# Summary – Chapter 7: Software Project Execution Control

**Source**: Chemuturi, M., & Cagley, T. (2010). *Mastering Software Project Management*

## 🎯 Purpose of the Chapter

To teach project managers **how to track, measure, and control project execution**, ensuring alignment between **planned outcomes and actual achievements**.

---

## 🔑 Key Concepts

| Concept | Description |
|---|---|
| **Planned vs. Actual Comparison** | Track tasks using metrics such as time, effort, cost, and deliverables. Constant comparison allows corrective actions. |
| **Monitoring Dimensions** | Includes schedule tracking, effort tracking, deliverables tracking, and quality tracking. |
| **Schedule Adherence** | Use tools like Gantt charts and milestone plans to visualize progress. |
| **Effort Tracking** | Track team effort in hours or story points — deviations indicate inefficiency or overrun. |
| **Deliverable Tracking** | Check if deliverables meet requirements — not just delivered but *validated*. |

| | |
|---|---|
| **Quality Tracking** | Monitor number and severity of defects; trends reveal code quality and testing effectiveness. |
| **Early Warning Signs** | Slippage in schedule, burn rate anomalies, and frequent change requests are early indicators of risk. |
| **Corrective Measures** | Can include resource reallocation, task reassignment, process improvement, or even scope revision. |
| **Change Management** | Formal change control helps isolate scope creep and evaluate change impact clearly. |
| **Status Reporting** | Standard, frequent reports to all stakeholders build transparency and shared understanding. |

## Connection to TTrack

| Area | What Chemuturi Suggests | How TTrack Applies |
|---|---|---|
| **Schedule Tracking** | Use milestone charts, track completion vs. expected dates | You've tracked key delivery points (e.g., .exe / .app builds, GUI load, transcript parsing). Consider using a Gantt chart or Kanban tool like Trello/Notion for visual feedback. |
| **Effort Tracking** | Record time spent vs. time planned per task | You may not log hours formally, but you've recognized which features (e.g., macOS packaging) took longer — tracking this could help in future estimation. |

| | | |
|---|---|---|
| **Deliverables Tracking** | Ensure actual output meets acceptance criteria | You test your .exe/.app builds, validate them with Dr. Atif, and include preloaded sample files — that's solid deliverables tracking. |
| **Quality Metrics** | Monitor bugs, rework, user feedback | You've adjusted based on feedback from stakeholders (e.g., parsing logic, UX tweaks). Could add a bug/issue tracker using GitHub or Notion. |
| **Change Control** | Evaluate each scope change for its impact | You've integrated features like sample templates mid-development. Writing down the reason/impact per change would align you with this best practice. |
| **Early Warning** | Catch issues before they grow — e.g., delayed features, repeated fixes | You noticed early UI/packaging blockers and adjusted scope and design priorities to meet academic timelines. |
| **Status Reporting** | Provide structured updates with "What's Done, What's Next, Risks" | You've sent updates to Dr. Atif (and could formalize this into a weekly update doc or Google Sheet for better transparency). |

# Suggested Additions to TTrack Based on the Chapter

| Area | Action |
|---|---|
| **Progress Control** | Use a lightweight task tracker (e.g. Notion board with columns: To Do / In Progress / Done) |

| | |
|---|---|
| **Time Logging** | Even rough logs (e.g. “2 hours on packaging issues”) can help with retrospective analysis |
| **Deliverables Matrix** | Track each key feature (Upload, Match Engine, Results Page) with status and owner |
| **Issue Register** | Start a Google Sheet or GitHub Issue tracker for bugs or blockers |
| **Weekly Report** | 1-pager or Slack/Notion update each week: “Progress”, “Next Steps”, “Blockers”, “Help Needed” |

## ✍️ Final Thought

This chapter echoes what you’re already doing intuitively with **TTrack**, especially in being hands-on, iterative, and transparent. But it also shows how adding **simple structure (dashboards, logs, change notes)** can boost professionalism and prepare you for **larger-scale, stakeholder-driven software projects**.