

LEADER · 7 MINUTE READ

Concurrent working: learn, build and use at the same time

Instead of requirements, build, test, then roll out, run short loops where real users learn, build, use and measure AI together.

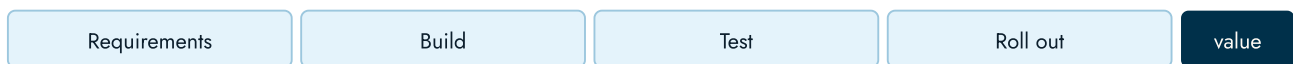
What it is

New technology is often introduced in a strict sequence: requirements, build, test, then roll out. Each stage waits for the last. Value arrives only at the end, when needs may have changed.

This guide borrows from two established practices. In product development, **concurrent engineering** overlaps design and manufacturing planning, with cross-functional teams (people from different departments) working together from the start, so problems surface earlier. Iterative, or **agile**, software practice adds short loops and a backlog.

Sequential versus concurrent (illustrative)

Sequential: one phase after another, value at the end



Concurrent: short loops where people learn, build, use and measure together



A person checks and approves at the end of every loop. High-risk systems: loops in a test environment only, and formal sign-off before use.

Applied to AI adoption, this becomes **concurrent working**: short **loops** of a few weeks in which real users learn the tool, build small things with it, use them in real work and measure the result, all at the same time. The first loop can produce some value, even if it is small. Each loop then improves on the last.

Think of learning to cook by making dinner every night, rather than reading a cookbook for a month first. You eat from the first evening, and each meal teaches you something.

Where it does not fit. Some systems carry high risk or are regulated: decisions about health, safety, credit, legal rights or employment, for example. For these: loops in a **test environment** only, and formal sign-off before use. A test environment is a safe copy of the system with made-up or approved test data.

Link to the J-curve. Results often dip before they rise when AI is introduced (see guide 30). Concurrent loops can help make that dip shorter and shallower, because learning, process change and measurement happen in small steps. They are not a guarantee: the review after each loop shows whether the curve is turning.

Why it matters

- Value can start in the first loop, not at the end of a long project.
- Real users shape the tool, so it fits the actual work rather than a guess about it.
- Problems with data, process or quality appear early, when they are cheaper to fix.
- Short loops with a measure at the end of each give leaders regular, honest evidence.

How to do it

1. **Choose a small scope.** One team, one or two tasks that happen often, such as drafting meeting notes or answering common customer questions (see guide 16).

2. **Involve real users from day one.** The people who do the work help design the prompts, steps and checks. Use only approved tools and approved data (see guide 10).
3. **Set a short loop length.** A few weeks is enough to try something real and still change course.
4. **Learn, build and use in the same loop.** A short session on the basics, then straight into building a saved prompt or simple tool, then using it on real tasks with a person checking each output.
5. **Measure at the end of every loop.** Record time and quality against the starting point (see guide 28).
6. **Have a person approve each loop.** A named owner reviews the measures and the risks, then approves the next loop, a change of direction or a stop.
7. **Keep a backlog.** A **backlog** is a simple, ranked list of ideas and fixes. Each loop takes the top few items. New ideas go on the list, not into the current loop.
8. **Agree stop rules in advance.** For example: stop if quality falls, if a data rule is broken, or if two loops in a row show no improvement.

Example

A customer service team wants AI help with answering common emails.

Sequential plan: two months of requirements, two months of building, one month of testing, then roll out. The team sees nothing useful for five months, and the first version misses much of what staff actually need.

Concurrent plan:

- **Loop 1 (a few weeks):** four service staff learn the approved assistant, write saved prompts for five frequent questions and use them with made-up examples first, then, if your data rules allow, on real emails. Every reply is checked by the staff member before sending. Measured result: time per reply for those five questions.
- **Loop 2:** the team adds a checklist for tone and policy, fixes two prompts that gave wrong answers and adds three new question types from the backlog.
- **Loop 3:** the prompts are shared with the wider team (see guide 29), with a short guide written by the first four staff.

A loop plan can be as short as this:

```
Loop [number], [dates]
Scope: [team], [tasks]
Build: [prompts or tools to create or improve]
Measure: [time per task], [quality check]
Human check: [who reviews each output]
Approver: [name] decides continue, change or stop on [date]
Stop if: [stop rules]
```

Common mistakes

- **Loops with no measure.** Fix: decide the measure before the loop starts.
- **Testers who are not real users.** Fix: include the people who do the work every day.
- **Letting scope grow inside a loop.** Fix: new ideas go on the backlog for the next loop.
- **Skipping the approval.** Fix: a named person signs off each loop, even briefly.
- **Using live loops on high-risk work.** Fix: loops in a test environment only, and formal sign-off before use.

Try it this week

1. Pick one repeat task in your team.
2. Write a one-page loop plan using the template above.
3. Name three real users and one approver.
4. Choose one speed measure and one quality measure.
5. Write two stop rules and share the plan with the team.

Going further

- **Run evaluations each loop.** Keep a small set of test cases and rerun them after every change (see guide 27).
- **Build the support around the AI.** Add saved instructions, checklists and review steps loop by loop (see guide 23).
- **Run loops in parallel teams.** Two teams on different tasks can compare notes at each review.

Key words

- **Concurrent engineering:** a product-development practice where design and manufacturing planning overlap, with cross-functional teams.
- **Concurrent working:** applying that idea to AI adoption, with users learning, building, using and measuring together.
- **Loop:** a short cycle of a few weeks that ends with a measure and a decision.
- **Backlog:** a ranked list of ideas and fixes waiting for a future loop.
- **Stop rule:** a condition, agreed in advance, that ends or pauses the work.
- **Test environment:** a safe copy of a system, with made-up or approved test data, for trying changes.