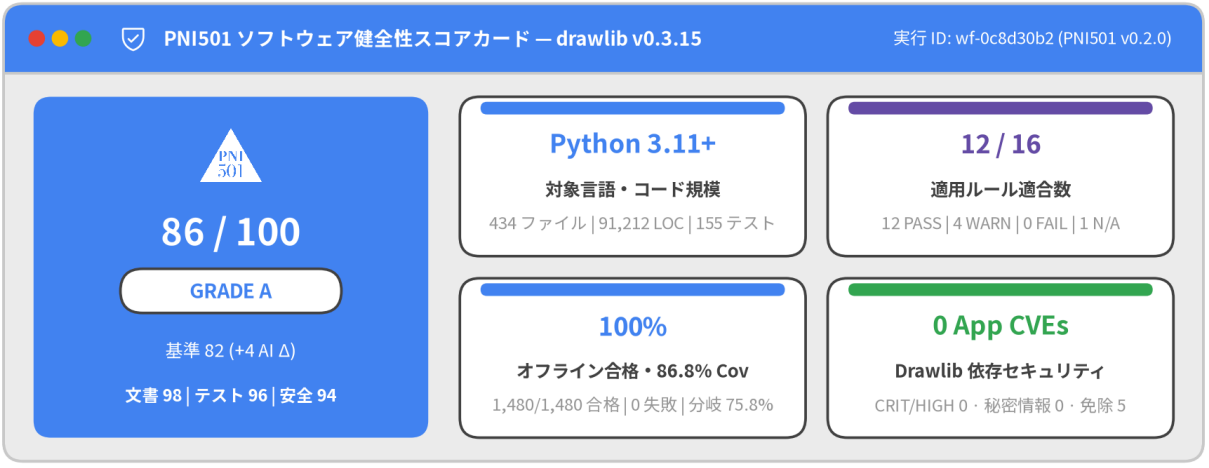
PNI ソフトウェア健全性診断レポート

対象システム: `drawlib` (`v0.3.15`) — 静的 AST・アーキテクチャ境界・オフラインコンテナ実行・サプライチェーンセキュリティ総合監査



Drawlib v0.3.15 — PNI501 v0.2.0 フル監査エグゼクティブ・スコアカード (実行 ID: wf-0c8d30b2)

診断スコープと監査メタデータ

`pni501` `v0.2.0` (`full_audit`) による `drawlib` (`v0.3.15`) のエンドツーエンド技術診断です。決定論的 Python AST 解析、階層アーキテクチャ境界検証、`--network=none` 隔離 Docker コンテナ実行テスト、Trivy `0.69.3` ファイルシステム / イメージ脆弱性スキャン、および `.pni/pni.toml` ポリシー検証を統合しています。

項目	内容
対象システム・種別	`drawlib` (`v0.3.15` , Python `>=3.11` [`3.11–3.14`], PEP 621 `uv` / `uv_build` , `py.typed`) — `library` (公開ファサード 38 ファイル / トップレベル遅延エクスポート 20 モジュール) + `cli` (9 コマンド群・39+ コマンド)
ワークフロー・ポリシー	`full_audit` (`wf-0c8d30b2` , PNI501 `v0.2.0` , 全 `18` ステップ / 論理 DAG `12` ステージ, `.pni/pni.toml` 有効アーキテクチャ免除 `5` 件) — Git `main@28aff03e` ・評価日 `2026-11-22`
コードベース規模	ソース `434` ファイル (`91,212` LOC・ `317` クラス・ `2,089` 関数) およびテスト `155` ファイル (pytest 収集 `1,485` 件 / トップレベル AST テスト `1,341` 件)
総合健全性スコア	**86 / 100** (`Grade A`) (決定論的ベースライン `82` -> AI 補正 `+4` 、各軸 `±10 pt` 上限) — ドキュメント `98` ・テスト `96` ・依存関係 `96` ・セキュリティ `94` ・型安全性 `90` ・アーキテクチャ `80` ・コード品質 `50`

項目	内容
ルール適合・検出件数	適用ルール `12 / 16` 件 `PASS`（`4 WARN`・`0 FAIL`・`1 N/A`、全体 `12 / 17`）— `0` CRIT・`0` HIGH・`446` MED・`14` LOW・`2` INFO（計 `462` 件。Drawlib 依存 CVE `0`・秘密情報 `0`・IaC 不備 `0`・免除 `5` 件が `454` 箇所に適合）

目次

1. エグゼクティブサマリー

2. アーキテクチャとソースコード構造

3. 推奨改善ロードマップ

4. 監査パイプライン（DAG）・標準適合・検出事項

5. 詳細分析：コード品質・テスト・コンテナ実行検証

6. 詳細分析：セキュリティ・サプライチェーン・API 攻撃面

1. エグゼクティブサマリー

Drawlib `v0.3.15` は、PNI501 `v0.2.0` のアンカー型フル監査メソドロジー（各評価軸 `±10 pt` 上限）のもと、**434 個の Python ソースモジュール（`91,212` LOC・`317` クラス・`2,089` 関数）**および **155 個のテストファイル（pytest 収集 `1,485` 件 / トップレベル AST テスト `1,341` 件）**全体で、総合健全性スコア **86 / 100（Grade A）**（決定論的ベースライン `82` から AI 補正 `+4` ）を達成しました。静的 AST 検証では **関数型カバレッジ 100.0%**、**Docstring・インポート衛生 99.3%**、L1–L7 層間の **境界違反 0 件**、および **バグリスク構文 0 件** を確認。さらに `--network=none` 隔離 Docker コンテナ実行で **1,480 / 1,485 テスト合格（失敗 `0`・エラー `0`・環境スキップ `5`、行カバレッジ `86.8%` / 分岐カバレッジ `75.8%` ）**、Trivy `0.69.3` スキャンで **CRITICAL/HIGH 脆弱性 0 件・Drawlib 自身の依存ツリー CVE 0 件・IaC 設定不備 0 件・秘密情報漏洩 0 件** を実証しています。

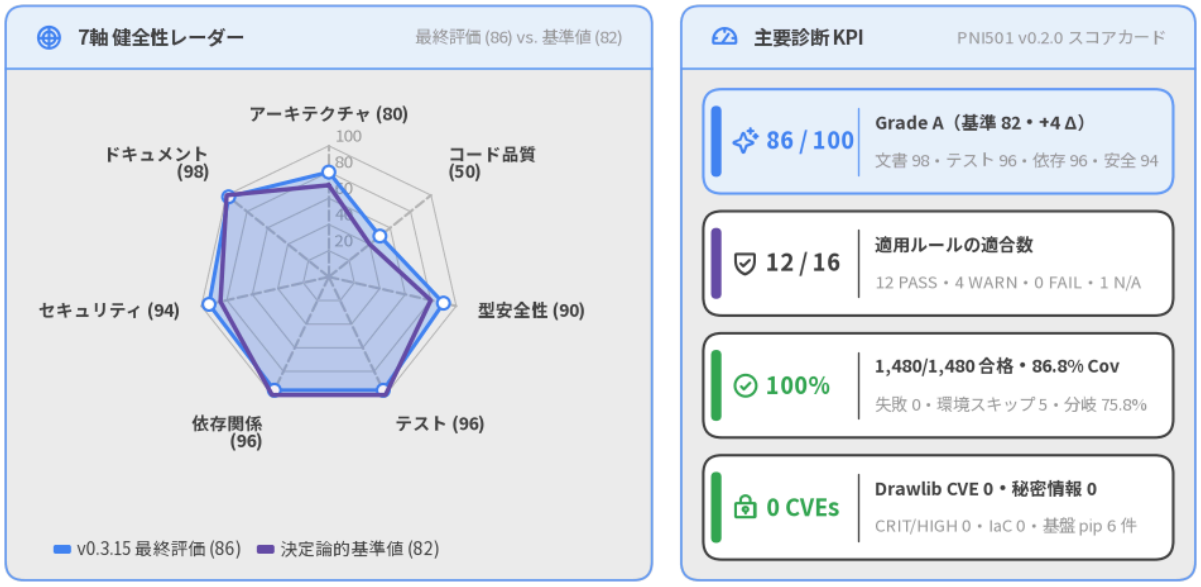


図 1.1 — 7 軸エンジニアリング健全性レーダー（最終評価 86 vs. 決定論的ベースライン 82）および主要診断 KPI

1.1 エグゼクティブ総括と診断ハイライト

PNI501 `v0.2.0` ワークフロー `full_audit`（`wf-0c8d30b2`、Git `main@28aff03e`）は、Drawlib `v0.3.15` に対し **全 18 ステップ（論理 DAG 12 ステージ）** を実行しました（`.pni/pni.toml` の有効免除 `5` 件が `454` 箇所に適用され、決定論的検出 `10` 件が AI トリアージにより再分類）。

- **PNI501 `v0.2.0` アンカー評価と総合水準**: 各評価軸の AI 補正幅を `±10 pt` 以内に厳格制限するアンカー評価方式のもと、決定論的ベースライン **82** から `+4` 補正の **86 / 100**（`Grade A`）を記録しました。**適用ルール `12 / 16` 件が** `PASS`（`4 WARN`・`0 FAIL`・`1 NOT_APPLICABLE` [`python/frameworks/fastapi`]、全 `17` ルール中 `12` 件合格）。全 `462` 件の統合検出項目のうち **CRITICAL `0` 件・HIGH `0` 件** であり、内訳は MEDIUM `446` 件（SIG 関数規模・複雑度 `442` 件 + 循環インポート群 `4` 件）、LOW `14` 件（ベースイメージ `pip@25.0.1` CVE `6` 件 + 改善項目 `8` 件）、INFO `2` 件です。
- **`v0.3.5` から `v0.3.15` への着実な品質向上**: コードベースが `424` モジュール（`85,323` LOC・`150` テスト）から `434` **モジュール（`91,212` LOC・`155` テスト / `1,485` pytest 項目）** へと拡大する中、`v0.3.5` で挙げた主要フォローアップ項目をすべて実装しました。具体的には、`SECURITY.md`（Illustration-as-Code 実行信頼モデルを明記）・`CODE_OF_CONDUCT.md`・週次 Dependabot（`uv` + `github-actions`）の整備、`.pni/dockerfiles/Dockerfile` へのヘッドレス Chromium（`libatomic1` 含む）および全 `83` 種のキャッシュアセット事前組み込みによるオフライン実行テスト合格率 `99.5%` -> `100.0%`（`1,480 / 1,480` **実行合格・失敗** `0`）と行カバレッジ `86.3%` -> `86.8%` の向上、さらに 6 つの日英バイリンガル Diataxis ポータルに対する 100% API/CLI 網羅性自動テストの確立です。`v0.3.5` の非アンカー `94`（`Grade S`）から `v0.3.15` の `86`（`Grade A`）への数値変動は品質低下ではなく、`v0.2.0` で導入された厳格な決定論的 SIG ベースライン（`コード品質` `40 -> 50`、`アーキテクチャ` `70 -> 80`）によるものであり、主要 5 検証軸（ドキュメント `98`・テスト `96`・依存関係 `96`・セキュリティ `94`・型安全性 `90`）はすべて `Grade S` 水準を維持・向上しています。
- **監査ステップ別スコア（低い順 -> 高い順）**: SIG 複雑度と `8` 件の静的循環インポートが集中する `python_ast_audit = 88/100 (Grade A)` が唯一の A 評価であり、残る 7 ステップはいずれも **Grade S** を獲得しています（`cli_guidelines_audit = 93`、`security_vulnerability_audit = 94`、`library_hygiene_audit = 94`、`supply_chain_audit = 95`、`python_test_audit = 95`、`repo_structure_audit = 96`、`container_runtime_audit = 96`）。
- **最優先の改善レバー**: `Grade S`（`>=90`）との差分の大半は `コード品質`（`50 / 100`）に集中しています。`_builder/_common/parallel.py`・`compiler/html.py <->`

`compiler/jupyter.py`・`_diagrams/_edge <-> _junction` の `8` 件の静的循環インポート解消、ビルダーオーケストレーター上位関数（`compiler/html.py`・`parallel.py`・`manifest.py`・`exporter_docx.py`）の分割、およびベースイメージ `pip@25.0.1 -> >=26.2.0` の更新により、アンカー評価基準のもとでも `Grade S` へ復帰できます。

1.2 7評価軸 エンジニアリング健全性ブレークダウン

評価カテゴリ	ベースライン	AI 補正 (±10pt)	最終スコア	監査根拠・主要エビデンス
ドキュメント	`100`	`-2`	**98 / 100**	Google スタイル Docstring カバレッジ `99.3%`（空 Docstring `0` 件）、6 つの日英バイリンガル Diataxis ポータル（`site`, `api`, `architecture`, `quickstart`, `about-drawlib`, `jupyter`）、`src/drawlib/_rules/` の組み込み AI ルール、`SECURITY.md` + `CODE_OF_CONDUCT.md`、および EN/JA API & CLI 100% 網羅性自動テスト。未記載の `@property` セッター `12` 件により `-2`。
テスト	`100`	`-4`	**96 / 100**	`--network=none` 隔離 Docker 実行で `1,480 / 1,485` テスト合格（失敗 `0`・エラー `0`・環境スキップ `5`）、行カバレッジ `86.8%`（`40,102 / 46,204`）・分岐カバレッジ `75.8%`・`155` テストファイル（トップレベル AST テスト `1,341` 件）による AST モジュール網羅率 `100.0%`、およびピクセル一致率 `>=97%` の視覚回帰検証。`parallel.py`（`61.1%` / `50.6%`）・`cli/cache.py`（`61.2%` / `30.0%`）・`__main__.py`（`0.0%`）の不足分により `-4`。
依存関係	`100`	`-4`	**96 / 100**	PEP 621 `pyproject.toml`（`uv` / `uv_build`）上の全 `13` ランタイム依存を SemVer `>=` 下限固定（未固定 `0` 件・Drawlib 依存 CVE `0` 件）、週次 Dependabot（`uv` + `github-actions`）稼働、`python/lockfile_committed` は免除適用済み。GitHub Actions の浮動 `@v7` タグと `109` 行の `docs_src/site/build_html.sh` により `-4`。
セキュリティ	`85`	`+9`	**94 / 100**	Trivy `0.69.3` のファイルシステムおよび解決済みコンテナイメージスキャンで CRITICAL `0`・HIGH `0`・Drawlib 依存 CVE `0`・IaC 不備 `0`（`DS-0026` 3 件免除）・秘密情報 `0`・未免除 SAST シンク `0`（`CWE-94` 8 件免除）を確認。`SECURITY.md`・SHA-256 アセット検証・最小権限 CI 5 ワークフローを完備。ベースイメージ内システム `pip@25.0.1` の CVE `6` 件（`26.2.0` で修正済み）により `94` 点で抑制。
型安全性	`80`	`+10`	**90 / 100**	全 `2,089` 関数で AST 型注釈カバレッジ `100.0%`、無限定 `# type: ignore` `0` 件、PEP 561 `py.typed` 同梱、`ty` 静的型検査エラー `0` 件。動的ファサードや `manifest.py` 等に `Any` や未パラメータ化 `dict` が `200` 箇所残存。

評価カテゴリ	ベースライン	AI 補正 (±10pt)	最終スコア	監査根拠・主要エビデンス
アーキテクチャ	`70`	`+10`	**80 / 100**	`434` モジュール（ `91,212` LOC ）の L1–L7 層間（ `l1_core -> l2_types -> l3_* -> l4_canvas -> domain -> builders -> PEP 562 facades` ）で境界違反 `0` 件。一方、静的循環インポート `8` 件（ `_builder/_common/parallel.py` の遅延ワーカーインポート `5` 件、 `compiler/html.py <-> compiler/jupyter.py` `1` 件、 `_diagrams` の `_edge <-> _junction` `2` 件）が検出対象。
コード品質	`40`	`+10`	**50 / 100**	厳格な決定論的 SIG 基準がビルダー / レンダラーオーケストレーター（ `compiler/html.py` , `compiler/jupyter.py` , `exporter_docx.py` , `parallel.py` , `manifest.py` ）の `>50` LOC または認知複雑度 `>15` の `442` 関数を減点。Docstring `99.3%` ・インポート衛生 `99.3%` ・バグリスク構文 `0` 件・Ruff+Ty クリーン・キーワード専用 API `353` 件の免除実績を踏まえ `+10` 補正。
総合健全性	`82`	`+4`	**86 / 100 (A)**	PNI501 `v0.2.0` の各軸 `±10 pt` アンカー制約下における 7 軸加重総合評価（全 `18` ステップ / 論理 DAG `12` ステージ、適用ルール `12 / 16` 件 `PASS` ・ `4 WARN` ・ `0 FAIL` ・ `1 N/A` ）。

1.3 検証済みの強みと優先リスク

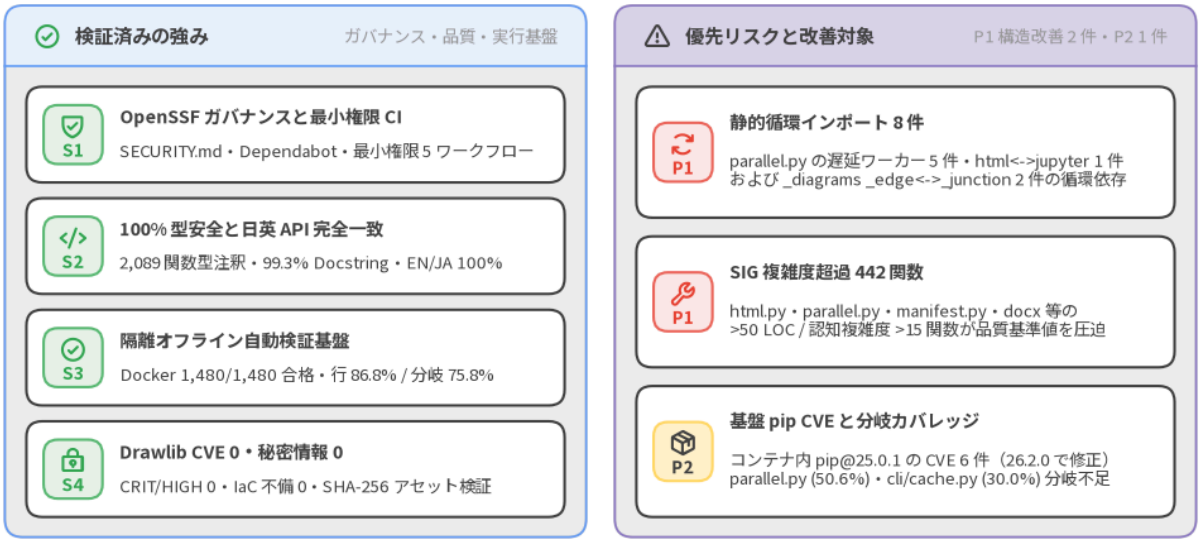
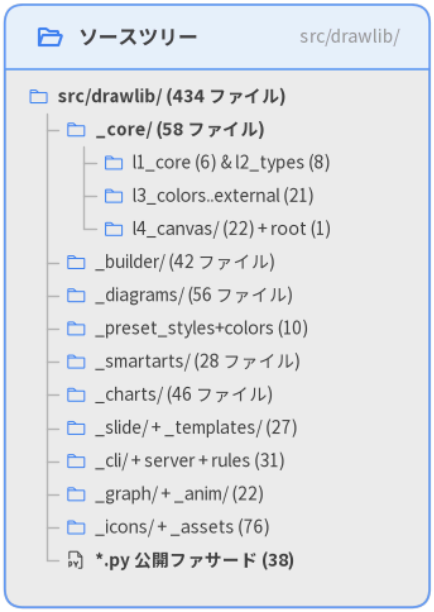


図 1.2 ― 検証済みのエンジニアリング上の強み（S1–S4）と優先改善リスク（P1–P2）

- **アンカー評価下での `Grade S` 復帰ロードマップ**: Drawlib `v0.3.15` はすでに `FAIL` ルール `0` 件・`CRITICAL` / `HIGH` 脆弱性 `0` 件・Drawlib 依存 CVE `0` 件・オフラインテスト `100%` 実行合格という堅牢な基盤を備えています。第 3〜6 章で詳述する `P1` 構造改善 2 件（`parallel.py` 昇格と `compiler/base.py` 分離、および上位オーケストレーター関数の分割）と、`P2` 仕上げ項目（ベースイメージ `pip` の `26.2.0` 更新、GitHub Actions の SHA 固定、`parallel.py`・`cli/cache.py` の分岐テスト強化）を実行することで、残る `WARN` ルール 4 件を解消し、`Grade S` 水準へ到達できます。

2. アーキテクチャとソースコード構造

Drawlib v0.3.15 は、**434 個の Python ソースファイル（`91,212` LOC・`317` クラス・`2,089` 関数・`155` テストファイル）**にわたり、**厳格な階層型コア + ドメイン可視化サブシステム + マルチターゲット文書コンパイルパイプライン（PEP 562 遅延公開ファサード配下）**を実装しています。コア層（`L1-L2` 〜 `L4`）およびドメイン可視化層（`L5`）は **境界違反ゼロ** の単方向依存を維持しており、PNI501 v0.2.0 テレメトリが特定した残存 **8 件の静的循環インポート** と **8 件の最高リスク複雑度ホットスポット（`risk_score = 1.00`）**は、L6 マルチターゲットビルダー群（`compiler/html.py`, `image_builder.py`, `exporter_docx.py`, `exporter_jupyter.py`, `parallel.py`）、L5 スライドコンパイラ（`_slide/compiler.py`）、および L4 図形ビルダー（`_symbol_builders.py`, `_container_builders.py`）に局所化されています。





パッケージ / 領域	件数	LOC	主なアーキテクチャ責務
_builder/	42	16,219	設定・SQLite WAL・並列・各種コンパイラ
_core/ (l1..l4)	58	13,641	設定・型定義・L3 アセット・L4 キャンバス
_diagrams/	56	13,147	6 種ダイアグラム（構成・フロー・UML・ER）
_preset_styles + _colors	10	12,997	セマンティックトークン・Google/モノクロ配色
_smartarts/	28	8,680	18 種 SmartArt（表・ツリー・地図・KPI 等）
_charts/	46	7,283	10 種チャート（Box・HeatmapOverlay 等）
drawlib/*.py ファサード	38	6,062	PEP 562 遅延公開 API とサブファサード
_slide/ + _templates/	27	5,764	16:9 HTML/PDF/PPTX と 6 種テンプレート
_cli/ + server + _rules	31	3,339	9 種 Typer CLI・プレビュー検証・AI ルール
_graph/ + _anim/	22	3,139	5 種決定論的グラフ配置・WebP アニメ
_icons/ + _assets	76	941	34 アイコン群・SHA-256 zip マニフェスト
合計 (src/drawlib/)	**434**	**91,212**	**100% 型付・L1〜L5 境界違反ゼロ**

図 2.1 — 物理ソースディレクトリ階層とモジュール構成インベントリ（434 ファイル・91,212 LOC）

2.1 システムプロファイルと物理モジュールインベントリ

Drawlib は **プログラマティック Python ライブラリ（`drawlib.*`）と ヘッドレスコンパイラ CLI（`drawlib`）**の二面を備え、Python `>=3.11`（`3.11〜3.14`、`uv` / `uv_build` PEP 621 管理）上で動作します。描画・出版基盤は、左下原点 `(0, 0)` のデカルト座標 2D ベクター / ラスター描画を担う **Matplotlib**（`>=3.8`）・**Pillow**（`>=13.0.0`）・**NumPy**（`>=1.26`）に、**Pydantic v2**（`>=2.8`）（`@validate_call` とイミュータブル `BaseModel` 設定）、**Typer**（`>=0.12.0`）**& Rich**（`>=13.0.0`）（9 コマンド CLI）、**markdown-it-py**（`>=3.0`）・**Jinja2**（`>=3.1`）・**Pygments**（`>=2.18`）・**MarkupSafe**（`>=2.1`）（Markdown/HTML/Jupyter コンパイル）、**Playwright**（`>=1.49.0`）（ヘッドレス Chromium による A4 PDF・スライド描画）、**python-docx**（`>=1.2.0`）**& python-pptx**（`>=1.0.2`）（Office OpenXML 出力）、および **標準ライブラリ** `sqlite3`（`.drawlib/cache.db` インクリメンタル SHA-256 キャッシュ）を組み合わせています。

パッケージ / サブシステム	ファイル数	LOC	階層	主な責務
`src/drawlib/_builder/`	42	16,219	**L6**	`drawlib.toml` 設定読込、SQLite WAL キャッシュ、並列事前描画、HTML/PDF/DOCX/Jupyter/MD/Image コンパイラ
`src/drawlib/_core/`（`l1..l4`）	58	13,641	**L1–L4**	`l1_core`（6）、`l2_types`（8）、`l3_*` サブシステム（21）、`l4_canvas`（22）、ルート（1）
`src/drawlib/_diagrams/`	56	13,147	**L5**	6 種テクニカルダイアグラム（`Architecture`, `Flow`, `Sequence`, `Class`, `ER`, `State`）
`src/drawlib/_preset_styles` + `_preset_colors`	10	12,997	**L3**	セマンティック・スタイルトークンと `Default` / `Google` / `Monochrome` パレット
`src/drawlib/_smartarts/`	28	8,680	**L5**	18 種 SmartArt（`Table`, `TreeNode`, `MindMapNode`, `Roadmap`, `KpiCards`, `GeoMap`, `Calendar` 等）
`src/drawlib/_charts/`	46	7,283	**L5**	`BoxChart` と `HeatmapOverlay` を含む 10 種定量チャート
`src/drawlib/*.py` ファサード（`charts/`, `diagrams/`）	38	6,062	**L7**	PEP 562 遅延ロード公開 API ファサード（`__init__.py`, `tools.py`, 各ドメイン公開窓口）
`src/drawlib/_slide/` + `_templates/`	27	5,764	**L5–L6**	16:9 HTML/PDF/PPTX スライドコンパイラと Jupyter を含む 6 種プロジェクトテンプレート
`src/drawlib/_cli/` + `_http_server` + `_rules`	31	3,339	**L6–L7**	9 種 Typer CLI モジュール、`--check` リンク検証付きプレビューサーバー、組込み AI ルール
`src/drawlib/_graph/` + `_anim/`	22	3,139	**L5**	5 種決定論的グラフレイアウトソルバーおよびマルチフレーム WebP アニメーション
`src/drawlib/_icons/` + `_assets`	76	941	**L3 / L5**	34 種アイコンスイート（アイコン / アセット管理 75 + `_cached_assets` 初期化 1）と SHA-256 zip マニフェスト

パッケージ / サブシステム	ファイル数	LOC	階層	主な責務
合計（`src/drawlib/`）	434	91,212	L1–L7	`317` クラス・`2,089` 関数・`155` テストファイル・`100%` 型付・コア境界違反ゼロ

外部ランタイム連携は以下の 4 境界に明確にカプセル化されています。

1. **GitHub Releases アセット配布**（`src/drawlib/_core/l3_external/_package.py`・`src/drawlib/_release_assets.py`）: ネットワーク通信前に `pkg.files` でファイル存在を検証し、`yuichi110/drawlib` から 83 種のフォント・アイコン・GeoJSON 地図 `.zip` パッケージを SHA-256 検証 + アトミック展開で遅延取得します。
2. **Playwright ヘッドレス Chromium**（`src/drawlib/_builder/_common/browser.py`）: クロスプラットフォームの `fcntl` / `msvcrt` OS ロックでブラウザ準備・A4 PDF 改ページ出力・16:9 スライド DOM / ラスター書き出しを調停します。
3. **Google Fonts および Google Colaboratory**: 生成 HTML・Jupyter ノートブック向けの任意フォントリンクおよび Colab 起動リンクを提供します。
4. **PyPI / TestPyPI 配布**: `uv_build` による PEP 621 ビルドと `./dcli docker test` によるクリーンルーム検証を実施します。

2.2 多層アーキテクチャ、依存境界およびリスク・ホットスポット

Drawlib は全 `434` モジュールを **6 つの記録済みアーキテクチャ階層**（`L1-L2`, `L3`, `L4`, `L5`, `L6`, `L7`）に分離しています。

- **コア境界違反ゼロ**: 描画コア層（`L1-L2 -> L3 -> L4`）およびドメイン可視化ライブラリ（`L5`）は、PEP 562 遅延公開ファサード（`L7`）の背後で厳格な単方向依存を維持しています。`40` 件の型参照サイクルはすべて `if TYPE_CHECKING:` で保護されており実行時影響はゼロです。
- **局所化された 8 件の静的循環インポート**（`ARCH-02` **改善対象**）: 静的 / 実行時の循環インポートは 2 領域に限定されています。L6 ビルダー制御の **6 件**（`src/drawlib/_builder/_common/parallel.py` 内のワーカー関数内遅延インポートによる上位ビルダー参照 5 件と、`compiler/html.py <-> compiler/jupyter.py` 間のアセット解決ヘルパー相互参照 1 件）、および `src/drawlib/_diagrams/{architecture,flow}` 内の軽微な兄弟サイクル **2 件**（`_edge.py`

`<-> _junction.py`）です。`parallel.py` を `_builder/_common` の上位へ引き上げ、共通 favicon/logo 解決ヘルパーを `compiler/base.py` へ集約することで完全非循環グラフを達成できます。

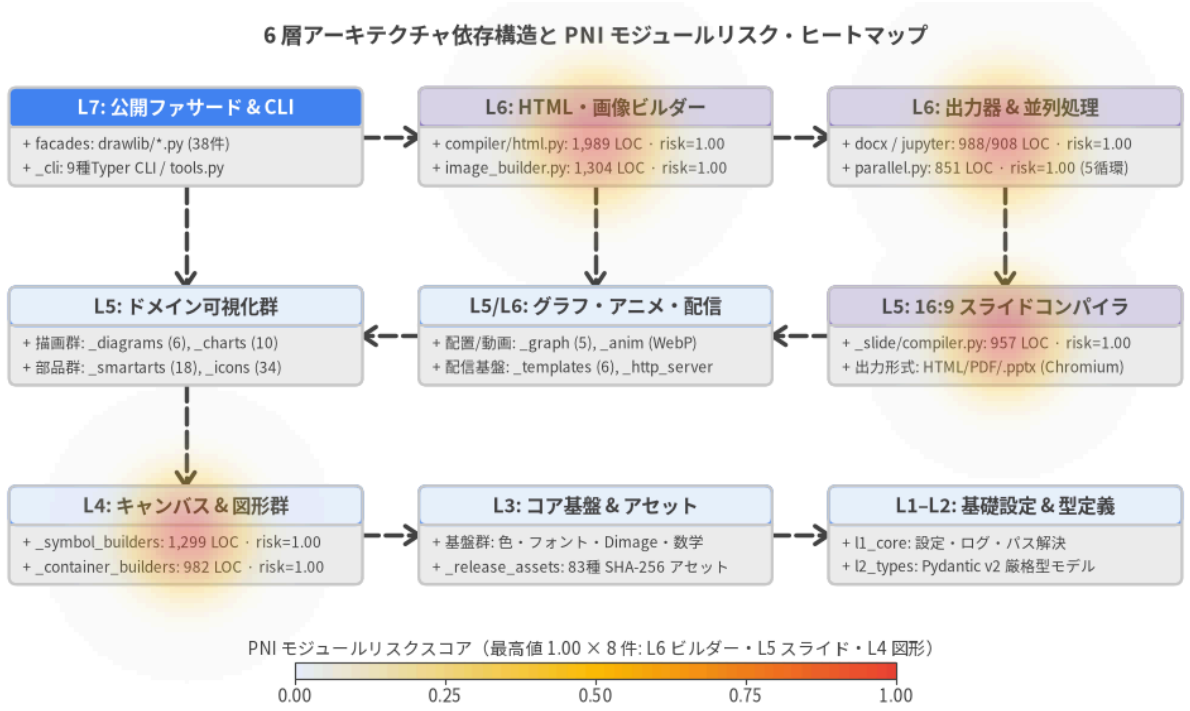


図 2.2 – 6 層アーキテクチャ依存関係と PNI リスク・ホットスポット分布（最高値 1.00 × 8 モジュール）

アーキテクチャ階層契約（`L7 -> L1-L2`）

階層	アーキテクチャ層	物理パス	許可される依存先
L7	公開ファサード & CLI 配信層	`src/drawlib/__init__.py`, `__main__.py`, `tools.py`, `charts/`, `diagrams/`, `_cli`	`L1–L6`、`typer`、`rich`
L6	マルチターゲット文書ビルダー & アセット基盤	`src/drawlib/_builder`, `_templates`, `_http_server`, `_rules`	`L1–L5`、`markdown-it-py`、`jinja2`、`playwright`、`python-docx`
L5	ドメイン可視化コンポーネント群	`src/drawlib/_icons`, `_smartarts`, `_charts`, `_diagrams`, `_graph`, `_anim`, `_slide`	`L1–L4`、`python-pptx`
L4	キャンバス描画エンジン	`src/drawlib/_core/l4_canvas`	`L1–L3`、`matplotlib`
L3	コアサブシステム & アセットエンジン	`src/drawlib/_core/l3_*`, `_preset_colors`, `_preset_styles`, `_release_assets.py`	`L1–L2`、`pillow>=13.0.0`、`numpy`

階層	アーキテクチャ層	物理パス	許可される依存先
L1–L2	コア基盤 & 型定義	`src/drawlib/_core/l1_core`, `_core/l2_types`	Python 標準ライブラリおよび `pydantic` のみ

リスク・ホットスポット一覧（`risk_score = 1.00`）

順位	モジュールパス	階層	LOC	リスク値	検出件数	該当ルール
1	`src/drawlib/_builder/doc_builder/compiler/html.py`	L6	1,989	`1.00`	中 18 件	`common/architecture`, `common/maintainability`
2	`src/drawlib/_builder/image_builder.py`	L6	1,304	`1.00`	中 15 件	`common/maintainability`
3	`src/drawlib/_core/l4_canvas/_shapes/_symbol_builders.py`	L4	1,299	`1.00`	中 9 件	`common/maintainability`
4	`src/drawlib/_builder/doc_builder/exporter_docx.py`	L6	988	`1.00`	中 7 件	`common/maintainability`
5	`src/drawlib/_core/l4_canvas/_shapes/_container_builders.py`	L4	982	`1.00`	中 9 件	`common/maintainability`
6	`src/drawlib/_slide/compiler.py`	L5	957	`1.00`	中 11 件	`common/maintainability`
7	`src/drawlib/_builder/doc_builder/exporter_jupyter.py`	L6	908	`1.00`	中 7 件	`common/maintainability`
8	`src/drawlib/_builder/_common/parallel.py`	L6	851	`1.00`	中 8 件・低 1 件	`common/architecture`, `common/maintainability`, `common/runtime_verification`

2.3 実行フロー、状態管理および並行処理モデル



図 2.3 — 記録済みランタイム実行パイプラインと状態・永続化・ロック設計（v0.3.15）

- **実行エントリーポイント**: ライブラリ利用者は `src/drawlib/__init__.py`・`src/drawlib/canvas.py`・`src/drawlib/tools.py` から呼び出し、CLI 自動化は `src/drawlib/__main__.py:main` -> `src/drawlib/_cli/_drawlib.py:app` および `src/drawlib/_cli/commands/build.py:build_app` を経由して 9 コマンド群（`init`, `build`, `show`, `serve`, `cache`, `css`, `styles`, `colors`, `rules`）を実行します。
- **状態管理・永続化モデル**:
  1. **アクティブキャンバスとテーマプロキシ**: `src/drawlib/_core/l4_canvas/_canvas.py` のプロセス内キャンバスシングルトンを `setup()` / `save()` / `clear()` で初期化・破棄し、`src/drawlib/_core/l3_styles` のアクティブ `Styles` / `Colors` プロキシと連動します。
  2. **コンテンツアドレス型 SQLite WAL キャッシュ**: `.drawlib/cache.db`（`src/drawlib/_builder/_common/cache.py` の `BuildImageCache` / `CliImageCache`）が SHA-256 ハッシュをキーに描画済み PNG/WebP バイナリを上限付きで保持します。
  3. **アトミックなローカルアセットキャッシュ**: `src/drawlib/_cached_assets/{fonts,fonticons,icons,maps,rules}` は一時ファイ

ル書き込み + `os.replace()` と OS ファイルロックにより安全に展開されます。

4. **ヘッドレス Chromium 調停**: Playwright ブラウザ取得・起動は `src/drawlib/_builder/_common/browser.py` のクロスプラットフォーム `fcntl` / `msvcrt` ロックで保護されています。

- **エラー制御と並行モデル**: 公開描画・ビルダー API は Pydantic (`@validate_call` およびイミュータブル `BaseModel`) で入力を即時検証し、契約違反時には明示的な `ValueError` / `FileNotFoundError` / `RuntimeError` (`jupyter` プロジェクトへの非対応ターゲット指定時の `ValueError` ガードを含む) を送出します。複数図版ブロックの事前描画は `src/drawlib/_builder/_common/parallel.py` の `ProcessPoolExecutor` で並列化されつつ単一プロセス逐次フォールバックと SQLite WAL 並行アクセスを備え、アセット展開・ブラウザ準備の OS ロックと併せてマルチワーカー破損を防止しています。

3. 推奨改善ロードマップ

Drawlib v0.3.15 は本 PNI501 v0.2.0 総合診断において **総合スコア 86/100（Grade A）**・`P0` **重大ブロッカー 0 件** を達成しました。前回の `v0.3.5` 診断で指摘された主要な `P0` / `P1` 課題（`pillow>=13.0.0` による 41 件の Pillow CVE 解消、非 root Docker `USER drawlib` `UID 10001` 化、`pkg.files` オフライン事前検証、Chromium + 83 アセットの事前同梱、ルート `SECURITY.md`・Dependabot 整備、ライブラリ内 `sys.exit(1)` の排除）はすべてリポジトリ内で完了・検証済みです。本章ではこの堅牢化された基盤のうえで、完全非循環の `ARCH-02 PASS` 達成、ビルダー認知複雑度の低減、CLI / ライブラリ純粋性・サプライチェーン固定化・コンテナ CVE ゼロ化・分岐カバレッジ `80%` 超を実現する **優先順位順の 10 アクション**（`0 P0`・`2 P1`・`8 P2`）を定義します。



図 3.1 — Drawlib v0.3.15 段階的改善タイムライン（0 P0・2 P1・8 P2）

3.1 改善戦略と優先度フェーズ

> `v0.3.15` **で完了・検証済みの** `v0.3.5` **マイルストーン**: 前回診断の緊急ブロッカーはすべて完了しています―― `pillow>=13.0.0` （ `uv.lock` の PyPI アプリ CVE を 41 件から `0` 件へ解消）、**非 root Docker** `USER drawlib` **（** `UID 10001` **）**（ `DS-0002` IaC 指摘ゼロ）、 `pkg.files` **オフライン事前メンバー検証**、 `.pni/dockerfiles/Dockerfile` **への Chromium + 83 アセット事前同梱**、 `SECURITY.md` ・ `.github/dependabot.yml` **の整備**、およびライブラリ層 `sys.exit(1)` **の排除**。

`P0` **ブロッカーゼロ** の健全なベースラインのうえに、 `v0.3.15` では以下の 3 段階・**10 件の優先アクション** を計画しています。

1. **フェーズ** `P1` **― アーキテクチャ完全非循環化とビルダー複雑度分割（アクション** `#1–#2` **）**: 残存する `8` 件の静的循環インポートを完全解消して `ARCH-02 PASS`（ `0` 循環）を達成し、L6 コンパイラ・オーケストレーションの大型関数（ `compiler/html.py` `1,989` LOC、 `_execute_block_render_job` 認知複雑度 `68` 、 `format_project_manifest_toml` 認知複雑度 `48` 、および `exporter_docx.py` / `compiler/jupyter.py` ）を `50` LOC 未満の凝集ヘルパーへ分割します。
2. **フェーズ** `P2-A` **― ライブラリ純粋性・CLI 操作性・型厳格化（アクション** `#3` **,** `#4` **,** `#10` **）**: CLI 一覧系コマンドへの `--json` 出力追加と `_ensure_overview_rules_cached()` の実行範囲限定、ビルダー層 `34` 箇所の直接 `print()` のロガー / コールバック化、 `manifest.py` の `dict[str, str]` 型厳格化、および `15` 箇所の `DRAWLIB_*` 環境変数参照の集約を行います。
3. **フェーズ** `P2-B` **― サプライチェーン堅牢化・コンテナ CVE 解消・分岐網羅率** `>80%` **（アクション** `#5–#9` **）**: 109 LOC の `docs_src/site/build_html.sh` を型付き Python `dcli` へ移行し、GitHub Actions を不変な 40 文字コミット SHA へ固定、コンテナベースイメージ内 `pip 25.0.1` の更新 / 除去（コンテナ内アプリ CVE `6 -> 0` ）、および `parallel.py` ・ `commands/cache.py` ・ `__main__.py` の分岐カバレッジを `80%` 超へ引き上げます。

3.2 アクションプラン・トレーサビリティマトリクス (P1 -> P2)

以下のマトリクスは、優先順位順に整列された **全 10 件の** `v0.3.15` **改善アクション** (`2 P1` + `8 P2`) について、対象ファイル・期待効果・関連ルール / CVE を対応付けています。

アクションプラン・トレーサビリティマトリクス — 優先順位順 10 件（0 P0・2 P1・8 P2）

優先度・#	改善アクション	対象ファイル	期待効果	関連ルール / CVE
[P1]・#1 構成	ビルダー・図版層の静的循環インポート 8 件を完全解消	parallel.py, html.py, jupyter.py, base.py	静的循環 8 -> 0 件; ARCH-02 PASS 達成	common/architecture, common/maintainability
[P1]・#2 品質	ビルダー上位の認知複雑度関数を小単位へ構造分割	compiler/html.py, parallel.py, manifest.py	関数 <50 LOC 化で SIG 保守性・試験性向上	common/maintainability
[P2]・#3 品質	CLI 一覧へ --json 追加とルール事前書込の範囲限定	_cli/_drawlib.py, cache.py, rules.py	clig.dev 連携向上と読取専用コンテナ安全化	common/cli_guidelines
[P2]・#4 品質	ビルダー内 34 件の print() をロガー／コールバックへ置換	progress.py, parallel.py, compiler/jupyter.py	drawlib.tools 組込時の stdout 汚染を防止	common/ library_guidelines
[P2]・#5 保守	109 LOC の build_html.sh を型付 Python dcli へ移行	docs_src/site/build_html.sh, dcli/docs/builder.py	唯一のレガシーシェル検出指摘を解消	common/supply_chain
[P2]・#6 保安	GitHub Actions のステップを不変 40 文字 SHA へ固定	quality.yml, release.yml, ut_template.yml	OpenSSF Scorecard Pinned-Deps を最大化	common/supply_chain
[P2]・#7 保安	コンテナ内ベース pip 25.0.1 を更新または除去	.pni/dockerfiles/Dockerfile, Dockerfile.test_pypi	コンテナイメージ内アプリ CVE を 6 -> 0 化	security_vulnerability, CVE-2026-8643/-13346
[P2]・#8 試験	CLI cache と並列ワーカーの残存カバレッジ不足を解消	test_cli_css.py, __main__.py, test_parallel_build.py	非生成モジュール <65% 行網羅の残件を解消	python/testing
[P2]・#9 試験	parallel.py・CLI cache の分岐カバレッジを集中的拡充	parallel.py, __main__.py, _cli/commands/cache.py	重要 CLI・並列経路の分岐網羅率を >80% へ	common/ runtime_verification
[P2]・#10 型付	未パラメータ化 dict/Any 厳格化と DRAWLIB_* 環境変数集約	manifest.py, browser.py, cache.py, progress.py	静的型精度の向上とライブラリ組込性強化	python/typing, python/modern_stack

図 3.2 — 優先順位順 全 10 改善アクション・トレーサビリティマトリクス (0 P0・2 P1・8 P2)

全 10 件 優先順位順アクション一覧

#	優先度	分類	アクション名	対象ファイル	関連ルール / CVE	期待効果
1	`P1`	`architecture`	**ビルダーおよびダイアグラムモジュールの静的循環インポート 8 件を解消**	`src/drawlib/_builder/_common/parallel.py`, `src/drawlib/_builder/doc_builder/compiler/html.py`, `src/drawlib/_builder/doc_builder/compiler/jupyter.py`, `src/drawlib/_builder/doc_builder/compiler/base.py`	`common/architecture`, `common/maintainability`	静的 / 実行時循環依存を `8 -> 0` に削減し、厳格な `ARCH-02 PASS` を達成。

#	優先度	分類	アクション名	対象ファイル	関連ルール / CVE	期待効果
2	P1	code_quality	ビルダー層の上位認知複雑度関数を分割	src/drawlib/_builder/doc_builder/compiler/html.py, src/drawlib/_builder/_common/parallel.py, src/drawlib/_builder/_common/manifest.py	common/maintainability	50 LOC 未満の凝集ヘルパーへ分割し、SIG 保守性と単体テスト容易性を向上。
3	P2	code_quality	CLI カタログ一覧コマンドへの --json 追加とルール事前キャッシュの範囲限定	src/drawlib/_cli/_drawlib.py, src/drawlib/_cli/commands/cache.py, src/drawlib/_cli/commands/rules.py	common/cli_guidelines	clig.dev 準拠のスクリプト連携性と読取専用コンテナでの安全性を確保。
4	P2	code_quality	プログラマティック・ビルダー層の直接 print() をロガー / 進捗コールバックへ置換	src/drawlib/_builder/_common/progress.py, src/drawlib/_builder/_common/parallel.py, src/drawlib/_builder/doc_builder/compiler/jupyter.py	common/library_guidelines	サービス・ノートブック・外部 CLI 組み込み時に drawlib.tools の標準出力をクリーンに維持。
5	P2	maintainability	肥大化した Bash スクリプト docs_src/site/build_html.sh を型付き Python dcli へ移行	docs_src/site/build_html.sh, tools/dcli/docs/builder.py	common/supply_chain	唯一のレガシー言語シェルスクリプト指摘を解消し、クロスプラットフォーム保守性を改善。
6	P2	security	GitHub Actions の各ステップを不変なコミット SHA へ固定	.github/workflows/quality.yml, .github/workflows/release.yml, .github/workflows/ut_template.yml	common/supply_chain	可変タグ改ざんリスクを排除し、OpenSSF Scorecard Pinned-Dependencies を最大化。
7	P2	security	コンテナ Dockerfile 内のベースイメージ pip 25.0.1 を更新または除去	.pni/dockerfiles/Dockerfile, tools/docker/Dockerfile.test_pypi	common/security_vulnerability, CVE-2026-8643, CVE-2026-13346	Trivy コンテナイメージ内アプリ CVE を 6 -> 0 へ削減。
8	P2	testing	CLI cache コマンドと並列ビルドワーカーの残存分岐カバレッジ不足を解消	tests/drawlib/cli/test_cli_css.py, tests/drawlib/doc_builder/test_parallel_build.py, src/drawlib/__main__.py	python/testing	非生成モジュールにおける <65% 行カバレッジ残件を解消。
9	P2	testing	parallel.py および CLI cache コマンドの分岐カバレッジを強化	src/drawlib/_builder/_common/parallel.py, src/drawlib/_cli/commands/cache.py, src/drawlib/__main__.py	common/runtime_verification	重要 CLI および並列オーケストレーション経路の分岐カバレッジを >80% へ向上。

#	優先度	分類	アクション名	対象ファイル	関連ルール / CVE	期待効果
10	`P2`	`typing`	未パラメータ化 `dict` / `Any` 型注釈の厳格化とビルダー環境変数・ログ管理の集約	`src/drawlib/_builder/_common/manifest.py`, `src/drawlib/_builder/_common/browser.py`, `src/drawlib/_builder/_common/cache.py`, `src/drawlib/_builder/_common/progress.py`	`python/typing`, `python/modern_stack`	型検査精度とライブラリ組み込み時の制御性を向上。

3.3 優先度別の詳細実装仕様

フェーズ `P1` — アーキテクチャ非循環化・ビルダー複雑度分割 (`2` 件)

アクション #1 (`[P1]` ・ `architecture`) : ビルダーおよびダイアグラムモジュールの静的循環インポート 8 件を解消

- **対象ファイル**: `src/drawlib/_builder/_common/parallel.py`, `src/drawlib/_builder/doc_builder/compiler/html.py`, `src/drawlib/_builder/doc_builder/compiler/jupyter.py`, `src/drawlib/_builder/doc_builder/compiler/base.py`
- **関連ルール**: `common/architecture`, `common/maintainability`
- **実装方針**: `parallel.py` を `src/drawlib/_builder/_common/` の上位 (例: `src/drawlib/_builder/parallel.py`) へ再配置し、ワーカー制御が `_common` および各コンパイラへ一方向に依存する構成へ改めることで `5` 件の遅延インポート循環を解消します。あわせて `resolve_favicon_href` と `resolve_logo_hrefs` を `compiler/base.py` へ移動して `compiler/html.py <-> compiler/jupyter.py` 間の循環を切り離し、`src/drawlib/_diagrams/{architecture,flow}` 内の `_edge.py <-> _junction.py` を共通接続エンドポイント `Protocol` で疎結合化します。
- **期待効果**: 全 `434` ソースモジュールの静的 / 実行時循環依存を `8` 件から `0` 件へ削減し、厳格な `ARCH-02 PASS` を達成します。

アクション #2 (`[P1]` ・ `code_quality`) : ビルダー層の上位認知複雑度関数を分割

- **対象ファイル**: `src/drawlib/_builder/doc_builder/compiler/html.py` , `src/drawlib/_builder/_common/parallel.py` , `src/drawlib/_builder/_common/manifest.py`
- **関連ルール**: `common/maintainability`
- **実装方針**: `compiler/html.py` (`1,989` LOC) からモード別コンパイル処理 (site・doc・catalog・バンドル Jupyter) を独立モジュールへ抽出します。同時に `parallel.py` の `_execute_block_render_job` (認知複雑度 `68`) および `manifest.py` の `format_project_manifest_toml` (認知複雑度 `48`) 、ならびに `exporter_docx.py` ・ `compiler/jupyter.py` の高分岐関数を `50` LOC 未満の凝集ヘルパー群へ分割します。
- **期待効果**: `risk_score = 1.00` 上位ホットスポットの SIG 保守性とレビュー容易性を大きく改善し、並列実行・マニフェスト直列化経路の単体テスト性を高めます。

フェーズ `P2` ― ライブラリ / CLI 洗練・サプライチェーン固定化・テスト網羅強化 (`8 件`)

アクション #3 (`[P2]` ・ `code_quality`) : CLI カタログ一覧コマンドへの `--json` 追加とルール事前キャッシュの範囲限定

- **対象ファイル**: `src/drawlib/_cli/_drawlib.py` , `src/drawlib/_cli/commands/cache.py` , `src/drawlib/_cli/commands/rules.py`
- **関連ルール**: `common/cli_guidelines`
- **実装方針**: `_ensure_overview_rules_cached()` の実行対象を `rules` および `init` コマンドに限定して読取専用 CLI 実行時の暗黙的ディスク書き込みを廃止し、 `list` 系コマンド (`cache list` , `rules list` , `styles` , `colors`) へ機械可読な `--json` 出力オプションを追加します。
- **期待効果**: CI パイプライン向け `clig.dev` スクリプト連携性を高め、読取専用コンテナ環境での安全な CLI 動作を保証します。

アクション #4 (`[P2]` ・ `code_quality`) : プログラマティック・ビルダー層の直接 `print()` をロガー / 進捗コールバックへ置換

- **対象ファイル**: `src/drawlib/_builder/_common/progress.py` , `src/drawlib/_builder/_common/parallel.py` , `src/drawlib/_builder/doc_builder/compiler/jupyter.py`
- **関連ルール**: `common/library_guidelines`
- **実装方針**: プログラマティック・ビルダー群に残る `34` 箇所の直接 `print()` 呼び出しを、 `quiet` 対応の構造化ロガーまたは CLI エントリーポイントから注入される進捗コールバックへ抽象化します。
- **期待効果**: Web バックエンド・Jupyter ノートブック・外部 CLI から `drawlib.tools.build_*()` を呼び出しても標準出力を汚染しない純粋ライブラリ品質を確立します。

アクション #5 (`[P2]` ・ `maintainability`) : 肥大化した Bash スクリプト `docs_src/site/build_html.sh` を型付き Python `dcli` へ移行

- **対象ファイル**: `docs_src/site/build_html.sh` , `tools/dcli/docs/builder.py`
- **関連ルール**: `common/supply_chain`
- **実装方針**: 109 LOC の多段階 HTML サイトビルド処理を `tools/dcli/docs/builder.py` の型付き Python 実装へ移管し、 `docs_src/site/build_html.sh` は `./dcli docs build site` を呼び出す `30` LOC 未満の薄いラッパーのみとします。
- **期待効果**: 唯一のレガシー言語シェルスクリプト検出指摘を解消し、クロスプラットフォーム保守性を統一します。

アクション #6 (`[P2]` ・ `security`) : GitHub Actions の各ステップを不変なコミット SHA へ固定

- **対象ファイル**: `.github/workflows/quality.yml` , `.github/workflows/release.yml` , `.github/workflows/ut_template.yml`
- **関連ルール**: `common/supply_chain`
- **実装方針**: `.github/workflows/*.yml` 内の浮動メジャーバージョンタグ (`@v7` 等) をすべて 40 文字の不変コミット SHA + バージョンコメントへ固定し、既存の `.github/dependabot.yml` (`github-actions` ルール) による週次自動更新で鮮度を維持します。

- **期待効果**: 可変タグ乗っ取りリスクを根絶し、OpenSSF Scorecard の `Pinned-Dependencies` スコアを最大化します。

アクション #7 (`[P2]` ・ `security`) : コンテナ Dockerfile 内のベースイメージ `pip 25.0.1` を更新または除去

- **対象ファイル**: `.pni/dockerfiles/Dockerfile` , `tools/docker/Dockerfile.test_pypi`
- **関連ルール・アドバイザリ**: `common/security_vulnerability` , `CVE-2026-8643` , `CVE-2026-13346`
- **実装方針**: `.pni/dockerfiles/Dockerfile` および `tools/docker/Dockerfile.test_pypi` でシステム `pip` を `>=26.2.0` へ更新 (または未使用のシステム `pip` を除去・最新の `uv` /Python slim ベースダイジェストへ刷新) します。
- **期待効果**: Trivy コンテナイメージ内のアプリケーション CVE を `6` 件から `0` 件へ解消し、Debian bookworm ハーネスの OS CVE 件数も低減します。

アクション #8 (`[P2]` ・ `testing`) : CLI cache コマンドと並列ビルドワーカーの残存分岐カバレッジ不足を解消

- **対象ファイル**: `tests/drawlib/cli/test_cli_css.py` , `tests/drawlib/doc_builder/test_parallel_build.py` , `src/drawlib/__main__.py`
- **関連ルール**: `python/testing`
- **実装方針**: `cache download` / `cache clear` の各種フラグ組合せに対するモックテスト、`parallel.py` 内のワーカープール例外フォールバック経路テスト、および `python -m drawlib` 直接起動の検証を追加します。
- **期待効果**: 非生成モジュールに残る行カバレッジ `65%` 未満の対象を解消します。

アクション #9 (`[P2]` ・ `testing`) : `parallel.py` および CLI cache コマンドの分岐カバレッジを強化

- **対象ファイル**: `src/drawlib/_builder/_common/parallel.py` , `src/drawlib/_cli/commands/cache.py` , `src/drawlib/__main__.py`
- **関連ルール**: `common/runtime_verification`
- **実装方針**: `parallel.py` の `ProcessPoolExecutor` 例外処理・ワーカー数 / `DRAWLIB_*` 環境変数上書き、 `commands/cache.py` のダウンロード・削除エラー分岐、および

`src/drawlib/__main__.py` ディスパッチ経路を対象とした分岐網羅テストを追加します。

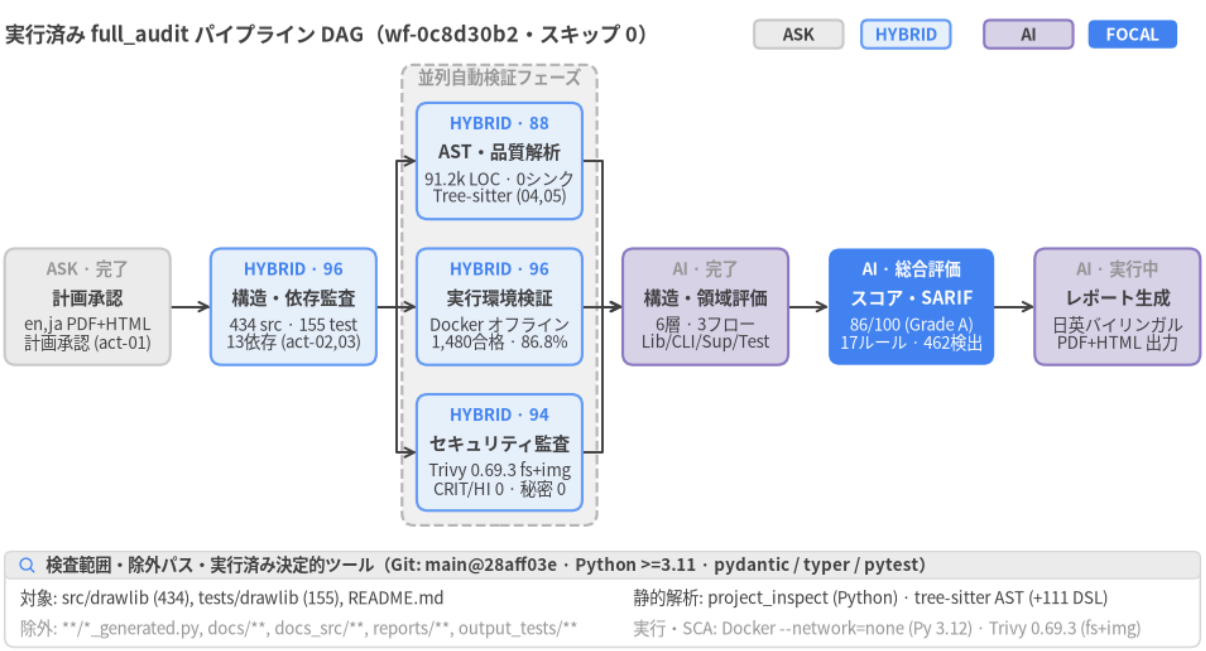
- **期待効果**: 重要な CLI および並列ビルドオーケストレーション経路の分岐カバレッジを `80%` 超へ引き上げます。

アクション #10 (`[P2]` ・ `typing`) : 未パラメータ化 `dict` / `Any` 型注釈の厳格化とビルダー環境変数・ログ管理の集約

- **対象ファイル**: `src/drawlib/_builder/_common/manifest.py`, `src/drawlib/_builder/_common/browser.py`, `src/drawlib/_builder/_common/cache.py`, `src/drawlib/_builder/_common/progress.py`
- **関連ルール**: `python/typing`, `python/modern_stack`
- **実装方針**: `manifest.py` の未パラメータ化 `dict` / `Any` 注釈を `dict[str, str]` 等へ厳格化し、ビルダー各所に分散する `15` 箇所の `DRAWLIB_*` 環境変数参照を型付き設定ヘルパーへ集約するとともに、診断出力を構造化ログ / コールバックへ統一します。
- **期待効果**: 静的型安全性を高め、プログラムからビルダーを呼び出す際の組み込み性と予測可能性を強化します。

4. 監査パイプライン（DAG）・標準適合・検出事項

本章では、以降の技術詳細ディープダイブに先立ち、**Drawlib v0.3.15**（`wf-0c8d30b2`、**86/100・Grade A**）の監査透明性と再現性を確立します。**実行済み 8 ステージのワークフロー DAG**（全 `18` アクション / `12` 論理ステージ中スキップ `0` 件）、**検査範囲と実行ツール**、**7 軸決定的ベースライン基準点**（`±10 pt`）、**8 監査ステップ別スコア**（`88`〜`96`）、**全 17 ルール適合マトリクス**（`0 FAIL · 4 WARN · 12 PASS · 1 N/A`）、**462 件の検出内訳・AI トリアージ記録**・`.pni/pni.toml` **免除設定 5 件**、および**機械可読な監査証跡アーティファクト一覧**を網羅します。



実行済み full_audit パイプライン DAG（wf-0c8d30b2: 18 アクション / 12 論理ステージ・スキップ 0）と検査範囲

4.1 実行済み監査パイプライン（DAG）・検査範囲・ツール

PNI501 v0.2.0 の `full_audit` ワークフロー（ `wf-0c8d30b2` ）は、**全 18 アクション・12 論理ステージをスキップ `0` 件ですべて完走**しました。静的リポジトリ検査、Tree-sitter による AST および付随 DSL 解析、完全オフライン（ `--network=none` ）Docker コンテナ実行テスト、Trivy 0.69.3 によるファイルシステム + ビルド済みイメージのスキャンをすべて完了したうえで、AI 総合評価とベースライン基準スコアリングを実施しています。

順序	論理ステージ（ DAG ノード ）	実行モード・状態	対応アクション	ルール数	検証済みステージ成果
1	`confirm_plan`	`ASK` ・ `完了`	`act-01`	`0`	Python ライブラリ＆CLI 監査計画、既存 `.pni/pni.toml` 免除 5 件、日英（ `en,ja` ）PDF + HTML 出力を承認。
2	`repo_and_deps`	`HYBRID` ・ `完了`	`act-02, act-03`	`5`	ソース `434` / テスト `155` 、SemVer `>=` 実行時依存 `13` 件、OpenSSF ガバナンス（ `LICENSE` , `SECURITY.md` , Dependabot ）確認、ステップスコア `96/100 (S)` 。
3	`ast_and_quality`	`HYBRID` ・ `完了`	`act-04, act-05`	`8`	`91,212` LOC（ `2,089` 関数）+ 付随 DSL `111` ファイルを Tree-sitter 解析。型付与 `100.0%` 、境界違反 `0` 、未免除シンク `0` 、ステップスコア `88/100 (A)` 。
4	`runtime_verification`	`HYBRID` ・ `完了`	`act-06, act-07`	`2`	Docker `--network=none` （Python 3.12）：ビルド `138.75s` 合格、 `1,480/1,485` 合格（失敗 `0` ・環境スキップ `5` ）、行 `86.8%` / 分岐 `75.8%` カバレッジ、スモーク `1.79s` 、スコア `96/100 (S)` 。
5	`security_audit`	`HYBRID` ・ `完了`	`act-08, act-09`	`2`	Trivy `0.69.3` （ `fs` + 解決済み `.pni/dockerfiles/Dockerfile` イメージ）：CRIT/HIGH `0` 件、Drawlib 依存 CVE `0` 件、設定不備 `0` 件、秘密漏洩 `0` 件、スコア `94/100 (S)` 。
6	`arch_and_domains`	`AI` ・ `完了`	`act-10..act-16`	`10`	根拠付きアーキテクチャ記録（ `6` 層・ `3` 実行フロー）+ 4 領域監査：ライブラリ衛生（ `94` ）、CLI ガイドライン（ `93` ）、サプライチェーン（ `95` ）、テスト健全性（ `95` ）。

順序	論理ステージ（DAG ノード）	実行モード・状態	対応アクション	ルール数	検証済みステージ成果
7	output_score	AI・総合評価	act-17	0	基準点アンカー 7 軸スコア & SARIF 生成：総合 86/100 (Grade A)、全 17 ルール（0 FAIL, 4 WARN, 12 PASS, 1 N/A）、462 検出、10 件の AI 再分類。
8	create_report	AI・実行中	act-18	0	日英バイリンガル（en, ja）Drawlib A4 PDF + HTML レポートの生成と視覚グリッド検証。

検査範囲・除外パス・実行済みツール

- **検査対象パス**: ルート `.` ・ ソース `['src/drawlib']` ・ テスト `['tests/drawlib']` ・ ドキュメント `['README.md']`
- **除外パス**: `['**/*_generated.py', 'docs/**', 'docs_src/**', 'reports/**', 'output_tests/**']`
- **言語・フレームワーク・リビジョン**: 主要言語 `python`（`>=3.11`） ・ フレームワーク `['pydantic', 'typer', 'pytest']` ・ Git リビジョン `main@28aff03e`
- **実行済み決定的ツール**:
  1. `project_inspect (python)` — PEP 621 メタデータ、依存衛生、CI ワークフロー権限、ガバナンス整備、Git 履歴の検証。
  2. `tree-sitter AST (python + 付随 DSL 111 ファイル)` — 型注釈、docstring、SIG 複雑度、インポート循環、付随 DSL SAST（`dockerfile=3`, `css=19`, `yaml=9`, `json=77`, `toml=1`, `xml=2`）。
  3. `container runtime harness (docker: .pni/dockerfiles/Dockerfile, Python 3.12, --network=none)` — クリーンルームビルド、`pytest` + `pytest-cov`、CLI/ライブラリ疎通スモーク検証（非 root `USER pni`, `UID 10001`）。
  4. `Trivy 0.69.3 (docker: fs + resolved image)` — SCA 脆弱性スキャン、Dockerfile IaC 検査、秘密情報検出。

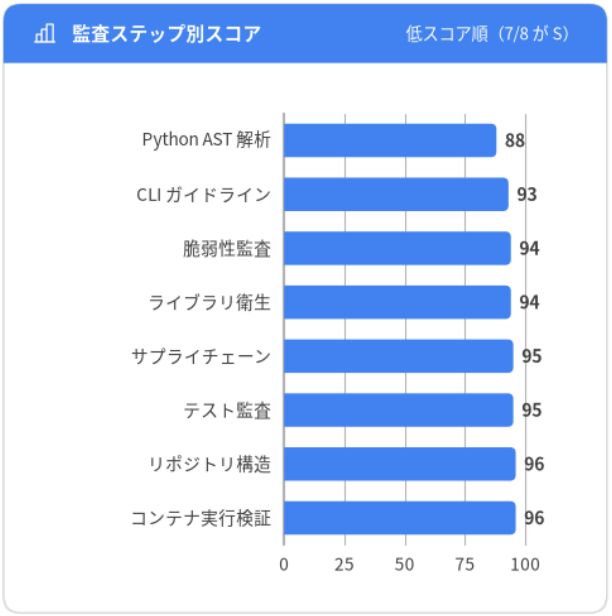
4.2 決定的ベースライン基準点（`±10 pt`）と監査ステップ別スコア

PNI501 v0.2.0 では、定性的な AI スコア調整幅をすべて決定的アナライザのベースラインから `±10` **点以内**に厳格に制限し、実測データとの乖離を防止しています。総合スコアは決定的ベースライン `82` から `+4` 点調整された `86/100 (Grade A)` となり、実行済み **8 監査ステップ中 7 ステップが Grade S（`>= 90`）**を達成しています。

7 軸ベースライン vs. 最終スコア（±10 pt 基準）

健全性評価軸	基準	AI Δ	最終
アーキテクチャ	70	+10	80
コード品質	40	+10	50
型安全性	80	+10	90
テスト品質	100	-4	96
依存関係管理	100	-4	96
セキュリティ	85	+9	94
ドキュメント	100	-2	98
総合スコア	**82**	**+4**	**86 (A)**

すべての軸で AI 調整幅は決定的ベースラインの ±10 点以内に制限



7 軸決定的ベースライン vs. 最終スコア（±10 pt 基準）と実行済み監査ステップ別スコア（低スコア順）

7 軸決定的ベースライン vs. 最終スコア（`±10 pt` アンカー比較表）

健全性評価軸	決定的ベースライン	AI Δ（`±10 pt`）	最終スコア	基準点に基づく評価根拠
アーキテクチャ	`70`	`+10`	`80`	PEP 562 公開ファサード背後の `434` モジュールで層境界違反 `0` 件。一方、静的循環 `8` 件（`parallel.py` 遅延ワーカー `5` 件、`html<->jupyter` `1` 件、`_diagrams` `_edge<->_junction` `2` 件）が残るため `80` 点。

健全性評価軸	決定的ベースライン	AI Δ (±10 pt)	最終スコア	基準点に基づく評価根拠
コード品質	40	+10	50	厳格な SIG 閾値により 442 関数 (>50 LOC または認知複雑度 >15) と非 CLI print() 34 件を検出。docstring 99.3% ・インポート衛生 99.3% ・バグリスク 0 ・Ruff+Ty クリーン・キーワード専用 API 免除 353 件を評価し +10 加点。
型安全性	80	+10	90	全 2,089 関数で 100.0% 型付与、無指定 ignore 0 件、 py.typed 同梱、Ty + Pydantic v2 適用。 Any / 未パラメータ化 dict 200 件および未型付け定数属性 162 件により 90 点。
テスト品質	100	-4	96	--network=none Docker 実行で 1,480/1,485 合格 (失敗 0 ・環境スキップ 5)、AST モジュール網羅率 100.0% 、行 86.8% / 分岐 75.8% 、 >=97% ピクセル回帰検証を実装。 parallel.py (50.6%) ・ commands/cache.py (30.0%) ・ __main__.py (0.0%) の分岐ギャップで -4 。
依存関係管理	100	-4	96	PEP 621 + uv_build 、SemVer >= 下限指定の実行時依存 13 件 (未固定 0 ・Drawlib CVE 0)、Dependabot 有効、 python/lockfile_committed 免除済み。 @v7 浮動タグと 109 行の docs_src/site/build_html.sh で -4 。
セキュリティ	85	+9	94	Trivy 0.69.3 (fs+img) で CRIT/HIGH 0 件、Drawlib 依存 CVE 0 件、設定不備 0 件 (DS-0026 3 件免除)、秘密情報 0 件、未免除シンク 0 件 (CWE-94 8 件免除)、SHA-256 + zip-slip 検証、 SECURITY.md 整備。ベースイメージ pip@25.0.1 の 6 CVE のみで 94 点。
ドキュメント	100	-2	98	docstring 99.3% (空 0)、日英 Diataxis 6 ポータル、組込み _rules/ 、CI 強制の 100% 日英 API/CLI 網羅テスト。SmartArt の未文書化 @property セッター 12 件で -2 。
総合スコア	82	+4	86 (A)	テスト・セキュリティ・依存・型・ドキュメントの極めて高い完成度に対し、ビルダー / オーケストレータの SIG 複雑度 (code_quality=50) が主たる改善余地。

実行済み監査ステップ別スコア（低スコア順）

順位	監査ステップ ID（`action_id`）	スコア・評価	Pass / Warn / Fail	対象ルール	主要診断サマリ
1	`python_ast_audit`（`act-05`）	88/100 (A)	`4 / 3 / 0`	`common/architecture`, `common/maintainability`, `common/security`, `common/security_vulnerability`, `python/code_quality`, `python/typing`, `python/modern_stack`, `python/frameworks/fastapi`	`434` モジュール（`91,212` LOC・`2,089` 関数）：型付与 `100.0%`、docstring `99.3%`、インポート衛生 `99.3%`、境界違反 `0`、未免除シンク `0`。静的循環 `8` 件（`parallel.py` 遅延 `5` 件）、SIG 複雑関数 `442` 件、`os.path` PTH ヒント `1,251` 件。
2	`cli_guidelines_audit`（`act-14`）	93/100 (S)	`2 / 0 / 0`	`common/cli_guidelines`, `common/documentation`	Typer + Rich CLI（`9` サブコマンド群）、`-h/--help`、`-v/--version`、排他詳細度フラグ、`no_args_is_help=True`、`1.79s` スモーク、日英 100% CLI リファレンス。`--json` 出力と `_ensure_overview_rules_cached()` の範囲限定が軽微な改善点。
3	`security_vulnerability_audit`（`act-09`）	94/100 (S)	`1 / 1 / 0`	`common/security_vulnerability`, `common/supply_chain`	Trivy `0.69.3`（fs+img）：CRIT/HIGH `0` 件、Drawlib 依存 CVE `0` 件、設定不備 `0` 件（`DS-0026` 3 件免除）、秘密情報 `0` 件、未免除 SAST シンク `0` 件（`CWE-94` 8 件免除）。ベースイメージ `pip@25.0.1` のみに MED `5` + LOW `1` 件。
4	`library_hygiene_audit`（`act-13`）	94/100 (S)	`2 / 1 / 0`	`common/library_guidelines`, `common/documentation`, `common/maintainability`	PEP 562 遅延 `__getattr__`、公開 `36` モジュールの明示的 `__all__`、`py.typed`、SemVer `>=` 境界、インポート時ネットワーク I/O ゼロ。プログラム呼出し経路 `_builder/` の直接 `print()` `34` 件のみ軽微減点。

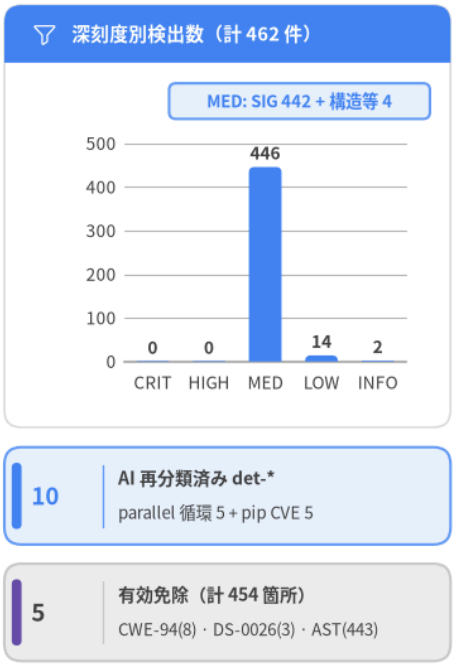
順位	監査ステップ ID (action_id)	スコア・評価	Pass / Warn / Fail	対象ルール	主要診断サマリ
5	supply_chain_audit (act-15)	95/100 (S)	3 / 0 / 0	common/supply_chain , common/security , common/git_workflow	LICENSE 、 SECURITY.md 、 .github/dependabot.yml (uv + github-actions) 、最小権限 CI ワークフロー 5 本、秘密情報 0 件、SHA-256 + zip-slip 検証。 @v7 Actions のコミット SHA 固定化が推奨 polish 項目。
6	python_test_audit (act-16)	95/100 (S)	1 / 0 / 0	python/testing	155 テストファイル・ 1,485 テスト、AST モジュール網羅率 100.0% 、行 86.8% / 分岐 75.8% 、autouse キャンバス初期化、 >=97% ピクセル回帰フック、Jupyter ドライランテスト。 parallel.py ・ commands/cache.py ・ __main__.py に局所的分岐ギャップ。
7	repo_structure_audit (act-03)	96/100 (S)	5 / 0 / 0	common/supply_chain , common/documentation , common/git_workflow , common/security , python/dependencies	OpenSSF ガバナンス完備、PEP 621 / uv_build 、SemVer >= 依存 13 件 (未固定 0) 、日英 Diataxis 6 ポータル、規約準拠の線形トランク履歴。軽微事項は docs_src/site/build_html.sh (109 LOC) のみ。
8	container_runtime_audit (act-07)	96/100 (S)	2 / 0 / 0	common/runtime_verification , common/production_readiness	完全オフライン --network=none Python 3.12 コンテナ：ビルド合格 (138.75s) 、pytest 1,480/1,485 合格 (失敗 0 ・環境スキップ 5 、 722.24s) 、行 86.8% / 分岐 75.8% 、非 root USER pni (UID 10001) 、スモーク合格 (1.79s) 。

4.3 全 17 ルール適合マトリクス（`0 FAIL · 4 WARN · 12 PASS · 1 N/A`）

全 **17 評価ルール**において、Drawlib v0.3.15 は `0 FAIL`、`4 WARN`、`12 PASS`、`1 NOT_APPLICABLE` を達成しています（適用対象 `16` ルール中の厳格合格率 `75.0%`、重大不合格ゼロ率 `100.0%`）。

全 17 ルール適合マトリクス（0 FAIL · 4 WARN · 12 PASS · 1 N/A）

判定	評価ルール ID	主要診断ゲート
WARN	common/architecture	境界違反 0 · 静的循環 8 件
WARN	common/maintainability	SIG 442 関数 · 非 CLI print 34
WARN	common/security_vulnerability	CRIT/HI 0 · ベース pip 6 CVE
WARN	python/modern_stack	Py>=3.11 · os.path PTH 1,251
PASS	common/cli_guidelines	9 コマンド群 · -h/-v · 1.79s
PASS	common/documentation	doc 99.3% · 日英 6 ポータル
PASS	common/git_workflow	線形 main · v0.3.15 · 秘密 0
PASS	common/library_guidelines	PEP 562 · 36 __all__ · py.typed
PASS	common/production_readiness	UID 10001 · オフライン · 回帰 0
PASS	common/runtime_verification	1,480/1,485 合格 · 行 86.8%
PASS	common/security	SECURITY.md · SHA-256 · シンク 0
PASS	common/supply_chain	Dependabot · 最小権限 CI 5 本
PASS	python/code_quality	99.3% 品質 · Ruff+Ty クリーン
PASS	python/dependencies	PEP 621 · >= 13 依存 · uv_build
PASS	python/testing	155 ファイル · AST 100% · >=97%
PASS	python/typing	100.0% 型付与 (2,089) · 無視 0
N/A	python/frameworks/fastapi	FastAPI 未使用 (typer/pydantic)



全 17 ルール適合マトリクス（0 FAIL · 4 WARN · 12 PASS · 1 N/A）、深刻度別検出数（計 462 件）と AI トリアージ / 免除サマリ

#	判定	ルール ID	監査元	要約評価	定量・診断エビデンス
1	`WARN`	`common/architecture`	`python_ast_audit`	層境界違反ゼロ。ビルダー・図表ヘルパに静的循環 8 件が残存。	`L1 -> L4 -> Domains -> PEP 562 Facades` で境界違反 `0` 件、`TYPE_CHECKING` 循環 `40` 件（実行時影響なし）、静的循環 `8` 件（`parallel.py` 遅延ワーカー `5` 件、`html<->jupyter` `1` 件、`_diagrams` `_edge<->_junction` `2` 件）。
2	`WARN`	`common/maintainability`	`python_ast_audit`	公開ファサードは高凝集・グローバル状態ゼロ。ビルダー部に SIG 複雑度が集中。	SIG 基準（`<=50` LOC または認知複雑度 `<=15`）超過が `442` 関数（`render_chapter=178`, `parse_block_info=157`, `_compile_jupyter_locales=156`, `_execute_block_render_job=68`, `format_project_manifest_toml=48`）。キーワード専用 API `353` 件は免除済み、非 CLI `print()` `34` 件。

#	判定	ルール ID	監査元	要約評価	定量・診断エビデンス
3	WARN	common/security_vulnerability	security_vulnerability_audit	CRIT/HIGH ゼロ・Drawlib 依存 CVE ゼロ。ベースイメージ pip の 6 CVE のみ残存。	Trivy 0.69.3（fs+img）：CRIT/HIGH 0 件、Drawlib 実行時/開発依存 CVE 0 件、設定不備 0 件（DS-0026 3 件免除）、秘密情報 0 件、未免除シンク 0 件（CWE-94 8 件免除）、付随 DSL 111 件クリーン。MED 5 + LOW 1 はすべてベースイメージ pip@25.0.1（26.2.0 で修正）。
4	WARN	python/modern_stack	python_ast_audit	Python 3.11+ モダン構成（Pydantic v2, Typer, uv, Ruff, Ty）。os.path ヒント残存。	requires-python >=3.11、PEP 562 遅延ファサード、未宣言依存 0 件。os.path PTH 近代化ヒント 1,251 件、非 CLI print() 34 件、散在 os.environ 参照 15 件が残存。
5	PASS	common/cli_guidelines	cli_guidelines_audit	9 サブコマンド群で高い clig.dev 準拠性と高速起動を実現。	-h/--help、即時 -v/--version、排他詳細度フラグ（--quiet/--verbose/--debug/--developer）、no_args_is_help=True、9 サブコマンド群、1.79s スモーク検証。
6	PASS	common/documentation	repo_structure_audit	Standard Readme・Diataxis 基準を超過達成し、日英完全性を CI で自動保証。	docstring 99.3%（空 0）、README.md（222 行・14 見出し）、日英 Diataxis 6 ポータル（site, api, architecture, quickstart, about-drawlib, jupyter）、日英 100% API/CLI 網羅自動テスト。
7	PASS	common/git_workflow	repo_structure_audit	Conventional Commits に準拠したクリーンなトランク履歴と秘密情報ゼロを維持。	線形 main トランク履歴、Conventional Commits、注釈付きリリースタグ（v0.3.14, v0.3.15）、.gitignore による .env/キャッシュ除外、秘密ファイル 0 件。
8	PASS	common/library_guidelines	library_hygiene_audit	PEP 562 遅延ロード、明示的 __all__ 契約、SemVer 衛生が模範的。	PEP 562 遅延 __getattr__、公開 36 モジュールの明示的 __all__、PEP 561 py.typed、全 13 実行時依存への SemVer >= 境界、インポート時ネットワーク I/O ゼロ。
9	PASS	common/production_readiness	container_runtime_audit	非 root・オフライン実行耐性、高速起動、リグレッションゼロを実証。	非 root USER pni（UID 10001）、事前キャッシュ後の --network=none 完全オフライン実行、1.79s CLI 起動、実行 1,480 テストで回帰 0 件、I/O タイムアウト未指定 0 件。
10	PASS	common/runtime_verification	container_runtime_audit	ハーメティック Docker ビルド・オフライン pytest・高カバレッジ・スモークすべて合格。	ビルド合格（138.75s）、オフライン pytest 1,480/1,485 合格（失敗 0・環境スキップ 5）、行 86.8%（40,102/46,204）/ 分岐 75.8% カバレッジ、スモーク合格（1.79s）。
11	PASS	common/security	repo_structure_audit	信頼モデル明文化、署名/ハッシュ検証付きダウンロード、境界バリデーションを完備。	Illustration-as-Code 信頼モデルを明記した SECURITY.md、秘密情報 0 件、未免除シンク 0 件、リリースアセットの SHA-256 + zip-slip 検証、Pydantic v2 バリデーション。
12	PASS	common/supply_chain	repo_structure_audit	OpenSSF ガバナンス整備、最小権限 CI、クリーンな依存ツリーを達成。	LICENSE、SECURITY.md、.github/dependabot.yml、最小権限 CI ワークフロー 5 本、未固定実行時依存 0 件、Drawlib CVE 0 件。

#	判定	ルール ID	監査元	要約評価	定量・診断エビデンス
13	`PASS`	`python/code_quality`	`python_ast_audit`	極めて高い docstring・インポート衛生とバグリスクゼロ、Ruff/Ty クリーンを達成。	docstring `99.3%`、インポート衛生 `99.3%`、空 docstring `0` 件、バグリスク構文 `0` 件、共有ミュータブルグローバル `0` 件、Ruff + Ty クリーン。
14	`PASS`	`python/dependencies`	`repo_structure_audit`	PEP 621 / `uv_build` による最新パッケージ構成と開発・監査グループの分離。	PEP 621 `pyproject.toml`、`uv_build`、SemVer `>=` 実行時依存 `13` 件、`[dependency-groups]` 分離、`python/lockfile_committed` 明示免除。
15	`PASS`	`python/testing`	`python_test_audit`	単体・ピクセル回帰・ドキュメント網羅・ノートブック実行検証を統合した高品質スイート。	`155` テストファイル・`1,485` テスト、AST モジュール網羅率 `100.0%`、行 `86.8%` / 分岐 `75.8%`、ピクセル一致検証フック（`>=97%`）、Jupyter ドライランテスト。
16	`PASS`	`python/typing`	`python_ast_audit`	全関数の完全型注釈と Ty 静的型検査クリーンを達成。	関数型付与率 `100.0%`（`2,089/2,089`）、無指定 `type: ignore` `0` 件、`py.typed` マーカー同梱、Ty クリーン。
17	`N/A`	`python/frameworks/fastapi`	`python_ast_audit`	Drawlib のライブラリ / CLI プロファイルには FastAPI ルール非該当。	検出フレームワークは `['pydantic', 'typer', 'pytest']` であり、FastAPI の宣言・インポートは一切なし。

4.4 統合検出事項の内訳・AI トリアージ記録・有効な免除設定

全検出件数は `462` **件**（決定的アナライザ `449` 件 + AI 定性評価 `13` 件）で、深刻度別には `CRITICAL 0`、`HIGH 0`、`MEDIUM 446`、`LOW 14`、`INFO 2` です。442 件の SIG 複雑度検出を羅列する代わりに、**(A) SIG 複雑度ホットスポットクラスタ（`442 MEDIUM`）**、**(B) 非 SIG 実行可能検出事項（`20` 件：`4 MEDIUM`, `14 LOW`, `2 INFO`）**、**(C) AI トリアージ記録（`det-*` 再分類 `10` 件 + 誤検知メモ `1` 件）**、および **(D) 有効な `.pni/pni.toml` 設計免除（`5` 件・計 `454` 箇所）** に体系化して整理します。

A. SIG ユニット規模・認知複雑度ホットスポットクラスタ（決定的 `MEDIUM` `442` 件）

サブパッケージクラスタ	件数	主なホットスポットファイルと最大複雑度 / 行数	主なリファクタリング方針
`src/drawlib/_builder`	`132`	`compiler/html.py`（`16` 件、`1,989` LOC）、`image_builder.py`（`15`）、`manifest.py`（`8`；`format_project_manifest_toml` 認知=`48`）、`parallel.py`（`7`；`_execute_block_render_job` 認知=`68`）、`exporter_docx.py`（`7`；`render_chapter` 認知=`178`）、`compiler/jupyter.py`（`_compile_jupyter_locales` 認知=`156`）、`processor/options.py`（`parse_block_info` 認知=`157`）	プロジェクトモード別（site/doc/catalog）の HTML/Jupyter コンパイラ分割と、DOCX / フェンス属性のトークン別ハンドラテーブル化。
`src/drawlib/_core`	`76`	`l4_canvas/_shapes/_container_builders.py`（`9`）、`_symbol_builders.py`（`9`）	分岐の多い図形ファクトリから共通幾何パス生成ヘルパを抽出。
`src/drawlib/_smartarts`	`61`	`_table.py`（`render` 認知=`74`）、`_tree.py`、`_mindmap.py`、`_geomap.py`	レイアウト座標計算と Matplotlib パッチ描画フェーズを分離。
`src/drawlib/_charts`	`49`	`_common/_legend.py`、`bar_chart/_renderer.py`、`_common/_axis.py`	軸目盛り算出と系列描画ロジックを小型ヘルパへ分割。
`src/drawlib/_diagrams`	`36`	`architecture/`, `flow/`, `sequence/`, `er/` のレイアウト・エッジルータ	共通エッジルーティングおよびノードアンカー計算を抽出。
`src/drawlib/_slide`	`24`	`_slide/compiler.py`（`11`）、`_pptx_writer.py`、`_pptx_dom.py`	スライド DOM 正規化と OpenXML 図形シリアライズを分割。
`src/drawlib/_cli` & `_graph`	`42`	`_cli`（`21`：`commands/build.py`, `commands/show.py`）、`_graph`（`21`：Sugiyama / グリッドソルバ）	CLI 引数正規化処理およびグラフ交差削減フェーズを分離。
その他（`_preset_*`, `_templates`, `_anim`, `_http_server`）	`22`	`_preset_styles`（`9`）、`_templates`（`8`）、`_preset_colors`（`3`）、`_anim`（`1`）、`_http_server`（`1`）	テンプレートスキャフォールド生成（`_agent.py`, `_project.py`）を分割。
SIG 複雑度検出 合計	**`442`**	**すべて `MEDIUM`（`common/maintainability`）、機能不具合ゼロ**	P1 優先施策として `_builder/` 最上位ホットスポットから段階分割。

B. 非 SIG 実行可能検出事項（20 件：4 MEDIUM · 14 LOW · 2 INFO）

#	深刻度	対象箇所（file:line）	ルールタグ	検出元	診断エビデンスと推奨アクション
1	MEDIUM	src/drawlib/_builder/_common/parallel.py:383	common/architecture	AI	_builder._common -> parallel -> processor/image_builder/_slide.compiler -> _common を経由する遅延静的循環が 5 件。parallel.py を _common/ 上位へ移動またはコールバック注入へ変更。
2	MEDIUM	src/drawlib/_builder/doc_builder/compiler/html.py	common/architecture	AI	compiler/html.py と compiler/jupyter.py 間の静的循環（resolve_favicon_href, resolve_logo_hrefs）。共通アセットヘルパを compiler/base.py へ移動。
3	MEDIUM	_builder._common -> parallel -> processor -> _common	common/architecture	決定的（再分類）	決定的循環検出を high -> medium へ再分類（関数内遅延インポートのためロード時 ImportError なし。_diagrams の _edge<->_junction 2 件と併せ P1 で解消）。
4	MEDIUM	src/drawlib/_builder/doc_builder/compiler/html.py	common/maintainability	AI	1,989 LOC のコンパイラモノリスおよび主要ビルダー高複雑度群（format_project_manifest_toml 認知=48, _execute_block_render_job 認知=68）。モード別に分割。
5–10	LOW (×6)	コンテナベースイメージ（pip@25.0.1）	CVE-2026-8643, 13346, 6357, 3219, 1703, CVE-2025-8869	決定的（5 件再分類）	6 CVE はすべて上流 ghcr.io/astral-sh/uv:python3.12-bookworm-slim のシステム pip@25.0.1 由来（Drawlib 依存ツリーは 0 件）。pip >= 26.2.0 更新または未使用 pip 除去を実施。
11	LOW	.pni/dockerfiles/Dockerfile	CVE-2026-8643	AI	解決済みコンテナイメージ上のベース pip@25.0.1 集約所見。Dockerfile のベースダイジェスト更新または pip 更新・削除で解消。
12	LOW	docs_src/site/build_html.sh	common/supply_chain	AI	100 行超の Bash スクリプト（109 LOC）。複数ステップのサイトビルド処理を tools/dcli/docs/builder.py（Python + Typer）へ統合。
13	LOW	src/drawlib/_builder/_common/manifest.py:72	python/typing	AI	未パラメータ化 dict 注釈（_resolve_localized_value, _parse_localized_str_or_map, _fmt_loc）と Any /弱型 200 件。dict[str, str] 等へ厳密化。
14	LOW	src/drawlib/_builder/_common/parallel.py	common/runtime_verification	AI	parallel.py の実測カバレッジが行 61.1% / 分岐 50.6%（commands/cache.py も 61.2% / 30.0%）。ワーカー例外フォールバックと環境変数解析テストを追加。
15	LOW	src/drawlib/_cli/commands/cache.py	python/testing	AI	commands/cache.py が行 61.2% / 分岐 30.0%、__main__.py（6 LOC）が 0.0%。キャッシュフラグのモックテストと python -m drawlib --version 疎通テストを追加。

#	深刻度	対象箇所 (`file:line`)	ルールタグ	検出元	診断エビデンスと推奨アクション
16	`LOW`	`src/drawlib/_builder/_common/progress.py:138`	`python/modern_stack`	AI	非 CLI `print()` `34` 件、散在 `os.environ` `15` 件、`os.path` PTH ヒント `1,251` 件。ロガー / コールバック経由化と `DRAWLIB_*` 環境変数読込の一元化を推奨。
17	`LOW`	`src/drawlib/_builder/doc_builder/compiler/jupyter.py:250`	`common/library_guidelines`	AI	`drawlib.tools` からのプログラム呼出し経路で `print()` が標準出力へ直接出力。`quiet` 対応ロガーまたは任意の `progress_callback` 注入へ移行。
18	`LOW`	`src/drawlib/_cli/_drawlib.py:129`	`common/cli_guidelines`	AI	`main_callback()` が `cache` 以外の全コマンドで `_ensure_overview_rules_cached()` を呼び出し、読取専用コマンドでも書込みが発生。`rules` / `init` 限定へスコープ縮小。
19	`INFO`	`src/drawlib/_cli/commands/cache.py`	`common/cli_guidelines`	AI	検査コマンド (`cache/styles/colors/rules list`) が Rich テーブル出力のみ。スクリプト / エージェント連携向けに `--json` オプション追加を推奨。
20	`INFO`	`.github/workflows/quality.yml:20`	`common/supply_chain`	AI	浮動メジャーバージョンタグ (`actions/checkout@v7` , `astral-sh/setup-uv@v7` , `actions/setup-python@v7`) 。40 文字の完全コミット SHA 固定化 (Dependabot で自動追随) を推奨。

C. AI 誤検知・深刻度トリアージ記録 (`det-*` 再分類 `10` 件 + インポート別名誤検知 `1` 件)

監査ステップ	決定的検出 ID	トリアージ判定	根拠付き技術判断
`act-05` (`python_ast_audit`)	`det-c049c75c` , `det-6fd1e2e0` , `det-cb51c1e4` , `det-ff1e113a` , `det-fe406c35` (`5` 件)	`high -> medium`	5 件はいずれも `_builder/_common/parallel.py` 内の関数内遅延ワーカーインポート経由であり、トップレベル読込時の `ImportError` を引き起こさないため `medium` へ再分類 (`parallel.py` の上位層抽出は P1 で追跡) 。
`act-09` (`security_vulnerability_audit`)	`det-13c60704` , `det-f749f183` , `det-88839930` , `det-dac9a878` , `det-dbf87858` (`5` 件)	`medium -> low`	`CVE-2026-8643` , `CVE-2026-13346` , `CVE-2026-6357` , `CVE-2025-8869` , `CVE-2026-3219` は上流 `ghcr.io/astral-sh/uv:python3.12-bookworm-slim` 同梱のシステム `pip@25.0.1` のみに起因し、`uv` 管理下の Drawlib 依存パッケージには含まれないため `low` へ再分類。
`act-05` *(無害な誤検知)*	`unused_dependencies=['python_docx']`	**無害な誤検知**	PyPI パッケージ名 (`python-docx`) とインポート名 (`import docx`) の差異による誤検知。`_docx_xml.py` および `exporter_docx.py` で実行時に必須利用されている。

D. 有効な設計免除設定（`.pni/pni.toml` `[[opt_out]]`、`5` 件・計 `454` 箇所）

免除 ID（`id`）	一致箇所数	適用スコープ（`scope`）	明文化されたエンジニアリング根拠（`reason`）
`CWE-94`	`8`	`src/drawlib/_builder/**`, `src/drawlib/_core/l4_canvas/_image_code.py`	Drawlib は「Illustration as Code」設計上、ユーザ記述の Python 作図ブロック・`styles.py`・`utils.py` を意図的に実行する。非信頼 Markdown は `--network=none` 非 root コンテナで隔離実行する前提。
`DS-0026`	`3`	`tools/docker/Dockerfile.*`, `.pni/dockerfiles/**`	バッチ CLI およびテスト用コンテナはコマンド終了と同時に終了するため、常駐デーモン向けの `HEALTHCHECK` 命令は非該当。
`python/lockfile_committed`	`0`	`repo-wide`	Python `3.11–3.14` を広くサポートする純 Python ライブラリとして、CI でのクロスバージョン依存解決を優先し `uv.lock` を Git 除外（SCA 監査時はローカル生成）。
`ast/private_submodule_import`	`90`	`src/drawlib/**`	PEP 562 公開ファサード（`drawlib/*.py`）と階層化内部パッケージが下向き DAG カプセル化を維持するため、先頭アンダースコア `_*` サブモジュールを意図的に再エクスポート。
`ast/excessive_parameters`	`353`	`src/drawlib/**`	宣言的スタイル・図表・チャート・レイアウト API は単一呼出しでの記述性と IDE 補完のため、Pydantic `@validate_call` 検証付きキーワード専用（`*`）引数を意図的に公開。

4.5 機械可読な監査証跡アーティファクト

本レポートのすべての数値・SARIF 位置情報・ワークフロー DAG 状態・根拠付きアーキテクチャグラフは、構造化アーティファクトとして永続化されています（生の JSON ペイロードを本文へ貼り付けるのではなく、再現・自動連携用の参照索引として管理します）。

アーティファクトパス	スキーマ / 形式	主な利用先と用途
`reports/full_audit_score.json`	`ScoreResult` v2.0 (JSON)	7 軸スコア・ベースライン・評価根拠・ルール適合・全検出事項・トリアージ記録・免除設定の完全構造化データ。
`reports/full_audit_summary.json`	`ScoreSummary` (JSON)	CI ゲート / 品質ポータルバッジ向けの軽量サマリ（総合スコア、グレード、深刻度件数、合格率）。
`reports/full_audit_findings.sarif`	`SARIF 2.1.0`	GitHub Code Scanning、VS Code SARIF Viewer、および IDE インライン診断連携用フォーマット。
`reports/full_audit_src/_facts/plan.json`	事前計画（JSON）	承認済み監査スコープ、除外パス、検出言語・フレームワーク、検証済みワークフロー DAG 定義。

アーティファクトパス	スキーマ / 形式	主な利用先と用途
`reports/full_audit_src/_facts/facts.json`	決定的ファクト集（JSON）	`project_inspect`、Tree-sitter AST + DSL 解析、Docker 実行ハーネス、Trivy `0.69.3` の生出力バンドル。
`reports/full_audit_src/_facts/architecture.json`	`ArchitectureRecord` (JSON)	根拠付き 6 層アーキテクチャ定義、モジュール台帳、実行フロー、およびホットスポットリスク値。

5. 詳細分析：コード品質・テスト・コンテナ実行検証

Drawlib v0.3.15 は、`434` のソースモジュール（`91,212` LOC・`317` クラス）にわたり **100.0%** の静的関数型アノテーション網羅率（`2,089 / 2,089` 関数）および **99.3%** の Google スタイル Docstring 規律を維持しつつ、行カバレッジ **86.8%**（分岐カバレッジ **75.8%**）のテストスイートを備えています。さらに `.pni/dockerfiles/Dockerfile` へヘッドレス Chromium（`libatomic1` + `chromium-headless-shell`）および全 `83` 個のキャッシュ済みアセットパッケージを事前組込したことで、完全オフライン `--network=none` コンテナ実行検証（`container_runtime_audit`：**96/100・Grade S**）において **実行テスト 100.0% 合格（`1,480 passed`・`0 failed`・`0 errors`・クリーン環境スキップ `5` 件）**を達成しました。

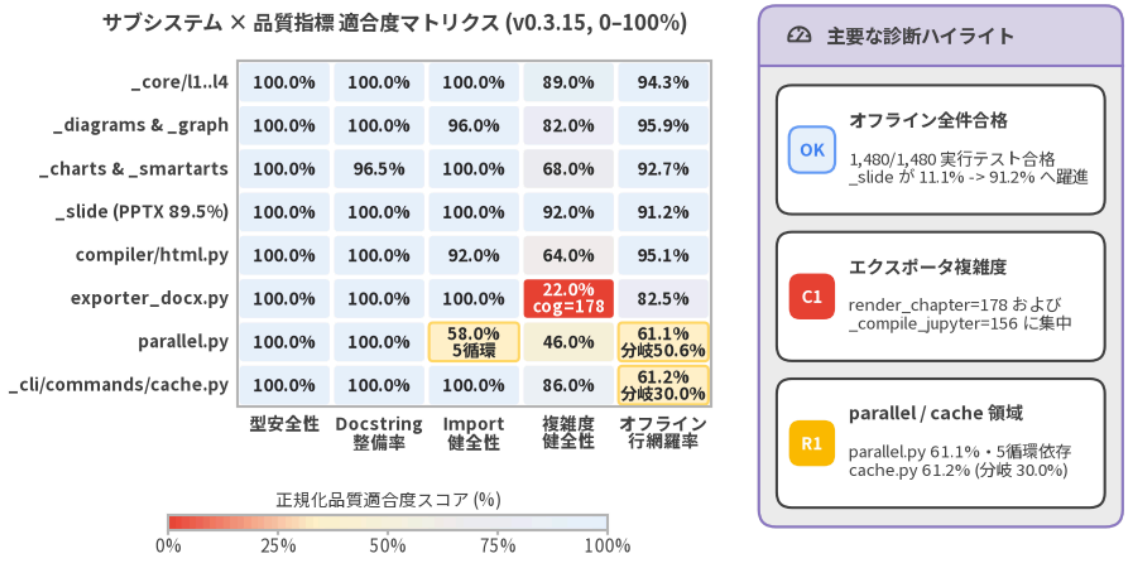


図 5.1 — サブシステム × 品質指標 適合度マトリクス（Drawlib v0.3.15）。均一な型安全性と _slide の 91.2% オフライン網羅率（PPTX 89.5%）、局所的なエクスポータ複雑度、parallel.py / cache.py の改善対象を可視化。

5.1 静的 AST コード品質・型安全性・保守性

PNI の決定論的 Python AST 監査（`python_ast_audit`：**88/100・Grade A**）は、自動生成アイコンラッパー（`**/*_generated.py`）を除く全 `434` ソースモジュール（`91,212` LOC・`317` クラス・`2,089` 関数 / メソッド）を精査しました。コア層（`_core/l1_core` 〜 `_core/l4_canvas`）およびドメインパッケージ（`_charts`・`_diagrams`・`_smartarts`・`_graph`・`_icons`・`_slide`・`_anim`）全体で極めて高い静的規律を維持している一方、多形式ドキュメント / Jupyter エクスポータの拡張に伴い、保守性改善対象がビルダー群へ局所化しています。

静的品質・保守性メトリクス	実測値	ルール基準 / 目標	判定・工学的解釈
関数型アノテーション網羅率	`100.0%`（`2,089/2,089`）	`>= 95.0%`（`python/typing`）	**PASS** — `src/drawlib/py.typed`（PEP 561）、無制限 `# type: ignore` `0` 件、`ty` エラー `0` 件、実行時 Pydantic `@validate_call` 検証を完備。軽微な整理対象として `Any` /未パラメータ化 `dict` `200` 件および定数クラス属性未注釈 `162` 件あり。
Docstring 網羅率	`99.3%`	`>= 90.0%`（`python/code_quality`）	**PASS** — 空 Docstring `0` 件。欠落は `_mindmap.py`・`_tree.py`・`_bulletpoints.py` の `@property` セッター `12` 箇所のみ。
Import 健全性・依存整合性	`99.3%`	`>= 95.0%`（`python/modern_stack`）	**PASS** — ワイルドカード import `0` 件、未宣言依存 `0` 件、Python `>=3.11` 構文不整合 `0` 件。`unused_dependencies=['python_docx']` は `import docx` とのパッケージ名差異による無害なエイリアス。
静的モジュールテスト網羅率	`100.0%`（`155` ファイル）	`>= 90.0%`（`python/testing`）	**PASS** — `1,341` 個の AST テスト関数（pytest 収集 `1,485` 件）が `src/drawlib/` をミラー構成で網羅し、アサーション欠如テストは `0` 件。
グローバル状態・バグリスク記法	`0` / `0`	`0`（`common/maintainability`）	**PASS** — キャンバス状態は `_core/l4_canvas/_base.py` にカプセル化され、共有可変グローバル状態 `0` 件・バグリスク記法 `0` 件。
静的循環インポート	`8` 件（**層違反 `0`**）	`0`（`common/architecture`）	**WARN** — `40` 件の型参照循環は `if TYPE_CHECKING:` 内に隔離済み。残る `8` 件は `_builder/_common/parallel.py` 内の遅延ワーカー import `5` 件、`compiler/html.py <-> compiler/jupyter.py` の遅延循環 `1` 件、`_diagrams/{architecture,flow}` の `_edge <-> _junction` 兄弟循環 `2` 件。
高複雑度関数（`> SIG`）	`442`	`<= 50` LOC / 認知複雑度 `<= 15`（`SIG`）	**WARN** — Markdown/OpenXML/Jupyter コンパイラ（`exporter_docx.py`・`options.py`・`compiler/jupyter.py`・`merger.py`・`parallel.py`・`manifest.py`）に集中。
多引数シグネチャ（`> 6` 引数）	`0` **未免除**（`353` 免除）	位置引数 `<= 6`（`SIG`）	**WAIVED**（`.pni/pni.toml`）— 宣言的キーワード専用（`*`）スタイル・タイポグラフィ・レイアウト API `353` 件を `ast/excessive_parameters` で明示免除。

静的品質・保守性メトリクス	実測値	ルール基準 / 目標	判定・工学的解釈
12-Factor / PRR 運用衛生	I/O タイムアウト欠落 `0`	`common/production_readiness`	**PASS / INFO** — I/O タイムアウト欠落は `0` 件。一方、散在する `os.environ` 参照 `15` 件、非 CLI モジュールの `print()` 呼出 `34` 件（`progress.py`・`parallel.py`・`compiler/jupyter.py`・`exporter_jupyter.py`・`processor.py`・`viewer.py`）、旧来 `os.path`（`PTH`）近代化ヒント `1,251` 件が残存。

認知複雑度ホットスポット上位 10 関数（`common/maintainability`）

基礎ジオメトリ・アイコン・キャンバスプリミティブが簡潔に保たれる一方、多形式ドキュメントコンパイル（`DOCX`・`HTML`・`Jupyter` および並列フェンスワーカー）では `markdown-it-py` トークンストリーム走査と各種出力形式の生成処理において深い分岐が蓄積しています。

#	ファイル & 行番号	関数 / メソッド	行数	分岐数	認知複雑度	主責務と分割リファクタリング方針
1	`src/drawlib/_builder/doc_builder/exporter_docx.py:583`	`render_chapter`	`284`	`56`	`178`	見出し・コールアウト・箇条書き・コードフェンス・画像・表を単一ループで出力するトークン走査処理。トークン種別ディスパッチテーブルへ分割。
2	`src/drawlib/_builder/doc_builder/processor/options.py:95`	`parse_block_info`	`104`	`45`	`157`	```` ```drawlib ```` フェンス属性（`file`・`width`・`align`・`code`・`caption` 等）の解析。キー別バリデータ関数へ抽出。
3	`src/drawlib/_builder/doc_builder/compiler/jupyter.py:457`	`_compile_jupyter_locales`	`421`	`46`	`156`	多言語 `.ipynb` コンパイル・アセット同梱・ドライラン検証・ランディングページ生成を一括統括。ロケール別パイプライン段階へ分割。
4	`src/drawlib/_builder/doc_builder/exporter_docx.py:315`	`_render_inline`	`128`	`36`	`102`	インライン装飾（`strong`・`em`・`code_inline`・`link`・数式 / HTML スパン）の状態遷移機械。スパン種別ハンドラへ抽出。
5	`src/drawlib/_builder/doc_builder/merger.py:150`	`build_merged_html`	`288`	`41`	`88`	複数章 HTML 結合・見出しレベルシフト・アンカー書換・目次合成。独立した AST 変換パスへ分離。
6	`src/drawlib/_smartarts/_table.py:490`	`_draw_border_lines`	`96`	`32`	`74`	外枠（`top` / `bottom` / `left` / `right`）・見出し罫線（`top2` / `left2`）・内部グリッド線の解決。外周ヘルパと内部グリッドヘルパへ分離。
7	`src/drawlib/_builder/_common/parallel.py:380`	`_execute_block_render_job`	`110`	`20`	`68`	スタイル注入・キャッシュ照合・エラーラップを伴う並列描画ワーカー実行部。プロトコルコールバックでジョブディスパッチを疎結合化。

#	ファイル & 行番号	関数 / メソッド	行数	分岐数	認知複雑度	主責務と分割リファクタリング方針
8	`src/drawlib/_builder/doc_builder/exporter_docx.py:444`	`_emit_table`	`100`	`20`	`57`	Markdown テーブルを列揃え・セル網掛け付き OpenXML `<w:tbl>` 構造へ変換。
9	`src/drawlib/_builder/doc_builder/exporter_docx.py:903`	`export_chapters_to_docx`	`227`	`26`	`52`	表紙生成・目次フィールド挿入・章連結・決定論的 OPC ZIP タイムスタンプ正規化を統括。
10	`src/drawlib/_builder/_common/manifest.py:832`	`format_project_manifest_toml`	`172`	`19`	`48`	正規化済み `drawlib.toml`（`[project]`・`[i18n]`・`[seo]`）を決定論的キー順序で直列化。セクション別フォーマッタへ分割。

ライブラリと CLI の境界衛生（`library_hygiene_audit`: 94/100・Grade S、`cli_guidelines_audit`: 93/100・Grade S）

`src/drawlib/__init__.py` は 20 サブモジュールに対する明示的 `__all__` と **PEP 562 遅延 `__getattr__` + `__dir__` ローディング**を実装しており、`import drawlib` は Matplotlib や Playwright の重い初期化やネットワーク副作用なしにミリ秒単位で完了します。`P2` として以下の軽微な境界結合 3 点の整理が推奨されます。

1. **ライブラリ層からの `sys.exit(1)` 直接呼出**: `src/drawlib/_core/l1_core/_settings.py`（`124`・`319`・`326`・`335` 行）および `src/drawlib/_builder/doc_builder/processor/viewer.py`（`193`・`197`・`249` 行）が直接 `sys.exit(1)` を呼び出しているため、`ValueError` または型付き `DrawlibError` を送出し `src/drawlib/_cli/_drawlib.py` のみで終了コードへ変換する設計が望まれます。
2. **非 CLI 層の `print()` 出力と副作用**: 内部ビルダー群に散在する `34` 件の `print()` 呼出（`progress.py`・`parallel.py`・`compiler/jupyter.py`・`exporter_jupyter.py`・`processor.py`・`viewer.py`）は構造化 `logging` または注入型レポータへ移行すべきであり、`src/drawlib/_cli/_drawlib.py:129` の `main_callback()` が非 `cache` コマンド実行毎に行う `_ensure_overview_rules_cached()` のディスク書込も見直し余地があります。
3. **サブプロセスタイムアウトと環境変数参照の集約**: `src/drawlib/_builder/_common/browser.py:306` の `install_headless_chromium()` は `subprocess.run(...)` に明示的 `timeout=300` を付与すべきであり、`_builder/_common/{browser.py,cache.py,office.py,parallel.py}`・`exporter_pdf.py`・`viewer.py`・`rules_builder.py`・

`_core/l3_external/_download.py` に分散する `15` 件の `os.environ` 参照を型付き設定ヘルパへ集約すると 12-Factor 整合性が高まります。

5.2 クリーンルームコンテナ検証と実測カバレッジ

静的テスト分析（`python_test_audit`: **95/100・Grade S**）は、`155` テストファイル・`1,341` AST テスト関数（パラメタライズ込み `1,485` pytest 項目）による **100.0% の静的モジュールミラー網羅率**と**アサーション欠如テスト 0 件**を確認しました。`tests/conftest.py` の `autouse` フィクスチャがテスト毎にキャンバス状態を初期化し、`pytest_runtest_call` フックが描画出力を `*_answers/*.png` ベースラインとピクセル単位で自動検証します。

PNI は `.pni/dockerfiles/Dockerfile`（`ghcr.io/astral-sh/uv:python3.12-bookworm-slim` + `pytest-cov`、非特権 `USER pni` UID `10001`、`libatomic1`・ヘッドレス Chromium シェル・全 `83` アセットパッケージを事前組込）上で外部通信を完全遮断（`--network=none`）し、**4 段階のコンテナ実行検証**（`container_runtime_audit`: **96/100・Grade S**、`status="passed"`）を実施しました。

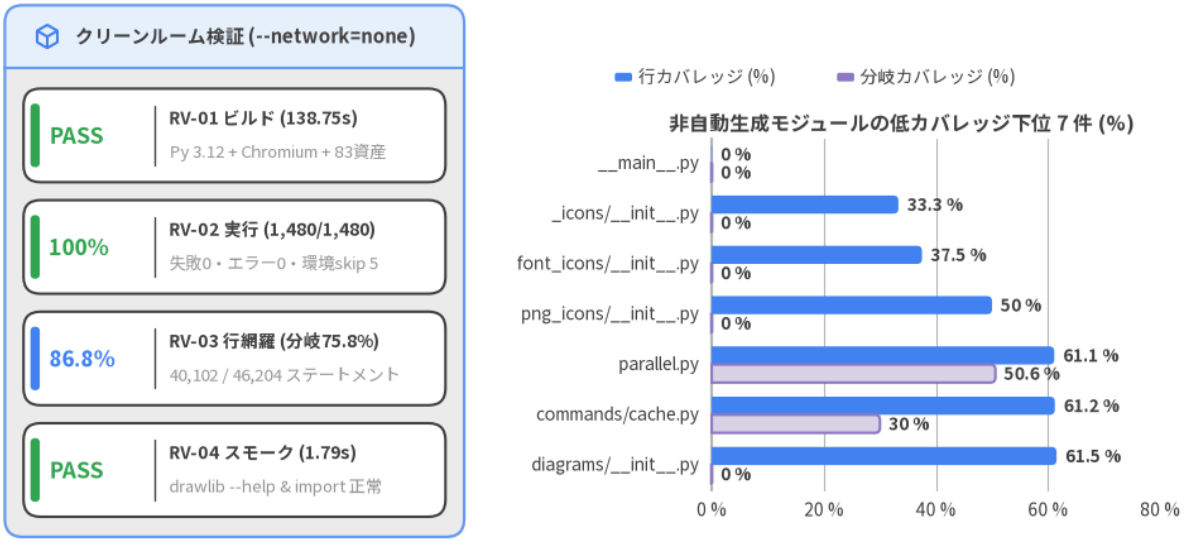


図 5.2 — 完全オフライン（--network=none）コンテナ実行検証 KPI（RV-01〜RV-04、実行 1,480/1,480 全件合格）と非自動生成モジュールの低カバレッジ下位 7 件。

サブシステム別・低カバレッジモジュール一覧（RV-03：行 86.8%・分岐 75.8%）

`src/drawlib/` の全 `46,204` 実行文のうち、`pytest-cov` は `40,102` 行の網羅（`86.8%`）および `75.8%` の分岐網羅を記録しました。主要サブシステムはいずれもオフライン実行下で高い網羅率を達成しています：`_diagrams & _graph`（`95.9%`・`9,394/9,792`）、`compiler/html.py`（`95.1%`・`716/753`）、`_core/l1..l4`（`94.3%`・`5,417/5,743`）、`_charts & _smartarts`（`92.7%`・`6,754/7,284`）、`cache.py`（`92.3%`・`298/323`）、`_slide`（`91.2%`・`1,479/1,621`。`_pptx_writer.py` は `11.1%` から `89.5%` へ、`_pptx_dom.py` は `58.6%` から `85.7%` へ劇的に向上）、`compiler/pdf.py`（`90.5%`・`95/105`）、`compiler/jupyter.py`（`83.1%`・`290/349`）、`exporter_docx.py`（`82.5%`・`514/623`）。

残る低カバレッジモジュールは、(1) 薄い CLI エントリ `__main__.py` と遅延 `__getattr__` ファサード、(2) 代表サンプル検証となる自動生成 `*_generated.py` アイコンラッパー（`51.6%`〜`52.7%`）、(3) 実質的な実行ロジックを持つ 2 モジュール（`_builder/_common/parallel.py`・`_cli/commands/cache.py`）の 3 群に整理されます。

モジュールパス（行カバレッジ昇順）	行カバレッジ	分岐カバレッジ	網羅 / 総文数	主な未網羅行と根本原因
`src/drawlib/__main__.py`	`0.0%`	`0.0%`	`0 / 6`	`12–23` 行：`python -m drawlib` エントリラッパー（CLI テストは `CliRunner` で Typer アプリを直接起動）。
`src/drawlib/_icons/__init__.py`	`33.3%`	`0.0%`	`7 / 21`	`99–109` 行：内部アイコンパッケージルートの遅延 `__getattr__` フォールバックおよび `AttributeError` 経路。
`src/drawlib/_icons/font_icons/__init__.py`	`37.5%`	`0.0%`	`6 / 16`	`24–38` 行：遅延属性解決および `dir()` イントロスペクション補助経路。
`src/drawlib/_icons/png_icons/__init__.py`	`50.0%`	`0.0%`	`6 / 12`	`57–66` 行：遅延 PNG アイコンスイートローダのエラー / フォールバック分岐。
`src/drawlib/_icons/font_icons/phosphor/_generated.py`	`51.6%`	`N/A`	`1,580 / 3,064`	自動生成 Phosphor グリフラッパー（視覚回帰テストで代表サブセットを検証）。

モジュールパス（行カバレッジ昇順）	行カバレッジ	分岐カバレッジ	網羅 / 総文数	主な未網羅行と根本原因
`src/drawlib/_icons/png_icons/aws/_generated.py`	`51.6%`	`N/A`	`342 / 663`	自動生成 AWS アーキテクチャアイコンラッパー（`azure/_generated.py` `52.1%` `263/505`、`gcp/_generated.py` `52.7%` `266/505`）。
`src/drawlib/_builder/_common/parallel.py`	`61.1%`	`50.6%`	`317 / 519`	`57, 125–133+` 行：マルチプロセスワーカープールのフォールバック、エラー伝搬、逐次 / 並列切替分岐。
`src/drawlib/_cli/commands/cache.py`	`61.2%`	`30.0%`	`41 / 67`	`40–43, 58–64+` 行：`drawlib cache` のダウンロード / 削除エラーハンドリングと対話確認分岐。
`src/drawlib/diagrams/__init__.py`	`61.5%`	`0.0%`	`8 / 13`	`39–43` 行：`drawlib.diagrams` 公開ファサードの遅延 `__getattr__` / `AttributeError` ガード。

5.3 完全オフライン検証の進展（v0.3.5 -> v0.3.15）とクリーンルームスキップ一覧

`github.com` や `cdn.playwright.dev` への外部通信が暗黙に成功してしまう通常の GitHub Actions ランナーとは異なり、`--network=none` の完全オフラインコンテナ内で `pytest` を実行すると隠れたネットワーク依存が即座に露呈します。v0.3.5 から v0.3.15 にかけて **前回特定された 2 件のオフライン不具合がいずれも完全解消** され、`722.24s` の実行で `1,480 passed` ・ `0 failed` ・ `0 errors` に到達しました。

1. **オフラインアイコン参照ガード（`解決済み`）**：`src/drawlib/_release_assets.py` の `find_package_for_resource_path()` および `find_package_for_icon_path()` がパッケージ返却前に `normalized in pkg.files` の存在確認を行い、`src/drawlib/_icons/png_icons/_base.py` の `resolve_png_icon_path()` が `download_if_not_exist()` 呼出前に不正アイコン名を即座に拒否する実装へ改修されました。存在しないアイコンの異常系テストは `0` DNS 参照で全件オフライン合格します。
2. **ヘッドレス Chromium と全 83 アセットの事前組込（`解決済み`）**：`.pni/dockerfiles/Dockerfile` がビルド段階（`RV-01`：`138.75s`）で `libatomic1`、Playwright OS 依存ライブラリ（`playwright install-deps chromium`）、`chromium-`

`headless-shell`（`playwright install --only-shell chromium`）、および全 `83` 個のフォント / アイコンキャッシュ（`drawlib cache download --all`）を事前配置したことで、`.pptx` DOM/OpenXML テスト・A4 PDF コンパイルテスト・日本語 `SourceCode` フォントテストがすべて `0` **失敗**で完走しました。その結果、`_pptx_writer.py` は `11.1%` -> `89.5%`、`_pptx_dom.py` は `58.6%` -> `85.7%`、`compiler/pdf.py` は `52.2%` -> `90.5%` へと大幅に引き上がっています。

クリーンルーム環境スキップテスト全件一覧（`RV-02`：`1,480` 合格・`0` 失敗・`5` スキップ）

収集された `1,485` テストのうちスキップされた `5` 件はすべて、最小実行コンテナイメージにリポジトリ執筆専用ディレクトリ（`tools/release_assets/v0.3`・`.agents/rules/drawlib.md`・`docs_src/jupyter`）を意図的に同梱していないことに起因する正常な環境スキップです。

#	状態	テストケース（`module::test_name`）	クリーンルーム未同梱パス	検証メモ
1	`SKIP`	`test_release_assets::test_all_declared_files_exist_on_disk`	`tools/release_assets/v0.3`	完全な Git チェックアウト上にのみ存在する未圧縮リリース資産の配置検証。
2	`SKIP`	`test_release_assets::test_deterministic_zip_hashes_match_all_packages`	`tools/release_assets/v0.3`	リリース資産を再 ZIP 化し `_release_assets.py` の決定論的 SHA-256 と照合。
3	`SKIP`	`test_cli_init_agent::test_repo_antigravity_rule_is_synced`	`.agents/rules/drawlib.md`	リポジトリ直下の `.agents/` ルールと組込 `drawlib rules` 出力の同期確認。
4	`SKIP`	`test_jupyter_docs_dryrun::test_official_jupyter_docs_dry_run`	`docs_src/jupyter`	リポジトリ内バイリンガル Jupyter ハンドブックソースに対するエンドツーエンドドライラン。
5	`SKIP`	`test_jupyter_docs_dryrun::test_official_jupyter_docs_build_outputs`	`docs_src/jupyter`	`docs_src/jupyter` のノートブックおよび HTML ランディングページ出力検証。

6. 詳細分析：セキュリティ・サプライチェーン・API 攻撃面

Drawlib v0.3.15 のファーストパーティソースコード・依存定義・コンテナパッケージングは、`security_vulnerability_audit` **94/100（Grade S、** `status="passed"` **）**、`supply_chain_audit` **95/100（Grade S）**、およびセキュリティ総合軸 **94/100** を達成しています。Drawlib 自身のランタイム依存 `13` 件および開発依存 `10` 件の **CVE 0 件、CRITICAL / HIGH アプリケーション CVE 0 件、有効 IaC 設定不備 0 件、** `434` ソースファイルおよび `111` 付随 DSL ファイルの **シークレット漏洩 0 件**に加え、完全な OpenSSF ガバナンス（`SECURITY.md`・`.github/dependabot.yml`・`CODE_OF_CONDUCT.md`・最小権限 GitHub Actions ワークフロー `5` 本）を完備しています。

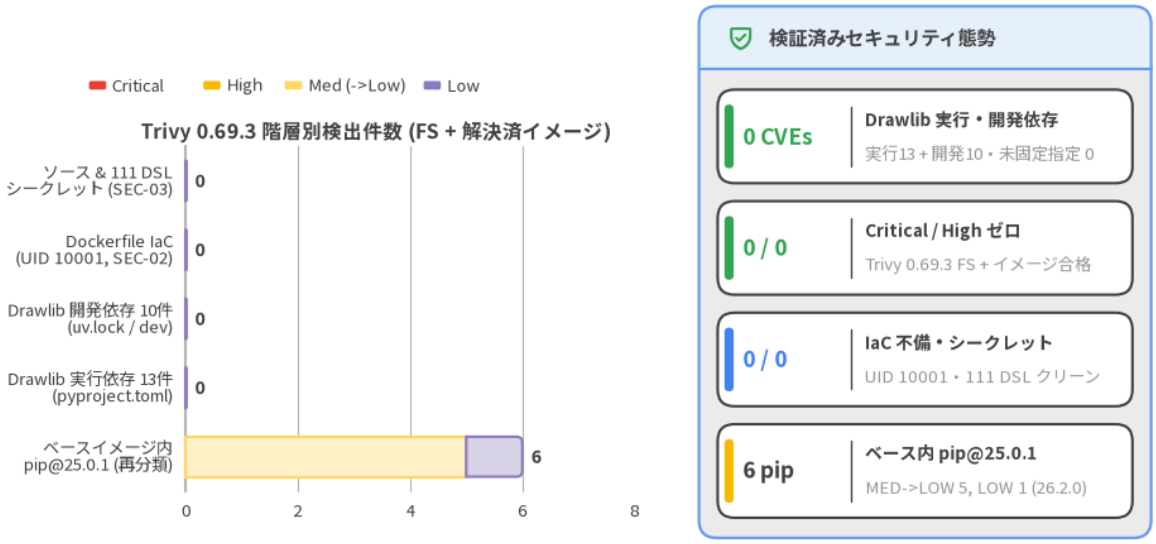


図 6.1 — Trivy 0.69.3 SCA・IaC・シークレット診断状況（v0.3.15）。Drawlib のランタイム 13 件・開発 10 件の依存、IaC、シークレットはすべて 0 件。ベースイメージ内システム pip@25.0.1 の 6 件は LOW へ再分類。

6.1 依存関係脆弱性・コンテナイメージスキャン (SEC-01: Drawlib 依存 CVE 0 件)

Trivy v0.69.3 によるソフトウェア構成解析 (SEC-01) は、プロジェクトファイルシステム (pyproject.toml ・ uv.lock) および .pni/dockerfiles/Dockerfile (ghcr.io/astral-sh/uv:python3.12-bookworm-slim ベース) から構築した解決済みコンテナイメージの双方を検証しました。

- **Drawlib 自身の依存ツリーは脆弱性 0 件 (0 CVEs)** : 直接ランタイム依存 13 件 (matplotlib>=3.8 ・ pillow>=13.0.0 ・ numpy>=1.26 ・ pydantic>=2.8 ・ pygments>=2.18 ・ markdown-it-py>=3.0 ・ jinja2>=3.1 ・ markupsafe>=2.1 ・ typer>=0.12.0 ・ rich>=13.0.0 ・ playwright>=1.49.0 ・ python-docx>=1.2.0 ・ python-pptx>=1.0.2) および開発依存 10 件はいずれも **未固定バージョン指定 0 件** かつ **CVE 0 件** (CRITICAL 0 ・ HIGH 0 ・ MEDIUM 0 ・ LOW 0) です。
- **ベースイメージ内システム pip@25.0.1 (6 件、act-09 にて LOW へ再分類)** : 解決済みイメージスキャンで検出された Python パッケージ脆弱性 6 件 (MEDIUM 5 件 ・ LOW 1 件) は、すべて ghcr.io/astral-sh/uv:python3.12-bookworm-slim に同梱されるシステム pip@25.0.1 に起因します。Drawlib は uv を採用しており実行時に pip をインポートも同梱もしないため、 act-09 にて MEDIUM 5 件を LOW へ再分類しました。Dockerfile 内でシステム pip >= 26.2.0 へ更新 (または不要な pip を削除) することで 6 件すべてが解消します。
- **コンテナ OS ベース層の範囲区分**: プロジェクト同梱の Dockerfile (tools/docker/Dockerfile.*) は **OS CVE 0 件** です。一方、PyPI ホイールには一切出荷されないローカル検証専用テストハーネス (.pni/dockerfiles/Dockerfile 、Debian bookworm パッケージおよび Chromium 依存 OS ライブラリ) では上流 Debian bookworm OS 勧告 740 件 (CRITICAL 11 ・ HIGH 181 ・ MEDIUM 304 ・ LOW 219) が計上されています。

ベースイメージ内システム pip@25.0.1 脆弱性一覧 (Drawlib 実行時未使用)

勧告 ID	元 -> 再分類 深刻度	CVSS	検出 -> 修正版	脆弱性概要とコンテナ是正策
CVE-2026-8643	MED -> LOW	8.0	pip@25.0.1 -> 26.1.2	不正なエントリポイント名を含む wheel インストール時のパストラバーサル。

勧告 ID	元 -> 再分類 深刻度	CVSS	検出 -> 修正版	脆弱性概要とコンテナ是正策
`CVE-2026-13346`	`MED` -> `LOW`	`6.5`	`pip@25.0.1` -> `26.2.0`	悪意あるパッケージインデックス経由の任意ファイル配置。
`CVE-2026-6357`	`MED` -> `LOW`	`5.8`	`pip@25.0.1` -> `26.1`	細工された wheel による任意コード実行または情報漏洩。
`CVE-2025-8869`	`MED` -> `LOW`	`5.3`	`pip@25.0.1` -> `25.3`	アーカイブ展開時のシンボリックリンク検証欠如。
`CVE-2026-3219`	`MED` -> `LOW`	`5.0`	`pip@25.0.1` -> `26.1`	不適切なアーカイブ処理による意図しないファイル配置。
`CVE-2026-1703`	`LOW`	`3.9`	`pip@25.0.1` -> `26.0`	細工された wheel アーカイブ展開時のパストラバーサルによる情報漏洩。

推奨是正策 (`P2`): `uv` のみを使用するコンテナビルド段階で `python -m pip install --no-cache-dir --upgrade "pip>=26.2.0"` を実行するか、未使用のシステム `pip` を削除してください。

6.2 コンテナ IaC ハードニング (`SEC-02`) とシークレット衛生 (`SEC-03`)

Dockerfile ハードニング (有効な設定不備 `0` 件)

Trivy IaC スキャンは `tools/docker/` 配下の本番・テスト用 Dockerfile `3` 本と `.pni/dockerfiles/Dockerfile` を検証しました。

チェック ID	深刻度	対象コンテナ定義	状態	実装内容と検証結果
`DS-0002`	`HIGH`	`tools/docker/Dockerfile.{prod,test_local,test_pypi}`, `.pni/dockerfiles/Dockerfile`	`PASS` (不備 `0` 件)	すべてのイメージで非特権ユーザー (`USER drawlib` / `USER pni` ・ `UID/GID 10001`) を作成・切替し、Playwright ブラウザ (`/opt/ms-playwright`) およびアセットキャッシュに非 root 読取権限を付与。Trivy `DS-0002` 不備は `0` 件。
`DS-0026`	`LOW` (`×3`)	`tools/docker/Dockerfile.{prod,test_local,test_pypi}`	`WAIVED` (`.pni/pni.toml`)	常駐デーモンではなく処理完了後に即終了するバッチ CLI / `pytest` コンテナのため、`[[opt_out]] id = "DS-0026"` により明示免除 (`opt_out_counts = {"DS-0026": 3}`)。

シークレットスキャンと付随 DSL 衛生 (`SEC-03` : シークレット `0` 件)

- **ソースコード内ハードコード秘密情報 `0` 件 (`PASS`)** : AST リテラルエントロピー解析 (`AstFacts.hardcoded_secrets = []`) と Trivy リポジトリスキャンの双方で、 `434` ソースファイルおよび `155` テストファイル全体から **API キー・認証情報・トークン 0 件、コミット済み秘密ファイル 0 件**を確認しました。
- **付随 DSL ファイル群 (`111` ファイル・シークレット `0` 件)** : Python 以外の設定・構成ファイル `111` 件 (`dockerfile=3` ・ `css=19` ・ `yaml=9` ・ `json=77` ・ `toml=1` ・ `xml=2`) すべてにおいて秘密情報検出は `0` 件 です。
- **デフォルト拒否 `.dockerignore` と `.pni/pni.toml` スコープ (`PASS`)** : ルート `.dockerignore` は先頭 `*` (全拒否) から必要ディレクトリのみを許可リスト指定し、 `.gitignore` と `.pni/pni.toml` (`skip_files = [".env", ".env.*", "**/.env"]`) が開発者ローカルの `.env` ファイルを保護します。

6.3 AST 危険コードシンク (`CWE-94`) と API 攻撃面



図 6.2 — CWE-94 exec()/eval() 全 8 箇所の棚卸し (.pni/pni.toml で免除済み) 、インジェクション / API / 111 DSL ゼロ検証、および非信頼 Markdown 向け多層サンドボックス境界。

`CWE-94` 該当箇所・インジェクション分類・API エンドポイント詳細評価

1. **設計上意図された 8 箇所の `CWE-94`（`exec` / `eval`）シンク（`[[opt_out]] id = "CWE-94"`・`matched=8`・未免除 `0` 件）**:
   - `src/drawlib/_builder/_common/styles_loader.py:78`（`load_styles`）および `styles_loader.py:158`（`load_utils`）：ドキュメントプロジェクト独自の `Styles` / `Colors` 定義や再利用マクロを読み込むため、`styles.py` / `utils.py` をコンパイル実行します。
   - `src/drawlib/_builder/doc_builder/processor/processor.py:121`（標準出力抑制モード）および `processor.py:123`（`verbose` / `developer` モード）：Markdown 内の ```` ```drawlib ```` フェンス Python コードを `save()` インターセプト付きで実行します。
   - `src/drawlib/_builder/doc_builder/exporter_jupyter.py:417`・`419`（末尾式 `eval`）・`428`（`exec`）：Jupyter AI ドライラン検証において生成済み `.ipynb` コードセルの実行可能性を事前検証する 3 箇所です。
   - `src/drawlib/_core/l4_canvas/_image_code.py:54`（`_worker_render_code`）：`get_dimage_from_code(code, timeout=...)` 向けに隔離された `multiprocessing` 子プロセス内でネスト描画スクリプトを実行します。
   - **アーキテクチャ評価**: Jupyter Notebook・Quarto・Sphinx のコード実行ディレクティブと同様、Python コードを実行すること自体が Drawlib（*"Illustration as Code"*）の中核価値です。この信頼モデルはルート `SECURITY.md` に明文化され、`.pni/pni.toml` の `[[opt_out]] id = "CWE-94"` として記録されています。信頼できない第三者製 Markdown をビルドする際は、`./dcli docker build --full` で構築した非 root・オフラインコンテナ（`--user 10001:10001 --network=none --read-only --pids-limit=128 --memory=1g`）内での隔離実行を徹底してください。
2. **インジェクションおよび安全でないデシリアライズ脆弱性 0 件（`PASS`）**:
   - `CWE-89`（**SQL インジェクション = `0`**）: `src/drawlib/_builder/_common/cache.py` の `BuildImageCache` / `CliImageCache` は静的スキーマ上で厳格な `?` パラメータ化 SQLite クエリのみを使用します。
   - `CWE-78`（**OS コマンドインジェクション = `0`**）: `src/drawlib/_builder/_common/browser.py:306` のサブプロセス起動は `shell=False` で明示的引数配列（`[str(exec_path), str(cli_path), *_INSTALL_ARGS]`）を渡しています。

- `CWE-79` / `CWE-502`（**XSS / 安全でないデシリアライズ = `0`**）: HTML テンプレートは `Jinja2` + `MarkupSafe` の自動エスケープを利用し、マニフェスト解析は標準ライブラリ `tomllib`、キャッシュは生 PNG/SVG バイト列と SHA-256 ダイジェストのみを保持します（ `pickle`・危険な `yaml.load` は `0` 件）。

3. **API エンドポイント攻撃面（ `SEC-05` / `SEC-06`: ルート `0` 件、`not_applicable`）**:
   - Drawlib のランタイム種別は `library+cli` であり、**本番 HTTP/gRPC サービスエンドポイントは 0 件**です。`drawlib serve` コマンドはローカルプレビューおよび `--check` リンク検証専用として `127.0.0.1` に厳格バインドされます。

6.4 OpenSSF サプライチェーン態勢およびガバナンスチェックリスト

Drawlib v0.3.15 は `supply_chain_audit` **95/100（Grade S）**および `repo_structure_audit` **96/100（Grade S）**を記録しました。v0.3.5 以降、`SECURITY.md` と週次 `.github/dependabot.yml` の追加によりガバナンス上の 2 大ギャップを解消し、最小権限 GitHub Actions ワークフローも `5` 本へ拡張しています。

サプライチェーン / ガバナンス統制	v0.3.15 判定	証跡箇所	検証内容・強化ポイント
オープンソースライセンス（`Apache-2.0`）	**PASS**	`LICENSE`	リポジトリルートおよび `pyproject.toml` に標準 Apache-2.0 ライセンスを明示宣言。
脆弱性開示ポリシー	**PASS** *(v0.3.5 以降新設)*	`SECURITY.md`	`0.3.x` サポート方針、*"Illustration as Code"* 実行信頼モデル、コンテナ隔離コマンド、GitHub Private Vulnerability Reporting 窓口を明記。
コミュニティ行動規範	**PASS**	`CODE_OF_CONDUCT.md`	リポジトリルートに行動規範およびエスカレーション指針を配置。
依存関係自動更新	**PASS** *(v0.3.5 以降新設)*	`.github/dependabot.yml`	`uv` Python 依存関係および `github-actions` の双方に対し週次グループ PR 更新を設定済み。
CI ワークフロー最小権限（`5` 本）	**PASS**	`.github/workflows/{quality,release,ut_github,ut_pypi,ut_template}.yml`	CI サプライチェーン指摘 `0` 件。全 `5` 本が明示的 `permissions: contents: read` を宣言し、`release.yml` の `publish-release` ジョブのみ `contents: write` へ厳格限定。

サプライチェーン / ガバナンス統制	v0.3.15 判定	証跡箇所	検証内容・強化ポイント
外部バイナリ資産の完全性 (`83` パッケージ)	**PASS**	`src/drawlib/_core/l3_external/_package.py` & `src/drawlib/_release_assets.py`	遅延ダウンロード対象のフォント・アイコン・GeoJSON アーカイブ全件に組込 **SHA-256 ダイジェスト検証**、Zip-Slip パス検証、`pkg.files` メンバー検査、`timeout=60` HTTPS 取得を実装。
コンテナ最小権限 (`USER`)	**PASS**	`tools/docker/Dockerfile.{prod,test_local,test_pypi}`, `.pni/dockerfiles/Dockerfile`	非 root `USER drawlib` / `USER pni` (`UID 10001`)、Trivy `DS-0002` 不備 `0` 件、デフォルト拒否 `.dockerignore` を徹底。
依存バージョン指定とロックファイル	**PASS / WAIVED**	`pyproject.toml`, `uv.lock`, `.pni/pni.toml`	全 `13` 実行依存に SemVer 下限指定 (未固定 `0` 件・Drawlib CVE `0` 件)。Python `3.11–3.14` ライブラリ柔軟性のため `uv.lock` の gitignore 方針は `python/lockfile_committed` で明示免除。
GitHub Actions コミット SHA 固定	**INFO** (`P2`)	`.github/workflows/{quality,release,ut_template}.yml`	現在はメジャーバージョンタグ (`actions/checkout@v7` ・ `astral-sh/setup-uv@v7` ・ `actions/setup-python@v7`) を参照。不変な出所保証のため 40 桁コミット SHA + バージョンコメント固定を推奨。