Chess Training Suite

Design and Implementation of a Modular Desktop Application

Prepared by: **Luis Armando Macías Moto**

Technical Documentation Report
Published: December 2025

This report documents the architecture, engineering decisions, and practical value of a modular chess training system developed as part of a professional software engineering portfolio.

Índice

1. Executive Summary

1.1 Project Context

The **Chess Training Suite** was developed to address a practical challenge faced by competitive players: the fragmentation of training tools across multiple applications.

Instead of relying on disconnected platforms for tactical exercises, analysis boards, rating projections, and study management, this project consolidates these needs into a unified desktop environment.

1.2 Objective

The goal was to design a modular and maintainable application capable of supporting structured chess improvement through specialized training modules.

The project emphasizes software architecture, usability, and performance, ensuring that each module can operate independently while contributing to a cohesive ecosystem.

1.3 Core Value

This system demonstrates how engineering principles can be applied to educational and performance-oriented software.

Its value lies not only in functionality, but in its architectural design, process isolation, and adaptability for future expansion.

2. Technical Architecture

2.1 Design Strategy

The application follows a modular architecture based on process isolation and responsibility separation.

1. **Launcher-Oriented Control:** A central application acts as the access point for all modules.
2. **Independent Execution:** Each training tool runs as a separate process.
3. **Shared Logic Components:** Core chess rules and validation systems are centralized.

This structure improves maintainability, reduces coupling, and prevents module failures from affecting the full system.

2.2 Technology Stack

- **Python:** Core programming language.
- **Tkinter / CustomTkinter:** User interface development.
- **Subprocess:** Independent module execution.
- **JSON:** Lightweight persistence layer.
- **Stockfish Engine:** Advanced chess analysis integration.

3. Functional Modules

3.1 Analysis Board

Provides manual position setup, move validation, and FEN generation for exporting and studying custom scenarios.

3.2 Rating Calculator

Implements the expected-score model used in competitive chess systems:

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}$$

This enables performance forecasting and tournament projection analysis.

3.3 Tactical Training

Includes modules for mate recognition, visualization speed, and decision-making under pressure.

These tools focus on repeatable practice and measurable progress.

4. Engineering Decisions

4.1 Process Isolation

A critical design decision was the use of child processes instead of a monolithic runtime.

This ensures that each module consumes resources independently and protects the launcher from crashes.

4.2 Native Logic Development

Instead of outsourcing all functionality to external libraries, core chess validation was implemented internally.

This approach provided greater control over rules, extensibility, and educational transparency.

4.3 Standards Compatibility

Support for FEN notation guarantees interoperability with industry-standard chess tools and engines.

5. Implementation Example

Independent Module Execution

The following snippet demonstrates the system's process-launching mechanism:

```
import subprocess
import sys

def open_module(self, module_name):
    try:
        subprocess.Popen([sys.executable, f"{module_name}.py"])
    except Exception as e:
        print(f"Error␣launching␣{module_name}:␣{e}")
```

This implementation allows each module to function autonomously, improving resilience and scalability.

6. Product Value and Future Development

6.1 Practical Impact

The suite transforms fragmented study workflows into a centralized training environment. This reduces context switching and enables structured improvement for serious players.

6.2 Expansion Opportunities

Future development may include:

- Cloud synchronization for training history
- Multiplayer sparring environments
- Machine-learning assisted tactical recommendations
- Advanced performance analytics dashboards

6.3 Portfolio Relevance

This project reflects applied software engineering skills in architecture, modularity, system integration, and domain-focused problem solving.

End of Report