*“How to Monitor and Control Project Execution”* by Wysocki (Chapter 7, *Effective Project Management*).

# Summary of the Reading

This chapter outlines **practical strategies for ensuring project execution stays aligned with the plan**. It emphasizes that **monitoring and control** are continuous, adaptive processes essential for delivering successful outcomes.

## Core Concepts

| Concept | Description |
|---|---|
| **Monitoring** | Collecting real-time data on scope, cost, schedule, and quality throughout project execution. |
| **Controlling** | Making adjustments based on data — reassigning tasks, updating estimates, or revising scope to stay on track. |
| **Earned Value Management (EVM)** | Combines cost, schedule, and work progress into unified performance indicators (e.g., SPI, CPI). |
| **Issue Management** | Logging and resolving deviations, bugs, risks, or resource issues systematically. |
| **Change Control** | Formal process to evaluate and approve/reject changes to scope, timeline, or features. |
| **Status Reporting** | Regular updates to stakeholders for transparency, risk communication, and alignment. |

# Relevance & Application to TTrack

| Category | Reading Concept | TTrack Implementation |
|---|---|---|
| **Execution Monitoring** | Track progress continuously using real metrics. | You monitored delivery checkpoints (e.g. .app/.exe, sample file loading, curriculum matching logic). Manual logs or Git commits acted as informal metrics. |
| **Control Process** | Actively adjust plan based on feedback or missed deadlines. | You adapted the UI to support macOS and Windows separately, and redefined logic flow when real data revealed parsing issues. |
| **Change Control** | Evaluate if a new request is feasible, then document it. | Mid-project, you introduced features like downloadable templates and preloaded samples — this added value while requiring refactoring. |
| **Issue Tracking** | Formal issue logs help track blockers. | You managed issues informally (e.g., file parsing bugs, compatibility quirks), but could benefit from a formal system like GitHub Issues or Notion. |
| **Earned Value Thinking** | Compare effort spent vs. progress achieved. | While not formally using EVM, you noticed time/resource drift (e.g., UI took longer than expected) and adjusted scope accordingly. |
| **Status Updates** | Regular, clear progress reports for stakeholders. | You shared updates with Dr. Atif and offered demo builds proactively — aligning with Wysocki's emphasis on stakeholder visibility. |

# Recommendations to Level Up Project Control in TTrack

| Area | Suggestion |
|---|---|
| **Monitoring** | Introduce a lightweight burndown chart or checklist to track remaining tasks per milestone. |
| **Issue Management** | Create a structured issue board (GitHub Projects, Notion) with tags like Bug, Feature, Blocker. |
| **EVM-lite** | Track time estimates vs. actual effort per feature — even manually — to better predict future workloads. |
| **Status Reporting** | Prepare a simple one-pager per week with: “What’s Done”, “What’s Next”, “Risks/Blockers” for meetings or portfolio. |
| **Change Control** | Create a log for new feature requests with status, decision (accepted/rejected), and rationale. |

## Final Thought

Wysocki’s framework is highly applicable to TTrack. You’re already intuitively using several of his strategies (adaptive iteration, scope adjustment, transparent demos). With minimal overhead, formalizing your **monitoring**, **issue tracking**, and **status communication** will:

- Increase your control over delivery velocity,
- Make stakeholder collaboration smoother,
- Strengthen your documentation for future assessment, showcasing solid **project management maturity**.