You leave with a decision you can keep true.

Not a report of one — the model the decision was made in, in your own repository, and the method to keep it current after we are gone.

THE PROBLEM

A system is retired. A diagram still shows it. A process changes. The document does not.

The same facts live in separate diagrams, documents and spreadsheets. Before a decision can use them, someone has to work out which copies are still true.

The work changes: people maintain the shared description and review changes, rather than reconcile separate copies.

HOW IT WORKS

1. **Describe the organisation once**

   Plain-text files name goals, processes, systems, owners and their relationships. Together, these records form the model.

2. **Generate what you need to see**

   Diagrams and reports show selected parts of that model. They are produced from the same records instead of being edited as separate copies.

3. **Review and check each change**

   A repository keeps the files and their history. Proposed changes are reviewed; automatic checks flag broken rules and references.

WHAT STAYS WITH YOU

A description you can inspect

Your team can read the records, see their history and propose the next change.

Connections you can follow

Trace the recorded links from a system to the processes and goals that depend on it.

A way to keep it useful

The model and review practice remain with your team after an engagement ends.

That is what **Architecture-as-Code for the enterprise** is for: the git-native, text-first discipline software already has, carried up to the strategy, business and architecture layers that code-scoped tools never reach. The enterprise is described once, in plain text, and every view — a goal tree, a capability map, a sequence diagram — is projected from that one description.

Against code-scoped tools. Architecture-as-Code exists today and stops at software. The layers where the money is committed — motivation, strategy, business — are the ones it never reaches, and they are the ones this covers.

Against context and memory layers. This is a formal model, not free-form notes and not opaque embeddings. It validates, it diffs, and a change to it is reviewed the way a change to code is reviewed. A machine can act on it because it is grounded, not because it was remembered.