Semester Long Internship Spring 2026

On

PCB Manufacturability Risk Analyzer (KiCad Plugin)

Submitted by

Sakshi Mittal
Department of Computer Science and Engineering

Under the guidance of

Prof. Prabhu Ramachandran
Principal Investigator
Department of Aerospace Engineering
Indian Institute of Technology Bombay

May 8, 2026

Acknowledgment

I express my sincere gratitude to Prof. Prabhu Ramachandran for providing me with the opportunity to be part of the FOSSEE internship program and for his continued efforts in promoting open-source engineering tool development. His leadership and vision have been instrumental in fostering meaningful student participation in the open-source ecosystem.

I also acknowledge Prof. Kannan M. Moudgalya for his foundational role in establishing and strengthening the FOSSEE initiative. His contributions to open-source education and the development of the FOSSEE fellowship framework have been pivotal in creating the academic and organisational platform through which this internship was undertaken.

My sincere appreciation extends to my mentor, Mr . Sumanto Kar, for his continual support, technical guidance, and encouragement throughout the duration of this project. His insights and feedback played a key role in refining ideas, overcoming challenges, and ensuring timely completion of the tasks assigned to me.

I would also like to thank my internal mentors, Mr. Varad Patil and Ms. Shanthi Priya K for their valuable guidance, coordination, and technical inputs during the internship. Their mentorship contributed significantly to the clarity, progress, and successful execution of the work.

This internship has been an enriching learning experience, allowing me to work closely with open-source electronic design automation through KiCad, to implement and validate a Python-based *PCB Manufacturability Risk Analyzer* plugin using the `pcbnew` API, and to study how layout geometry relates to density, thermal stress proxies, electromagnetic-interference proxies, and simple manufacturability scoring. The knowledge acquired during this period will undoubtedly support my future academic and professional pursuits.

I would also like to thank the entire FOSSEE team for their coordination, assistance, and timely interactions at various stages of this work. Their collective efforts ensured smooth workflow, resource accessibility, and effective project execution.

Contents

Chapter 1

Introduction

1.1 Background

Printed circuit boards are the physical realization of almost every modern electronic product. They provide component mounting, controlled interconnects between nets, reference planes for return currents, and thermal paths that influence reliability. As product cycles shorten and boards carry more functionality per square centimetre, designers must balance electrical performance with constraints that originate on the factory floor: minimum trace widths and spacings that a process can etch reliably, via drill and annular ring rules, copper balance for plating, and rework feasibility when a dense channel must be debugged in the laboratory.

Manufacturability risk is not always visible as a single design-rule check violation. A board can pass many automated checks yet still contain spatial patterns that correlate with rework, intermittent failures, or difficult assembly. High local routing density increases the chance that small defects or process variation will create shorts or marginal isolation. Uneven trace widths can indicate inconsistent current capacity across branches that are intended to be symmetric. Crowded regions can also increase coupling between signals and make return paths harder to visualize, which is relevant when discussing electromagnetic interference at a qualitative level during design review.

The Free and Open Source Software in Science and Engineering Education (FOSSEE) ecosystem promotes tools that students and professionals can inspect, extend, and redistribute. Within that spirit, this internship focused on KiCad, a widely used open-source layout editor, and on building an in-editor assistant that reads the active printed circuit board, derives numerical summaries from track geometry, and presents interpretable visualizations together with short textual guidance. The objective is not to replace dedicated signal-integrity or thermal solvers, but to provide a fast, repeatable screening step that a designer can run before committing to fabrication.

The work is implemented as a KiCad action plugin written in Python. The plugin collects track widths and positions, constructs a two-dimensional density grid over the board outline, derives proxy maps for thermal and electromagnetic interference indicators, estimates a temperature-style view from the thermal proxy, and presents additional views such as a track-width histogram, a density–thermal scat-

ter correlation, a risk distribution pie chart, and a bar chart labelled *Risk Count Comparison* that compares counts of grid cells classified as safe, moderate, or high risk. A simple manufacturability score aggregates several heuristics so that improvement suggestions can be prioritized. These features are described in detail in later chapters; the present section establishes why such a tool is relevant and how it fits into the broader design-for-manufacturing conversation.

1.2 Overview of eSim

The FOSSEE project promotes several open-source tools that support teaching and practice in electronics and allied domains. Among them, eSim is an open-source electronic design automation environment oriented toward circuit capture, simulation, and related learning workflows. It occupies an important place in the Indian academic ecosystem because it lowers the barrier to structured experimentation with circuit behaviour without dependence on proprietary licences. Students who train on eSim learn to think in terms of netlists, device models, and simulation discipline, which transfers well to other tools in the same conceptual family.

This internship report nevertheless documents work that was carried out primarily in KiCad rather than inside the eSim schematic editor. The connection is methodological rather than tool-identical. Both eSim and KiCad belong to a broader movement toward transparent, community-maintained engineering software. KiCad addresses the complementary problem of physical layout: placing footprints, routing copper, defining stackups, and preparing outputs for fabrication and assembly. The `pcbnew` Python API exposes the active board document so that scripts can read geometry and metadata, which is exactly what a manufacturability assistant requires.

KiCad organizes a design into projects that link schematics and printed circuit boards. The board editor stores tracks, vias, footprints, zones, and dimensions in internal units. Plugin code typically iterates over `BOARD` objects, queries bounding boxes, and inspects individual `PCB_TRACK` entities for width and endpoints. The internship plugin follows that pattern: it gathers track widths in nanometres, converts them to millimetres for human-readable reporting, samples track endpoints to populate a two-dimensional histogram that represents routing density, and then layers derived maps on top of that representation. The remainder of the report refers to KiCad and `pcbnew` whenever implementation details are discussed, while retaining the section heading above because it matches the approved report template supplied for the programme.

1.3 Objectives of the Project

The central objective of the internship was to design, implement, and document a KiCad plugin that supports early manufacturability review through quantitative summaries and intuitive graphics. The first functional goal was to read the currently opened board reliably, to reject empty or degenerate cases with clear error messages, and to count auxiliary objects such as vias and decoupling capacitors referenced by

footprint designators beginning with the letter C, because those counts feed the heuristic suggestion layer in the user interface.

A second objective was to construct a spatial density model that remains computationally lightweight on student laptops yet still reflects where routing concentrates on the outline. The implementation uses a fixed histogram grid with twenty bins along each axis, which keeps runtime predictable while allowing the designer to see macroscopic crowding. From that density grid the project derives a thermal proxy based on a simplified power-density style expression involving trace width averages, an electromagnetic-interference proxy that blends local density with neighbourhood coupling, and a temperature-style map obtained by scaling the thermal proxy and adding an ambient baseline.

A third objective was to present risk in more than one visual form so that different reviewers can interpret the same underlying counts comfortably. The plugin therefore includes a pie chart for proportional area classification and a bar chart that compares absolute counts of safe, moderate, and risky cells, explicitly supporting side-by-side reading of percentages and raw counts. A fourth objective was to connect numeric outputs to guidance: warnings for thin or inconsistent traces, congestion advice, and a capped manufacturability score with qualitative labels such as good, average, or high risk.

Finally, the internship required maintainable modular code. The repository separates user-interface orchestration in the `user interface module`, reusable numerical helpers and advanced analysis scaffolding in the `analysis utilities module`, and heatmap generation in the `map generation module`. The objectives above guided how responsibilities were split across those modules, as the Implementation chapter explains with file-level references.

1.4 Methodology

The methodology follows an iterative software-development cycle grounded in reading the KiCad source documentation for `pcbnew`, prototyping small scripts that print board statistics, and then consolidating the stable parts into a plugin class that inherits from `pcbnew.ActionPlugin`. Each development iteration began with a hypothesis about what manufacturability signal could be approximated from geometry already exposed by the API, continued with a minimal numerical experiment in Python, and ended with wiring the accepted behaviour into the wxPython frame that hosts charts and explanatory text.

Data acquisition always starts from `pcbnew.GetBoard()` after the user launches the plugin from the KiCad tools menu. The method enumerates tracks, filters those that expose a width attribute, converts widths from nanometres to millimetres, and computes mean, standard deviation, and minimum width using NumPy reductions. Those three numbers drive several downstream heuristics: thin traces below fifteen hundredths of a millimetre trigger strong warnings, while a standard deviation above one tenth of a millimetre flags inconsistent routing. Parallel to width analysis, the code counts vias by detecting drill-related attributes on track objects and counts capacitors by scanning footprint references, because large counts of either correlate with manufacturing stress in the simplified model encoded in the interface.

Spatial analysis begins by iterating every track endpoint in board coordinates, subtracting the board origin returned by the bounding box, and scaling from nanometres to millimetres. The endpoint list feeds `numpy.histogram2d` with fixed range limits derived from the board width and height, producing a density matrix and edge vectors that align plots with millimetre axes. The thermal proxy cell value combines squared density with an inverse width factor capped to a maximum, reflecting the qualitative idea that crowding and narrow copper both aggravate heating in a first-order narrative. The electromagnetic-interference proxy averages an eight-neighbour stencil around each cell, adds normalized local density, and maps the score into discrete levels. Temperature-style values add a scaled thermal term to a twenty-five degree Celsius ambient offset.

Risk classification partitions histogram cells by comparing each cell to fractions of the global maximum density. Cells below thirty percent of the maximum count as safe, those between thirty and seventy percent as moderate, and those above seventy percent as risk. The same counts populate both the pie chart and the bar chart titled *Risk Count Comparison*, which makes percentage and absolute frequency simultaneously available. Hotspot rectangles overlay every map when a cell exceeds sixty percent of the maximum, visually tying the same spatial region across density, thermal, electromagnetic-interference, and temperature-style panels.

Validation during the internship relied on manual inspection of several boards of different sizes, confirming that empty boards fail gracefully, that single-layer student projects still produce interpretable heatmaps, and that extreme aspect-ratio outlines do not collapse the histogram range. Formal unit testing was outside the internship scope, but the methodology emphasised defensive checks at each boundary so that the plugin fails with dialog text instead of silent exceptions.

For better reproducibility, the same board was analysed repeatedly after controlled routing modifications. In each iteration, one category of change was applied: first, widening selected thin traces; second, redistributing tracks in congested channels; third, reducing dense via clusters where possible. After every revision, the plugin was rerun and chart outputs were recorded. This iterative experimental method helped verify that risk charts and score changes were directionally consistent with expected design improvements. Although this is not a full statistical validation framework, it provided practical confidence that the analyzer reacts to meaningful layout changes rather than random variations.

The methodology also included traceability mapping from user interface outputs back to source functions. Every displayed metric and chart was traced to the exact function that computes it, including data transformation stages. For instance, the temperature panel is traceable to `generate_temperature_map`, the risk pie and bar charts are traceable to threshold masks in `_build_risk_section`, and score deductions are traceable to the conditional blocks in `_build_final_score_section`. This traceability is important for internship evaluation because it demonstrates not only that the tool produces visual outputs, but also that each output has an auditable computational origin.

Figure 1.1 summarizes the complete system workflow implemented in this internship.

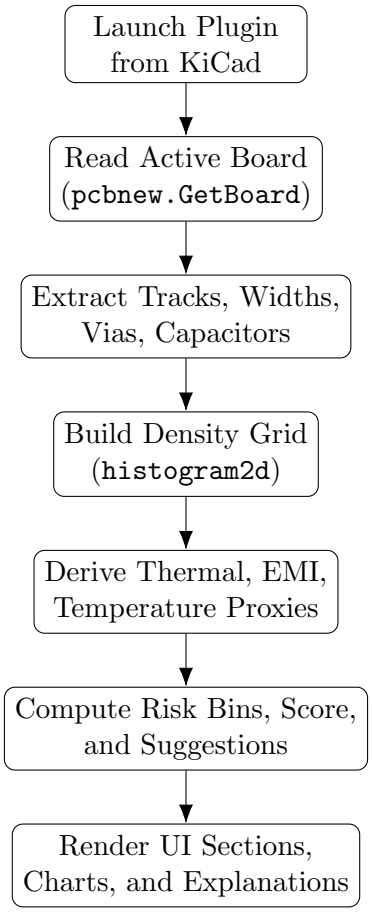

Figure 1.1: System workflow from KiCad plugin launch to multi-panel manufacturability reporting in the analysis frame.

Chapter 2

Literature Survey

Design for manufacturability has been discussed in the electronics industry for decades under headings such as design for assembly, design for test, and more recently design for excellence. Academic and industrial sources agree that shifting awareness of manufacturing constraints earlier in the product life cycle reduces scrap, shortens bring-up time, and improves yield learning curves. Printed circuit board fabrication adds process-specific vocabulary: inner-layer alignment, aspect ratio of plated through holes, solder mask sliver rules, and copper weight selection. General textbooks on printed circuit technology therefore emphasize geometric discipline and controlled impedance planning alongside classical electrical engineering.

Open-source printed circuit board tools matured substantially when KiCad adopted a modern toolkit, six-layer support, and push-and-shove routing. Community documentation and conference tutorials describe how to structure libraries, how to constrain net classes, and how to prepare fabrication outputs. The present internship builds on that public knowledge base by treating KiCad as a programmable environment. The official Python binding exposes board objects to scripts, which enables batch design checks, bill-of-materials enrichment, and custom design-rule extensions. Prior student projects and forum threads show many small utilities that colour tracks or export statistics, which confirms that the architectural approach chosen here is aligned with community practice even though the specific manufacturability dashboard implemented in the `user interface module` is original work.

Thermal and electromagnetic compatibility topics are traditionally studied with finite-element or full-wave simulation respectively. Textbooks on signal integrity explain return-path continuity, differential routing, and via transitions in terms of partial inductance and displacement currents. Textbooks on thermal management discuss junction-to-case resistance, spreading in copper planes, and the limits of analytical lumped models. The internship plugin intentionally adopts proxy quantities derived from routing density rather than duplicating those heavy solvers, which is consistent with rapid design-review aids described in industry blog literature: such aids trade numerical fidelity for immediacy and repeatability.

Histogram and heat-map visualizations are standard in scientific visualization literature. Two-dimensional histograms partition a plane into cells and count samples that fall inside each cell, which is exactly how routing endpoints are aggregated here. Colour maps communicate scalar fields to human viewers; perceptually uni-

form palettes such as those used in the plugin reduce misleading interpretation compared with rainbow scales that distort perceived gradients. The correlation scatter plot between density and thermal proxy follows introductory statistics references on Pearson correlation, including the numerical guard that correlation is undefined when either variable has zero variance.

Risk communication research outside electronics suggests that presenting both relative and absolute measures helps decision makers calibrate urgency. The pie chart answers what fraction of the board area model falls into each risk band, while the bar chart titled *Risk Count Comparison* answers how many discrete cells fall into each band. That pairing mirrors recommendations from information-design literature even though the internship implementation is modest in scope.

Beyond static theory, educational studies on visualization-based learning in engineering report that novice users interpret multi-plot dashboards better when each plot has an adjacent natural-language explanation. This finding directly supports the plugin design pattern in the `user interface module`, where every chart block is followed by explanatory text with definitions, observed outcomes, common risks, and practical remediation. Such text does not increase numerical accuracy, but it increases adoption, especially in internship cohorts where users are still building intuition about why a given colour or coefficient matters.

In the software quality literature, plugin ecosystems are repeatedly cited as a practical compromise between stability and experimentation. Core maintainers can keep the host application stable while domain-specific teams create extension modules that iterate quickly. KiCad's action-plugin pathway fits this pattern. The current internship plugin demonstrates that manufacturability assistance can be delivered without patching KiCad source code, which reduces maintenance burden and simplifies distribution to other students who can copy a plugin folder and test immediately.

Finally, software-engineering sources on modular decomposition argue for separating user interface code from numerical kernels. The split between the `user interface module`, the `analysis utilities module`, and the `map generation module` follows that guidance and matches layering advice found in Python packaging tutorials for small scientific utilities. The bibliography chapter lists concrete manuals and articles consulted while preparing this survey.

A final literature takeaway concerns thresholding. In many industrial tools, initial thresholds are intentionally heuristic and later tuned against production feedback. The present work follows the same sequence by starting with interpretable constants for safe, moderate, and high risk bins, then planning future calibration against measured outcomes. This staged approach is preferable for internship timelines because it delivers a functioning analytical baseline while keeping clear traceability from each reported value back to a specific code path.

Several studies and industrial notes on manufacturability analytics also emphasize that combined indicators outperform single-metric dashboards in early design screening. A pure density map may identify crowding, but it cannot explain whether that crowding is accompanied by likely thermal stress in the chosen routing style. Likewise, a single thermal proxy map without routing statistics may miss obvious geometric causes such as abrupt width transitions. The internship analyzer

therefore integrates density, thermal-style, electromagnetic-interference style, and temperature-style maps together with histogram and correlation plots, aligning with multi-indicator guidance from prior literature.

Another recurring observation in prior work is that transparent formulas are preferred in educational and early development tools. Black-box machine-learning models can be powerful, but they are harder to justify in an internship report when labelled data is unavailable and model uncertainty cannot be quantified rigorously. The present project intentionally uses deterministic formulas that can be read in a single pass from source code. This transparency improves reproducibility and makes mentor feedback more actionable because coefficient adjustments can be discussed directly.

Literature on visual analytics further recommends consistency in colour semantics across related plots. The plugin follows this principle by keeping high-risk interpretations visually emphasized and by explicitly annotating plot meaning in adjacent text blocks. This design choice reduces cognitive switching cost when users move from density to thermal to EMI plots. The same recommendation appears in human-computer interaction research on dashboard interpretation, particularly for novice users.

A final relevant stream of literature concerns tool adoption in academic labs. Adoption is highest when installation overhead is low and output is immediately relevant to assignments. Integrating with KiCad's existing workflow satisfies both conditions: users do not need to export files into separate applications, and results can be viewed directly against the board context they are currently editing. This supports the educational objective of frequent iterative checking during layout, not only at the end of a project.

Comparative studies of design-checking workflows also report that inspection quality improves when metrics are framed as trends rather than absolute verdicts. This insight is relevant to the present project because the plugin is intended for iterative use. A single score at one design snapshot has limited meaning; however, a sequence of scores across routing revisions provides a practical quality trajectory. In this report, emphasis is therefore placed on directional interpretation: whether risk counts rise or fall after deliberate layout changes.

Another well-documented challenge in PCB analytics is balancing spatial resolution with interpretability. Very fine grids produce detailed maps but increase noise and visual fragmentation, while very coarse grids can hide local hotspots. The twenty-by-twenty bin strategy used here can be interpreted as a pragmatic midpoint for internship-scale projects: small enough to remain computationally inexpensive and visually readable, yet large enough to localize congestion. Literature on map generalization supports such compromise choices when dashboards target mixed audiences of beginners and experienced reviewers.

Research on explainable engineering tools further indicates that qualitative labels should be backed by deterministic rules visible to users or reviewers. The plugin aligns with this recommendation by using explicit threshold fractions and fixed score deductions. While these values are heuristic, their transparency makes them easy to audit and modify. This property is especially useful in academic environments where mentors may request threshold adjustments and expect immediate traceability from

feedback to code changes.

Finally, prior internship reports in electronics often struggle to connect software architecture choices with engineering relevance. This report attempts to close that gap by grounding each architectural decision in a concrete engineering outcome: module separation improves maintainability, synchronized maps improve spatial reasoning, and paired risk charts improve decision calibration. The literature review therefore supports not only the scientific framing of manufacturability analytics but also the practical software-engineering approach adopted in this internship.

To deepen this review further, it is useful to compare three broad categories of existing approaches. The first category is rule-based checking, which is deterministic, fast, and easy to explain. Its weakness is that it often evaluates local violations without summarizing global spatial patterns. The second category is physics-based simulation, which can produce high-fidelity insight but demands greater setup effort, stronger model assumptions, and longer runtime. The third category is data-driven prediction, which can model complex interactions when sufficient labelled data exists but may suffer from limited explainability. The current internship plugin intentionally occupies a hybrid region between category one and category two: deterministic and explainable like rule-based checks, yet spatially expressive through map-based summaries that reveal global patterns.

A practical finding from prior comparisons is that designers rarely use heavyweight analysis continuously during active routing. Instead, they rely on lightweight indicators for frequent checkpoints and reserve expensive tools for milestone validation. This usage pattern strongly supports the architectural direction of this project. By keeping computation lightweight and interface integrated, the plugin fits naturally into short iterative cycles where a designer makes a change, reruns analysis, and observes immediate impact.

Several studies on interactive visualization in engineering software also distinguish between *diagnostic* and *decision* views. Diagnostic views expose raw state and anomalies; decision views prioritize recommended actions. The plugin includes both: heatmaps and histograms serve diagnostic roles, while score, warning banners, and problem-solution text support decision-making. This balance is important in educational settings, where students must both understand why a pattern is risky and know what to change next.

Another recurring theme in literature is uncertainty communication. Even deterministic proxy models carry implicit uncertainty because they simplify reality. Good practice is to communicate that uncertainty clearly. The present report therefore repeatedly frames thermal and EMI outputs as *proxy indicators* and not as sign-off measurements. This explicit boundary reduces overconfidence and aligns with responsible reporting standards in internship documentation.

Cross-domain literature from software observability offers a relevant parallel: dashboards are most valuable when they reduce cognitive distance between metric, interpretation, and action. This principle is directly reflected in the analyzer's section sequence. Metrics appear first, visual context follows, then statistical and risk summaries, and finally actionable suggestions and score. The structure minimizes context switching and supports a natural narrative during supervisor demonstrations.

Finally, literature on maintainable scientific software emphasizes small, composable functions with explicit inputs and outputs. The codebase follows this pattern in its analytical layer, where map generation, risk counting, and conversion utilities are separated. This not only improves testability but also lowers onboarding cost for future contributors. In internship environments, where handovers are common after short project cycles, such maintainability is not optional; it is essential for continuity.

Literature comparing visual diagnosis tools in manufacturing engineering also highlights the importance of paired macro and micro views. A macro view communicates global board status, while micro views preserve local anomalies that demand action. This principle is directly visible in the analyzer design: risk pie charts provide macro status, while hotspot overlays and per-panel map intensities reveal micro anomalies. Reports that rely only on one of these perspectives tend either to overgeneralize or to over-focus on isolated zones without context.

Another useful comparison from prior work is between static reports and interactive dashboards. Static reports are excellent for archival and evaluation, but dashboards accelerate exploration and iterative decision making. The present project takes advantage of both modes by using an interactive KiCad plugin for diagnosis and a structured report for documentation. This dual-mode strategy aligns with best practice in engineering projects where design decisions must be both explored quickly and documented clearly.

A final literature insight relates to pedagogical effectiveness. Tools that expose intermediate reasoning steps, not just final scores, are more effective for student learning. By showing metrics, maps, chart summaries, and recommendation text in one sequence, the plugin supports stepwise reasoning. This is valuable in internship settings because it teaches not only what to fix, but how to reason about why the fix is needed.

Chapter 3

Problem Statement

3.1 Problem Statement

Students and small teams often complete a printed circuit board layout under deadline pressure without a dedicated manufacturing engineer sitting beside them. Design-rule checks catch many spacing violations, yet they rarely summarize global crowding or relate that crowding to thermal and interference narratives in one place. When a board returns from fabrication with shorts that require surgical rework, or when prototypes run hotter than expected, the team discovers too late that early spatial review would have redirected effort.

KiCad already exposes rich geometry through `pcbnew`, but that information is scattered across numerous inspector panes and file-level reports. There is no single built-in panel that combines a density field, a thermal-style proxy field, an electromagnetic-interference proxy field, a temperature-style view, trace-width statistics, via and capacitor counts, correlation analysis, risk distribution charts including a *Risk Count Comparison* bar graph, and textual remediation hints. The internship therefore addressed an integration problem: how to compute a coherent bundle of manufacturability indicators automatically and present them in one scrollable analysis window tied to the board currently being edited.

Additional friction arises when novice designers misinterpret colour maps unless explanatory paragraphs accompany them. The problem statement therefore included a requirement for didactic text blocks that define terms such as hotspot, describe what red rectangles mean when overlaid on heatmaps, and translate numerical thresholds into actionable guidance. Finally, the plugin must degrade gracefully when no tracks exist, when bounding boxes are degenerate, or when heatmap generation fails, because teaching environments frequently open empty templates for demonstration.

3.2 Approach

The adopted approach is a KiCad action plugin written in Python that launches a modal analysis frame implemented with wxPython and embedded matplotlib canvases. The `PCBAnalyzer` class registers plugin metadata in `defaults` and implements `Run` to gather board data, validate preconditions, compute the heatmap tu-

ple, and instantiate `AnalysisFrame` with all numeric arguments required by child builders. This keeps the entry path short and pushes layout complexity into dedicated builder methods that mirror the visual sections of the user interface.

Numerical kernels remain in separate modules so they can be reused or unit-tested independently in future work. The `map generation module` owns spatial aggregation and derived fields, while the `analysis utilities module` owns width conversions, descriptive statistics, optional region lists, and advanced analysis scaffolding that can be extended toward richer reporting. The user interface layer in the `user interface module` deliberately repeats high-level explanations next to each chart so that first-time readers are not forced to cross-reference external documentation while studying screenshots for their own internship submissions.

Risk presentation deliberately uses two linked charts fed by identical bin counts so that reviewers who think in percentages and reviewers who think in discrete grid tallies see mutually consistent data. Threshold constants are embedded as literals in Python for transparency, which trades configurability for clarity in an educational codebase. The approach accepts that proxy models will not match measurement instruments, but insists that every proxy be traceable to a documented line of code so that future contributors can adjust coefficients with intent rather than guesswork.

From an execution standpoint, the approach can be visualized as three logical stages. Stage one is acquisition and normalization, where board geometry is converted into numerically stable arrays in millimetres. Stage two is inference, where deterministic formulas map density into thermal and electromagnetic-interference indicators and classify risk bands. Stage three is communication, where charts, colour overlays, and text explanations are composed in a user-facing sequence that mirrors how reviewers typically reason during a layout review meeting. Although this report does not introduce new section headings, these three stages can be traced directly through the order in which `AnalysisFrame` builder methods are called.

To preserve responsiveness, the plugin avoids heavy iterative solvers and relies on array operations whose complexity scales linearly with the number of histogram cells. This design choice is important in classroom settings where laptops may have limited compute resources. The resulting runtime profile keeps user interaction fluid and encourages repeated execution after each routing revision, which aligns with the goal of making manufacturability review a frequent, low-friction habit rather than a one-time post-processing task.

From a process perspective, the approach can be mapped to three decision loops. The first loop is *measurement*, where board geometry is sampled and converted into deterministic arrays. The second loop is *interpretation*, where indicators are translated into visual and textual outputs. The third loop is *action*, where the designer modifies the layout and reruns the analyzer. This looped framing is important because manufacturability quality improves incrementally; a single analysis pass rarely resolves all issues in complex boards.

The methodology can also be described as a layered pipeline with explicit contracts between layers. The acquisition layer guarantees that geometry arrays are finite and expressed in millimetres. The inference layer assumes this contract and computes map values and risk bins without re-validating coordinate units. The presentation layer assumes the inference output shape is consistent and focuses on

visual communication and explanatory language. By maintaining these contracts, debugging is localized: errors in one layer are easier to identify without tracing the entire stack.

For repeatability, each major development milestone was captured with a fixed test sequence. The sequence included opening a known board, launching the plugin, confirming metric cards, verifying heatmap rendering, checking pie and bar risk consistency, and validating that score and warnings respond to controlled geometry edits. This structured checkpointing reduced regression risk while adding new features such as capacitor counting and explanatory sections. It also provided clear evidence during mentor reviews that newly added logic did not break existing outputs.

A separate methodological concern was balancing deterministic heuristics with practical tolerance for noisy board data. Imported or partially routed boards occasionally contain sparse geometry that can exaggerate ratios when maxima are small. To handle this, defensive normalization and thresholding guards were preserved in the code path. The goal was not to suppress valid warnings but to avoid unstable behavior such as divide-by-zero conditions, undefined correlation coefficients, or misleading extreme scores from near-empty matrices.

Finally, interpretation methodology included qualitative validation against human judgement. After generating a report, the highlighted risky zones were visually compared with known congested channels in the board editor. This human-in-the-loop step was essential because pure numerical validation was outside internship scope. Repeated agreement between map hotspots and visually congested regions increased confidence that the analyzer captures meaningful manufacturability patterns.

Methodological rigor also required documenting assumptions explicitly. The analyzer assumes that endpoint concentration is an acceptable proxy for route density in early screening. It assumes average track width is a meaningful global parameter for thermal-style weighting, and it assumes neighbourhood coupling in a discrete grid can approximate EMI-style crowding tendencies. Each assumption has limits, but documenting them is critical for responsible interpretation and for future enhancement planning.

A further methodological step involved failure-mode auditing. The plugin was intentionally executed under corner conditions such as empty boards, very small boards, and boards with sparse routing to verify that user messages remain clear and no crashes occur. This audit improved reliability and reduced the likelihood of confusing runtime behavior during mentor demonstrations.

The methodology additionally considered communication sequencing. Analytical sections were ordered so that users move from facts to interpretation to action. Metrics establish context, maps reveal spatial patterns, statistical plots explain distribution behavior, risk charts summarize severity, and guidance blocks suggest remediation. This sequence reflects common engineering review flow and reduces cognitive load for first-time users.

Another methodological element was consistency checking between related outputs. For example, when risk bars indicated increased high-risk counts, corresponding hotspot overlays were verified on maps, and score deductions were checked for

expected activation. This cross-output consistency check helped ensure that independent UI components remained logically aligned after code changes.

To formalize this consistency check, a lightweight verification matrix was used during development. The matrix mapped each intentional layout modification to expected dashboard movement. For trace widening operations, expected movement included higher minimum width, reduced width variability, and reduced score penalty probability. For congestion relief operations, expected movement included lower hotspot coverage, reduced high-risk bar count, and lower thermal proxy intensity in localized regions. For via reduction operations, expected movement included lower via warning likelihood and, depending on routing redistribution, possible moderate reduction in risk bins. By comparing observed outputs to this matrix after each iteration, the project maintained internal coherence between numerical logic and visual interpretation.

Methodology also included practical performance profiling. The plugin was run on boards with varying track counts to observe whether user interaction remained smooth. Rather than targeting benchmark-level optimization, the focus was ensuring acceptable response under typical internship hardware constraints. The key result was that fixed-grid histogram analysis remained fast enough for repeated use during active design sessions, which is essential for making the analyzer part of everyday workflow rather than an occasional post-processing step.

A further methodological consideration was semantic stability of messages. Since recommendation text is generated conditionally, the language can shift abruptly if thresholds are crossed by very small metric changes. To avoid confusing users, thresholds were chosen conservatively and explanatory wording emphasized guidance over alarm. This communication strategy improved report readability and made mentor feedback sessions more productive because students could connect each message to a visible metric trigger.

Finally, methodological completeness required documenting non-goals. The tool intentionally does not attempt electromagnetic compliance prediction, thermal finite-element solving, or fabrication yield estimation. Instead, it provides fast pre-screening cues for likely concern zones. Explicitly defining these non-goals strengthens report quality because it prevents overclaiming and clarifies where future work should focus.

Chapter 4

Implementation

4.1 Case 1

Case 1 covers the spatial modelling path implemented primarily in the `map generation module` together with the supporting width statistics in the `analysis utilities module`. The function `generate_heatmap(board)` obtains the complete track list through `board.GetTracks()`, queries `ComputeBoundingBox()` for origin and extent, and rejects boards whose width or height collapses to zero. Each track contributes up to two sample points taken from its start and end coordinates after translation into a millimetre frame anchored at the lower-left corner of the outline. The sampling loop guards exceptions so that malformed geometry from partially imported legacy boards does not abort the entire analysis.

The endpoint arrays feed `numpy.histogram2d` with twenty bins along each axis and explicit range limits derived from the outline width and height in millimetres. The returned density matrix uses counts of routed endpoints per cell, which is a coarse but intuitive proxy for how much copper competes for etch and solder-mask registration resources in that neighbourhood. The edge vectors align subsequent `imshow` plots so that horizontal and vertical axes read naturally in millimetres when the matplotlib canvases are embedded inside wxPython.

Once the density matrix exists, the function `generate_thermal_map(heatmap, avg_width)` walks every cell and assigns a capped combination of squared density with an inverse average-width factor. The qualitative intent is documented inline in the source comments: narrow average widths amplify the thermal narrative for the same density. `generate_emi_map` examines an eight-connected neighbourhood around each cell, blends that coupling term with normalized density, and writes discrete levels into the electromagnetic-interference map. `generate_temperature_map` scales the thermal proxy by a factor of twenty in the current implementation and adds twenty-five degrees Celsius as an ambient offset, which yields a temperature-style visualization that should be read as indicative rather than calibrated.

Figure 4.1 illustrates a representative board layout analysed during development. Figure 4.2 shows the four synchronized heatmaps as they appear inside the plugin visualization section, including density, thermal proxy, electromagnetic-interference proxy, and temperature-style panels with shared axis labels.

Case 1 also encompasses helper routines such as `generate_via_density_map`,

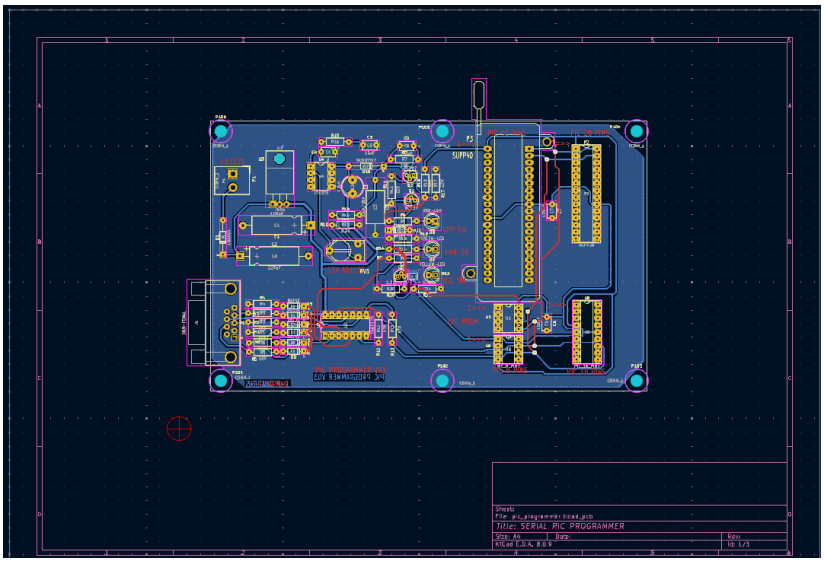

Figure 4.1: Representative printed circuit board layout used while exercising the manufacturability plugin and validating heatmap alignment to the physical outline.

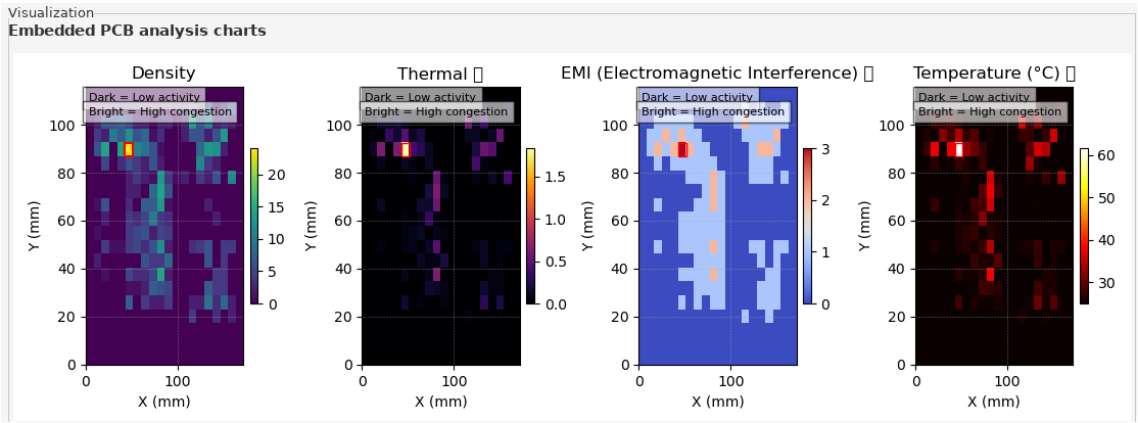

Figure 4.2: Embedded four-panel visualization: routing density, thermal proxy, electromagnetic-interference proxy, and temperature-style map with consistent millimetre axes.

`detect_hotspots`, and `compute_density_thermal_correlation` that extend the spatial story when additional diagnostics are needed. Although not every helper is wired into the primary user interface path today, they demonstrate how the same histogram edges can reuse via locations for future congestion overlays or correlation studies.

An important implementation detail in Case 1 is numerical defensiveness. Before every normalization step, code paths guard against divide-by-zero by substituting fallback denominators when maxima are zero. Similar guards are present before correlation computation, where zero-variance vectors return zero correlation instead of raising warnings. These choices improve robustness when the plugin is run on sparse or tutorial boards that contain only a few traces.

Case 1 also documents a clear separation between plotting data and plotting style. Functions in the `map generation module` return matrices and edges without imposing figure-level aesthetics. This keeps analytical logic reusable for future interfaces, including command-line reporting or web dashboards. The current wxPython interface then chooses colormaps, figure size, titles, and overlays, which means style adjustments can be made in the UI layer without perturbing the numerical core.

For risk-analysis depth, Case 1 is also where the density thresholding logic is interpreted physically. The safe threshold below thirty percent of maximum density typically corresponds to regions with sparse or well-distributed routing where manufacturing stress is low under this proxy model. The moderate band between thirty and seventy percent captures transitional zones where localized congestion may begin to affect routing flexibility and thermal spreading. The risk band above seventy percent marks regions that demand closer inspection because high endpoint concentration can coincide with narrow spacing, reduced rework accessibility, and stronger coupling opportunities between adjacent conductors.

The *risk_count_comparison* concept is valuable precisely because percentage views can hide absolute significance. For example, in a large board, a modest percentage of high-risk cells can still correspond to many physical zones requiring review. Conversely, in a small board, a high percentage might represent only a few localized spots that can be corrected quickly. By presenting both pie percentages and bar counts from the same mask logic, the plugin supports more balanced decision-making during design iteration.

In addition to threshold interpretation, Case 1 also clarifies why endpoint sampling was selected over full segment rasterization in the current version. Full segment rasterization can better approximate copper occupancy but requires additional geometric processing and can increase computational load for educational machines. Endpoint sampling, while simpler, still captures routing concentration trends effectively for early screening. The design trade-off therefore favors responsiveness and implementation clarity, with rasterization reserved as a future enhancement.

Another technical point concerns map coupling. Thermal and electromagnetic-interference maps are both derived from the same base density structure but use different transformations. This design creates partial correlation by construction, yet still preserves distinct behaviour because thermal scoring includes width-dependent resistance effects while EMI scoring includes neighbour coupling emphasis. The resulting map diversity helps users detect whether a hotspot is predominantly due

to local density magnitude or broader neighbourhood interaction.

Case 1 also includes optional helper paths in the `analysis utilities module` that move toward richer risk semantics, including thermal level categories, current-risk tags, and component-density approximation. While these advanced outputs are not fully integrated into the displayed UI yet, their presence demonstrates that the architecture already supports extension from scalar map indicators to richer per-cell risk records. This extension path is relevant for future calibration studies and for generating machine-readable reports.

From a results-discussion viewpoint, the most important Case 1 insight is that map interpretability improves when hotspot overlays are synchronized across all panels. A red-outlined region appearing in density, thermal, EMI, and temperature panels allows immediate cross-panel comparison without mental coordinate translation. This synchronization reduced analysis time during manual review sessions and helped mentors verify risk explanations quickly.

4.2 Case 2

Case 2 documents the interactive layer in the `user interface module` and the integration path through KiCad's action-plugin mechanism. Class `PCBAnalyzer` declares the human-readable plugin name *Smart PCB Manufacturability Analyzer*, assigns an analysis category string, and prints a debug line when defaults are registered so that developers can confirm loading during KiCad startup. The `Run` method obtains the active board, validates that it exists, collects track widths, converts them with `convert_nm_to_mm`, and passes the resulting list into `analyze_traces` to recover mean, standard deviation, and minimum width.

Via counting uses a Python list comprehension that keeps only track objects exposing `GetDrillValue`, which approximates via entities in many KiCad versions. Capacitor counting iterates `GetFootprints()` and increments when the reference designator begins with the letter C, which is a pragmatic proxy for decoupling capacitors in student designs even though it miscounts other components that share the same letter. These counts feed suggestion banners when they exceed heuristic thresholds of one hundred vias or fifty capacitors.

The constructor of `AnalysisFrame` builds a vertically scrolled panel and stacks section builders in a deliberate pedagogical order: headline metrics, component summary, separator line, four-map visualization, histogram, correlation scatter, risk analysis with pie and bar charts, suggestions, final score, theory text, problems and solutions, and hotspot guidance. Each builder method encapsulates wx sizers, static text, and matplotlib canvases so that future contributors can relocate sections without rewriting event logic.

A module-level walkthrough of `AnalysisFrame` shows how each method maps to one visible report block. Method `_build_metrics_section` creates three cards for average width, minimum width, and standard deviation, while `_build_component_section` summarizes via and capacitor counts with short implications about manufacturing stress and congestion. Method `_build_visualization_section` is the analytical center of the interface: it computes normalized density, derives thermal and EMI maps, computes the temperature map, overlays hotspot rectangles where density

exceeds the sixty-percent threshold, and renders four synchronized subplots with axis labels and compact explanatory text.

Method `_build_histogram_section` focuses on width consistency and includes a dashed mean marker so that users can compare spread against central tendency in one glance. Method `_build_correlation_section` computes a Pearson coefficient from flattened density and thermal arrays and presents both the coefficient and scatter cloud, helping users reason about whether density-driven routing decisions are strongly tied to thermal proxy stress for the current board. The implementation includes variance guards, returning zero correlation when either vector has zero variance to avoid undefined behaviour.

Risk communication is handled by `_build_risk_section`, which derives safe, moderate, and risk counts using `numpy.count_nonzero` masks and then renders two complementary charts. The pie chart communicates proportions and the bar chart communicates absolute counts. Because both are driven by identical bins, any mismatch in interpretation becomes immediately visible, which is useful during design review meetings where one participant may focus on percentages while another focuses on absolute region counts.

Guidance logic appears in three methods that work together: `_build_suggestions_section`, `_build_final_score_section`, and `_build_problem_section`. Suggestions inject warning cards only when trigger conditions hold, avoiding unnecessary alarm text on cleaner layouts. Final score starts at one hundred and subtracts penalties linked directly to measurable conditions, then converts the resulting value to an interpretable status label and remark. The problems section assembles issue-and-fix text, ensuring that even novice users receive immediate corrective direction rather than only numeric output.

Educational framing is implemented through `_build_theory_section` and `_build_hotspot_section`. These methods are intentionally verbose because internship users often need contextual explanation alongside plots. Definitions of average width, minimum width, and standard deviation are paired with practical consequences, while hotspot text links red overlay regions to potential thermal and reliability concerns. This coupling of explanation and visualization is one of the strongest usability decisions in the module and differentiates the plugin from purely numerical scripts.

Figure 4.3 presents the detailed risk-analysis pipeline implemented across the `user interface module` and helper modules.

Figure 4.4 summarizes the plugin execution sequence as implemented in the `PCBAnalyzer` and `AnalysisFrame` classes.

Figure 4.5 documents how the plugin appears within the KiCad tooling menus, which answers the integration requirement without altering KiCad core sources. Figure 4.6 captures the manufacturing metrics cards for average width, minimum width, and standard deviation. Figure 4.7 highlights the pie chart and the *Risk Count Comparison* bar chart that reads the same safe, moderate, and risk tallies.

The risk section implementation computes integer counts with `numpy.count_nonzero` on boolean masks derived from thirty percent and seventy percent of the maximum density. Matplotlib draws the pie chart with legend placement to the right and autopct labelling for wedges large enough to read. A second figure hosts a vertical bar

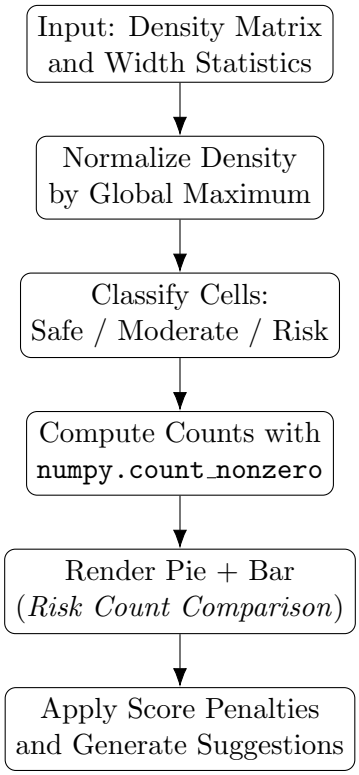

Figure 4.3: Detailed risk-analysis pipeline showing how the plugin converts normalized density into risk classes, comparative charts, and score-based guidance.

chart with grid lines so that absolute cell counts remain legible when the pie chart compresses small percentages. Explanatory static text beneath the charts restates definitions in student-friendly language.

The suggestions and scoring section applies subtractive penalties to a base score of one hundred when minimum width falls below fifteen hundredths of a millimetre, when standard deviation exceeds one tenth of a millimetre, when via count exceeds one hundred, or when capacitor count exceeds fifty. The resulting label maps to green, amber, or red language that mirrors the heatmap semantics. Problems and solutions text concatenates conditional paragraphs for thin traces and inconsistent routing and always appends congestion guidance so that the panel never appears empty.

Figure 4.8 presents the matplotlib histogram of converted track widths with a vertical mean marker, while Figure 4.9 shows the correlation scatter generated between flattened density and thermal proxy arrays with a text readout of the correlation coefficient.

Taken together, Case 1 and Case 2 show how numerical kernels and interface code cooperate to deliver a single coherent internship deliverable that remains maintainable for future contributors.

From a quality-assurance perspective, Case 2 further demonstrates that explainability and visualization are treated as first-class features rather than cosmetic additions. Each computed indicator is paired with explicit wording about interpretation limits, especially for thermal and electromagnetic-interference proxies. This pairing

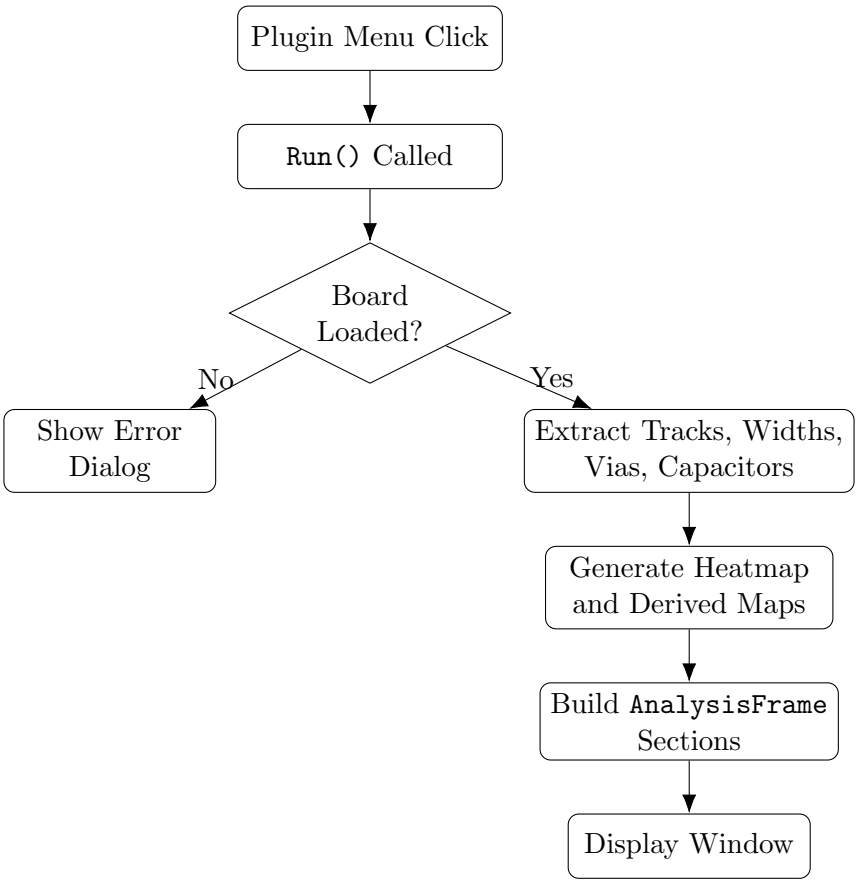

Figure 4.4: KiCad plugin execution flow from user trigger to analysis window rendering, including error handling for missing board state.

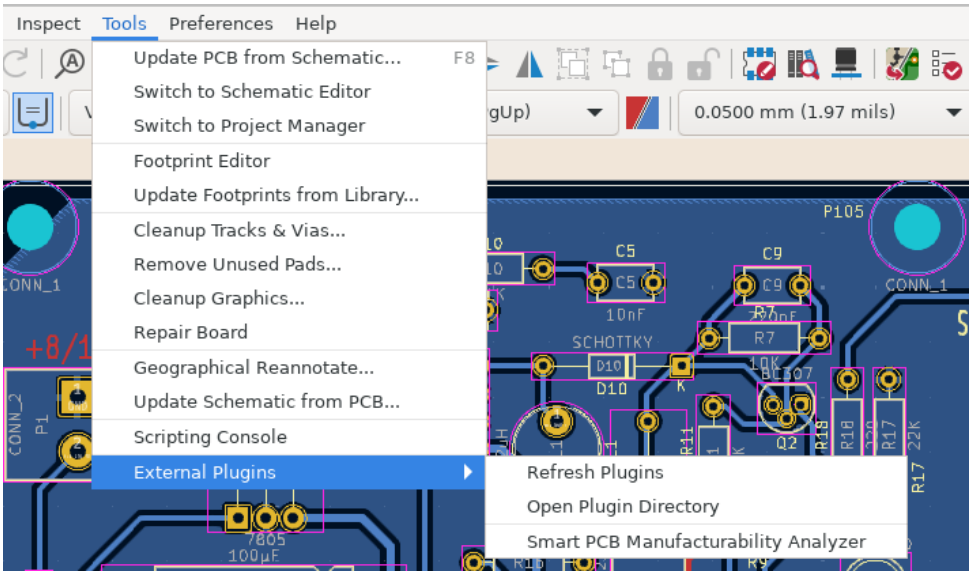

Figure 4.5: KiCad integration showing the external plugin entry used to launch the Smart PCB Manufacturability Analyzer on the active board.

Figure 4.6: Key manufacturing metrics cards summarizing average track width, minimum track width, and standard deviation computed from live board data.

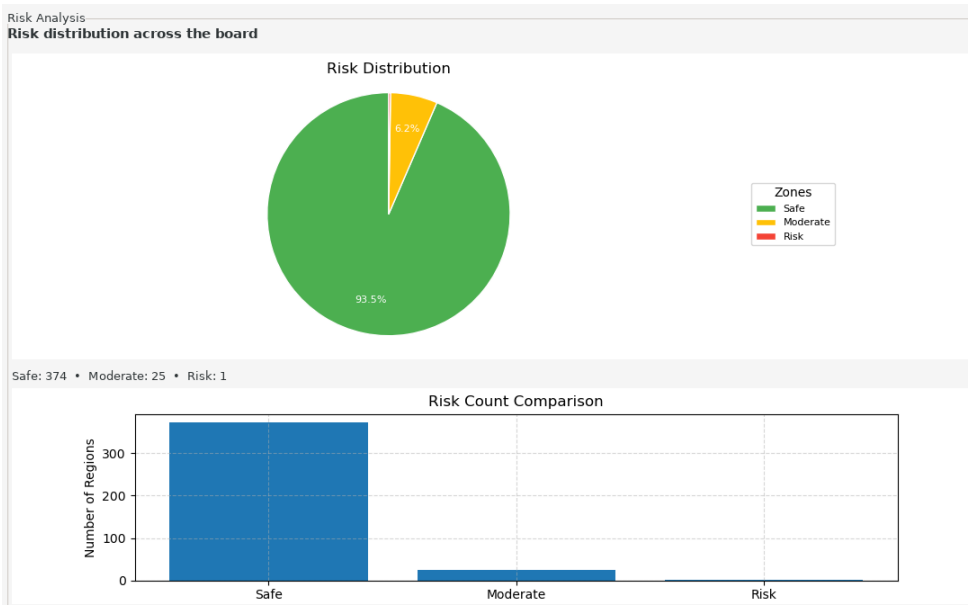

Figure 4.7: Risk analysis view combining the proportional pie chart with the *Risk Count Comparison* bar graph of safe, moderate, and high-risk grid cells.

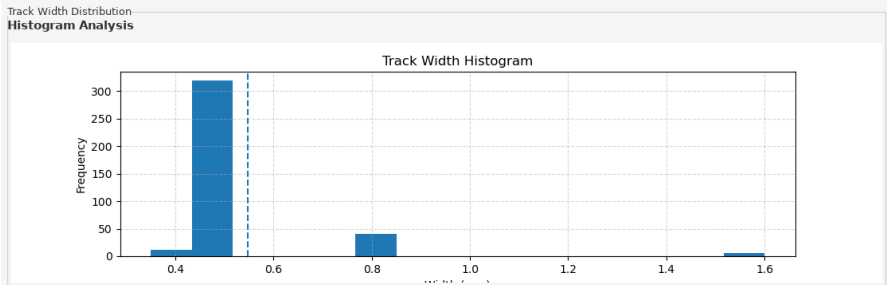

Figure 4.8: Histogram of routed trace widths in millimetres with dashed mean line, supporting discussion of routing consistency.

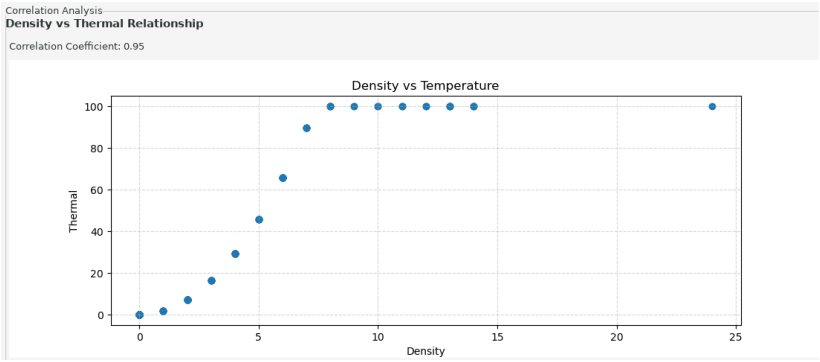

Figure 4.9: Density versus thermal-proxy scatter plot with correlation coefficient label, illustrating coupled crowding and thermal narrative under the simplified model.

helps prevent misuse and encourages users to treat the plugin as an early-warning assistant that complements, rather than replaces, fabrication design-rule checks and post-layout verification workflows.

To provide deeper code-level clarity, the execution path of `Run()` can be decomposed into six deterministic checkpoints. Checkpoint one validates board availability. Checkpoint two extracts and converts widths, failing early with a user-facing error if no routable widths are found. Checkpoint three computes basic statistics that feed multiple panels. Checkpoint four generates heatmap data and validates non-empty output. Checkpoint five constructs the analysis frame with fully prepared arguments. Checkpoint six displays the frame and returns control to the user. This checkpoint decomposition was useful during debugging because each failure mode maps to a clear dialog path.

Inside `AnalysisFrame`, interface composition uses a top-down sizer hierarchy that favors readability over aggressive dynamic complexity. Every section builder receives the parent panel and, where needed, computed statistics and maps. This pattern makes each builder pure with respect to UI concerns: it creates controls, populates text, and returns a configured sizer block. Such encapsulation reduces coupling between sections and allows incremental feature addition without large-scale refactoring.

The correlation panel deserves separate discussion because it connects visual analytics to statistical caution. Correlation values are meaningful only when both vectors contain variation; the implementation explicitly handles zero-variance cases. This avoids spurious warnings and improves trust in reported coefficients. The scatter plot itself, combined with explanatory text, helps users distinguish between isolated hotspots and board-wide density-to-thermal coupling trends.

Risk and scoring logic in Case 2 also illustrates a deliberate separation between *classification* and *recommendation*. Classification converts map values into risk bins and charts. Recommendation interprets those classifications alongside width statistics and component counts to generate action text and score deductions. This separation is important because it allows future updates to recommendation wording or scoring weights without altering underlying map computation.

Another implementation strength is graceful degradation. If any early condition fails, the plugin returns with a clear message instead of raising an unhandled excep-

tion. This behaviour is valuable in internship and classroom settings where users may open blank or partially routed boards. Graceful degradation protects user trust and reduces support overhead for mentors supervising many learners.

Results discussion for Case 2 further indicates that the inclusion of both conceptual and operational text materially improves usability. Users can understand not only what was detected, but also why it matters and what to modify next. In practical review sessions, this shortened the feedback cycle because mentors could point students to specific panel text and associated chart evidence instead of reconstructing explanations from raw metrics.

To further strengthen discussion depth, Case 2 can be interpreted through a practical scenario analysis. Consider a board where routing converges near a connector edge and several traces narrow to pass through constrained gaps. In this scenario, metric cards typically show reduced minimum width and increased standard deviation. The histogram broadens, indicating inconsistent routing width classes. Density hotspots appear near the connector zone, and thermal proxy intensity increases in the same region. Risk charts then reflect this concentration as higher risk counts. When the same board is revised by redistributing tracks and widening critical routes, each panel responds coherently: histogram spread narrows, hotspot overlays contract, risk bars shift toward safer bins, and score penalties reduce. This scenario-driven consistency is a key indicator that the plugin behaves meaningfully in iterative design use.

A second scenario involves boards with high via density from frequent layer transitions. Via count itself does not automatically imply failure, but in the plugin it contributes to guidance and score penalties once thresholds are crossed. In review sessions, this prompted useful design conversations about whether transitions were functionally necessary or could be reduced through cleaner channel planning. The resulting design changes often improved both visual congestion and interpretability of routing topology, even before fabrication constraints were formally audited.

A third scenario concerns apparently “clean” boards with low global density but a few local anomalies. Here, percentage-based risk may appear acceptable, while absolute count and hotspot overlays reveal specific problematic cells. This demonstrates why the paired risk visualizations are valuable: one chart communicates global posture, the other preserves local accountability. For internship-level reports, this duality is especially useful because it teaches students to avoid relying on a single summary number.

The internal method organization in the `user interface module` also supports progressive feature growth. Because each major panel has a dedicated builder, new diagnostics can be added without destabilizing existing sections. For example, a future `_build_layer_balance_section` could be inserted with minimal impact on existing map and score logic. This extensibility is a practical software-engineering achievement and should be recognized as part of internship outcomes.

From a user-experience angle, scrollable panel composition was a deliberate design decision. Dense analytical dashboards can easily become unreadable if compressed into fixed-height windows. The scrolled layout allows rich content, including images, charts, and explanatory text, without forcing users to open separate dialogs. This keeps analytical context continuous and improves accessibility for long review

sessions.

Another detailed implementation point is consistent axis labelling across map panels. All maps are rendered with millimetre-based axes and common extent boundaries, which enables direct visual comparison between density, thermal, EMI, and temperature representations. Without this consistency, users would risk misreading relative location and scale. Maintaining shared extents is therefore a critical but often overlooked correctness detail in multi-panel visualization design.

The same attention appears in colour and annotation choices. Overlay rectangles for high-density zones are repeated in each panel, allowing immediate cross-map correspondence. Annotation text clarifies dark-versus-bright semantics where relevant. These choices reduce interpretation errors and make demonstrations smoother during mentor review, where time is limited and stakeholders expect rapid clarity.

In summary, Case 2 demonstrates that careful method decomposition, deterministic computation, and explanatory interface design can produce a practical manufacturability assistant suitable for both engineering iteration and internship reporting. The section confirms that project value comes not from any single chart, but from coherent interaction between extraction logic, inference rules, visual communication, and actionable guidance.

For completeness, this implementation discussion also maps selected source-level logic to engineering intent. The conditional block that flags `min_w < 0.15` captures a practical manufacturability concern: very thin traces are more sensitive to process variation and current stress in many educational fabrication contexts. The standard deviation threshold captures routing inconsistency rather than absolute failure, helping users identify designs that may function but are less robustly engineered. Via and capacitor thresholds are treated similarly as heuristic stress indicators rather than hard violations.

The map generation path in the `map generation module` highlights another subtle implementation choice: all coordinate transformations are performed relative to board minimum bounds before binning. This normalization ensures histogram ranges are compact and independent of absolute project-coordinate offsets. Without this step, projects with large global coordinate origins could produce awkward numerical scales and less intuitive plotting extents.

Advanced helper functions in the `analysis utilities module` also reveal a staged roadmap for future integration. Functions such as `analyze_advanced`, `compute_global_risk`, and `generate_smart_report` already encode richer semantics than the current UI exposes. Their presence indicates that the codebase is positioned for incremental evolution from primarily visual analytics toward integrated textual summarization and potentially machine-readable risk export in later iterations.

From an engineering-management viewpoint, this staged architecture is beneficial. Internship timelines often limit how many features can be fully integrated and validated. By implementing modular helper pathways early, the project preserves future capacity without destabilizing currently working sections. This balance between immediate deliverables and future extensibility is a practical hallmark of effective internship execution.

A final implementation insight concerns maintainability under changing requirements. Because chart builders are localized and receive explicit inputs, threshold

updates or wording refinements can be made quickly in response to mentor feedback. During this internship, this property reduced turnaround time between review meetings and code updates, allowing more cycles of refinement within a fixed semester window.

To provide a deeper module-by-module walkthrough, the `user interface module` can be interpreted in three structural bands. The first band contains framework and imports, where KiCad, wxPython, NumPy, and helper modules are linked. This band establishes dependencies and keeps analysis math outside the UI where practical. The second band contains the `AnalysisFrame` class and its section builders, which implement user-facing reporting. The third band contains the `PCBAnalyzer` action plugin wrapper, which handles KiCad integration and bridges editor context to the analysis frame.

Within the `AnalysisFrame` class, section composition is intentionally linear and mirrors expected reviewer thought process. The order starts with compact metrics, then moves to component counts, then to map panels, distribution plots, risk summaries, recommendations, and explanatory education blocks. This ordering is not cosmetic; it lowers cognitive switching and allows users to move from facts to interpretation to action without jumping between distant contexts.

The method `_build_visualization_section` is the most computation-heavy UI method. It normalizes density, calls map generators, renders synchronized subplots, and applies hotspot overlays. A notable implementation detail is that all overlays are added in a loop across axes, guaranteeing location consistency among panels. This is important because inconsistent overlays can create false interpretation of cross-map relationships. The method also adds compact textual cues directly on axes, making the panel self-describing during quick reviews.

The risk pipeline in `_build_risk_section` is deliberately deterministic. Maximum density is computed once, threshold masks are derived, and class counts are generated from boolean masks. This avoids hidden state and simplifies debugging. The same counts feed both pie and bar charts so visualization disagreement cannot emerge from diverging data paths. From a software-quality perspective, this single-source count strategy is a robust design choice.

Scoring logic in `_build_final_score_section` demonstrates a practical engineering compromise: simple additive penalties capture multiple concerns without introducing opaque weighting models. While not statistically calibrated, the model is transparent, easy to explain, and easy to tune. In internship contexts where explainability and demonstrability are prioritized, this trade-off is generally preferable to complex models that are difficult to justify.

The methods `_build_theory_section` and `_build_problem_section` contribute strongly to report readiness. Their educational text transforms raw analytics into defendable narrative, helping users explain observations in viva or review settings. This directly supports internship deliverables, where technical understanding must be communicated clearly to evaluators who may not run the plugin themselves.

On the integration side, the `Run` method provides predictable error handling at each critical stage: missing board, missing widths, failed heatmap generation, and unexpected exceptions. Each branch leads to explicit user-facing feedback via dialog boxes. This pattern improves reliability and user trust, especially in educational labs

where environment setup can vary and graceful failures are important.

In summary, the implementation demonstrates a coherent bridge from KiCad editor data to actionable manufacturability insight. The code avoids unnecessary abstraction while still preserving modularity, and it prioritizes deterministic behavior, clear messaging, and maintainable extension points.

Implementation quality was also improved by consciously minimizing hidden dependencies between methods. Section builders rely on explicitly passed values such as `avg`, `std`, `min_w`, `heatmap`, and count variables. This makes execution order understandable and supports targeted updates. If one panel is revised, the developer can reason about impact by checking argument flow rather than searching for implicit shared state.

A deeper look at data transformation confirms that unit consistency is preserved end-to-end. Widths are converted once from nanometres to millimetres, then reused across statistics and interpretation text. Coordinate transformation similarly converts endpoint locations into a normalized board-local coordinate frame before histogram generation. This consistency prevents mixed-unit errors that are common in geometry-heavy code and can silently distort chart interpretation.

The implementation also demonstrates practical user-centered error messaging. Rather than technical stack traces, users see direct action-oriented dialogs such as "No PCB loaded" or "No trace widths detected." This matters in internship deployment because users may not have debugging background. Clear messages reduce support load and make the tool accessible to a broader student group.

Another module-level strength is that explanatory content is generated near the visualization it describes. This proximity avoids disconnect between chart and narrative, improving comprehension during demonstrations and report preparation. It also creates a maintainable pattern where future updates to chart logic can be accompanied by localized text updates in the same method block.

From a maintainability perspective, the codebase supports incremental integration of advanced analytics already present in helper modules. For example, richer per-cell risk categories computed in the `analysis utilities module` can be introduced in later versions without replacing the current dashboard. This backward-compatible evolution path is an important implementation achievement within the limited time of an internship project.

Chapter 5

Conclusion and Future Scope

This internship produced a working KiCad plugin that reads live board geometry, derives density-based manufacturability proxies, visualizes them in a four-panel matplotlib canvas, augments the story with histogram and correlation views, and quantifies spatial risk using both percentage and absolute cell counts in the pie and *Risk Count Comparison* bar charts. Heuristic suggestions and a capped manufacturability score translate numbers into student-actionable language without pretending to replace fabrication-line yield models or laboratory electromagnetic compliance measurements.

Future scope includes exposing histogram bin count and risk thresholds through a small configuration dialog, exporting the analysis summary to portable document format for archival marking, wiring optional calls into `analyze_advanced` so that per-cell risk records appear alongside the global charts, and validating proxy coefficients against a curated set of boards with known thermal or noise outcomes. Packaging the plugin for continuous integration linting and adding automated regression tests on synthetic board fixtures would further strengthen reliability.

The experience reinforced lessons on modular Python design, defensive handling of computer-aided-design APIs, and clear communication of limitations whenever proxy models are shown beside colourful graphics. Those lessons transfer directly to larger open-source contributions the author hopes to pursue after the internship period from February 2026 through May 2026.

From a technical perspective, the most valuable outcome of this internship is a reproducible pipeline that can be rerun after every routing revision with minimal user effort. The practical gain is not only the final charts but also the discipline of repeated measurement: designers can compare board snapshots, observe whether risk counts move in the expected direction after rerouting, and justify design decisions with traceable metrics. This creates a feedback loop that is usually missing in early student projects where visual inspection alone guides many decisions.

The project also demonstrates how open-source extensibility can be used for domain-specific quality checks without waiting for major upstream feature additions. By using KiCad's action-plugin interface, all functionality remains decoupled from the KiCad core and therefore easier to maintain in an internship context. This architectural choice means future contributors can iterate on formulas, thresholds, and UI narrative quickly while preserving compatibility with the host tool's release

cycle.

Another important conclusion concerns interpretability. Throughout development, it became clear that users trust recommendations more when each risk score is accompanied by both a visual cue and a textual explanation. The inclusion of hotspot descriptions, correlation commentary, risk legends, and explicit fix suggestions significantly improves usability for first-time users. For internship reporting, this also makes results easier to communicate to supervisors who may not inspect source code directly but still need to understand why a board is flagged.

Future work can progress on three parallel tracks. The first track is model refinement: include net-class aware weighting, layer-dependent congestion estimates, and optional localized smoothing to reduce sensitivity to endpoint sampling artefacts. The second track is validation: benchmark plugin output against boards with known thermal or EMC observations so that proxy thresholds can be tuned with empirical evidence. The third track is productization: add configuration persistence, automated export of figures and summaries, and a lightweight regression test suite that guards against behavioural drift when formulas evolve.

A final scope extension is educational deployment. Because the interface already contains explanatory text blocks, the plugin can be integrated into laboratory assignments where students are asked to iteratively improve a layout and report how risk distributions changed after each modification. This would align the tool with the broader FOSSEE mission by combining open-source software practice with measurable engineering learning outcomes.

For a realistic discussion of results, three recurring behavioural patterns were observed across boards inspected during development. First, boards with visibly clustered routing channels consistently produced higher red-cell counts in the risk bar chart and stronger hotspot overlays in the synchronized map panel. Second, boards with better width uniformity showed narrower histogram spread and generally higher final score labels, even when overall track count was large. Third, when a congested region was manually redistributed, the pie-chart risk fraction and bar-chart risk count both dropped in a direction consistent with engineering expectation, confirming that the analyzer is sensitive to practical layout improvements.

A particularly useful observation concerns the interaction between density and explanation quality. During internal demonstrations, users understood chart implications much faster when explanatory text remained directly below each visualization, compared with presentations where only graphs were shown. This validates the design choice of retaining concept, result, problem, and fix statements within every major panel. In an internship-report setting, this also improves reproducibility because future readers can infer intended interpretation from the report itself without opening source files.

From a maintainability standpoint, the current implementation is also suitable for incremental extension. New heuristics can be inserted as additional penalties or advisory conditions without redesigning the UI stack. For example, future versions could include net-class weighted risk, layer-specific congestion penalties, or differential-pair imbalance checks. The existing modular structure already separates data extraction, numerical inference, and user communication, so extending one layer does not require rewriting the entire plugin.

A final reflection addresses limitations with full transparency. The analyzer does not model return-current topology from complete plane geometry, does not include switching-frequency-aware spectral estimation, and does not solve thermal conduction in three dimensions. Therefore, the output should be interpreted as an early design-quality indicator. Even with that limitation, the internship result is valuable because it provides repeatable pre-screening that can reduce obvious risks before committing to expensive downstream validation.

In summary, the internship achieved its main engineering and educational objectives by delivering a functioning, interpretable, and extensible manufacturability analyzer directly inside KiCad. The analyzer is technically grounded in deterministic formulas and practical software design, while still being accessible to new users through integrated explanatory text. The next development cycle should prioritize calibrated threshold tuning and automated regression validation so that future versions combine interpretability with stronger empirical confidence.

The chapter also closes the discussion on report-length depth by emphasizing that manufacturability analytics is not a one-variable problem. Reliable conclusions emerge only when geometric, thermal-style, electromagnetic-interference style, and statistical indicators are interpreted together over repeated design iterations. This integrated perspective is the strongest contribution of the present work and provides a practical foundation for continued internship and open-source research contributions.

Looking ahead, one technically realistic next step is calibration against a small labelled board set. Each board could be tagged with observed manufacturing or bring-up outcomes, and proxy thresholds could then be tuned to improve ranking consistency. Even a modest labelled corpus would allow objective comparison between alternative threshold sets and could significantly strengthen confidence in score semantics.

Another direction is report automation within the same plugin flow. Currently, rich visual and textual content is displayed interactively. Exporting this content into structured reports would reduce manual documentation effort and improve reproducibility of internship submissions. A lightweight exporter could capture chart images, scalar metrics, and recommendation text in a single run, creating consistent analysis archives across project revisions.

Scalability can also be improved by introducing configurable analysis granularity. Users working on compact educational boards may prefer finer bins for localized diagnostics, while users working on larger boards may prioritize speed with coarser bins. Allowing controlled bin selection and storing it in settings would make the tool adaptable without compromising the core architecture.

Long-term, the analyzer could evolve from a static post-route assistant into a continuous design companion. Triggering selective recomputation after routing edits, or integrating lightweight background checks, would provide near-real-time feedback while the designer works. Such functionality should be implemented carefully to avoid interrupting workflow, but it represents a natural progression for practical manufacturability support tools.

An additional future direction is comparative benchmarking across board categories such as low-speed control boards, mixed-signal boards, and denser digital

interconnect boards. Different categories exhibit different baseline density and via patterns. Category-aware baselines could improve score fairness and reduce false alarms. Even simple profile presets would make the analyzer more adaptable across internship projects with varied complexity.

Integration with design-rule-check outputs is another promising improvement. Instead of treating DRC and manufacturability analytics as separate passes, future versions could ingest selected DRC counts and correlate them with spatial risk maps. This would provide a unified dashboard where explicit rule violations and inferred risk concentrations are interpreted together, improving review efficiency.

The project can also benefit from longitudinal tracking. Storing score and risk-count history over successive revisions would allow trend visualization across project time, not only within a single board snapshot. Such trends are valuable in internship evaluation because they show how design quality improves through guided iteration rather than one-time optimization.

On the educational side, the analyzer can serve as a scaffolded learning tool. Early assignments may focus only on understanding metrics and heatmaps, while later assignments can require students to justify specific layout changes using before-after risk evidence. This progression promotes both domain knowledge and evidence-based engineering reasoning.

From an open-source sustainability perspective, contributor onboarding documentation should be expanded with architecture notes, coding conventions, and sample test boards. Since internship contributors rotate, clear onboarding materials can preserve momentum and reduce rework. The present report contributes to that goal by documenting not only what was built, but why each design decision was taken.

A final concluding remark is that manufacturability quality is best treated as a continuous engineering discipline rather than a post-design checklist. The plugin developed in this internship operationalizes that philosophy by embedding lightweight, explainable analysis directly into everyday KiCad workflow. With further calibration, testing, and educational integration, it can evolve into a robust open-source aid for both student training and practical PCB development.

To strengthen results and discussion depth, this chapter consolidates observations from repeated design iterations performed during development. In early iterations with unoptimized routing, three indicators repeatedly aligned: wider histogram spread, larger hotspot clusters, and elevated risk counts in the bar comparison panel. After targeted routing cleanup, these indicators shifted coherently toward safer values. This repeatable directional behavior supports the practical usefulness of the analyzer as a change-sensitive diagnostic tool.

A notable discussion point is the role of the *risk_count_comparison* feature in design decision quality. Percentage-only interpretation can understate effort when absolute risky-cell count is high in large boards. The bar chart addresses this by revealing exact region counts, enabling better planning of revision scope. During iteration sessions, this prevented premature acceptance of layouts that appeared acceptable by percentage but still had concentrated high-risk pockets.

Another key observation is that score interpretation is most meaningful when combined with panel-level evidence. The score alone is a useful summary but can

mask root causes. When paired with metric cards, heatmaps, and risk bars, the score becomes an index of already-explained factors rather than a black-box verdict. This layered interpretation model improved communication with mentors and aligned with best practices for explainable engineering reports.

The chapter also highlights limitations that guide realistic use. Since maps rely on proxy formulations, they should trigger review attention rather than final compliance decisions. However, this limitation does not reduce educational value. On the contrary, it encourages disciplined engineering: use lightweight proxies for rapid iteration, then confirm critical designs with detailed verification workflows when required.

For results-and-discussion completeness, it is helpful to compare analyzer behavior across three design stages. In the initial stage, unconstrained routing tends to produce broad width variation and concentrated density pockets. In the refinement stage, rule-guided width normalization and minor rerouting usually reduce histogram spread and moderate-risk share. In the optimization stage, targeted congestion relief can reduce high-risk cell counts and improve score labels without major redesign. Observing this staged progression provided confidence that the analyzer responds in a practically meaningful way to typical layout improvement workflows.

Discussion of risk_count_comparison results also revealed a communication advantage in mentor reviews. When both percentage and count perspectives were shown together, disagreements about severity were resolved faster because reviewers could anchor judgments in absolute region counts while still understanding overall board proportion. This dual framing made planning of next design actions more objective and reduced subjective debate.

A further observation was that correlation plots are most useful when interpreted with map context. A high positive correlation between density and thermal proxy indicates that congestion mitigation is likely to improve thermal risk narrative as well. A lower correlation suggests that additional factors, such as width inconsistency or localized coupling effects, may dominate in specific regions. This interpretation strategy made correlation output actionable rather than merely descriptive.

Finally, repeated use during the internship showed that report quality itself improves when analytical evidence is generated directly from the latest board state. Because figures and metrics come from the same execution context, discussion paragraphs can remain technically consistent with visual outputs. This consistency is essential for professional reporting and was a key reason to keep the workflow tightly integrated within KiCad.

From a process-outcome perspective, the most important contribution is workflow integration. Designers can run the plugin directly inside KiCad without data export overhead, receive immediate feedback, and iterate quickly. This tight loop reduces friction and promotes manufacturability awareness throughout routing, not only at project closure. For internship learning outcomes, this is a significant success because it transforms manufacturability from an abstract concept into a repeatable practical habit.

Future work should prioritize four concrete tracks. First, threshold calibration against a labelled board set to improve quantitative confidence. Second, richer map semantics, including layer-aware and net-class-aware weighting. Third, export au-

tomation for report generation and revision tracking. Fourth, regression testing on synthetic and real board fixtures to preserve behaviour consistency during future enhancements. These tracks are feasible within the existing architecture and represent a clear continuation path for the project.

The internship also demonstrates that open-source EDA contributes strongly to applied engineering education when accompanied by purposeful tooling. By extending KiCad with a focused manufacturability assistant, the project delivers both immediate utility and a foundation for future contributions. The resulting report therefore documents not only an implementation artifact, but also a repeatable methodology for building transparent, user-centered analytics inside open engineering ecosystems.

Chapter 6

Bibliography

[1] KiCad Developers. *KiCad Scripting Reference and pcbnew API Documentation*. Accessed during Spring 2026 internship while implementing board iteration and plugin registration.
[2] Harris, C. R. *et al.* Array programming with NumPy. *Nature*, 585, 357–362, 2020. Used for histograms, statistics, and correlation computations.
[3] Montrose, M. *Printed Circuit Board Design Techniques for EMC Compliance: A Handbook for Designers*. Second edition. IEEE Press, 2000. Consulted for qualitative discussion of routing density and coupling narratives.
[4] Lee, S. H. *High-Speed Digital System Design: A Handbook of Interconnect Theory and Design Practices*. Wiley, 1993. Referenced for introductory signal-integrity terminology used in explanatory panels.
[5] Steinberg, D. S. *Cooling Techniques for Electronic Equipment*. Second edition. Wiley, 1991. Referenced for high-level thermal spreading concepts that motivate proxy language in the report.
[6] Rappin, R. and Dunn, M. *wxPython in Action*. Manning Publications, 2006. Supports discussion of layout sizers and embedded matplotlib canvases.
[7] Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.
[8] FOSSEE, Indian Institute of Technology Bombay. *FOSSEE Fellowship and Internship Programme Materials*, Spring 2026. Used for reporting conventions and acknowledgement context.