Critical path method explained: find what really sets your delivery date

20 November 2026 · 6 min read · blogs.iictp.org/critical-path-method-explained

Most projects contain a few chains of work that decide the end date and many tasks that have room to slip. The critical path method finds the chain with no room. Knowing it tells you where a day lost is a day lost from the whole project, and where it is not.

TL;DR The short version

- The critical path is the longest chain of dependent tasks, and its length is the shortest the project can take.
- List tasks, durations and dependencies to find it; a day lost on it is a day lost from the project.
- Focus attention on critical tasks, shorten the path by splitting or parallelising work, and add one buffer at the end rather than padding every task.

Read online, listen to it, or change the text size: https://blogs.iictp.org/critical-path-method-explained

The idea in one sentence

The critical path is the longest chain of dependent tasks from start to finish; its length is the shortest possible duration of the project.

How to find it

1. List the tasks and estimate each one’s duration.
2. Record dependencies: which task must finish before another can start.
3. **Forward pass:** working from the start, each task’s earliest start is the latest earliest-finish of the tasks it depends on.
4. **Backward pass:** working from the end, each task’s latest finish is the earliest latest-start of the tasks that depend on it.
5. **Float** (or slack) is latest start minus earliest start. Tasks with zero float form the critical path.

A worked example

- A. Design API: 3 days, no dependency.
- B. Build backend: 5 days, after A.
- C. Build frontend: 4 days, after A.
- D. Integration testing: 2 days, after B and C.

- Path A-B-D takes 10 days; path A-C-D takes 9. The critical path is A-B-D, so the project needs at least 10 days. C has 1 day of float: it can slip a day without moving the end date. B cannot slip at all.

What to do with it

- Put your strongest people and closest attention on critical tasks.
- To finish sooner, shorten the critical path: split a task, run work in parallel, or remove a dependency. Speeding up a task with float changes nothing.
- Re-check after every change. When a non-critical task slips past its float, the critical path moves.

Software-specific traps

- Hidden dependencies: a shared test environment, a security review, another team's API. Write them down as tasks.
- Estimates are guesses. Add a buffer at the end of the critical path rather than padding every task.
- Integration is often the real critical task. Integrate early so it is not one big step at the end.

Related credential: Program Technical Practitioner. Explore the exam facts and syllabus.

Sources and further reading

1. IICTP. TPBOK, Engineering Governance & Agile Delivery. iictp.org/standards/tpbok.

Published by IICTP, an independent technical credential body, on 20 November 2026. The current version of this post is at https://blogs.iictp.org/critical-path-method-explained. More guides: blogs.iictp.org.