, CMake 3.30.5, and Ninja 1.12.1. The build used the Debug configuration and `code/build/`. CTest reported one registered test, `core_foundations`, passed (1/1). The retained execution summary is `code/tests/evidence/first-run-2026-10-20.md`; reproduction commands are in `code/README.md`.

Table 10.4: First Scaffold Check: Observations and Limits

Check	Observed result and limit
Native build	All application and test targets compiled and linked. This establishes a working local toolchain, not product acceptance.
Foundation test	Known coordinate expectations, inverse conversion, invalid mapping configuration, guarded byte edits, stale revisions, overlaps, and atomic rejection passed in the test executable. These cover part of TC-028 and source-edit foundations; zoom/pan and GUI gestures were not tested.
First GUI observation	The launched process reported a responsive `TikZRefinery` window. Vincent opened `boxes-initial.tex` and supplied a screenshot showing the source pane, empty canvas, and graphical-editing-unavailable status. He confirmed that the first display looked good. This was a source-loading observation, not a byte-comparison or save test.
Outstanding checks	Parsing, scene geometry, dragging, filesystem saves, preview compilation, responsiveness measurements, and canvas/PDF agreement remain untested. Proposed product/developer cases retain their existing statuses; a passing build does not complete all of TC-030.

At that checkpoint, rectangle parsing and display were the next proposed increment. The bounded result is recorded below; whole-box dragging and the separated-box contact workflow remain pending.

10.6 Rectangle Parsing and Canvas Display

On October 21, 2026, Vincent authorized the rectangle display increment. The existing Qt/MinGW Debug build succeeded; CTest passed 2/2 entries.
`core_foundations` and `rectangle_display` both passed.
Execution details and reproduction commands are retained in this file:
`code/tests/evidence/rectangle-display-2026-10-21.md`.

The rectangle check covers both box corpus files, numeric byte ranges, revision-scoped IDs, unchanged source bytes, signed/reversed corners, comments, UTF-8/BOM/CRLF, default/explicit strokes, alternate mapping, and refusal of unsupported contexts without partial or stale geometry. The bounding path sets a separate fixed extent and creates no graphics item. The initial scene rectangles are $(200, 200, 100, 100)$ and $(400, 200, 100, 100)$ in (x, y, w, h) form; the touching-reference right rectangle is $(300, 200, 100, 100)$. These implement the one-centimetre dimensions and gap at 100 scene units per centimetre with origin $(200, 300)$ and inverted y.

An automated offscreen window check triggers the actual Open action and file dialog, verifies the source pane, two canvas items, clean state, disabled Save, and display-only status. It does not test native mouse/keyboard gestures.

Both untouched corpus documents compiled with shared TinyTeX. A separate comparison wrapper places each unchanged picture on a fixed 7×5 cm page at the origin. Poppler at 254 dpi and Qt scene rendering both yield 700×500 pixels. No fitted translation or independent tight crop is used. Before execution, the prototype threshold was set to grayscale below 128 for dark edge pixels and a one-pixel counterpart allowance in either direction. Both comparisons reported zero unmatched edge pixels. The retained comparison sheet, JSON metrics, and window image were visually inspected.

These checks support rectangle display and part of TC-028; they do not complete a product case. Loading `boxes-touching.tex` is not an executed drag. TC-025, TC-027, and TC-029 retain their interaction and acceptance work. Selection, dragging, source update on release, save/reopen, zoom/pan, broader parsing, compiler/PDF integration, timing, and human acceptance remain pending. The raster threshold is a prototype check, not an approved product tolerance.

10.7 Line and Arrow Display

Later on October 21, 2026, Vincent prioritized line/arrow display ahead of dragging and reported that the examples did not compile. Both untouched initial files compiled with TinyTeX; the editor's rectangle-only parser had refused their syntax. The parser now recognizes one absolute-centimetre straight segment per drawing command, with optional forward, backward or double-ended default arrowheads. Source bytes and numeric ranges remain intact.

The Debug build succeeded and all three registered CTest entries passed: the existing foundation and rectangle checks, plus `segment_display`. The latter covers all six line/arrow fixtures, independently expected endpoints, arrow flags, revisions and byte ranges, mixed and reversed/vertical geometry, alternate mapping, clean state, refusal behavior and startup-file windows. Curves, polylines, custom tips/styles and transformed contexts remain unsupported. Zero-length segments are rejected; shafts too short for their arrow-tip insets fail display with a diagnostic and an empty scene.

The scene uses the installed PGF default curved `to` tip geometry, round tip strokes and butt-cap shaft shortening, with 72.27 TeX points per inch and 2.54 cm per inch. Each arrow, including both tips, remains one graphics item. Bounding paths still supply a separate comparison extent. An optional command-line source path loads the file on startup; the native application was launched with the initial arrow example already open. Offscreen window checks and retained screenshots cover the same startup route. They do not establish native gesture behavior or Vincent's acceptance.

All eight unchanged corpus files compiled independently. Fixed 7×5 cm comparison pages and the mapped Qt extent produced 700×500 rasters at 100 pixels per centimetre. The prior prototype threshold was retained: grayscale below 128 for dark pixels and a one-pixel counterpart allowance in both directions, without fitted shifts, rescaling or independent tight crops. All eight comparisons reported zero unmatched pixels. The four comparison sheets and initial line/arrow window images were visually inspected.

Execution details, reproduction commands, metrics and image paths are retained in the following evidence record:
`code/tests/evidence/segment-display-2026-10-21.md`.

These results extend display evidence for TC-027 and TC-028; they do not complete edited-output or interaction cases. Selection, dragging, source updates on release, saving and reopening, zoom and pan, integrated compilation and PDF preview, timing and acceptance remain pending.

10.8 Whole-Element Editing and Saved Output

On October 21, 2026, Vincent authorized the next increment: selection and whole-element dragging, source updates on release, cancellation, save/reopen and the two-box contact comparison. The Debug build succeeded; CTest passed all four entries, including the new `whole_element_drag` check.

The test delivers pointer press, move and release events to a Qt Graphics View viewport. Preview moves the selected item while source, revision and dirty state remain unchanged. Release changes the source once and emits one document notification. The right-box move by $(-100, 0)$ scene units produces exactly the expected contact source: corners $(1, 0)$ and $(2, 1)$ cm, unchanged left box, and unchanged sizes. Equivalent line/target-arrow moves produce the expected translated corpus bytes. Separate backward/double-arrow gestures retain direction and unrelated elements.

Escape, focus loss and saving during a preview cancel that proposal; a later release does not commit it. Click-only selection, stale revisions, unsupported endpoint gestures and out-of-extent movement leave source unchanged. A repeated contact gesture at $2\times$ view scale after panning gives the same source displacement. This does not establish future zoom/pan controls or range limits.

Displacement rounds to 0.000001 cm on release. Exact decimal string addition preserves original precision and dimensions; unchanged axes keep their original spelling. A fixture with 18 fractional digits, UTF-8 comments and CRLF checks the guarded byte result. Candidate source and display are validated before applying the real batch. File saving preserves expected bytes, reopening is clean, and an observed external change prevents overwriting the destination. The actual window source pane, modified state and Save action are also checked.

The three actual saved/reopened outputs were independently compiled with TinyTeX. At the existing fixed 7×5 cm extent and 100 pixels per centimetre, all three comparisons had zero unmatched dark pixels using the prior one-pixel allowance in both directions. No fitted shift or independent crop was used. The images and edited-window screenshot were visually reviewed. The native application was launched with a separate demo copy for Vincent; the corpus was preserved.

Retained evidence, saved sources, metrics and reproduction commands: `code/tests/evidence/whole-element-drag-2026-10-21.md`.

These bounded checks support portions of TC-021, TC-022, TC-025, TC-027, TC-028 and TC-029, together with guarded saving. They use Qt viewport events, not operating-system pointer injection. Native human acceptance, endpoint edits, undo/redo, broader syntax, integrated compilation/PDF preview and timing remain pending. Product statuses are unchanged.

10.9 Canvas Contrast Check

On November 5, the rebuilt window displayed the two-box fixture in controlled light and dark Qt palettes. In both captures, black TikZ outlines remained visible on the white canvas. Labels, spacing and the canvas border distinguished it from the source pane. The window and source pane used their respective palettes. The four existing CTests passed after the view change. Captures and the reproduction boundary are recorded in `code/tests/evidence/canvas-contrast-2026-11-05/`. The palette captures used the offscreen platform, not a native-theme acceptance test. No resize or plugin test was executed.

The follow-up fit check uses the same 7 by 5 cm fixture in small and large windows. The actual view scales in both directions and shows all four editing edges with a grey surround. The source and editing extent stay unchanged. The whole-element test also verifies that an out-of-bounds preview stays within the paper and release preserves the source. All four CTests passed. Offscreen captures are retained under `code/tests/evidence/canvas-fit-2026-11-05/`.

Vincent then observed faint drag trails that disappeared on a larger repaint. A clipped raster probe compared the old fills with the corrected view at 1, 1.25 and 1.5 device scales. The old method left 55, 69 and 83 nonwhite pixels on blank paper; the corrected method left zero in all three cases. The probe verified that smoothing remained enabled for item drawing. The four CTests passed. This is bounded evidence for TC-065; native drag observation remains pending.

10.10 Windows Prototype Package Check

On October 23, 2026, a separate Windows x64 Release package was built with the existing CMake application version 0.1.0 and Qt 6.12.0. Qt's existing deployment tool copied runtime dependencies; the package includes an example and manifest. A dedicated process started with the example and a Windows-only PATH, remained running after three seconds, and produced no captured standard-error output. Only that smoke-test process was terminated. The development build and editable demos were preserved. This is bounded startup evidence, not a clean-machine deployment check, a new interaction-test result or product acceptance. The actual package-directory screenshot is in Week Report 8; details are retained in `code/tests/evidence/package-2026-10-23.md`.

10.11 Transaction Contract Verification Plan

Cases TC-034–TC-043 are new planned checks for the October 22 contract. They have not been executed. Existing four-entry CTest results cover the implemented subset only; they do not verify generic transactions, undo/redo, plugin capabilities or integrated compilation. The behavior is defined in the category registers in Chapter 5. The proposed design is in Section 7.8, with responsibilities in Section 7.8.2 and activity in Figure D.7. Cases should retain exact source diffs, history state, revision/notification counts, and the environment used.

10.12 Rectangle Resize Check

On November 12, 2026, all five Debug and Release CTests passed. The bounded resize increment adds `code/tests/ResizeTests.cpp` for TC-060 and updates the boundary release check for TC-064. The fixture independently specifies all four source-coordinate slots, high-precision originals, reversed corners, single/both-axis crossing, 0.01 cm minimum dimensions, thin filler geometry and stroke-aware maximum bounds. Pointer events exercise actual handles, repeated crossing in one gesture, unchanged preview source, cancellation, view transforms and save/reopen. Stale/unsupported/nonfinite requests preserve source; an unchanged constrained release preserves revision. Evidence and reproduction commands are retained under `code/tests/evidence/resize-2026-11-12/`. These checks cover explicit-coordinate outlines only. They do not verify generic undo/redo, plugin loading, integrated compilation, or human acceptance.

10.13 Rectangle Plugin Extraction Check

On November 20–21, both Debug and Release passed six CTests after extraction. The original five suites now exercise rectangle display, movement, corner resizing, crossing, minima, bounds, cancellation, exact source preservation and save/reopen through the loaded module; built-in lines/arrows remain covered. The new `rectangle_plugin_contract` suite loads the actual module path, checks presentation/selection/four handles and independently expected move/resize coordinates, then rejects unknown schemas/operations, stale targets, nonfinite displacement and invalid constraints. A real incompatible-metadata module and a missing module both leave opened source intact without rectangle fallback; a segment-only source still displays. Host proposal tests reject changed identity, appearance, bounds, outcome consistency, fixed corners and move dimensions.

The same suite checks version `0.1.2-alpha`, hash-free title and About's short/full checkpoint, source state and provider identity. A Windows-only-PATH package launch and native capture show the actual UI and loaded DLL. Evidence is under the following retained directories:
`code/tests/evidence/plugin-extraction-2026-11-20/`
`documents/assets/plugin-extraction-2026-11-20/`

This executes TC-061 for the bounded rectangle and portions of TC-067, TC-068, TC-071–TC-074. It does not execute their broader multi-provider, conversion, label, lifecycle or generic-history scenarios and does not establish general product acceptance.

10.14 Deployment Label Fixture

Vincent identified opaque arrow-label backgrounds and border intersections in the November 16 deployment review as useful future label-editing data. An exact pre-fix standalone source is retained at:
`code/tests/fixtures/label-editing/deployment-before-label-fix.tex`. The report diagram uses transparent labels with adjusted placement. This complete TikZ diagram

is outside the current parser's editable subset. It is candidate data for the planned label/provider work, not an executed editor test or evidence that label editing is implemented.

10.15 Sidebar Shell Check

The November 23 `workspace_sidebar` suite exercises the implemented portion of TC-079: persistent strip, icon/accessible identification, page selection, active-tab/header collapse, retained expanded width, disabled creation and unchanged source/dirty state. It checks that window destruction releases the sidebar/pages, covering only the UI lifetime part related to TC-082. Its standalone Files page emits one Open request; the main window connects it to its existing guarded Open method.

All seven Debug and seven Release CTest suites passed for the increment. The older fit/drag test now gives the large editor area its former width plus the sidebar width, preserving the original fit, source, save and geometry assertions. Catalogue alternatives, creation, shared-service ownership and participant validation remain planned. Actual final logs and native captures are retained in `code/tests/evidence/sidebar-shell-2026-11-23/`. This bounded evidence does not mark the full requirements or cases accepted.

10.16 Session History Check

The November 24 session-history suite exercised the bounded `0.1.4-alpha` coordinate editing implementation. All eight Debug and eight Release suites passed. Exact BOM/UTF-8/CRLF/numeric spelling, multi-step move/resize restoration, monotonically increasing revisions and one change notification per accepted action were checked. Save retains entries and changes the dirty baseline only on success; Undo/Redo do not write disk. External-write refusal, failed Open and successful reload were also exercised.

The suite checks redo branching, cancellation/no-op/stale rejection preserving redo, unavailable action states, source/geometry synchronization, failed parse restoration preserving state, and Undo while a pointer gesture is active. A 150-edit model session can be undone completely; this checks absence of an automatic entry limit, not unlimited physical memory or measured memory scaling. Native package review exercised Ctrl+Z/Ctrl+Y, the Edit menu and About identity. A viewport repaint after accepted changes addresses stale foreground handles when restoration leaves the canvas bounds unchanged.

These are bounded portions of TC-035–TC-038, TC-042, TC-043 and TC-050. Generic transactions, deliberate memory-failure injection, file-store doubles and provider-dependent retention remain untested. Participant validation is still planned. Vincent must review the evidence before requirement or verification statuses change. Retained logs, hashes and native captures are at `code/tests/evidence/undo-redo-2026-11-24/`.

Chapter 11

Traceability

This matrix is the current ID-level map from user stories to requirements, use cases, and proposed tests. It makes missing links visible as the registers grow. The matrix records planned coverage, not passing test evidence or stakeholder approval. A cross-cutting code-organization rule may have no direct user story or use case; that relationship is marked explicitly. Requirement and test wording remains in their respective chapters. Developer conventions have no direct user story or use case. The source-module rule retains REQ-A-005 after moving to the Developer register; new DEV rows map to proposed source/design reviews and build checks.

The MVC and class views in Chapter 7 identify their requirement links; Table D.1 in Appendix D maps the supporting activities. Design links do not constitute executed test evidence; the limited first-run evidence is in Section 10.5.

11.1 Milestone 2 Representative Traceability Paths

The full matrices follow and appear in Chapter 12 and Chapter 13. These paths explain how needs shape design; links describe planned coverage rather than a passed acceptance chain.

Table 11.1: Representative Need-to-Design-to-Evaluation Paths

User need	Requirement and design response	Requirement verification	User validation
US-003	REQ-D-001, REQ-A-011; byte-preserving bindings and guarded document acceptance (Figure 7.16).	TC-004, TC-034, TC-042; exact diffs and failure injection.	VAL-003; author continues a realistic source-editing task.
US-007	REQ-D-002, REQ-A-012; document owns history and publishes both states together (Figure D.8).	TC-035, TC-038, TC-056; exact restoration and lifecycle traces.	VAL-007; author explores and recovers layouts confidently.

User need	Requirement and design response	Requirement verification	User validation
US-009, US-010	REQ-F-021, REQ-I-008, REQ-I-009; separate label binding/attachment and point/area selection (Figure 7.18).	TC-044, TC-047, TC-053, TC-054; host preservation and obscured-target checks.	VAL-009, VAL-010; author improves readability and selects hidden objects.

11.2 Traceability Matrix

Table 11.2: Provisional Product Traceability

User story	Requirement / verification	Use case	Test case(s)
US-002	REQ-B-001 VER-001	UC-002	TC-011
US-005	REQ-B-002 VER-002	UC-003	TC-011
US-001	REQ-F-001 VER-003	UC-001	TC-001
US-001	REQ-F-002 VER-004	UC-001	TC-002
US-002	REQ-F-003 VER-005	UC-002	TC-003
US-002, US-003	REQ-F-004 VER-006	UC-002	TC-004, TC-027, TC-029
US-004	REQ-F-005 VER-007	UC-001, UC-002	TC-005, TC-006
US-005	REQ-F-006 VER-008	UC-003	TC-007, TC-008
US-004	REQ-F-007 VER-009	UC-001, UC-002, UC-004	TC-012, TC-039
US-001	REQ-F-008 VER-010	UC-001	TC-013
US-002	REQ-F-009 VER-011	UC-002	TC-021, TC-022
US-002	REQ-F-010 VER-012	UC-002	TC-023, TC-024
US-002	REQ-F-011 VER-013	UC-002	TC-025
US-007	REQ-F-012 VER-014	UC-004	TC-035, TC-037
US-007	REQ-F-013 VER-015	UC-004	TC-038
US-007	REQ-F-014 VER-016	UC-004	TC-036

Provisional Product Traceability (continued)

User story	Requirement / verification	Use case	Test case(s)
US-007	REQ-F-015 VER-017	UC-002, UC-004	TC-037
US-007	REQ-F-016 VER-018	UC-003, UC-004	TC-038
US-007	REQ-F-017 VER-019	UC-001, UC-004	TC-038
US-007	REQ-F-018 VER-020	UC-001, UC-003, UC-004	TC-038
US-007	REQ-F-019 VER-021	UC-004	TC-038
US-007	REQ-F-020 VER-022	UC-001, UC-004	TC-038
US-009	REQ-F-021 VER-023	UC-005	TC-044
US-009	REQ-F-022 VER-024	UC-002, UC-005	TC-045
US-009	REQ-F-023 VER-025	UC-005	TC-046
US-003, US-009	REQ-F-024 VER-026	UC-005	TC-047
US-002	REQ-F-025 VER-073	UC-002	TC-060
US-008	REQ-F-026 VER-081	UC-007	TC-069
US-008	REQ-F-027 VER-082	UC-002, UC-007	TC-070
US-006, US-008	REQ-F-028 VER-083	UC-008	TC-075
US-008	REQ-F-029 VER-084	UC-008	TC-076
US-008	REQ-F-030 VER-085	UC-008	TC-077
US-004, US-008	REQ-F-031 VER-086	UC-008	TC-078
US-004, US-008	REQ-F-032 VER-087	UC-001, UC-002, UC-007	TC-074
US-008, US-011	REQ-F-033 VER-088	UC-009	TC-079
US-003, US-011	REQ-F-034 VER-089	UC-009	TC-080
US-011	REQ-F-035 VER-106	UC-009	TC-080
US-003, US-005	REQ-D-001 VER-027	UC-002, UC-003, UC-009	TC-004, TC-007, TC-021, TC-022, TC-023, TC-024, TC-025, TC-080
US-007	REQ-D-002 VER-028	UC-002, UC-004	TC-034, TC-042
US-007	REQ-D-003 VER-029	UC-002, UC-004	TC-034, TC-036

Provisional Product Traceability (continued)

User story	Requirement / verification	Use case	Test case(s)
US-007	REQ-D-004 VER-030	UC-002, UC-004	TC-035
US-007	REQ-D-005 VER-031	UC-003, UC-004	TC-038
US-007	REQ-D-006 VER-032	UC-003, UC-004	TC-038
US-007	REQ-D-007 VER-033	UC-004	TC-035, TC-037, TC-042
US-007	REQ-D-008 VER-034	UC-002, UC-004	TC-034, TC-040
US-008, US-009, US-010	REQ-D-009 VER-035	UC-002, UC-005, UC-006	TC-048
US-003, US-008, US-009	REQ-D-010 VER-036	UC-002, UC-005	TC-049
US-007	REQ-D-011 VER-037	UC-003, UC-004	TC-050
US-008, US-009, US-010	REQ-D-012 VER-038	UC-005, UC-006	TC-048, TC-053
US-002, US-008	REQ-D-013 VER-090	UC-002	TC-072
US-002, US-008	REQ-D-014 VER-091	UC-002	TC-072
US-002, US-004, US-008	REQ-D-015 VER-092	UC-002	TC-073
US-008	REQ-D-016 VER-093	UC-007, UC-008	TC-068, TC-070
US-003, US-008	REQ-D-017 VER-094	UC-007, UC-008	TC-071
US-008, US-011	REQ-D-018 VER-105	UC-009	TC-079
US-002	REQ-I-001 VER-039	UC-002	TC-026
US-007	REQ-I-002 VER-040	UC-004	TC-043
US-007	REQ-I-003 VER-041	UC-002, UC-004	TC-034, TC-036, TC-042, TC-043
US-007	REQ-I-004 VER-042	UC-002, UC-004	TC-043
US-007	REQ-I-005 VER-043	UC-004	TC-043
US-004, US-007, US-009, US-011	REQ-I-006 VER-044	UC-002, UC-004, UC-005, UC-009	TC-051, TC-080

Provisional Product Traceability (continued)

User story	Requirement / verification	Use case	Test case(s)
US-004, US-009, US-011	REQ-I-007 VER-045	UC-002, UC-005, UC-009	TC-052, TC-080
US-010	REQ-I-008 VER-046	UC-006	TC-053
US-010	REQ-I-009 VER-047	UC-006	TC-054
US-010	REQ-I-010 VER-048	UC-006	TC-055
US-002	REQ-I-011 VER-075	UC-002	TC-062
US-002	REQ-I-012 VER-076	UC-002	TC-063
US-002, US-011	REQ-I-013 VER-077	UC-002, UC-009	TC-064, TC-080
US-002	REQ-I-014 VER-078	UC-002	TC-065
US-008, US-011	REQ-I-015 VER-100	UC-009	TC-079
US-004, US-008	REQ-I-016 VER-101	UC-008	TC-077
US-001, US-004, US-008	REQ-I-017 VER-102	UC-001	TC-081
US-001, US-011	REQ-I-018 VER-107	UC-001, UC-009	TC-079
US-001, US-011	REQ-I-019 VER-108	UC-001, UC-009	TC-079
US-001, US-011	REQ-I-020 VER-109	UC-001, UC-009	TC-079
US-011	REQ-I-021 VER-110	UC-009	TC-080
US-004	REQ-S-001 VER-049	UC-001	TC-009
US-001, US-002	REQ-A-001 VER-050	UC-001, UC-002	TC-014
US-001	REQ-A-002 VER-051	UC-001	TC-015
US-001	REQ-A-003 VER-052	UC-001	TC-016
US-001, US-002	REQ-A-004 VER-053	UC-001, UC-002	TC-018
US-002, US-003	REQ-A-006 VER-054	UC-002	TC-020
US-002, US-003	REQ-A-007 VER-055	UC-002	TC-028, TC-029
US-007, US-008	REQ-A-008 VER-056	UC-002, UC-004	TC-034, TC-040, TC-042

Provisional Product Traceability (continued)

User story	Requirement / verification	Use case	Test case(s)
US-008	REQ-A-009 VER-057	UC-001, UC-002	TC-041
US-007	REQ-A-010 VER-058	UC-002, UC-004	TC-039
US-007	REQ-A-011 VER-059	UC-002, UC-004	TC-034, TC-040, TC-042
US-007, US-009	REQ-A-012 VER-061	UC-002, UC-004, UC-005	TC-056
US-007, US-009	REQ-A-013 VER-062	UC-002, UC-003, UC-004, UC-005	TC-056
US-005, US-007	REQ-A-014 VER-063	UC-001, UC-003, UC-004	TC-057
US-008, US-009, US-010	REQ-A-015 VER-064	UC-001, UC-002, UC-005, UC-006	TC-058
US-008, US-009	REQ-A-016 VER-065	UC-002, UC-005	TC-059
US-008, US-009	REQ-A-017 VER-066	UC-002, UC-005	TC-058
US-008	REQ-A-018 VER-074	UC-002	TC-061
US-008	REQ-A-019 VER-095	UC-007	TC-068, TC-071
US-008	REQ-A-020 VER-096	UC-007, UC-008	TC-071
US-008	REQ-A-021 VER-097	UC-002, UC-008	TC-076, TC-077
US-008	REQ-A-022 VER-098	UC-008	TC-078
US-002, US-008	REQ-A-023 VER-099	UC-002	TC-060, TC-072
US-008, US-011	REQ-A-024 VER-111	UC-002, UC-009	TC-082
Cross-cutting	REQ-A-005 VER-060	*Cross-cutting*	TC-019
Cross-cutting	REQ-DEV-001 VER-067	*Cross-cutting*	TC-030
Cross-cutting	REQ-DEV-002 VER-068	*Cross-cutting*	TC-031
Cross-cutting	REQ-DEV-003 VER-069	*Cross-cutting*	TC-032
Cross-cutting	REQ-DEV-004 VER-070	*Cross-cutting*	TC-033
Cross-cutting	REQ-DEV-005 VER-079	*Cross-cutting*	TC-066
Cross-cutting	REQ-DEV-006 VER-080	*Cross-cutting*	TC-067

Provisional Product Traceability (continued)

User story	Requirement / verification	Use case	Test case(s)
Cross-cutting	REQ-DEV-007 VER-103	*Cross-cutting*	TC-067
Cross-cutting	REQ-DEV-008 VER-104	*Cross-cutting*	TC-067
US-002	REQ-P-001 VER-071	UC-002	TC-017
US-006	REQ-C-001 VER-072	UC-001, UC-002, UC-003	TC-010

11.3 Reverse Coverage

Every requirement row points to its verification entry; every test points to the verification entries it supports. Verification rows name requirements and tests, and validation rows name their stories. The story register provides reverse validation links. Developer rules remain explicitly cross-cutting rather than acquiring invented user stories.

Chapter 12

Verification

12.1 Purpose and Evidence Boundary

Verification checks each stated requirement against its acceptance criterion. Validation separately asks whether users can accomplish their intended tasks (Chapter 13). The register below is the full requirement-to-check map, not a list of passed requirements. Every requirement links back here; every test links to the verification rows it supports. Vincent reviews the evidence before changing requirement status.

All rows remain Planned: bounded prototype checks support parts of the design, but no complete product requirement has been signed off. Expected evidence is specified per row and by Chapter 10. A blocked prerequisite is recorded as Blocked, never converted into a passing result.

12.2 Full Requirement Verification Map

Table 12.1: Requirement-to-Verification Map

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-001	REQ-B-001	Inspection / test TC-011	A representative supported element reaches the preselected target through the GUI.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-002	REQ-B-002	Inspection / test TC-011	The saved revision opens in a standard editor and compiles externally with the intended change.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-003	REQ-F-001	Inspection / test TC-001	Read source matches the selected file exactly; opening changes zero input bytes.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-004	REQ-F-002	Inspection / test TC-002	The displayed compiled result corresponds to the selected supported source and successful compiler run.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-005	REQ-F-003	Inspection / test TC-003	The selected supported object reaches the specified target position.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-006	REQ-F-004	Inspection / test TC-004, TC-027, TC-029	Only intended source coordinates change; canvas and fixed-extent output satisfy the declared comparison oracle.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-007	REQ-F-005	Inspection / test TC-005, TC-006	Each invalid/unsupported fixture reports the failure and retains exact source.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-008	REQ-F-006	Inspection / test TC-007, TC-008	A successful save reopens with exact accepted bytes; failed writes retain the unsaved revision.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-009	REQ-F-007	Inspection / test TC-012, TC-039	After a failed new compilation, the previous successful preview remains explicitly marked older.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-010	REQ-F-008	Inspection / test TC-013	Available local dependencies are included; a missing dependency is named and no false current preview appears.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-011	REQ-F-009	Inspection / test TC-021, TC-022	Both endpoints translate by the same specified vector; arrow direction and unrelated bytes are preserved.	Dated run record; expected/actual outputs and source/state differences.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-012	REQ-F-010	Inspection / test TC-023, TC-024	Only the selected endpoint changes; the opposite endpoint and unrelated source remain fixed.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-013	REQ-F-011	Inspection / test TC-025	The rectangle translates to the target while both dimensions remain unchanged.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-014	REQ-F-012	Inspection / test TC-035, TC-037	One complete action per undo/redo; restored source equals the expected snapshot byte for byte.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-015	REQ-F-013	Inspection / test TC-038	History is session-only; saved source embeds no history and application restart restores none.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-016	REQ-F-014	Inspection / test TC-036	Undo during an active drag first cancels the preview, then restores the last completed action.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-017	REQ-F-015	Inspection / test TC-037	Only a newly accepted changed edit discards the old redo branch.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-018	REQ-F-016	Inspection / test TC-038	Successful Save changes no existing undo/redo entries or position.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-019	REQ-F-017	Inspection / test TC-038	Successful document replacement creates empty history for the newly loaded document.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-020	REQ-F-018	Inspection / test TC-038	Cancelled/failed I/O preserves state, including a prior explicit successful Save.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-021	REQ-F-019	Inspection / test TC-038	Undo/redo changes in-memory state only; disk bytes remain unchanged until an explicit successful Save.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-022	REQ-F-020	Inspection / test TC-038	Cancel or failed requested Save prevents Open/Reload from replacing current state.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-023	REQ-F-021	Inspection / test TC-044	The label reaches the target through supported relative-placement edits; host coordinates and unrequested text, anchor, rotation, alignment and style bytes are unchanged.	Dated run record; expected/actual outputs and source/state differences.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-024	REQ-F-022	Inspection / test TC-045	The host translates once and the label follows its retained attachment rule; no stale reference is used.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-025	REQ-F-023	Inspection / test TC-046	Every drag retains attachment and host geometry. Detachment requires a distinct explicit request; unsupported detachment performs no rewrite. Implementation of detachment remains later work.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-026	REQ-F-024	Inspection / test TC-047	The supported move changes only local placement; shared definitions and unrelated elements are byte-identical. The unsupported move is unavailable with a reason and no state change.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-027	REQ-D-001	Inspection / test TC-004, TC-007, TC-021, TC-022, TC-023, TC-024, TC-025, TC-080	Zero unintended byte changes outside approved edits, including UTF-8 comments and line endings.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-028	REQ-D-002	Inspection / test TC-034, TC-042	Every injected prepublication failure leaves source, history entries and position unchanged.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-029	REQ-D-003	Inspection / test TC-034, TC-036	Exactly one history entry for a changed accepted action; zero for noncommits.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-030	REQ-D-004	Inspection / test TC-035	Each accepted changed edit/restoration advances revision; old requests remain stale.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-031	REQ-D-005	Inspection / test TC-038	Saved baseline equals bytes actually written; failure changes no baseline bytes.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-032	REQ-D-006	Inspection / test TC-038	Dirty is true exactly when current and saved source bytes differ.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-033	REQ-D-007	Inspection / test TC-035, TC-037, TC-042	Undo/redo moves the existing history position without appending an entry.	Dated run record; expected/actual outputs and source/state differences.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-034	REQ-D-008	Inspection / test TC-034, TC-040	Every proposal includes identity/revision/description and pre-edit byte ranges; insert/delete/replace use the declared semantics.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-035	REQ-D-009	Inspection / test TC-048	The element representation must separate a revision-scoped ElementDescription (identity, shape family/type, schema version, interpreted geometry, appearance and relevant content) from SourceBinding, presentation/selection data, attachments and per-element available operations.	Source/contract inspection and linked execution records when implemented.	Planned
VER-036	REQ-D-010	Inspection / test TC-049	Source bindings, provider bindings and attachment references must identify their document and revision; accepted edits shall invalidate old records and rebuild relationships from accepted source.	Source/contract inspection and linked execution records when implemented.	Planned
VER-037	REQ-D-011	Inspection / test TC-050	Exact snapshots support restoration in the session without automatic eviction in the initial prototype. Save retains history without serializing it; successful replacement/restart starts empty history. Failure paths retain current state.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-038	REQ-D-012	Inspection / test TC-048, TC-053	Each selectable element must provide selection geometry and a readable description independent of source spelling and graphics-item pointers; presentation coordinates shall be distinct from viewport pixels.	Source/contract inspection and linked execution records when implemented.	Planned
VER-039	REQ-I-001	Inspection / test TC-026	Supported vector shapes redraw at declared test zooms without enlarged bitmap artifacts.	Dated run record; expected/actual outputs and source/state differences.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-040	REQ-I-002	Inspection / test TC-043	Undo/Redo is disabled exactly when its corresponding entry is unavailable.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-041	REQ-I-003	Inspection / test TC-034, TC-036, TC-042, TC-043	One successful change notification per changed acceptance/restoration and zero for noncommits.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-042	REQ-I-004	Inspection / test TC-043	Post-acceptance parse and canvas describe the accepted source revision.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-043	REQ-I-005	Inspection / test TC-043	Injected restoration failure is visible and leaves source/history unchanged.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-044	REQ-I-006	Inspection / test TC-051, TC-080	Changed returns Applied; byte-equivalent returns Unchanged; failure returns Rejected with a reason. Cancellation/no-op shows no error; only the changed action records history.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-045	REQ-I-007	Inspection / test TC-052, TC-080	No preview update produces a rejection banner. One rejected attempt produces one red-accented bottom strip with bright red icon and reason; it stays until dismissal or the next completed operation. Canvas remains usable.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-046	REQ-I-008	Inspection / test TC-053	Label wins ordinary hit selection. Every intersecting element appears once with a readable description; hovering identifies it and the chosen obscured host can be moved without moving blockers first.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-047	REQ-I-009	Inspection / test TC-054	Both gestures show a retained rectangle; candidates include partial intersections and obscured objects. Region creation makes no source/history change or automatic group selection; chosen entry selects one element and Escape clears the region.	Dated run record; expected/actual outputs and source/state differences.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-048	REQ-I-010	Inspection / test TC-055	Candidate inclusion follows geometry intersection with the 6-pixel-radius screen circle. Zoom does not change its logical screen size. Record user feedback before changing the provisional value; these zooms are test inputs, not a product range claim.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-049	REQ-S-001	Inspection / test TC-009	Captured compiler arguments contain no shell-escape enablement without explicit user configuration.	Dated run record; expected/actual outputs and source/state differences.	Planned
VER-050	REQ-A-001	Inspection / test TC-014	An unrelated UI interaction completes while slow compilation is still running; compilation is outside the UI thread.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-051	REQ-A-002	Inspection / test TC-015	Compilation adds zero intermediate files to the opened document directory.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-052	REQ-A-003	Inspection / test TC-016	Compiler tests operate with a substituted display component and no concrete PDF-widget dependency.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-053	REQ-A-004	Inspection / test TC-018	Each concern has a documented responsible component and code follows the stated boundaries.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-054	REQ-A-006	Inspection / test TC-020	Interpretation and edit tests complete without starting Widgets or LaTeX.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-055	REQ-A-007	Inspection / test TC-028, TC-029	Known points, inverse mappings and equivalent zoom/pan gestures match independent expectations.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-056	REQ-A-008	Inspection / test TC-034, TC-040, TC-042	Built-in/provider insertions, deletions and replacements use the same atomic reversible entry path.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-057	REQ-A-009	Inspection / test TC-041	Element-only, syntax-only and combined providers participate through the same source/history contract.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-058	REQ-A-010	Inspection / test TC-039	A later compiler failure preserves the accepted source and existing history entry.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-059	REQ-A-011	Inspection / test TC-034, TC-040, TC-042	Wrong identity/revision/bytes, bounds/conflicts and unsupported rules are rejected before publication.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-060	REQ-A-005	Inspection / test TC-019	Each primary project class has its matching source module; helpers have a documented subordinate role.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-061	REQ-A-012	Inspection / test TC-056	Controller owns edit service/file store and only coordinates; document owns history. Preparation cannot mutate live state. A successful edit/restoration updates source and history together; injected failure changes neither.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-062	REQ-A-013	Inspection / test TC-056	Controller owns edit service/file store and only coordinates; document owns history. Preparation cannot mutate live state. A successful edit/restoration updates source and history together; injected failure changes neither.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-063	REQ-A-014	Inspection / test TC-057	File store touches no history. Only successful I/O permits corresponding load/baseline updates; baseline equals written bytes. Failed operations preserve state, including any preceding explicit successful Save.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-064	REQ-A-015	Inspection / test TC-058	Syntax providers must interpret source and prepare bound rewrites. Shape providers shall consume shared descriptions and return presentation, selection and interpreted operation proposals, without directly mutating document, history, scene or files; the scene shall create and own graphics items.	Source/contract inspection and linked execution records when implemented.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-065	REQ-A-016	Inspection / test TC-059	Existing bindings route to the same responsible instances throughout the gesture; stale requests fail without mutation. Missing-provider behavior follows TC-074.	Source/contract inspection and linked execution records when implemented.	Planned
VER-066	REQ-A-017	Inspection / test TC-058	An element operation must be available only when the shape provider, syntax/source binding and current document context support it; the host shall check both when enabling the interaction and when processing its request, and explain unavailable operations.	Source/contract inspection and linked execution records when implemented.	Planned
VER-067	REQ-DEV-001	Inspection / test TC-030	All project class/file references match the agreed tkr naming convention and build.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-068	REQ-DEV-002	Inspection / test TC-031	Inspected methods and variables match the coding guide, declared types and coordinate spaces.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-069	REQ-DEV-003	Inspection / test TC-032	Every project-defined inheritance path has at most three levels, excluding vendor ancestry.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-070	REQ-DEV-004	Inspection / test TC-033	Every owned/borrowed object has one consistent lifetime mechanism; borrowers cannot outlive references.	Interface/code/diagram review plus linked run record and failure-path evidence.	Planned
VER-071	REQ-P-001	Inspection / test TC-017	Geometry updates without compilation; pointer movement, pause and release launch none. A successful Save requests compilation of the saved revision; failed Save requests none.	Source/contract inspection and linked execution records when implemented.	Planned
VER-072	REQ-C-001	Inspection / test TC-010	Open/preview/edit/save completes with networking disabled and no network-service dependency.	Dated run record; expected/actual outputs and source/state differences.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-073	REQ-F-025	Inspection / test TC-060	All four corners, crossing, minimum dimensions and stroke-aware limits match independent expected results. Only intended coordinate bytes change; cancellation, stale requests and no-op preserve source/revision.	Dated gesture run, source diff and saved/reopened fixture.	Planned
VER-074	REQ-A-018	Inspection / test TC-061	The actual loaded provider supplies rectangle geometry/presentation and resize behavior through the common API. Baseline behavior is retained; missing/incompatible modules fail explicitly without a hidden rectangle implementation.	November 20 module-load, call-site and six-suite Debug/Release evidence; bounded execution recorded, product sign-off pending.	Planned
VER-075	REQ-I-011	Inspection / test TC-062	Pane labels and separation are visible, and the black outlines remain legible.	Screenshots and user inspection.	Planned
VER-076	REQ-I-012	Inspection / test TC-063	The entire canvas remains visible at both sizes, with unchanged proportions, source bytes and edit limits.	Viewport observations and source comparison.	Planned
VER-077	REQ-I-013	Inspection / test TC-064, TC-080	When a document declares a fixed editing extent, the editor must distinguish it visually and constrain move, resize and creation preview and acceptance to valid geometry inside it, including stroke; reaching a limit must not be reported as rejection.	Before/after images and source comparison.	Planned
VER-078	REQ-I-014	Inspection / test TC-065	Only the current outline and any selection marks remain visible; the drag path contains no residual marks.	Incremental repaint check and native drag observation.	Planned
VER-079	REQ-DEV-005	Inspection / test TC-066	Numeric literals in project-owned application code are confined to named definitions. Names identify the purpose and units where applicable.	Dated source-review record and any findings.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-080	REQ-DEV-006	Inspection / test TC-067	Feature/revision changes and `-alpha` labels follow policy. The built baseline is truthful even when the document advances; folder/title omit hashes, About includes short/full identity and source state, and historical artifacts are unchanged.	Source/contract inspection and linked execution records when implemented.	Planned
VER-081	REQ-F-026	Inspection / test TC-069	The application must present duplicate declared plugin IDs with candidate name, version and file location, and let the author choose which candidate to activate; conflicting candidates shall not be selected by discovery order.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-082	REQ-F-027	Inspection / test TC-070	The application must allow compatible providers for the same shape family to coexist, bind each element to its chosen provider instance/version, and preserve that binding during an operation.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-083	REQ-F-028	Inspection / test TC-075	Available compatible plugin updates must normally be applied at application startup before diagrams are opened or restored; an unavailable update service shall not prevent the local editing workflow.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-084	REQ-F-029	Inspection / test TC-076	During a session the application must prepare an update in the background and activate it only after validation and a fresh check that its provider is unused.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-085	REQ-F-030	Inspection / test TC-077	An update discovered for an in-use provider must leave that version active and let the author defer the update or restart through the unsaved-work guard; deferred updates shall be considered at the next startup.	Contract inspection, prepared test data and dated execution record when implemented.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-086	REQ-F-031	Inspection / test TC-078	Plugin installation and activation must retain the previous working version until replacement succeeds; failure shall preserve document source and provider choices and report the reason.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-087	REQ-F-032	Inspection / test TC-074	A missing or incompatible provider must preserve source, identify unavailable support and its reason, and shall not silently substitute another provider.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-088	REQ-F-033	Inspection / test TC-079	The application must enumerate catalogue entries offered by active compatible providers, with shape family/type, readable name, preview and creation support, while distinguishing provider alternatives.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-089	REQ-F-034	Inspection / test TC-080	The application must create supported elements using a selected catalogue/provider entry and a supported syntax-insertion route; host validation and document acceptance must preserve unrelated source and assign accepted element identity.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-090	REQ-D-013	Inspection / test TC-072	Operation requests must identify host-defined common operations such as `tkr.move` and `tkr.resize`, or provider-qualified custom operations; parameters and their schema version shall be separate fields.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-091	REQ-D-014	Inspection / test TC-072	An existing-element edit request must contain target document/element identity and revision, the initial element description, operation-specific parameters and host constraints in document centimetres, without mouse events, widget pointers or source byte ranges.	Contract inspection, prepared test data and dated execution record when implemented.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-092	REQ-D-015	Inspection / test TC-073	Shape preparation must return Prepared change with proposed interpreted changes, Unchanged, or Cannot prepare with a reason; it shall not claim that source or history has been applied.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-093	REQ-D-016	Inspection / test TC-068, TC-070	Declared stable plugin/provider identity must be distinct from the host-assigned runtime instance and loaded-version identity used by element bindings and outstanding operations.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-094	REQ-D-017	Inspection / test TC-071	A replacement provider must accept existing element data or use an explicitly defined validated conversion preserving relevant information and used capabilities; otherwise the previous provider shall remain available and no implicit lossy conversion shall occur.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-095	REQ-A-019	Inspection / test TC-068, TC-071	Provider registration must declare stable ID, display name, release version, element/syntax role, supported types and schemas, API compatibility and supported operations; incompatible registrations shall remain inactive with a reason.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-096	REQ-A-020	Inspection / test TC-071	Provider compatibility must check the API contract, element-data schema and required capabilities; the first rectangle extraction shall require the explicitly supported API version and matching documented build environment.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-097	REQ-A-021	Inspection / test TC-076, TC-077	A provider must remain available while any open document, active gesture or pending operation depends on it; activation or unloading shall recheck those dependencies at a safe point.	Contract inspection, prepared test data and dated execution record when implemented.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-098	REQ-A-022	Inspection / test TC-078	Plugin updates must be associated with an explicit update origin and installed and validated separately from the active version before activation; detailed distribution, authentication and recovery mechanisms require later design.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-099	REQ-A-023	Inspection / test TC-060, TC-072	Preview and release must calculate shape proposals from initial gesture geometry and total document-space displacement under the same constraints, rather than accumulate changes from earlier previews.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-100	REQ-I-015	Inspection / test TC-079	The Shapes page must show provider-distinguishable previews and names within the collapsible workspace sidebar, and expose creation only when both the shape and syntax-insertion route support it.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-101	REQ-I-016	Inspection / test TC-077	An update notification for an in-use provider must be nonmodal and offer Later or Restart and update; restart shall respect Save, Discard and Cancel without interrupting an active edit by replacing its provider.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-102	REQ-I-017	Inspection / test TC-081	When compiler/PDF integration exists, a diagram missing graphical provider support must be viewable through a successful read-only compilation when dependencies are available; the displayed result shall identify its source revision and any stale status.	Contract inspection, prepared test data and dated execution record when implemented.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-103	REQ-DEV-007	Inspection / test TC-067	Normal application titles and package-folder labels must show the implemented baseline as `0.feature.revision-alpha` without a commit hash; later report revisions shall not relabel an existing package.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-104	REQ-DEV-008	Inspection / test TC-067	About must show the implemented baseline followed by a Git-generated shortened checkpoint, retain the full checkpoint in details and build metadata, and identify clean, modified or unavailable source state truthfully.	Contract inspection, prepared test data and dated execution record when implemented.	Planned
VER-105	REQ-D-018	Inspection / test TC-079	Catalogue entries must describe offered shape types, readable names, preview data and creation support separately from document elements; the host must associate provider identity and version.	Planned source/UML inspection and dated execution after implementation; no current pass claimed.	Planned
VER-106	REQ-F-035	Inspection / test TC-080	The editor must support click placement using provider defaults and rectangle drag placement using opposite corners; clicked rectangles must be centred where valid and shifted inward without changing default size to fit a declared editing extent.	Planned source/UML inspection and dated execution after implementation; no current pass claimed.	Planned
VER-107	REQ-I-018	Inspection / test TC-079	The workspace sidebar must collapse to a visible tab strip through which its pages can be reopened.	Bounded shell checks: Section 10.15; broader TC-079/catalogue acceptance remains planned.	Planned
VER-108	REQ-I-019	Inspection / test TC-079	Shapes and Files tabs must have distinct icons, readable identification and clear active-state feedback.	Bounded shell checks: Section 10.15; broader TC-079/catalogue acceptance remains planned.	Planned
VER-109	REQ-I-020	Inspection / test TC-079	The sidebar must retain its selected page and previous expanded width during the window session.	Bounded shell checks: Section 10.15; broader TC-079/catalogue acceptance remains planned.	Planned

Check ID	Requirement	Method / tests	Measurable criterion	Planned evidence	Status
VER-110	REQ-I-021	Inspection / test TC-080	After accepted creation, the editor must select the new element and return to selection; Escape must cancel, while rejected placement must retain the creation tool for retry and report its reason once per attempt.	Planned source/UML inspection and dated execution after implementation; no current pass claimed.	Planned
VER-111	REQ-A-024	Inspection / test TC-082	The main window must own shared plugin services and the catalogue; consumers must borrow access and be released before the services on which they depend.	Planned source/UML inspection and dated execution after implementation; no current pass claimed.	Planned

12.3 Evidence Already Obtained

The November 23 sidebar shell adds seven-suite Debug/Release execution and native UI evidence for the bounded collapse, tab identification and session-state behaviour in VER-107–VER-109; see Section 10.15. Catalogue discovery, creation and the shared-service refactor remain planned; requirement statuses are unchanged.

The rectangle extraction adds bounded execution evidence for VER-074 and rectangle-specific portions of VER-080, VER-090–VER-096, VER-099, VER-103 and VER-104; see Section 10.13. Their wider scenarios and product acceptance remain pending; a six-suite pass does not mark the full requirement register verified. These dated records are separate from the planned product register. They are not new executions for this milestone and do not establish native acceptance.

Table 12.2: Existing Bounded Prototype Evidence

Date / check	Observed result / retained record	Coverage boundary
October 20–22: native build / CTest	Six-class scaffold built; four registered tests passed on October 22. See Section 10.5 and `code/README.md`.	Foundation, mapping, parsing/display and whole-element editing checks; no generic history/provider implementation.
October 21: display	Eight fixed-extent TinyTeX comparisons: zero unmatched dark edge pixels with one-pixel allowance at 100 pixels/cm. Section 10.7.	Explicit cm rectangles and straight segments with default tips; not arbitrary TikZ or text rendering.
October 21: saved edits	Three actual saved outputs compiled and matched the prototype comparison; byte edits, cancellation and observed external-write refusal checked. Section 10.8.	Whole-element movement only; endpoint handles and integrated PDF preview remain unfinished.
October 23: package	Application 0.1.0 Windows x64 package stayed running for three seconds with Windows-only PATH and no captured stderr. Section 10.10.	Dependency/startup smoke check only; not a clean-machine deployment, full acceptance or controlled release.

Chapter 13

Validation

13.1 Stakeholder Method and Acceptance

Validation checks whether the solution addresses a user need in context. Professor Muresan is the identified initial external stakeholder; an available TikZ author may also participate. Vincent prepares the scenarios and records feedback. Owner agreement with a design is not a completed participant task. All scenarios below are Planned; availability and dates remain to be arranged.

Before a session, agree representative diagrams and target outcomes, record toolchain and revision, and confirm which features are implemented. Ask the participant to attempt each task with minimal coaching; record every prompt or assistance. Success requires the stated observable result and explicit participant confirmation of usefulness. A need may fail validation even when all associated requirements pass verification. Capture negative feedback as a design action, then repeat the affected task after correction. No claimed time saving, satisfaction score or completed user study is invented.

A scenario is complete only when every acceptance condition for its linked story is exercised. If a required feature or agreed comparison setting is unavailable, record the scenario as Blocked or Partial with the missing checks; a bounded demonstration does not establish full story validation.

13.2 Full User-Story Validation Map

Acceptance criteria below operationalize the acceptance examples in the story register. Every story links to its row and each row links back to that story. Numerical performance targets remain subject to stakeholder review.

Table 13.1: User-Story-to-Validation Map

Validation	Story	Scenario	Acceptance criterion	Method / planned evidence	Status
VAL-001	US-001	Open a selected supported source using Open and, once its interactions are defined, the Files page; exercise the shared Save/Discard/Cancel guard. Identify the compiled result and source revision.	Participant identifies the selected drawing correctly; original input is byte-identical and participant confirms that preview supports choosing a refinement.	Observe task and compare source/preview; retain screen record, file hashes, participant response and unresolved issues.	Planned
VAL-002	US-002	Place a line, arrow and box at preselected targets; adjust one endpoint and resize the box to a specified size, check the editing boundary and drag feedback at two window sizes in both palettes, inspect shapes at the declared zoom settings, and compare the saved diagram with independently compiled output.	Targets, resized dimensions and geometry intended to stay fixed match the task. Participant can identify the fitted editing area, sees clean drag feedback, and confirms that shapes remain clear under zoom, the compiled layout matches the intended edits, and graphical refinement is useful without manually finding coordinates.	Observe the complete task; retain zoom settings, before/after screenshots, source diffs, saved source and external compiler output, assistance and participant feedback. Fix comparison settings before execution.	Planned
VAL-003	US-003	Refine an object in a diagram containing comments and unrelated supported or preserved source.	No unrelated bytes change; participant can continue source editing and confirms that the result retains the original authoring intent.	Joint source-diff review and realistic follow-up edit; retain annotated diff and participant judgment.	Planned
VAL-004	US-004	Encounter unsupported syntax and a failed preview/edit, then recover or continue working.	Participant correctly explains which operation failed and what source remains current, can locate the reason, and loses no accepted work.	Observe recovery and ask participant to identify current state; retain messages, state hashes and response.	Planned
VAL-005	US-005	Save a graphical correction, reopen it in a standard editor and compile outside TikZRefinery.	External source remains editable, compiles with the intended correction, and participant confirms it fits the existing LaTeX workflow.	Demonstrate external workflow; retain saved file, compiler result, screenshots and feedback.	Planned

Validation	Story	Scenario	Acceptance criterion	Method / planned evidence	Status
VAL-006	US-006	Complete open, preview, edit and save on a configured computer with networking disabled.	All four task stages complete without network service and participant confirms the workflow meets the offline/private-document need.	Observe disconnected task; retain network condition, task record and participant confirmation.	Planned
VAL-007	US-007	Explore two layouts, undo/redo, save, undo again and identify saved versus current state.	Participant recovers the intended layout, predicts dirty state correctly, and confirms confidence in recovering edits; snapshots are exact.	Observe scenario and ask for state explanation; retain history/state trace and participant feedback.	Planned
VAL-008	US-008	Use additional-element/alternate-syntax providers, compare shared catalogue entries, choose rectangle variants, resolve a duplicate-ID choice and respond to an in-use update notification.	The author distinguishes providers and capabilities, retains intended source and active bindings, and understands deferral and unavailable-operation reasons.	Observe author tasks and developer setup; retain choices, source diffs and feedback. Automatic updater execution awaits implementation.	Planned
VAL-009	US-009	Reposition an attached label in both axes and move its host afterward.	Participant improves readability without altering host geometry during label movement; attachment and unrequested appearance persist, and participant confirms the interaction meets the need.	Observe label/host tasks and review source diff; retain positions, relationship check and feedback.	Planned
VAL-010	US-010	Choose an obscured host among at least three overlapping elements, using point and area menus.	Participant identifies and selects the requested object without moving blockers, understands highlighting, and judges the method usable.	Observe both selection routes; retain candidate lists, highlights, assistance and participant response.	Planned

Validation	Story	Scenario	Acceptance criterion	Method / planned evidence	Status
VAL-011	US-011	In a later supported creation context, identify sidebar icons, collapse/reopen Shapes and compare entries; click-place and drag-size, reach bounds, cancel, reject/retry, then accept/save/reopen.	The author distinguishes providers and tabs, understands the retained sidebar state and preview, preserves editable source, and sees accepted creation selected in normal selection mode. New-document/insertion prerequisites must be settled before executing the complete scenario.	Observe the complete later workflow; retain source, external compilation, assistance and author feedback. Persona validation and all creation execution remain Planned.	Planned

Appendix A

Document History

The newest revision appears first. From November 19, 2026 the project uses `0.feature.revision-alpha`. Completed document updates and code refinements advance revision; a completed feature release advances feature and resets revision to zero. Weekly reports no longer determine the middle number. This report starts the new convention at `0.1.0-alpha`; rectangle extraction implements `0.1.2-alpha`. The later `0.2.0-alpha` milestone is shape discovery and creation, not a delivered feature.

The initial report `0.1.3-alpha` added planned design without rebuilding the retained packages. Vincent subsequently rejected an identity-only demo rebuild and explicitly requested implementation at this baseline. The November 23 sidebar shell and matching rebuilt package now supply actual new behavior; the earlier report PDFs are retained as a dated baseline. Each application build retains the architectural baseline it implements. The document may advance without relabelling that build. Package folders and normal titles omit commit hashes; About supplies the version plus shortened Git identity, full-hash details and truthful source state. The frozen `0.10.3` package and matching reports retain their original labels. The new 0.1.2-alpha package implements the title/About rule. The frozen executable retains its earlier format.

Historical `1.x` and weekly-based `0.10.x` entries are preserved. Vincent authorized their earlier conventions and the targeted `1.8.1.1` assignment correction. The new baseline is an explicit numbering transition, not a claim of version-order continuity or product acceptance. **Author initials:** VS — Vincent Sun; CDX — OpenAI Codex. AI-assisted contributions remain subject to Vincent Sun's review and academic responsibility.

Table A.1: Document Update History

Date / Version	Author	Change	Location
November 24, 2026 / 0.1.4-alpha Front-matter lists	VS, CDX	Reviewed Vincent's earlier-course Overleaf template and the retained Cornell sources. Adopted Contents, List of Tables, List of Figures order with generated captions, numbering and page links; captioned fourteen existing tables so they enter the list. Retained the companion diagram source/status index and refreshed its physical-page locations. Continued the same baseline; no application or package change.	Tables; Figures; RPT-049
November 24, 2026 / 0.1.4-alpha Diagram audit	VS, CDX	Reconciled active diagrams with current code and reviewed design: current 3D deployment, candidate/history acceptance, Open/Save history effects and typed lifelines. Added current restoration sequence and maintained diagram navigation. Preserved old evidence, application/package bytes and this version by Vincent's explicit instruction; code rename and broader features remain pending.	Deployment; Restoration; Figures
November 24, 2026 / 0.1.4-alpha Submission preparation	VS, CDX	Moved the full revision history to an appendix following Vincent's account of instructor feedback; retained a distinct front-matter version paragraph and direct link. Refreshed Demo 3 for this Alpha, with the live demonstration after progress and current history/class evidence. Recorded Milestone 3 coverage; source-control commit and assignment submission remain pending. No application change or package relabelling.	Version; Milestone 3

Continued on next page

Date / Version	Author	Change	Location
November 24, 2026 / 0.1.4-alpha Code/report	VS, CDX	Implemented bounded session undo/redo against this architectural baseline; document-owned exact snapshots, atomic prepared-state acceptance and standard actions. Approved no automatic eviction pending measurements; added an eighth suite and matching package. Corrected stale-handle repainting; preserved the pre-implementation report baseline and older packages.	Checks; History class
November 24, 2026 / 0.1.4-alpha	VS, CDX	Reconciled agreed, unimplemented undo/redo: corrected ownership/class labels, expanded document operations, separated new-edit/restoration sequences and clarified provider/compile-on-save boundaries. Strengthened AI-REQ-017 and diagram-content checks. Preserved matching 0.1.3-alpha reports, app and Demo 3; no code or new runtime results.	Design; sequence; Chapter 5
November 23, 2026 / 0.1.3-alpha	VS, CDX	Implemented the authorized sidebar shell and refreshed its demo package; creation/folder browsing and shared services remain planned. Added three bounded UI classes, a seventh test suite and current UML/UI evidence. Preserved earlier packages and pre-shell reports.	Shell; checks
November 22–23, 2026 / 0.1.3-alpha	VS, CDX	Documented reviewed catalogue, click/drag creation, collapsible sidebar and planned window-owned shared services. Added seven Open requirements and one planned test with reciprocal maps; retained current implementation views. Renamed the displayed AI register to Directives and added the bidirectional impact-review rule. Report date remains November 22; document implementation/QA occurred November 23. Packages/code are unchanged.	Sidebar; Maps; internal AI records; internal AI records

Continued on next page

Date / Version	Author	Change	Location
November 20–21, 2026 / 0.1.2-alpha Code and report	VS, CDX	Implemented the separate rectangle DLL and value-record API, metadata/build compatibility, host proposal validation and hash-free title/folder with detailed About identity. Six Debug and Release CTests pass. Replaced the UI wireframe with an actual package screenshot; 0.2.0-alpha now targets discovery/creation. Prior report snapshots and frozen 0.10.3 package are retained.	Plugin; Tests; UI; Week 12
November 20, 2026 / 0.1.2-alpha	VS, CDX	Added Week 12, initial persona PERS-001 and a matching current-interface wireframe/mapping. Linked deployment, plugin design and requirements/use-case progress; retained Week 11's original build-version caption and archived report. No application change.	Week 12; Persona/UI; Deployment; RPT-057
November 20, 2026 / 0.1.1-alpha	VS, CDX	Prepared the missing Week 11 retrospectively from November 12 resize evidence, with an explicitly current follow-up plan and versioned report autobuild evidence. No Week 12 report, application change or assignment submission.	Week 11; Resize evidence; RPT-056
November 19, 2026 / 0.1.0-alpha	VS, CDX	Recorded reviewed plugin data, operations, selection and lifecycle contracts; separated first extraction from later creation/updates; introduced feature-based versioning and UML elbow routing. Application and frozen package unchanged.	Plugin API; Chapter 5; Chapter 6; Chapter 10; Chapter 11; Version policy
November 18, 2026 / 0.10.4	VS, CDX	Added the agent entry point, separated README and handoff roles, and aligned continuation/review procedures. Retained the frozen 0.10.3 demo package; version-policy redesign is deferred.	RPT-010
November 16, 2026 / 0.10.3	VS, CDX	Added the reviewed Alpha deployment diagram with transparent, repositioned arrow labels. Preserved the original TeX as a future label-editing fixture.	Deployment; Fixture

Continued on next page

Date / Version	Author	Change	Location
November 16, 2026 / 0.10.2	VS, CDX	Aligned the Alpha application and architecture versions. Added visible version/source identity, retained historical versions, and linked the build-identity check.	Version rule; TC-067; VER-080
November 12, 2026 / 1.10.1	VS, CDX	Refined corner crossing, minimum dimensions and constrained boundary release. Added the edit service, resize interfaces and linked checks; retained future save-triggered PDF comparison as deferred design.	REQ-F-025; REQ-I-013; UC-002; Edit service; TC-060; Evidence
November 6, 2026 / 1.10.0	VS, CDX	Added Week 10, the implemented drag sequence, and the fixed-boundary alternative for UC-002. Recorded the canvas class, requirement progress and versioned report autobuild.	Week 10; Sequence; UC-002; RPT-055
November 5, 2026 / 1.9.5	VS, CDX	Added the developer requirement for named numeric values and linked its source inspection. Existing code is unchanged.	Rule; Test; Review
November 5, 2026 / 1.9.4	VS, CDX	Registered canvas visibility, default fit, editing limits and clean drag feedback. Documented the canvas view class and the background repaint correction.	Interface requirements; Canvas view; Canvas checks
November 5, 2026 / 1.9.3	VS, CDX	Matched the white paper to the prototype's fixed editing area and fitted the canvas to the pane at each size. Recorded the viewport and drag boundary checks.	Main window; Canvas checks
November 5, 2026 / 1.9.2	VS, CDX	Distinguished the prototype canvas from its surroundings in light and dark palettes. Recorded the bounded display checks and approved corner-resize behavior.	Main window; Resize plan; Canvas check
November 3, 2026 / 1.9.1	VS, CDX	Recorded instructor direction as reported by Vincent: resize the existing box, then extract it as the first shape plugin. Added planned resize and plugin parity checks, and recorded one shape API with optional internal factories.	Shape plugin plan; Chapter 5; Chapter 12; Chapter 8

Continued on next page

Date / Version	Author	Change	Location
October 30, 2026 / 1.9.0	VS, CDX	Added Week Report 9 with links to the reviewed design and Milestone 2 updates, prior-plan outcomes, next steps, actions and risks. Recorded Vincent's confirmation of the 1.8.1.1 submission; implementation remains deferred. Updated the summary and weekly-link obligation; rebuilt and reviewed both editions.	Week 9; Abstract; Appendix C
October 30, 2026 / 1.8.1.1	VS, CDX	Rechecked every Milestone 2 assignment item. Completed US-002's validation plan with zoom clarity and independently compiled-output agreement, and made partial/blocked validation explicit. Used Vincent's requested correction version; no product execution is claimed.	Chapter 4; Chapter 13; M2 coverage
October 30, 2026 / 1.8.2	VS, CDX	Restored clickable navigation from the Milestone 2 history entry to the affected chapters and design appendix; checked link destinations in both report editions.	History; Chapter 11
October 30, 2026 / 1.8.1	VS, CDX	Prepared Milestone 2: added reciprocal story/validation and requirement/verification maps, measurable checks and test execution plan; codes/standards applicability, constraints/tradeoffs and design iterations. Registered reviewed collaborator, label, selection and provider boundaries with updated use cases, planned class/sequence/activity diagrams. Isolated unfinished choices in a temporary Open Design Decisions chapter. Existing prototype evidence and implementation limits are preserved.	Chapter 3; Chapter 4; Chapter 5; Chapter 6; Chapter 7; Chapter 8; Chapter 9; Chapter 10; Chapter 11; Chapter 12; Chapter 13; Appendix D
October 23, 2026 / 1.8.0	VS, CDX	Added Week Report 8 against the Canvas assignment, with completed prototype work, next-week plans, actions, risks and a screenshot of the versioned Windows prototype package directory. Built application package 0.1.0 with Qt dependencies and recorded a startup smoke check; application and report numbering remain separate.	Appendix B; Appendix C; internal AI records

Continued on next page

Date / Version	Author	Change	Location
October 22, 2026 / 1.7.8	VS, CDX	Moved the agreed transaction/history behavior into individual Functional, Data, Interface and Architectural requirement rows with reciprocal story, use-case and test mappings. Retained proposed data/classes in architecture and aligned UC-004 with the existing TikZ actor, boundary and note-panel design.	Chapter 5; Chapter 6; Chapter 7; Chapter 10
October 22, 2026 / 1.7.7	VS, CDX	Recorded the agreed common transaction and undo/redo contract, plugin capability scope, proposed class separation and mapped verification cases. Distinguished planned design from implemented editing. Updated review guidance and authorized a local checkpoint of the accumulated application, report and evidence.	Chapter 5; Chapter 7; Chapter 10; internal AI records
October 21, 2026 / 1.7.6	VS, CDX	Adopted weekly-report-based versions and retroactively renumbered all history entries. Updated report metadata, roadmap version references and the ongoing authoring convention. Historical dates, changes and requirement statuses are retained.	Document History; Appendix C; internal AI records
October 21, 2026 / 1.7.5	VS, CDX	Added whole-element selection, live dragging, minimal coordinate updates on release and cancellation. Recorded four passing checks, guarded save/reopen and three compiled saved-output comparisons. Updated current design and retained an editable demo.	Chapter 10; Chapter 7; Appendix D; internal AI records
October 21, 2026 / 1.7.4	VS, CDX	Added line and default-arrow parsing/display ahead of dragging at Vincent's request. Recorded three passing checks and fixed-extent agreement for all eight corpus files. Added startup-file opening and retained same-chat workflow preference.	Chapter 10; Chapter 7; application; internal AI records
October 21, 2026 / 1.7.3	VS, CDX	Implemented conservative rectangle parsing and mapped outline display. Recorded two passing automated checks and a fixed-extent TinyTeX comparison; aligned current architecture views and handoff. Dragging and product acceptance remain pending.	Chapter 10; Chapter 7; application sources; internal AI records

Continued on next page

Date / Version	Author	Change	Location
October 20, 2026 / 1.7.2	VS, CDX	Moved MVC, ownership, and linked class diagrams into Software Architecture; retained supporting activities in the design appendix. Added a focused-chat handoff procedure and current checkpoint. Recorded the first native build, passing foundation test, and Vincent's GUI observation with explicit limits.	Chapter 7; Chapter 10; Appendix D; internal AI guide and records
October 20, 2026 / 1.7.1	VS, CDX	Added AI-REQ-015: read TeX sources for report information and reserve the PDF skill for visual inspection. Aligned the guide, checklist, and repository instructions.	Internal AI requirements and guide; repository README; internal AI records
October 17, 2026 / 1.7.0	VS, CDX	Added Week Report 7, covering developer conventions and stakeholder findings while carrying forward pending diagram review, native build, and prototype work.	Appendix B; internal AI records
October 15, 2026 / 1.6.1	VS, CDX	Added Developer Requirements and proposed review checks; moved the source-module rule while retaining its ID. Applied the confirmed tkr class prefix to source, files, build references, and diagrams. Clarified a maximum of three project-defined inheritance levels without adding subclasses.	Chapter 5; Chapter 10; Chapter 11; Chapter 7; internal AI records
October 9, 2026 / 1.6.0	VS, CDX	Added Week Report 6, distinguishing prepared code and diagrams from pending review and execution. Checked the new assignment's documentation fields and converted the existing linked glossary to a table.	Appendix B; glossary; internal AI records
October 8, 2026 / 1.5.5	VS, CDX	Added an MVC allocation diagram, revised the class overview around responsibilities and exchanges, and linked all six classes to individual attribute/operation diagrams. Recorded source contracts, ownership, and unfinished behavior.	Chapter 7; internal AI records

Continued on next page

Date / Version	Author	Change	Location
October 8, 2026 / 1.5.4	VS, CDX	Addressed reported Milestone 1 feedback: linked overview goals to their use-case entries, added one annotated TikZ diagram per use case, and added reciprocal table/figure links. Proposed flows and existing identifiers are preserved.	Chapter 6; internal AI records
October 6, 2026 / 1.5.3	VS, CDX	Added TikZ class and activity diagrams derived from the current scaffold and agreed design. Distinguished existing open/save/close logic from the planned drag workflow.	Chapter 7; Appendix D; internal AI records
October 6, 2026 / 1.5.2	VS, CDX	Aligned current planning and requirements context with the selected C++/Qt direction, class responsibilities, naming, ownership, and editing contracts. Product requirements remain unverified.	Chapter 3; Chapter 5; Chapter 2; glossary; internal AI records
October 6, 2026 / 1.5.1	VS, CDX	Recorded the initial six-class C++/Qt scaffold, shared records, coding guide, and prepared foundation checks. Parsing and graphical editing remain unfinished; the native build toolchain was not found, so compilation and test execution remain unverified.	internal AI records
October 2, 2026 / 1.5.0	VS, CDX	Added Week Report 5, connecting the Week 4 plan to reference examples, stakeholder review, the C++/Qt decision, and proposed class design. Included next steps and a class-boundary reminder; prototype implementation remains pending.	Appendix B; internal AI records
October 1, 2026 / 1.4.7	VS, CDX	Defined the first explicit-coordinate test subset. Added line, arrow, and box reference sources with expected edits; broader syntax coverage remains a later goal.	Chapter 3; Chapter 5; Chapter 10; internal AI records
October 1, 2026 / 1.4.6	VS, CDX	Strengthened independent coordinate-mapping checks and added a separated-box contact test with compiled TikZ reference illustrations, common comparison bounds, and reciprocal requirement links.	Chapter 3; Chapter 5; Chapter 10; Chapter 11; internal AI records

Continued on next page

Date / Version	Author	Change	Location
October 1, 2026 / 1.4.5	VS, CDX	Added a Table of Contents bookmark to the PDF outline for both report editions, preserving the printed contents and its chapter links.	Table of Contents; internal AI records
September 29, 2026 / 1.4.4	VS, CDX	Added provisional straight-line, arrow, and rectangular-box drag behavior, zoom clarity, and canvas-coordinate mapping requirements. Expanded proposed tests for gestures, source preservation, zoom, and compiled-output agreement with reciprocal links.	Chapter 5; Chapter 4; Chapter 6; Chapter 10; Chapter 11; Appendix C; internal AI records
September 29, 2026 / 1.4.3	VS, CDX	Added provisional object-oriented component responsibilities, one-primary-class-per-module organization, and UI-independent source-edit logic, with proposed design checks and traceability. Kept detailed interface contracts and language-specific naming open.	Chapter 5; Chapter 4; Chapter 6; Chapter 10; Chapter 11; internal AI records
September 29, 2026 / 1.4.2	VS, CDX	Added provisional preview recovery and dependency requirements, architectural boundaries for compilation and display, and responsive-drag behavior, with proposed tests and reciprocal story, use-case, and traceability links. Saving architecture remains for later review.	Chapter 5; Chapter 4; Chapter 6; Chapter 10; Chapter 11; Appendix C; internal AI records
September 29, 2026 / 1.4.1	VS, CDX	Expanded the provisional PySide6 feasibility approach with a measured core workflow, a conditional focused C++/Qt comparison, and separate product performance criteria pending stakeholder review.	Chapter 3; internal AI records
September 25, 2026 / 1.4.0	VS, CDX	Drafted Week Report 4 from documented work and the Week 3 plan, separating completed report work from the pending TikZ subset, architecture sketch, and framework feasibility test.	Appendix B; internal AI records

Continued on next page

Date / Version	Author	Change	Location
September 25, 2026 / 1.3.21	VS, CDX	Prepared separate Milestone 1 submission and developer editions. The submission edition omits the three internal AI appendices while retaining the AI-assisted authorship record; the developer edition retains the appendices and prompt log.	Chapter 3; Appendix C; internal AI records
September 25, 2026 / 1.3.20	VS, CDX	Added a provisional phase-level work breakdown and TikZ Gantt chart based on Vincent's tentative milestones; marked inferred planning windows and retained the controlled-chart and resource questions as open.	Chapter 3; Appendix C
September 24, 2026 / 1.3.19	VS, CDX	Added Vincent's tentative targets for concept development, preliminary design, implementation and alpha evaluation, and beta refinement and final evaluation. Distinguished these planning targets from confirmed course deadlines.	Chapter 3
September 24, 2026 / 1.3.18	VS, CDX	Added two provisional business outcomes for positional editing and reusable LaTeX source, a proposed end-to-end test and reciprocal mappings, and a future UML capability noted from Vincent's account of the professor discussion.	Chapter 5; Chapter 4; Chapter 6; Chapter 10; Chapter 11; Chapter 3
September 24, 2026 / 1.3.17	VS, CDX	Split the candidate product requirements into template-style functional, data, security, and constraint registers while preserving all IDs, fields, wording, and traceability links.	Chapter 5
September 23, 2026 / 1.3.16	VS, CDX	Clarified that the project edits LaTeX diagrams using TikZ, matched the course-recorded team name while retaining the TikZRefinery product name, corrected the TikZ glossary definition, and explained constraint requirements and existing coverage.	Chapter 1; Chapter 2; Chapter 5; Glossary
September 23, 2026 / 1.3.15	VS, CDX	Standardized capitalization and punctuation in Development Plan and AI Agent Guide action lists, and updated the Guide's active-source list to include the current story, use-case, test, and traceability chapters.	Chapter 3; internal AI records

Continued on next page

Date / Version	Author	Change	Location
September 23, 2026 / 1.3.14	VS, CDX	Added the currently identified stakeholder and his role and interest to the Development Plan; clarified that provisional needs live in User Stories pending stakeholder review, and aligned the report roadmap.	Chapter 3; Chapter 4; Appendix C
September 23, 2026 / 1.3.13	VS, CDX	Capitalized and punctuated the provisional stakeholder-review questions and increased space before AI prompt records. Whether to retain the questions remains under Vincent's review.	Chapter 4; internal AI records
September 23, 2026 / 1.3.12	VS, CDX	Audited Demo 1 coverage, expanded provisional story descriptions and review questions, and added an offline operating constraint with reciprocal story, use-case, test-case, and traceability links. Stakeholder interview evidence remains unconfirmed.	Chapter 4; Chapter 5; Chapter 6; Chapter 10; Chapter 11
September 23, 2026 / 1.3.11	VS, CDX	Corrected AI Prompt Log model attribution using Vincent's account: GPT-5.6 Sol Medium before the Week 3 Report prompt and GPT-6 Sol Medium from that prompt onward.	internal AI records
September 23, 2026 / 1.3.10	VS, CDX	Audited the controlled report roadmap against the current drafts; updated stale statuses, locations, and remaining evidence for requirements, stories, tests, traceability, citations, and navigation.	Appendix C
September 23, 2026 / 1.3.9	VS, CDX	Added template-style implementation version and test-case fields to product requirements, proposed tests and traceability, and a TikZ use-case overview; named the co-drafted user stories explicitly. Existing glossary and index formats were retained.	Chapter 4; Chapter 5; Chapter 6; Chapter 10; Chapter 11
September 23, 2026 / 1.3.8	VS, CDX	Applied the reviewed template use-case table style to the remaining provisional use cases, retaining story and requirement mappings and Vincent's recollection attribution.	Chapter 6
September 23, 2026 / 1.3.7	VS, CDX	Reformatted provisional UC-001 as a labeled use-case table adapted from the course template for comparison with the remaining narrative use cases.	Chapter 6

Continued on next page

Date / Version	Author	Change	Location
September 23, 2026 / 1.3.6	VS, CDX	Added provisional Demo 1 stories, product requirements, and mapped use cases; introduced cited sources, glossary and index examples. Editor interaction is attributed to Vincent's recollection pending review.	Chapter 4; Chapter 5; Chapter 6
September 22, 2026 / 1.3.5	VS, CDX	Reviewed the Development Plan with Vincent's feedback: recorded provisional milestone timing, the instructor's Qt recommendation, incremental test definition, the basic application priority, and the unavailable prior-project link.	Chapter 3; internal AI records
September 22, 2026 / 1.3.4	VS, CDX	Applied instructor feedback from the full-score Week 3 grading by ordering weekly reports from newest to oldest and recording that convention in the controlled roadmap and AI guide.	Appendix B; Appendix C; internal AI records; internal AI records
September 22, 2026 / 1.3.3	VS, CDX	Drafted the project-specific Development Plan, including roles, iterative method, workspace, communication, milestone sequence, testing strategy, risk register, and assumptions. Updated the active TinyTeX build guidance.	Chapter 3; Appendix C; internal AI records
September 18, 2026 / 1.3.2	VS, CDX	Added review-ready Development Plan and Requirements chapter structures without baselining unapproved project details or product requirements.	Chapter 3; Chapter 5; Appendix C; internal AI records
September 18, 2026 / 1.3.1	VS, CDX	Migrated the report roadmap and binding AI-agent requirements into controlled LaTeX appendices with stable IDs, priorities, statuses, version fields, links, and review rules. Reduced the Markdown roadmap and README to navigation roles.	Appendix C; internal AI records; internal AI records; internal AI records
September 17, 2026 / 1.3.0	VS, CDX	Added Week Report 3, connecting the Milestone 0 project definition to the planned requirements, supported TikZ subset, component architecture, and PySide6 feasibility work.	Appendix B; internal AI records

Continued on next page

Date / Version	Author	Change	Location
September 17, 2026 / 1.2.6	VS, CDX	Added Milestone 0 with the TikZRefinery team identity, project proposal, mission statement, key drivers, and key constraints. Updated the abstract, roadmap, and AI records.	Chapter 2; internal AI records
September 15, 2026 / 1.2.5	VS, CDX	Converted the Location column to navigable internal references for affected chapters, appendices, and front/back matter.	Document History
September 15, 2026 / 1.2.4	VS, CDX	Aligned every AI Prompt Log record with the course six-field schema and reconciled the revision history with the Week 1 report corrections, including the original glossary version.	Document History; internal AI records
September 15, 2026 / 1.2.3	VS, CDX	Added a required template-review gate for report structure and authoring conventions, including connected-use searches, adaptation decisions, and Vincent Sun's review.	internal AI records; internal AI records
September 15, 2026 / 1.2.2	VS, CDX	Connected the existing glossary entries to controlled terms throughout the report following the course implementation pattern and added a persistent roadmap for staged report sections.	Chapter 1; Appendix B; Glossary
September 15, 2026 / 1.2.1	VS, CDX	Audited the current and deferred report structure against the course examples, refined the staged chapter plan, and formalized AI-assisted authorship using the course initials convention.	Document History; internal AI records
September 15, 2026 / 1.2.0	VS, CDX	Added Week Report 2, documenting the independent desktop direction, Qt and PySide6 investigation, preliminary application workflow, project risks, and next steps.	Appendix B
September 15, 2026 / 1.1.2	VS, CDX	Added the project update history, adapted the abstract, established the repository document structure, expanded the project-specific AI guide and prompt log, and added the initial project glossary.	Abstract; Document History; internal AI records; internal AI records; Glossary

Continued on next page

Date / Version	Author	Change	Location
September 15, 2026 / 1.1.1	VS, CDX	Replaced generic course content with TikZRefinery metadata, added the independent developer introduction, and compiled and visually verified the Week 1 report.	Chapter 1; Appendix B
September 4, 2026 / 1.1.0	VS, CDX	Recorded the initial problem definition, platform investigation, planned work, action items, and project risks for Week 1.	Chapter 1; Appendix B

Appendix B

Weekly Reports

B.1 Week Report 12 (November 20, 2026)

Work Completed During the Past Week

I prepared the Windows Alpha deployment view and reviewed plugin records, operations, capabilities, provider choice and lifecycle policy in separate UML views (plugin contract). Requirements/use cases now cover provider choice, updates and creation with reciprocal planned checks. The report contains 104 product requirements and 81 test definitions; those counts do not mean all tests have executed.

I then extracted rectangle presentation, selection, movement and resizing into a separately loaded native module through a Qt Core API (implemented extraction). The host retains syntax/source rewriting and guarded document acceptance. All six Debug and all six Release CTest suites passed, including actual DLL loading, missing and incompatible modules, geometry parity and invalid-proposal rejection (evidence). The separate `0.1.2-alpha` package started with a Windows-only PATH; the original 0.10.3 package remains frozen. Feature-based numbering reserves `0.2.0-alpha` for later shape discovery and creation.

For this assignment I added draft PERS-001, a TikZ-literate report author, rather than claiming an interview result. Its interface mapping and actual application screenshot connect layout, source preservation and saving needs to the fitted canvas, corner handles, read-only source pane and file controls. Current host/provider class, MVC and deployment views distinguish this implementation from future design. I also prepared the missing Week 11 entry from dated resize evidence, explicitly marking its retrospective preparation.

Work Planned for Next Week

Prepare the Alpha demonstration and preliminary-design submission; review shape discovery, creation placement and source-insertion contexts before implementing the next feature milestone. Review the package with intended users.

Action Items

Rehearse Open–move/resize–Save–reopen; confirm persona/interface priorities; retain the frozen demo and regression fixtures while reviewing the new package.

Issues and Risks

The native API requires the matching toolchain and is not a sandbox. Syntax variants and source preservation remain risks. Provider choice/update UI, creation, undo/redo and integrated compilation remain pending. The passing bounded suites do not establish general plugin or product acceptance.

B.2 Week Report 11 (November 13, 2026)

Prepared retrospectively on November 20 from dated project records. Completed work below covers November 7–13, not the later plugin review.

Work Completed During the Past Week

Following Week 10's plan, I implemented rectangle corner resizing with corner crossing, independent 0.01 cm minimum dimensions and stroke-aware boundary clamping. Move and resize stop at the furthest valid geometry rather than rejecting the whole gesture. A changed release applies one guarded coordinate batch; cancellation and unchanged releases preserve the source revision.

All five Debug and all five Release CTests passed on November 12. Checks covered all corners, thin rectangles, crossing, bounds, cancellation, stale requests, view transforms and save/reopen, preserving the stationary box and unrelated source. The saved resize sample also compiled with TinyTeX. These are retained results, not tests rerun for this report; see the resize evidence.

I updated `REQ-F-025`, `REQ-I-013`, `UC-002` and their checks. The edit-service class view, move/resize sequence and activity document responsibilities and interaction. The resize work is checkpoint `5a776cb`. Plugin loading, undo/redo and integrated compiler/PDF viewing remain unimplemented.

Work Planned for Next Week

No contemporaneous Week 11 plan was retained. My follow-up plan, recorded now, is to review the rectangle-plugin interface, presentation records and lifetime wiring before extraction, and prepare the Alpha demonstration. The existing working package will remain available while extraction uses a separate package.

Action Items and Risks

Confirm the extraction interfaces, rehearse the supported resize/save workflow, and retain regression fixtures. Arbitrary TikZ support and plugin compatibility remain risks; keep the first extraction narrow and require source-preservation and geometry-parity checks. Late report preparation is explicitly dated above.

```
November 20, 2026 - actual report autobuild output
TikZRefinery report autobuild | Version 0.1.1-alpha
PASS  TikZRefinerySubmission.pdf | TinyTeX, bibliography, index
PASS  TikZRefinery-0.1.1-alpha.zip | 1064474 bytes
ZIP contains TikZRefinery-0.1.1-alpha.pdf
```

First November 20 report autobuild (offscreen log view), `0.1.1-alpha`; versioned PDF ZIP. This is current documentation evidence, not a November 13 build or new executable.

B.3 Week Report 10 (November 6, 2026)

Work Completed During the Past Week

I prioritized canvas feedback over Week 9's planned history implementation. The prototype now fits the canvas to the pane, shows the fixed editing area, and repaints the box without leaving trails in a clipped raster check. Four automated tests passed. I refreshed Windows prototype 0.1.0 and checked that it closed and reopened with a Windows-only PATH. The canvas and tests are in local checkpoint `0aeef51`.

I linked the canvas requirements and the named numeric values rule (Chapter 5) to checks and clarified the out-of-bounds branch in `UC-002`. The canvas view class diagram and whole-element drag sequence document the implemented prototype. The automated report build produces a versioned ZIP. See Document History for this version.

Work Planned for Next Week

I will review the packaged drag on Windows, settle the minimum usable box size, and implement and test a corner resize as one guarded source edit. The rectangle plugin follows a working resize baseline.

Issues and Risks

Native drag acceptance remains pending. I will use the two-box fixture for that review and check that resizing preserves unrelated source. History and provider registration remain deferred.

```
TikZRefinery report autobuild | Version 1.10.0
PASS  TikZRefinerySubmission.pdf | TinyTeX, bibliography, index
PASS  TikZRefinery-1.10.0.zip | 797962 bytes
ZIP contains TikZRefinery-1.10.0.pdf
```

First automated 1.10.0 package run. The final ZIP was rebuilt with this capture included.

B.4 Week Report 9 (October 30, 2026)

Work Completed During the Past Week

I completed Week 8's review of transaction records and ownership: the document will control atomic source/history updates, the edit service prepares proposals, the file store handles I/O and the controller coordinates. Reviewed rejection feedback, attached labels, overlap selection and provider boundaries are now in requirements (Chapter 5), use cases (Chapter 6), planned class views (Chapter 7) and activity/sequence diagrams (Appendix D). Unfinished registration, ambiguity and lifetime choices remain in the temporary Chapter 8.

I prepared and submitted Milestone 2 version 1.8.1.1. I updated the project constraints (Chapter 2), tradeoffs and design iterations (Chapter 3), stories (Chapter 4), standards applicability (Chapter 9), test plan (Chapter 10) and reciprocal mappings (Chapter 11). The 72 requirements and 59 planned test cases map to verification (Chapter 12); ten stories map to validation (Chapter 13). The final audit added zoom clarity and compiled-output agreement to VAL-002. Both editions built and passed visual review.

Transaction/history implementation was deferred for review and documentation. No new feature, product test run or participant validation is claimed. This weekly starts report 1.9.0; the abstract, linked Document History and report roadmap (Appendix C) are updated.

Work Planned for Next Week

I will review provider registration, competing interpretations, compatibility and lifetime, then begin the narrow transaction/history prototype and exact-source undo/redo checks after resolving its required interfaces.

Action Items

1. Review one label expressed in two source forms to test provider boundaries.
2. Settle required record/interface details and implement the agreed history path.
3. Obtain Milestone 2 feedback and clarify the package-screenshot scope.

Issues and Risks

Syntax variants, provider lifetimes and history memory remain risks. I will retain the bounded parser and atomic acceptance boundary, review ambiguity explicitly and measure storage before choosing limits. Labels, menus, plugins and undo/redo remain planned. Instructor feedback and product acceptance are pending.

B.5 Week Report 8 (October 23, 2026)

Work Completed During the Past Week

I completed Week 7's Qt/C++ setup, build and initial parsing objectives. The running prototype displays explicit-coordinate rectangles, lines and arrows, supports whole-element dragging/cancellation, preserves unrelated source and saves/reopens edits. Four automated checks pass; eight reference drawings and three saved outputs matched compiled TikZ within the prototype's one-pixel allowance.

I clarified transaction/history and plugin behavior in linked requirements, use cases and planned tests, aligned UC-004's TikZ diagram, and separated proposed collaborators from current classes. Detailed ownership review remains pending; undo/redo and plugins are unimplemented.

I packaged prototype 0.1.0 with Qt 6.12.0 and checked startup without the development Qt PATH. Its directory appears below. Report 1.8.0 uses separate weekly numbering; Document History is updated.

Work Planned for Next Week

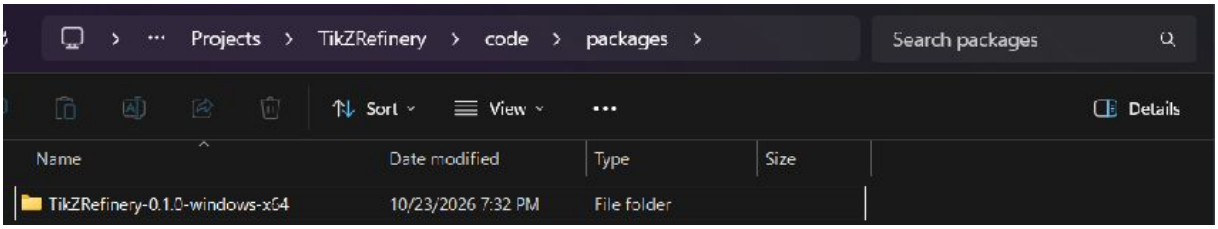
I will review records, validation and ownership, update class diagrams, then begin the common transaction/history implementation and exact-source undo/redo checks.

Action Items

1. Resolve transaction collaborator responsibilities and ownership.
2. Implement and test agreed history behavior after review.
3. Confirm the instructor's intended built-package screenshot scope.

Issues and Risks

Broader syntax, element insertion, endpoint editing and integrated PDF preview remain pending. I will retain the narrow subset and regression checks while reviewing plugin lifetimes and history storage. Native acceptance and performance targets remain unverified; a package startup check does not establish acceptance.



Prototype package 0.1.0, Windows x64; Qt 6.12.0; October 23, 2026.

B.6 Week Report 7 (October 16, 2026)

Work Completed During the Past Week

I added Developer Requirements for source organization, naming, inheritance, and ownership, with proposed checks and traceability. I applied the agreed `tkr` prefix to the six classes, matching files, build references, and design diagrams. I clarified the maximum of three project-defined inheritance levels: base, child, and grandchild. The current design needs no additional subclasses.

I clarified my professor's diagram-authoring workflow and submitted the IDE stakeholder-validation response. He estimated 30–60 minutes for Visio diagram creation and about five minutes for editing existing diagram TeX files. These describe different tasks, not measured savings. He described increasing manual TikZ creation effort with complexity and requested adding elements to AI-generated diagrams, a need for later scope review.

Week 6's planned diagram review, Qt/C++ setup, scaffold build, and foundation tests remain pending. The parsing and box-drag prototype has not begun.

Work Planned for Next Week

Next week, I will review the MVC, class, and activity diagrams for responsibilities and ownership, configure the Qt/C++ toolchain, build the scaffold, and run its foundation tests. If successful, I will begin explicit-coordinate parsing for the separated-box contact workflow. I will assess the new creation need against the initial editing scope.

Action Items

1. Finish diagram review and record corrections.
2. Verify the Qt/C++ build and foundation tests before extending code.
3. Start explicit-coordinate parsing and the box-contact prototype.

Issues and Risks

Limited availability and companion coursework delayed review and implementation. I will resume with review and build checks while retaining the narrow initial subset. Parsing, dragging, source preservation, response time, and canvas/PDF agreement remain unverified. Stakeholder estimates require further investigation before establishing performance targets or broader scope.

B.7 Week Report 6 (October 9, 2026)

Work Completed During the Past Week

I advanced Week 5's class-design plan by defining responsibilities, interfaces, ownership, naming conventions, and a shallow inheritance structure. I prepared the initial six-class C++/Qt scaffold, build configuration, coding guide, and foundation tests, and aligned the Development Plan with these decisions. The application has not yet been compiled or run.

I added TikZ class and activity diagrams, then expanded the design documentation with a model–view–controller allocation, a responsibility and interaction overview, and linked details for each class. Following reported Milestone 1 feedback, I also linked the use-case tables to the overview and individual use-case diagrams. These diagrams are prepared, but my review is still in progress.

I drafted the companion IDE course's initial project understanding with supporting research. I also checked the new weekly-assignment documentation instructions: requirements and user-story tables already exist, and the glossary now uses a linked table of terms and definitions.

Work Planned for Next Week

Next week, I will finish reviewing the design diagrams and resolve any responsibility, interface, or ownership discrepancies. I will install and verify the Qt/C++ toolchain, build the scaffold, and run the prepared foundation tests. If the build succeeds, I will begin explicit-coordinate parsing and the box-drag prototype planned in Week 5, which remains pending.

Action Items

1. Finish diagram review and record needed corrections.
2. Configure the Qt/C++ kit and run the scaffold and foundation tests.
3. Begin parsing and the separated-box contact workflow.

Issues and Risks

Limited availability delayed diagram review and prototype work. I will resume with review and build verification before expanding implementation. Parsing, dragging, source preservation, response time, and canvas/PDF agreement remain unverified; planned tests and reference diagrams are not passing application test evidence.

B.8 Week Report 5 (October 2, 2026)

Work Completed During the Past Week

I reviewed the Development Plan and provisional requirements with my professor's feedback, clarifying dragging, basic shapes, offline operation, saving, and immediate visual feedback. I examined the previous group's report and repository for lessons about compilation, preview, plugins, and agreement between the editor and compiled output.

I defined an explicit-coordinate subset for lines, arrows, and rectangles, compiled reference examples and expected edits, and expanded proposed coordinate-conversion and output-comparison tests. The examples include moving separated boxes into contact. I also explored a TikZ loop with a compiled DVI example. These are reference demonstrations; application tests remain unexecuted.

I selected C++ with Qt and discussed six initial classes, folders, shallow inheritance, and ownership. I agreed on descriptive names, variable type prefixes, and `m_` private-member markers. This advances Week 4's subset, stakeholder, and architecture work. The planned PySide6 test was not built; Qt/C++ feasibility work will replace it.

Work Planned for Next Week

Next week, I will review class boundaries, interfaces, and ownership, document the language and naming decisions, and generate initial class files and build configuration. I will begin a prototype to evaluate dragging, coordinate conversion, minimal source updates, and compiled-output agreement.

Action Items

1. Finalize class responsibilities and ownership.
2. Verify the Qt/C++ build environment and document naming rules.
3. Start the box-contact prototype and retain comparison evidence.

Issues and Risks

Interaction speed, source preservation, and rendering agreement remain unverified. I will evaluate immediate canvas feedback separately from compilation, check coordinate conversions independently, and compare saved output with reference geometry. Broader syntax remains deferred.

Reminder for Next Session: Resume review of the six proposed class boundaries, ownership, and interfaces before generating files.

B.9 Week Report 4 (September 25, 2026)

Work Completed During the Past Week

I prepared and presented Demo 1, then continued the Development Plan for Milestone 1. I drafted provisional user stories, product requirements, use cases, test cases, and a traceability matrix. I revised the use-case and requirement tables to follow the course template and added a use-case diagram. The plan now includes a phase-level work breakdown and a provisional TikZ Gantt chart based on my tentative milestone dates. I also prepared separate submission and developer editions of the report after learning that the AI prompt log may be omitted from the submitted PDF.

This completed the initial requirements and use-case drafting planned in Week 3. The precise supported TikZ subset, component and data-flow sketch, and PySide6 feasibility test remain to be done. The proposed requirements and tests are drafts, not approved scope or executed test results.

Work Planned for Next Week

Next week, I will review the Milestone 1 Development Plan for gaps, including resources and skills, and seek feedback on the provisional stories and requirements. I will define a first supported TikZ object and the source preservation boundary, then sketch the major application components and begin a focused PySide6 feasibility test for graphical interaction and external LaTeX compilation. I will assess the previous group's materials if access becomes available.

Action Items

1. Review the Milestone 1 plan and request stakeholder feedback.
2. Specify the first editable TikZ subset and source-preservation rules.
3. Sketch the architecture and run the first PySide6 feasibility check.

Issues and Risks

The supported syntax and stakeholder expectations are not yet confirmed, and the framework has not been tested in a prototype. I will keep the first workflow narrow, seek feedback using representative diagrams, and use the feasibility check before committing to the implementation approach. Access to the previous project remains pending, so current planning does not depend on it.

B.10 Week Report 3 (September 18, 2026)

Work Completed During the Past Week

This week, I advanced the project definition and scope planned in Week 2 even though the previous group's materials were not yet available. I completed the Milestone 0 Team Declaration, which established the TikZRefinery team identity, project proposal, mission statement, key drivers, and key constraints. This work clarified the business intent: reduce the effort required to refine rough or AI-generated TikZ diagrams while preserving readable, editable source.

I also refined the initial prototype boundary. The project remains a local desktop application centered on source editing, external LaTeX compilation and preview, basic graphical manipulation, and reliable round-trip editing for a deliberately limited TikZ subset. I documented source synchronization, the external toolchain, the senior-design schedule, and single-developer capacity as constraints that must shape the requirements and architecture. Qt 6 and PySide6 remain implementation candidates rather than approved choices.

Finally, I maintained the project report and repository records so that the proposal, glossary, revision history, AI-use records, and future chapter plan remain consistent.

Work Planned for Next Week

Next week, I will turn the approved proposal into initial platform-independent requirements and use cases, define the first supported TikZ object and editing subset, and draft the application's component and data-flow boundaries. I will also build a small PySide6 feasibility test for application startup, a graphics scene, and external compilation without treating the result as a final framework decision. If the previous group's materials arrive, I will review them for reusable lessons, constraints, and failure modes.

Action Items

1. Draft and review initial requirements and representative use cases.
2. Define the minimum TikZ subset and source-preservation rules.
3. Sketch the component architecture and run the PySide6 feasibility test.
4. Review prior-project evidence when it becomes available.

Issues and Risks

Delayed access to the previous project may hide useful lessons, so current work will remain independent and later findings will be evaluated rather than adopted automatically. Scope remains the primary risk for a single developer; the prototype will prioritize a verifiable end-to-end workflow over broad TikZ support.

B.11 Week Report 2 (September 11, 2026)

Work Completed During the Past Week

This week, I resolved the platform and ownership question from Week 1. I separated from the original group so that Turan and Sergio could continue with their web direction, while I began pursuing the graphical TikZ editor as an independent desktop project. I selected the name TikZRefinery to reflect the goal of turning rough or AI-generated TikZ diagrams into carefully adjusted, editable results.

I investigated Qt 6 with PySide6 as the leading desktop framework. Qt Widgets could provide the application shell, while QGraphicsScene and QGraphicsView could support selectable and movable diagram elements. I also identified that Qt would launch and display results from a separate LaTeX toolchain rather than compile LaTeX itself. This clarified a preliminary workflow from TikZ source, through external compilation, to PDF preview and graphical adjustment.

I also learned that a previous senior-design group attempted this project using an existing open-source application. I requested access to its repository so I can review its approach and failures before committing to a final architecture. The minimum viable prototype remains centered on source editing, compilation and preview, and manipulation of a deliberately limited subset of TikZ objects.

Work Planned for Next Week

Next week, I will review the previous implementation if it becomes available, write initial platform-independent requirements, define the supported TikZ subset for the minimum prototype, and draft the desktop application's major components and data flow. I will also prepare a small PySide6 feasibility test without treating the framework choice as final.

Action Items

1. Obtain and assess the previous group's repository and lessons learned.
2. Define the minimum supported editing and compilation workflow.
3. Draft initial requirements and a preliminary component architecture.

Issues and Risks

The previous repository may be delayed, so I will continue with requirements that do not depend on it. Supporting arbitrary TikZ would create excessive parsing and round-trip-editing risk for one developer; the first prototype will therefore use a documented subset and preserve unsupported source where possible. The project scope will be reviewed against the available schedule.

B.12 Week Report 1 (September 4, 2026)

Work Completed During the Past Week

This week, I selected the professor-provided idea of a graphical TikZ editor and began defining the problem. AI-generated diagrams may contain spacing, alignment, or scale errors that require precise human correction, while editing those details directly in LaTeX can be difficult. The proposed application would provide graphical manipulation while retaining editable TikZ source.

I discussed the platform with Turan and Sergio. A web application aligned with Turan's experience, while I was more interested in desktop development. I identified graphical editing, source synchronization, LaTeX compilation, preview, and EPS export as major needs. Reliable round-trip editing between the graphical view and TikZ source appears to be the central technical challenge.

Work Planned for Next Week

Next week, I will clarify ownership of the web and desktop directions, define a minimum viable product, configure and test the LaTeX report, compare desktop user-interface frameworks, and begin outlining the application architecture.

Action Items

1. Resolve the platform direction and project ownership.
2. Define the minimum prototype and test the LaTeX and desktop tools.

Issues and Risks

Unresolved platform scope could divide effort; I will establish clear ownership before implementation. Because arbitrary TikZ is difficult to rewrite safely, early prototypes will target a limited subset.

Appendix C

Report Roadmap

C.1 Purpose and Authority

This appendix is the authoritative plan for the contents of the TikZRefinery Senior Design Core Report. It replaces the former Markdown roadmap and records which report obligations are satisfied, underway, planned, conditional, or no longer applicable. Course assignments, instructor feedback, and Vincent Sun's approved decisions take precedence over this roadmap.

The developer's edition includes the AI Directives, AI Agent Usage Guide, and AI Prompt Log. These records are omitted from the submission PDF: Vincent reports that Professor Muresan said the prompt log may be left out of a submitted report. Vincent chose to omit the other two AI appendices from that edition. Requirements RPT-009–RPT-011 apply to the developer's edition and its maintained source.

The identifiers in this appendix govern report content, not the behavior of the TikZRefinery application. Product requirements use separate identifiers and are mapped to stories, use cases, and proposed tests in Chapter 11.

C.2 Adaptation of the Course Pattern

The course requirements register was reviewed as the structural example. Its stable identifiers, linked references, controlled priorities, lifecycle statuses, version field, and multipage table format are adopted here. The status meanings are adapted to document work. Product use-case and test-case mappings are omitted because a report-content obligation is verified by its controlled report location, assignment, or inclusion trigger rather than by application behavior.

C.3 Priority and Status Model

Priority and status are independent. Priority records the importance or timing of a report obligation. Status records its progress in the report lifecycle. The controlled priorities are Must, Should, Could, and Would.

Table C.1: Report Requirement Status Definitions

Status	Meaning	In Version Rule
Open	The obligation applies, but its project-specific content has not been started. A later trigger does not make the requirement inactive.	Enter ∅.
In Progress	Drafting, review, recurring maintenance, or integration has begun, but the current obligation is not yet complete.	Enter ∅.
Done	The controlled report satisfies the obligation, its links are current, and the complete report has been built and reviewed.	Enter the first report version that satisfied the obligation, using the numbering convention in Document History.
Retired	The obligation was superseded, withdrawn, or found not to apply. Its identifier and rationale remain in the register and Document History.	Retain the original version if previously Done; otherwise enter ∅.

Only **Done** asserts that the current baseline satisfies an obligation. Done content may still mature; later revisions preserve the stable identifier. Retired identifiers shall not be reused.

C.4 Report Requirements Register

Table C.2: Report Requirements Register

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
Current and recurring report content					
RPT-001	The report shall identify the project, document version, status, owner, organization, date, and copyright holder.	Must	**Done**	1.1.1	`tkrPrologue.tex`
RPT-002	The report shall provide a current abstract summarizing the project and the evidence contained in the report.	Must	**Done**	1.1.2	Abstract
RPT-003	The report shall retain a newest-first history of controlled document revisions and their locations.	Must	**Done**	1.1.2	Document History; distinct front-matter summary and link
RPT-004	The report shall provide a navigable table of contents.	Must	**Done**	1.1.0	`tkrPrologue.tex`
RPT-005	The report shall introduce the project and its current developer or team.	Must	**Done**	1.1.1	Chapter 1
RPT-006	The report shall contain the approved team declaration, project proposal, mission, key drivers, and key constraints.	Must	**Done**	1.2.6	Chapter 2
RPT-007	The report shall contain every assigned weekly status report of at most one page with the required date, completed work, next plan, actions, and applicable risks, ordered from newest to oldest.	Must	**In Progress**	∅	Appendix B; recurring course assignment and instructor feedback
RPT-008	The report shall maintain this controlled roadmap of report obligations, statuses, versions, locations, and triggers.	Must	**In Progress**	∅	Appendix C; pending item-by-item review

Continued on next page

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-009	The report shall define stable directives governing material AI-agent assistance.	Must	**In Progress**	∅	internal AI records; pending item-by-item review; internal AI records controls bidirectional change-impact review
RPT-010	The report shall provide operating procedures that implement the AI-agent requirements.	Must	**Done**	1.1.2	internal AI records
RPT-011	The report shall maintain a prompt log for material AI assistance using the course record fields.	Must	**In Progress**	∅	internal AI records; recurring
RPT-012	The report shall define controlled technical terms and link their substantive uses to the glossary. The glossary shall present terms and definitions in a table.	Must	**Done**	1.6.0	Glossary and acronyms; Week 6 table-format instruction applied in 1.6.0
Project definition and first design proposal					
RPT-013	The report shall state the mission, intended benefit, and measurable outcome.	Must	**In Progress**	∅	Initial statement in Chapter 2; refine when success criteria are approved
RPT-014	The report shall explain the motivating problem, prior work, and supporting evidence.	Must	**In Progress**	∅	Motivation in Chapter 1 and Chapter 2; prior-work analysis and supporting evidence remain to be added

Continued on next page

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-015	The report shall identify stakeholders and their roles and interests.	Must	**In Progress**	∅	Initial stakeholder list, role, and interest in Chapter 3; additional stakeholders and their evidence remain to be confirmed
RPT-016	The report shall analyze stakeholder needs, priorities, conflicts, and evidence.	Must	**Open**	∅	Provisional needs appear in Chapter 4; analyze evidence, priorities, and conflicts after stakeholder review
RPT-017	The report shall define project scope, resources, skills, tools, schedule, and relevant constraints.	Must	**In Progress**	∅	Initial resources, tools, schedule, and constraints in Chapter 3; scope and skills need review
RPT-018	The report shall define the work breakdown, responsibilities, schedule, risks, review gates, and change-control approach.	Must	**In Progress**	∅	Phase-level work breakdown, roles, schedule, risks, gates, and change control in Chapter 3; detailed task planning and instructor feedback remain pending

Continued on next page

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-019	The report shall define stable, singular, feasible, unambiguous, and verifiable system requirements with controlled priority, status, version, and traceability fields.	Must	**In Progress**	∅	Category-specific provisional entries and test links in Chapter 5, including Developer conventions; supported syntax and measurable criteria need stakeholder review
RPT-020	The report shall provide user stories and use cases derived from reviewed stakeholder goals and mapped requirements.	Must	**In Progress**	∅	Drafts in Chapter 4 and Chapter 6; overview and nine focused TikZ diagrams have reciprocal specification links following reported Milestone 1 feedback; goals and flows await confirmation
RPT-021	The report shall record project constraints and assumptions, including the supported TikZ subset and external LaTeX toolchain.	Must	**In Progress**	∅	Initial assumptions in Chapter 3 and constraints in Chapter 2; initial explicit-coordinate syntax boundary in Chapter 5 and reference corpus in Chapter 10; broader coverage and comparison tolerances remain open

Continued on next page

© Stevens Institute of Technology

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-022	The report shall identify applicable codes, standards, regulations, policies, and ethical considerations.	Must	**In Progress**	∅	Chapter 9; selected applicability and derived checks reviewed for Milestone 2, full compliance not claimed
RPT-023	The report shall compare feasible concepts and record screening criteria, trade-offs, stakeholder input, and the selection rationale.	Must	**Open**	∅	Add Concept Development and Selection for Milestone 1
RPT-024	The report shall include a team-organization appendix showing current project roles and known advisor or sponsor relationships.	Must	**Open**	∅	Add when the course organization format and relationships are confirmed
RPT-025	The report shall include a controlled Gantt chart for the approved milestone schedule.	Must	**Open**	∅	Provisional phase chart in Chapter 3; confirmed dates, detailed dependencies, critical path, and status remain pending for the controlled Gantt appendix
Preliminary design and alpha planning					

Continued on next page

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-026	The report shall define the architecture overview and system context.	Would	**In Progress**	∅	MVC, ownership, and linked class views in Chapter 7; selection in Section 3.3.1; system context and fuller architectural views remain pending; reviewed plugin boundary; planned sidebar/shared-service ownership
RPT-027	The report shall document the user-interface design and principal interaction flows.	Would	**In Progress**	∅	Initial persona/UI, current application screenshot and mapping; activities in Appendix D; persona validation and detailed interface review remain pending; later Shapes/creation in UC-009, not implemented; expanded/collapsed planned UI; actual screenshot remains current

Continued on next page

© Stevens Institute of Technology

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-028	The report shall maintain consistent logical, process, development, and physical architectural views.	Would	**In Progress**	∅	MVC and class views in Chapter 7; activities in Appendix D; full views and baseline pending; current/planned UML package and provider views remain separate; planned sidebar MVC; current views remain source-based

Continued on next page

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-029	The report shall document preliminary and detailed design, including components, data flow, interfaces, compilation, error handling, and persistence.	Would	**In Progress**	∅	Initial component and editing contracts in Section 3.3.1; class attributes, operations, and contracts in Section 7.6; detailed parser, compilation, and display design remain pending; versioned request/result records; planned method roles; catalogue records; agreed history architecture, planned contracts and restoration sequence; bounded session history in Section 10.16; measured memory limits and broader lifetimes remain open
RPT-030	The report shall address security and privacy for local file processing and external tool execution.	Would	**Open**	∅	During architecture and detailed design
RPT-031	The report shall define and retain design evaluation methods and concept-selection evidence.	Would	**Open**	∅	With concept evaluation and prototype planning

Continued on next page

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-032	The report shall define the alpha prototype scope, integration plan, demonstration criteria, budget, and test plan.	Would	**In Progress**	∅	Alpha planning milestone; bounded extraction plan; full alpha planning remains open; reviewed creation activity; insertion/New prerequisites remain open
RPT-033	The report shall include controlled design documents when substantive diagrams, layouts, or analyses exist.	Would	**In Progress**	∅	Class/MVC diagrams in Chapter 7, including the reviewed Alpha deployment view; activities in Appendix D; November 24 active-diagram/source audit and maintained figure index complete; broader design/validation remain open; applicable UML with orthogonal routes under diagram convention; editable planned UML and wireframes; reviewed new-edit activity and two-panel history interaction

Continued on next page

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-034	The report shall include human-readable citations, a References Cited appendix, and a generated bibliography when formal external claims are introduced.	Would	**In Progress**	∅	Citations and generated bibliography present; the References Cited appendix remains to be added
Implementation and evidence					
RPT-035	The report shall document implementation and controlled release information when implementation exists.	Would	**In Progress**	∅	Build: Section 10.5; editing: Section 10.8; release record pending; frozen 0.10.3 plus separate 0.1.2-alpha rectangle module; 0.2.0-alpha discovery/creation remains planned; 0.1.3-alpha adds the bounded sidebar shell (Section 7.10.1); catalogue/creation and shared-service ownership remain planned

Continued on next page

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-036	The report shall define tests and record testing and evaluation results without confusing expected outcomes with actual evidence.	Would	**In Progress**	∅	Proposed cases in Chapter 10; bounded display/edit and sidebar-shell results included; product evaluation pending; lifecycle and creation checks remain planned; TC-079 and UI lifetime aspects of TC-082 have bounded shell evidence; TC-080 remains unexecuted
RPT-037	The report shall maintain a requirements traceability matrix with complete reciprocal mappings.	Would	**In Progress**	∅	Draft ID matrix in Chapter 11; review completeness when requirements are approved; plugin/new-use-case reciprocal maps included; 111 product requirements, 82 test definitions and 111 verification rows; reciprocal mappings updated

Continued on next page

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-038	The report shall distinguish and retain verification and validation evidence.	Must	**In Progress**	∅	Chapter 12, Chapter 13; Milestone 2 explicitly requires plans and bidirectional maps; completed evidence remains separate; provider and creation user validation remain Planned; VAL-011 includes later sidebar/creation tasks; no participant execution claimed
RPT-039	The report shall record alpha and beta prototype results, defects, corrective actions, and revisions.	Would	**Open**	∅	Prototype evaluation milestones
RPT-040	The report shall record high-performance or realistic-environment testing when required by the course or project risks.	Would	**Open**	∅	Applicable later testing milestone
RPT-041	The report shall provide the final controlled design specification and known limitations.	Would	**Open**	∅	Final design and report milestones
Entrepreneurship and business development					
RPT-042	The report shall address business objectives and risks when assigned.	Would	**Open**	∅	Add as assigned
RPT-043	The report shall document competitive intelligence and market analysis when assigned.	Would	**Open**	∅	Add as assigned
RPT-044	The report shall include a Lean Canvas business model when assigned.	Would	**Open**	∅	Add as assigned

Continued on next page

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-045	The report shall provide financial analysis when assigned.	Would	**Open**	∅	Add as assigned
RPT-046	The report shall address intellectual property and open-source licensing obligations when evidence exists.	Would	**Open**	∅	Add as assigned or when a dependency decision creates an obligation
RPT-047	The report shall retain Innovation Expo materials, feedback, and resulting actions when assigned.	Would	**Open**	∅	Innovation Expo work
Navigation and supporting matter					
RPT-048	The report should provide an Executive Summary after meaningful design decisions and prototype or evaluation results exist.	Should	**Open**	∅	Add when a decision-and-evidence summary can be supported; update near final submission
RPT-049	The report could provide a List of Tables when substantive technical tables exist beyond document-control tables.	Could	**Done**	0.1.4-alpha	List of Tables follows the contents and precedes the List of Figures, adopting the reviewed course-template order. Captions, numbers and page links are generated from the report

Continued on next page

© Stevens Institute of Technology

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-050	The report could provide a List of Figures when the first substantive project figure is included.	Could	**Done**	0.1.4-alpha	List of Figures; generated source/status/page index in `documents/DIAGRAM_INDEX.md`. User authorized the navigation addition on November 24; full design review remains required
RPT-051	The report could provide an index when stable artifacts and concepts make it useful.	Could	**Done**	1.3.6	Generated index at the end of the report
RPT-052	Week Report 8 shall include a screenshot of the directory of built packages showing version numbers.	Must	**In Progress**	∅	Week Report 8; Vincent approved using the prototype package version; confirm the instructor's intended package scope at the next class
RPT-053	Milestone 2 shall map all stories to validation and all requirements to verification in both directions, with acceptance criteria, methods and evidence; include workflows/classes/sequences where needed, test levels/data/oracles, applicable standards, constraint tradeoffs and a feedback-to-design-to-recheck iteration.	Must	**In Progress**	∅	October 30 assignment; Table 3.11. Vincent confirmed submission of 1.8.1.1; instructor review remains pending
RPT-054	Week Report 9 and Document History shall link the report locations updated during the reporting period.	Must	**Done**	1.9.0	Week Report 9; Document History; October 30 weekly assignment

Continued on next page

© Stevens Institute of Technology

Report Requirements Register (continued)

ID	Report Requirement	Priority	Status	In Version	Location or Trigger
RPT-055	Week Report 10 shall cover completed work and next-week plans in at most one page, add at least one class and one sequence diagram, and continue requirements and use cases. It shall show an autobuild screenshot with a Major.Minor.Changelist version and ZIP output containing at least the versioned report PDF.	Must	**In Progress**	∅	Week Report 10; class view; drag sequence
RPT-056	Week Report 11 shall cover completed work and a next-week plan in at most one page, identify documentation/4+1-view updates, show the document version and an autobuild screenshot with a versioned ZIP containing at least the PDF report.	Must	**In Progress**	∅	Week 11; November 13 assignment, read November 20. Retrospective preparation and current follow-up plan are explicit; submission and instructor acceptance remain pending
RPT-057	Week Report 12 shall cover completed work and next-week plans in at most one page; include a deployment diagram, continued requirements/use cases and relevant 4+1 views, and an explicitly identified initial persona with a matching user interface.	Must	**In Progress**	∅	Week 12; Deployment; Persona/UI; November 20 assignment. Persona is a hypothesis; user validation and submission remain pending
RPT-058	Milestone 3 shall provide the updated Architecture Report supporting an Alpha demonstration, matching application and report versions mapped to source control, and use-case, responsibility/class, sequence and deployment diagrams.	Must	**In Progress**	∅	November 24 assignment, read November 24; coverage and remaining submission checks

C.5 Milestone 3 Submission Coverage

The November 24, 2026 Canvas assignment specifies the Alpha demonstration topics and explicitly requests the updated Architecture Report as a PDF. The separate Demo 3 assignment uses the same six topics for the presentation. Both list November 24 at 11:59 p.m. as their deadline. This is a preparation map, not evidence of submission, a timed presentation or instructor acceptance.

Table C.3: Milestone 3 Topic Coverage

Assigned topic	Current evidence / location
Basic Alpha functionality	Bounded session-history checks, earlier display/edit checks and the runnable `code/packages/0.1.4-alpha/` package. Live presentation remains for Vincent.
Matching version and source identity	Report/application baseline `0.1.4-alpha`; About and build metadata retain parent Git checkpoint `7a2965a4f14e` plus modified-working-tree state. A commit containing the current work remains pending; the parent hash alone does not identify these edits.
Use-case diagram	UC-004: Undo or redo; Chapter 6 retains the other focused actor-goal views.
Classes, responsibilities and naming	Current class overview, linked individual responsibility views and document-owned history; project classes use `tkr` in C++ namespace `TikzRefinery`.
Sequence	Current edit sequence; broader new-edit/restoration design remains explicitly distinguished from the bounded implementation.
Deployment	Current rectangle-plugin deployment; the current physical view is Figure 7.1; older deployments remain historical evidence.

Demo 3 material is maintained locally under `documents/slides/demo3/`. Its current flow is progress, live Alpha, use case, class responsibilities, provider collaboration, sequence, deployment, evidence and limits. The script allocates ten minutes to the live Alpha and five to slides within the assigned fifteen-minute slot; Vincent still needs to rehearse that timing.

C.6 Inclusion and Review Rules

A planned section enters the active report only when it contains project-specific, reviewable content or a course assignment explicitly requires it. Before adding, removing, replacing, or substantially restructuring a feature, review the relevant course entry point and component files and search the complete template for connected definitions, uses, output, dependencies, and build steps. Classify the course pattern as adopted, adapted, deferred, or omitted and record the project-specific reason.

When a roadmap requirement changes, preserve its identifier, update its priority, status, version, location or trigger, reciprocal references, Document History, and the AI Prompt Log as applicable. Material choices and uncertain adaptations require Vincent Sun's review.

Appendix D

Design Documents

D.1 Scope and Evidence

Additional reviewed, planned sidebar ownership/MVC, catalogue data, creation activity and UI sketches are linked from Section 7.10. They do not replace the current implementation activities below.

This appendix retains activity diagrams for opening, whole-element dragging, saving, and closing. A separately labelled planned new-edit activity and two-panel interaction sequence record the agreed transaction/history contract, reconciled on November 24. MVC, ownership, and class views appear in Chapter 7, adapting the course distinction between architecture chapters and supporting design artifacts.

The activities follow source inspection and agreed design. The first-build check (Section 10.5) did not exercise save or close-guard branches. Bounded drag/save evidence now appears in Section 10.8. Endpoint editing and compiler/PDF integration remain unfinished.

Table D.1: Design Diagram Evidence and Requirement Links

Diagram	Basis and boundary	Related requirements
Open source	Existing file-read and subset-display paths; unsupported source yields diagnostics and an empty scene.	REQ-F-001, REQ-F-005, REQ-D-001
Drag and source update	Whole-element preview, guarded commit and cancellation; endpoint editing remains planned.	REQ-F-004, REQ-F-009–REQ-F-011, REQ-D-001, REQ-A-007, REQ-P-001
Save source	Guarded save after graphical edits; save/reopen and observed external-change refusal checked in the bounded prototype.	REQ-F-006, REQ-D-001

Close window	Existing unsaved-change guard and stack/QObject lifetimes; future child processes need a separate shutdown design.	REQ-F-006, REQ-D-001

Notation. Component names follow Chapter 7. Filled/ringed circles mark start/end; rounded boxes are actions; diamonds are decisions or merges. Brackets show conditions. No concurrency fork or join is implied.

D.2 Open Source Activity

This follows the current subset-display and unsupported-source paths. A successful file read is reported as an open success even when graphical interpretation is unavailable. Supported elements are displayed; unsupported source clears records and items and returns a diagnostic. The offscreen Open check is recorded in Section 10.6; startup-file checks follow in Section 10.7.

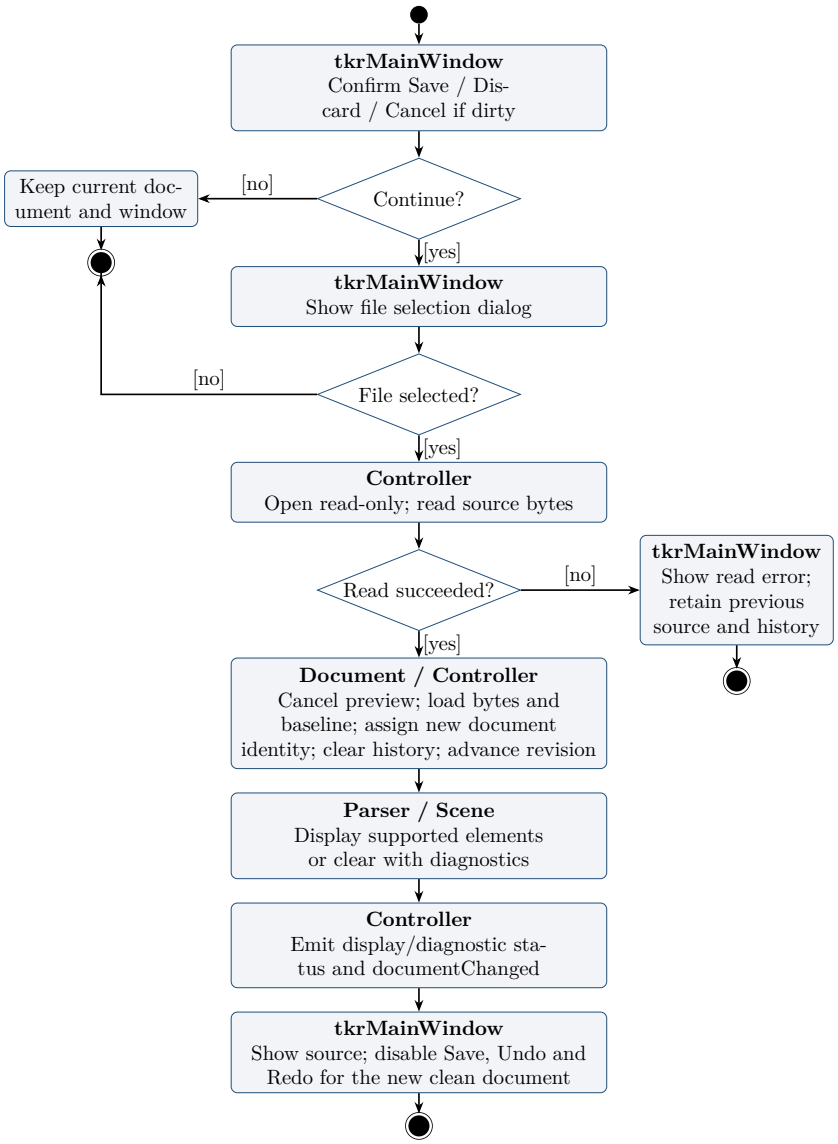


Figure D.1: Current open-file control flow, including cancellation and read failure. Source remains authoritative; supported elements permit whole-body movement.

D.3 Move/Resize and Source-Update Activity

Precondition: the parser and scene have produced supported editable records. Move and rectangle-corner gestures use scene events, service-prepared constraints, candidate checks and guarded source edits. Whole-element moves translate both defining points; corner resizing keeps the opposite corner fixed. Endpoint editing remains deferred. Escape, focus loss, and cancelling an active interaction restore the unchanged document geometry. Source changes occur once on release. A future saved-revision compilation request after successful Save is governed by REQ-P-001; compiler/display integration remains pending.

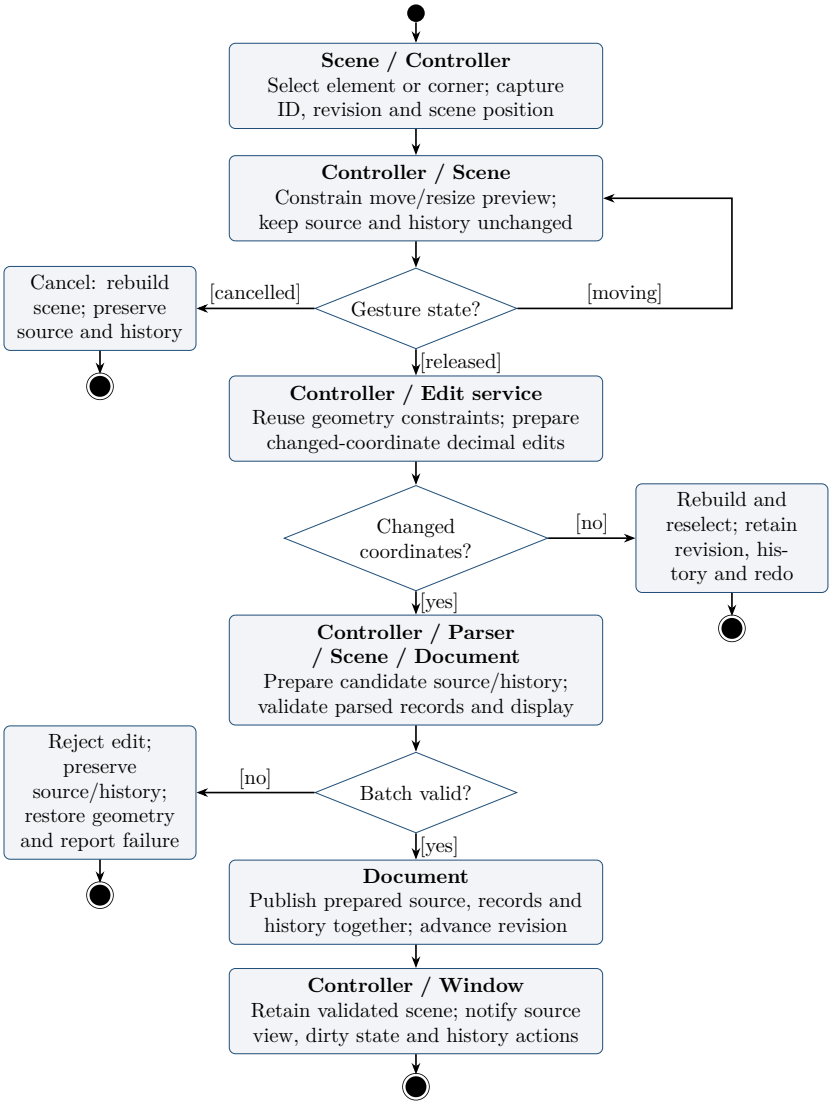


Figure D.2: Current move/resize feedback with one source-update batch on release. Cancelling or rejecting an edit preserves source. No LaTeX run occurs inside the movement loop.

D.4 Prototype Move/Resize Sequence

This sequence follows the current scene and controller for UC-002. The drag preview clamps the displayed item to the fixed paper when present (REQ-I-013). Release reuses the same constrained geometry. A changed, accepted move or resize publishes candidate source, parsed records and history atomically, then updates the source pane. Unchanged/cancelled gestures preserve revision and redo. Boundary contact is a valid edit; stale or unsupported requests preserve the prior source.

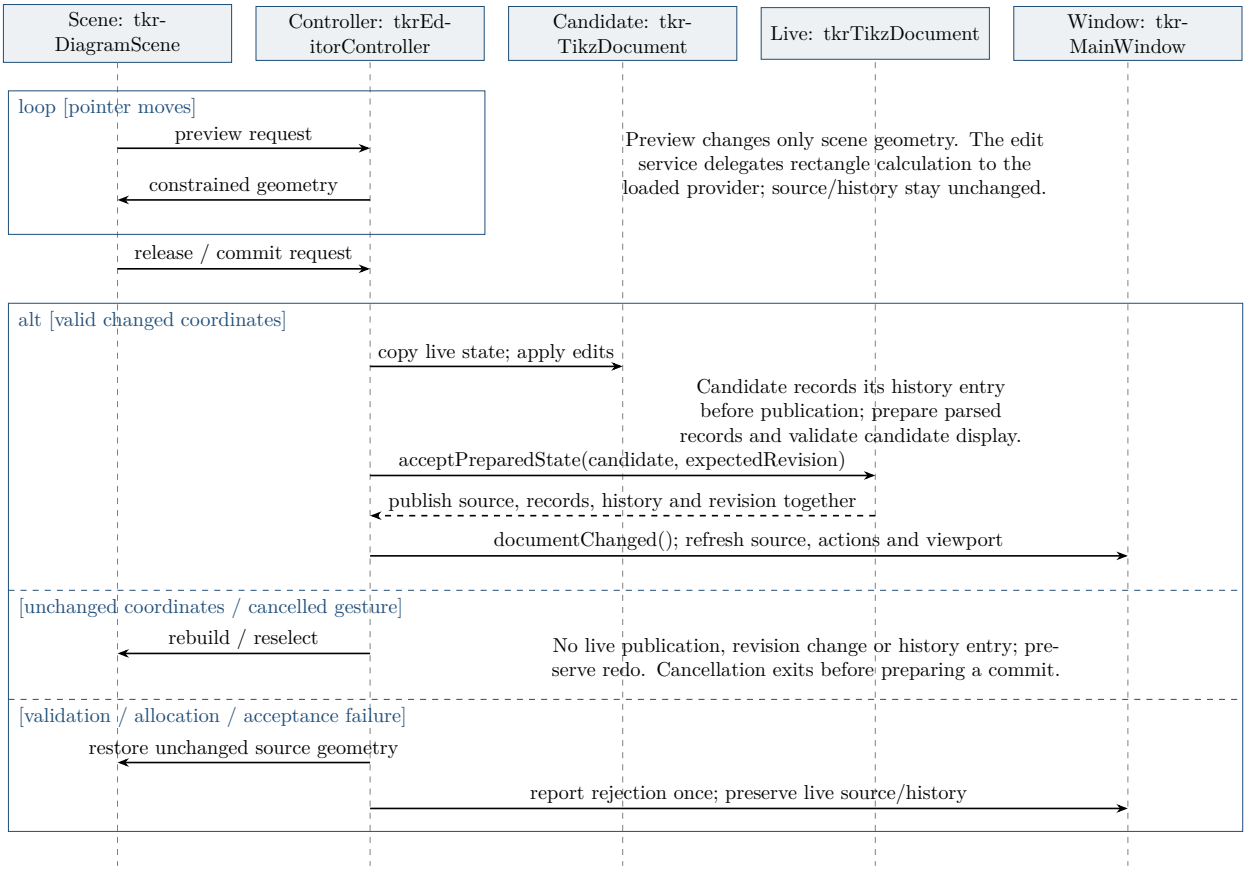


Figure D.3: Implemented ordering for movement and rectangle resizing. The candidate source is checked before the authoritative document changes.

D.5 Current Undo/Redo Sequence

The bounded implementation of UC-004 copies the live document, restores a target in the candidate, validates interpretation/display, and then accepts the entire prepared state. The candidate owns its own history value; no live cursor is advanced before acceptance. This differs from the proposed generic method names in Figure D.8.

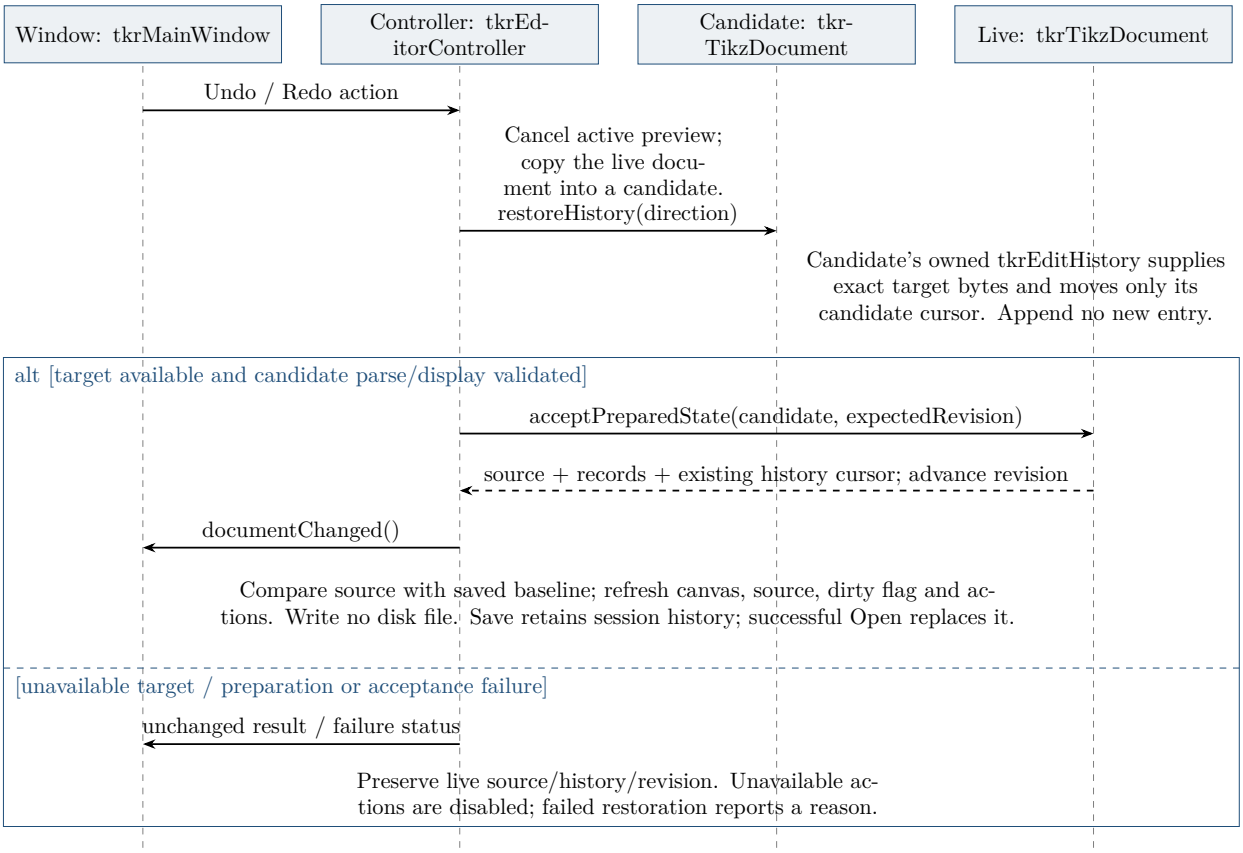


Figure D.4: Current snapshot restoration and atomic publication. Save retains history; successful Open clears it. No new history entry or disk write occurs during restoration.

D.6 Save Source Activity

This models the existing `saveDocument()` logic. The changed-document path is reachable after a whole-element move. Active previews are cancelled before saving. The baseline comparison detects observed external changes; it does not eliminate a concurrent-write race after that comparison.

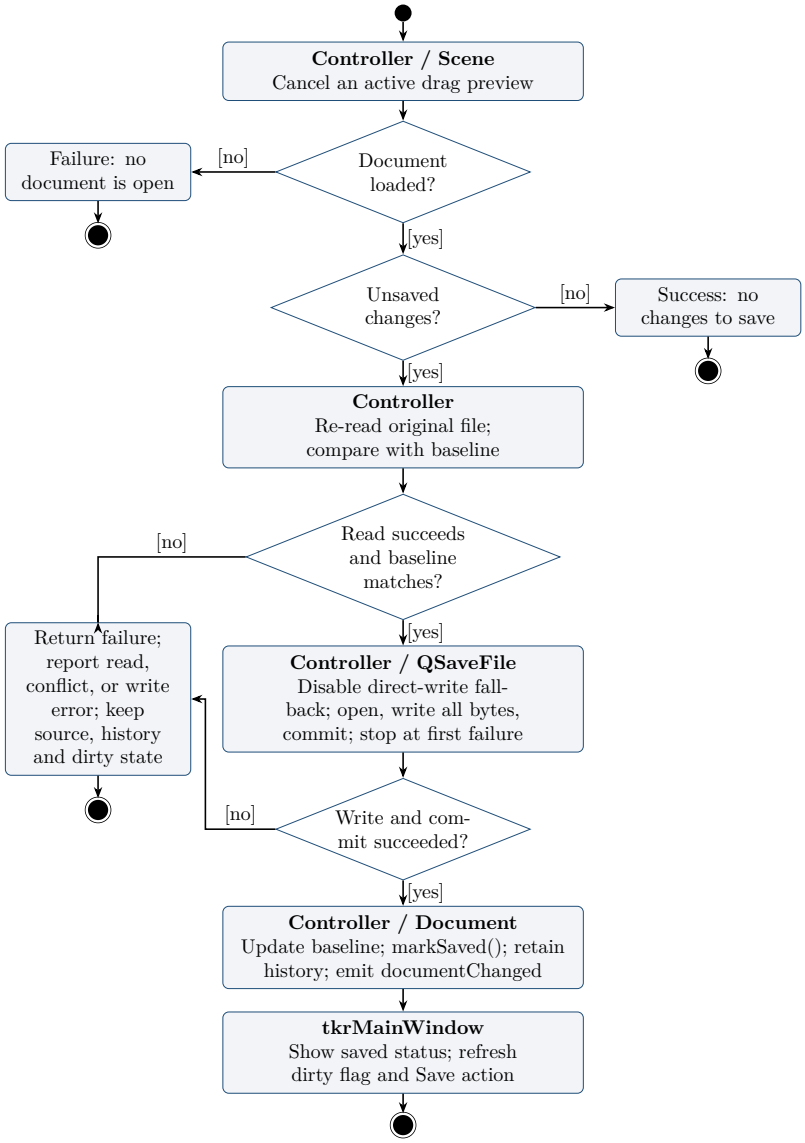


Figure D.5: Current guarded save: failures retain dirty state, and only a successful commit updates the saved baseline; all paths retain session history. Opening, writing, and committing stop immediately on the first failure.

D.7 Close and Lifetime Activity

This models the single stack-owned window in `code/src/main.cpp`. An active drag preview is cancelled before the unsaved-change guard runs. Cancelling or failing a requested save prevents closing. An accepted final close exits the event loop, after which the stack window is destroyed.

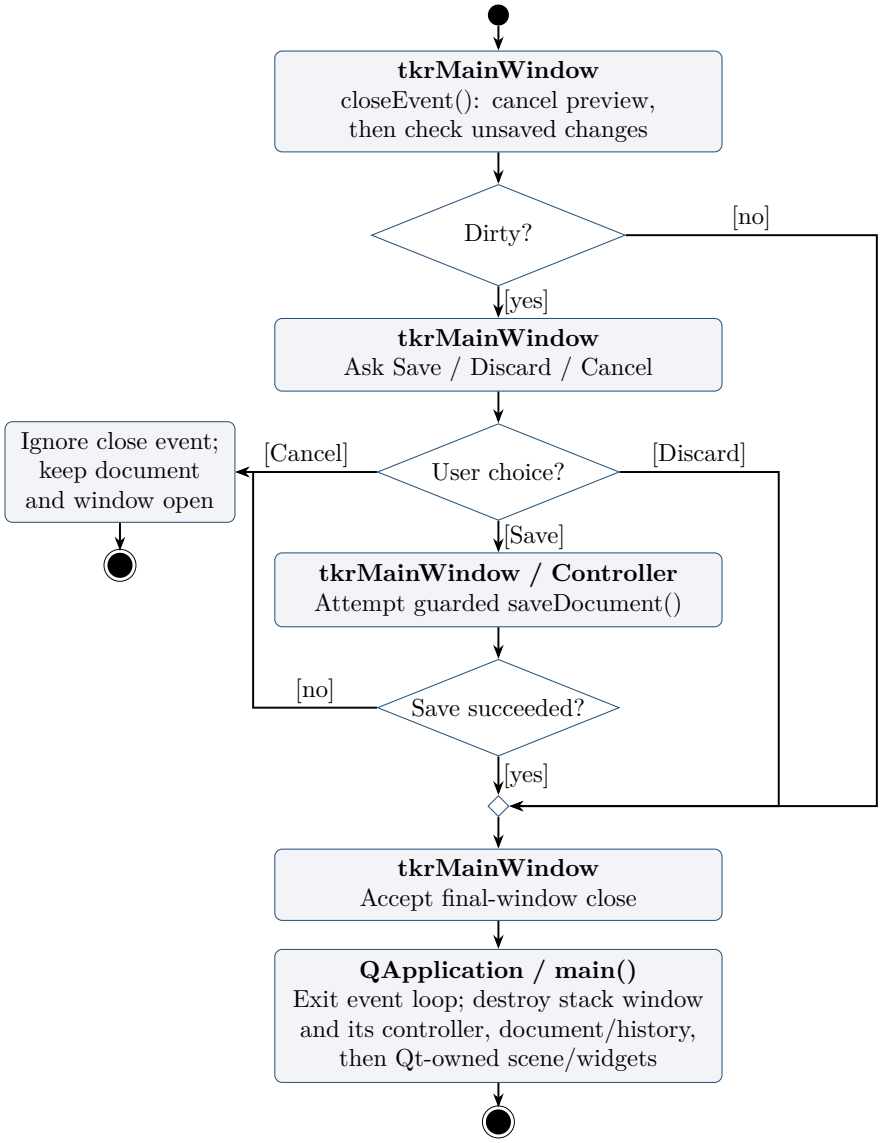


Figure D.6: Current close guard and component lifetime. Discard allows closing; it does not write source to disk. See Figure D.5 for the save subactivity.

D.8 Planned Common Transaction Activity

This activity illustrates acceptance of a new edit under the registered requirements, not current runtime behavior. Undo/Redo restores an existing target through panel (b) of Figure D.8; it creates no new entry. Requirements REQ-D-002 and REQ-A-011 define atomic publication and validation. REQ-I-003 defines notification. Validation and history preparation complete before source/history publication. The planned collaborators are shown in Figure 7.16. The bounded coordinate-edit implementation follows the same acceptance rule; its source-based activity is Figure D.2. Generic transaction signatures and the file-store extraction remain planned.

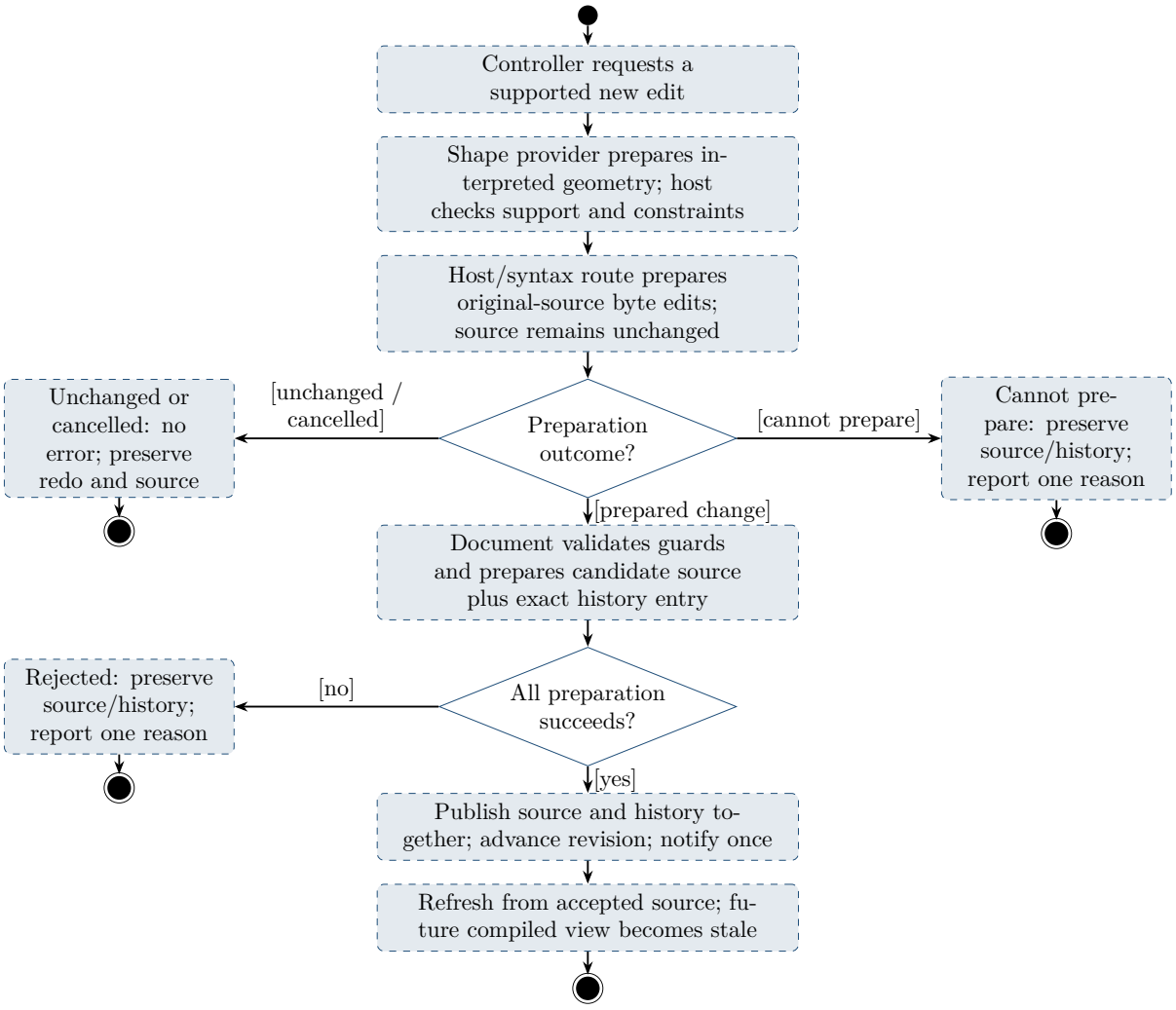


Figure D.7: Reviewed planned new-edit acceptance. Unchanged/cancelled actions preserve redo without an error; rejection preserves source/history with a reason. Future compilation belongs to successful Save and cannot roll back acceptance.

D.9 Planned Transaction Interaction Sequence

Use cases UC-002, UC-004 and UC-005 share this core. Preparation precedes the atomic publication boundary; history retrieval does not advance the position prematurely. Save/Open are separate I/O workflows and do not write disk during undo/redo. Panel (a) prepares a new edit; provider geometry and host/syntax rewrites remain distinct. Panel (b) cancels any active gesture preview before Undo/Redo, retrieves the existing target without advancing history prematurely, and prepares restoration before atomic publication. Availability/failure paths preserve state. Neither restoration nor failed preparation appends a new history entry.

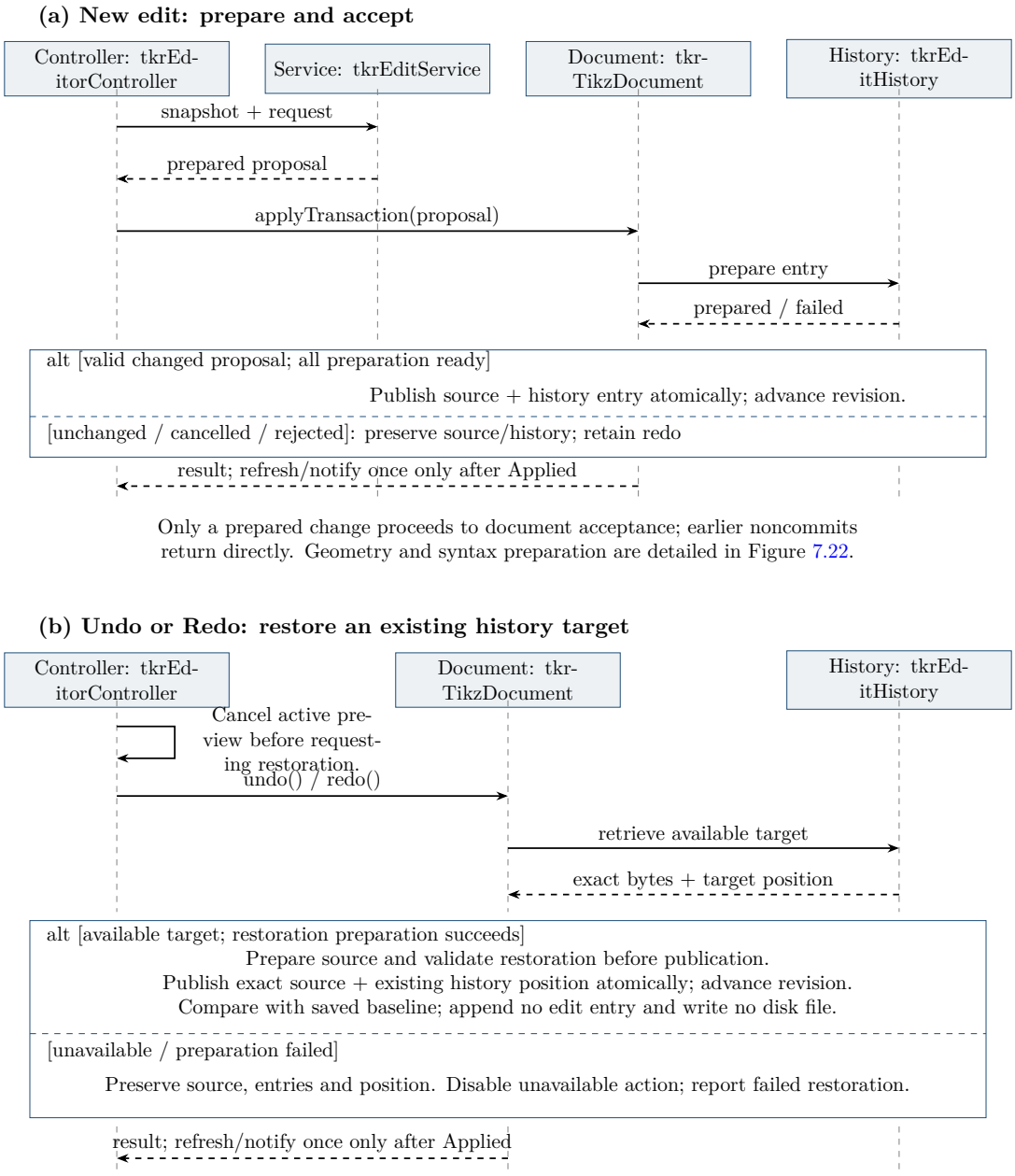


Figure D.8: Reviewed planned interactions: (a) new-edit acceptance and (b) Undo/Redo restoration. Ordering follows REQ-D-002, REQ-A-012 and REQ-I-006; no runtime execution is claimed.

D.10 Planned Label and Selection Activity

This supplements the existing source-based drag activity; it does not imply that label parsing or the selection menus are implemented. Unchanged proposals exit without publication or an error, as REQ-I-006 requires. Candidate selection follows UC-006; label adjustment follows UC-005. Detachment is a separate later action.

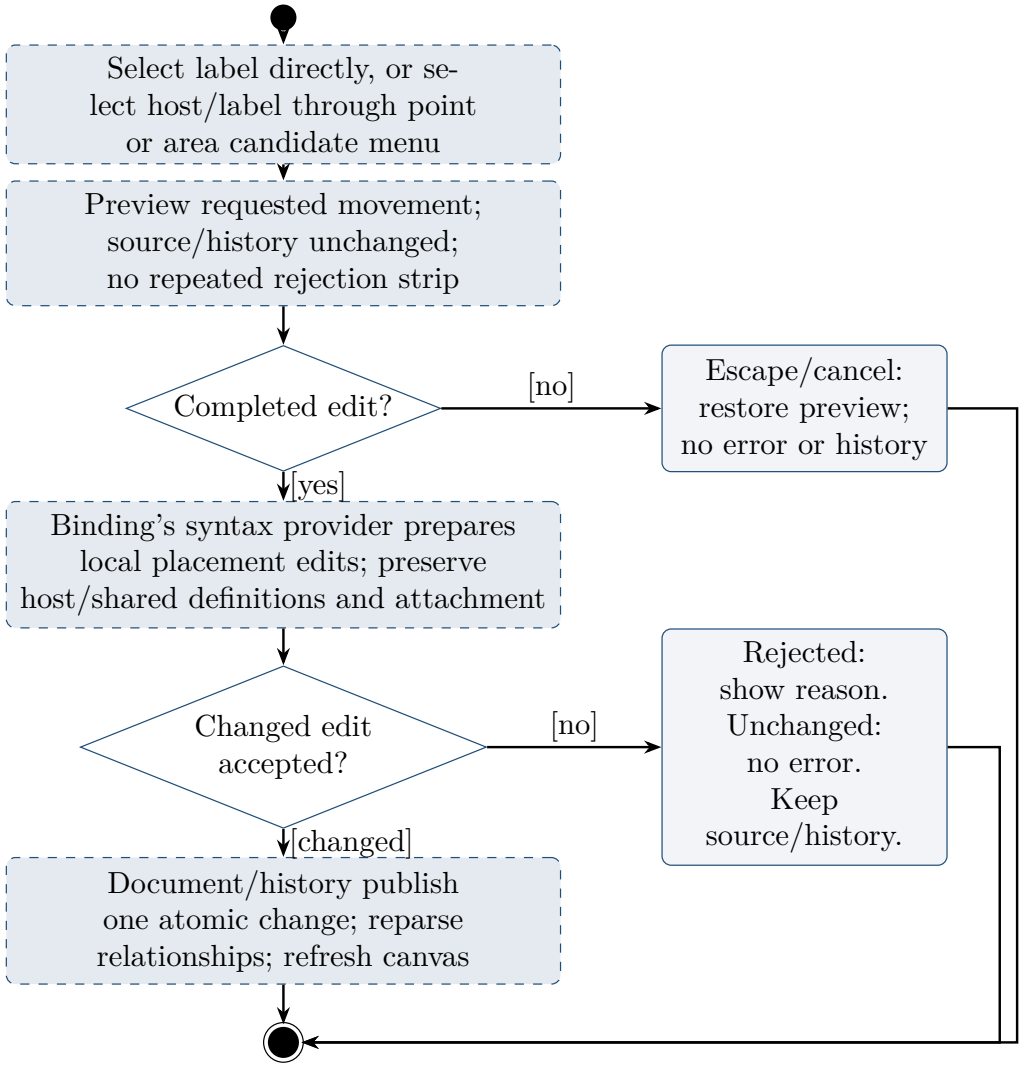


Figure D.9: Planned label movement and failure/cancellation boundaries for REQ-F-021–REQ-F-024, REQ-I-007–REQ-I-009.

Bibliography

[1] Martin Fowler. Model view controller. https://martinfowler.com/eaaCatalog/modelViewController.html. Accessed October 8, 2026.

[2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design patterns: Abstract factory. https://www.informit.com/articles/article.aspx?p=1398599, 2009.

[3] International Organization for Standardization. ISO/IEC/IEEE 29148:2018, Systems and Software Engineering—Life Cycle Processes—Requirements Engineering, 2018. https://www.iso.org/standard/72089.html. Accessed September 23, 2026.

[4] Robert C. Martin. The single responsibility principle, 2014. May 8.
Author's article. Accessed October 22, 2026.

[5] MGB-UML contributors. MGB-UML Plugin SDK and Element Plugin Interface, 2026. Revision `ab47abd0bf04`.
Pinned SDK source. Source inspected October 22, 2026.

[6] Object Management Group. *OMG Unified Modeling Language, Version 2.5.1*, 2017. Official specification listing.
https://www.omg.org/spec/UML/2.5.1/About-UML/. Accessed October 30, 2026.

[7] The Qt Company. *Graphics View Framework*. Qt 6 documentation. https://doc.qt.io/qt-6/graphicsview.html. Accessed September 23, 2026.

[8] The Qt Company. *Model/View Programming*. Qt 6 documentation. https://doc.qt.io/qt-6/model-view-programming.html. Accessed October 8, 2026.

[9] The Qt Company. *QProcess Class*. Qt 6 documentation. https://doc.qt.io/qt-6/qprocess.html. Accessed September 23, 2026.

[10] Qt Group. *How to Create Qt Plugins*. Qt 6 documentation, accessed November 3, 2026.

[11] Alan Shalloway and James R. Trott. Design patterns explained: The strategy pattern. https://www.informit.com/articles/article.aspx?p=1398602&seqNum=6, 2009.

[12] Till Tantau. *The PGF/TikZ Manual*. Online documentation. https://tikz.dev/. Accessed September 23, 2026.

[13] The LaTeX Project. Latex documentation. https://www.latex-project.org/help/documentation/. Accessed September 23, 2026.

[14] F. Yergeau. *RFC 3629: UTF-8, a Transformation Format of ISO 10646*, 2003. https://www.rfc-editor.org/info/rfc3629/. Accessed October 30, 2026.

Glossary

Term	Definition	Pages
acceptance criterion	An observable condition used to decide whether a story or requirement has been met	21, 240
architecture	The organization of a system into major components, their responsibilities, interfaces, dependencies, and data flows	240
change-impact review	A recorded bidirectional check identifying controlled artifacts updated, reviewed unchanged or explicitly deferred after a material design or code change	240
Could	A desirable requirement satisfied when time, evidence, and resources permit	209, 222, 223, 240
graphical editor	An application that lets a user select and manipulate visual objects directly rather than changing every object only through source text	1, 2, 240
LaTeX	A document preparation system that converts structured source files into typeset output such as PDF	1, 2, 4, 28, 45, 240
minimum viable prototype	The smallest implementation that can test the project's central technical assumptions and demonstrate its core workflow	2, 206, 240

Term	Definition	Pages
Must	A requirement necessary for the current controlled report or release; the applicable result is unacceptable without it	21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 209, 210, 211, 212, 213, 214, 221, 223, 224, 240
parameter schema	A versioned definition of the fields and meanings of one operation's parameters, separate from plugin release and API versions	109, 240
placement anchor	The document-space reference point used to position a provider's default shape geometry during creation	116, 240
PySide6	The official Python bindings for Qt 6, considered earlier for TikZRefinery before C++ with Qt was selected	206, 240
Qt	A cross-platform application framework for desktop windows, controls, graphics scenes, process management, and related application services	206, 240
requirement	A singular, feasible, unambiguous, and verifiable statement of a needed system capability or constraint	206, 240
round-trip editing	The ability to move between TikZ source and a graphical representation while preserving supported meaning and unrelated source	2, 205, 207, 240
shape catalogue	A host-associated collection of offered shape-type descriptions and provider identities, distinct from accepted elements in a document	116, 240
shape provider	A provider that interprets a shared element description to prepare presentation, selection and geometric operations without directly mutating document or scene	105, 240

Term	Definition	Pages
Should	An important requirement expected to be satisfied unless a documented constraint prevents it	22, 27, 29, 36, 38, 39, 43, 209, 222, 240
source binding	Document- and revision-scoped information identifying the responsible syntax provider and original source locations used for a safe rewrite	113, 240
source synchronization	The process of keeping the textual TikZ source and the graphical editing view consistent after either representation changes	3, 240
TikZ	A TeX graphics language supplied by PGF and commonly used in LaTeX documents to describe diagrams with nodes, paths, coordinates, styles, and other visual elements	iii, 1, 2, 4, 240
TikZ subset	The explicitly documented portion of TikZ syntax that a given prototype can parse, display, and edit reliably	2, 27, 45, 205, 206, 240
traceability	The ability to follow a controlled artifact backward to its source and forward to its implementation, related artifacts, and evidence	240
use case	A structured description of how an actor interacts with a system to achieve a goal, including normal and exceptional flows	48, 240
validation	Confirmation that the completed system satisfies stakeholder needs in its intended context; building the right system	240
verification	Confirmation that specified requirements have been fulfilled; building the system right	240
workspace sidebar	A planned tabbed container for Shapes and Files pages that retains an identifiable strip while its content is collapsed	116, 240
Would	A requirement assigned to a later milestone or version after higher-priority work	209, 215, 216, 217, 218, 219, 220, 221, 222, 240

Acronyms

Term	Definition	Pages
AI	artificial intelligence	iii, 2, 240
EPS	Encapsulated PostScript	240
PDF	Portable Document Format	240
UI	user interface	240

Index

acceptance criterion, 21
activity diagram, 226

class diagram, 77
    tkrCoordinateMapper, 87
    tkrDiagramScene, 83
    tkrDiagramView, 82
    tkrEditorController, 85
    tkrEditService, 88
    tkrMainWindow, 81
    tkrTikzDocument, 80
    tkrTikzParserExplicit, 86
class naming, 43
class responsibilities, 77
creation, 116

deployment diagram, 71

inheritance depth, 43

Model-View-Controller, 75, 76

ownership, 75

persona, 72
plugin API, 89, 105

requirement, 27
    architectural, 39
    business, 27
    constraint, 45
    data, 32
    developer, 43
    functional, 27
    interface, 36
    performance, 45
    security, 38
requirements traceability matrix, 153

shape catalogue, 116
shape provider, 89, 105
software architecture, 69
source synchronization, 48
standards, 127

test case, 130
traceability, 153
transaction, 94

undo and redo, 94
use case, 48
use case diagram, 48
    UC-001, 51
    UC-002, 53
    UC-003, 56
    UC-004, 58
user interface, 72
user story, 21

validation, 177
verification, 160

Windows package, 71
workspace sidebar, 116