



ARCHITECTURE & GOVERNANCE · ONE BOOK, EVERYONE IN TIME

Capable teams, each playing its own tune? One book gets everyone in time.

We help you draw one shared map of which software does what, and agree a few house rules with your teams. Then we set up checks that run while the work happens, so rule questions get answered as you go, not the night before launch.

Worth knowing: It's a journey, not a quick fix: we start with one map and one rule, and build from there with your teams. Automatic checks catch many problems, not all of them. Only rules written so a computer can read them can be checked this way, and people still make the hard calls.

One book, everyone in time.

Start with one map: email contact@algoshred.com with what your teams run and where things clash, and we'll suggest the first corner to map.



The problem, and the idea.

Every team picked its own tools, and now nothing quite fits together.

Nobody can draw how our systems connect, or the drawing is years old.

We pay for several tools that do the same job.

Our house rules live in a long document, and every reviewer reads it differently.

We find out we broke a rule the night before launch, or at review time.

Nobody remembers why we chose this, so the same argument keeps coming back.

Architecture and governance means one shared map of which software does what and how it connects, plus a few agreed house rules that are checked while the work happens, not the night before launch.

The story, decoded. In our story, the orchestra is your teams and the software each one runs, the music book is one shared map of which software does what and how it connects, Sione, the young conductor, is you or your tech lead, Tock the cricket's shared beat is the few choices everyone agrees, the listening gadget on each stand is an automatic check, the safety inspector stands for the outside rules your business must follow (at work, your auditor), and the oboe player's kazoo is a tool nobody agreed on.



01 One shared map of which software does what

One current map of your software, who looks after each piece and how they connect. Each team zooms into its part, and new projects plan to fit before they start.

Experts call this: enterprise architecture, solution architecture

Worth knowing It's a journey, not a quick fix: we start with one map and one rule, and build from there with your teams.

02 A few agreed choices, and the reason written down

Everyone agrees a short list of choices, such as which tools to use for common jobs and where customer details are kept. Each group decides its small things and writes down why.

Experts call this: standards, principles, architecture decision records

Worth knowing Teams can make a call after asking the people the decision affects; asking is not the same as everyone agreeing. Writing down why means fewer old arguments come back, not none.

03 House rules checked as you go

The rules that matter most are written so a computer can check them each time someone changes the software. A bit off: a hint. Way off: a person decides. A few big rules simply stop it. You set, rule by rule, which of the three a slip gets.

Experts call this: policy as code, guardrails, continuous compliance

Worth knowing Automatic checks catch many problems, not all of them. Only rules written so a computer can read them can be checked this way, and people still make the hard calls. Most rules start as hints; only a few become a firm stop.

04 Three questions for anything new

Before a new tool or supplier joins: does it fit, what does it replace, what if it breaks? The pieces you already rely on get a plan B too.

Experts call this: technology governance, technology risk



Own tune, or one book.

SECTION, TEAM

EVERYONE ON THEIR OWN TUNE

ONE BOOK, EVERYONE IN TIME

Strings the website team	We picked our own tools and our own way of storing customer details.	We use the shared map, and our part plugs into the rest.
Woodwind the customer-service team	We added a new chat tool, and nobody checked how it fits.	New tools get a quick look first, so they fit with the rest.
Brass the finance team	We pay for a tool that does the same job as two others.	The map shows the overlap, so we agree which tool to keep.
Percussion the warehouse team	We built our own stock list, and it doesn't match the website's.	Our stock list is on the map, so the website knows where to look.

An illustration, not a measurement.

Set how strict each house rule is

● Hint ■ Ask a person ▲ Stop

AT WORK

STARTS AS

WHY

Each new system is added to the shared map, with a named owner In the orchestra: Play from the shared book	● Hint	Easy to put right and low risk, so a reminder is usually enough.
Use the agreed tools for common jobs In the orchestra: Keep to the shared beat	● Hint	Teams sometimes have a good reason to differ, so a nudge beats a wall.
Name things the agreed way, so others can find them In the orchestra: Label your part the agreed way, so anyone can find it	● Hint	A naming slip is quick to fix, so a gentle blink is plenty.
A big design choice gets a short note saying why In the orchestra: Pencil a note when your group changes a part	● Hint	The note matters most later, so a reminder at the time is usually enough.
A new outside tool or supplier gets a quick look from a named person In the orchestra: A new instrument asks Sione first	■ Ask a person	Something new can touch a lot, so a person takes a quick look before it joins.
Customer details are kept where we agreed to keep them In the orchestra: Each section keeps its sheet music on its own agreed shelf	▲ Stop	It protects your customers and the promises you made, so it is a firm stop. Where a law sets the place, the rule follows the law, not a lower setting.
No passwords written into the code In the orchestra: No spare key taped to the stage door	▲ Stop	A password in the code can be copied by anyone who sees it, so this one stops.

When does the inspector get her answers?

	FIRST REHEARSAL	NEXT WEEK	THE WEEK AFTER	DRESS REHEARSAL	THE NIGHT BEFORE THE CONCERT
Checked the night before	Everyone plays. Nobody checks anything yet.	A note is a bit off. Nobody notices.	The questions keep piling up, unanswered.	Still nothing written down.	Everyone digs through old emails to answer every question at once.
Checked as you go	The gadgets listen from the first note, and the first check goes in the folder.	A note is a bit off. The gadget blinks, and it's put right while it's small.	Another week of checks goes in the folder.	The record keeps building as the work happens.	An ordinary evening. The record is already in the book.

Checked the night before. Surprise visit. Nothing is ready yet: everyone starts digging through old emails.

Checked as you go. Surprise visit. Sione just opens the book: the record is clipped inside. The inspector still reads it herself and decides.

Worth knowing

Most rules start as hints; only a few become a firm stop.

Worth knowing

Laws and regulators' rules are not optional; "few and clear" is about your own house rules.

Worth knowing

Checks leave a record of what was looked at. That record is a head start, not a certificate: your auditors still decide.

Worth knowing

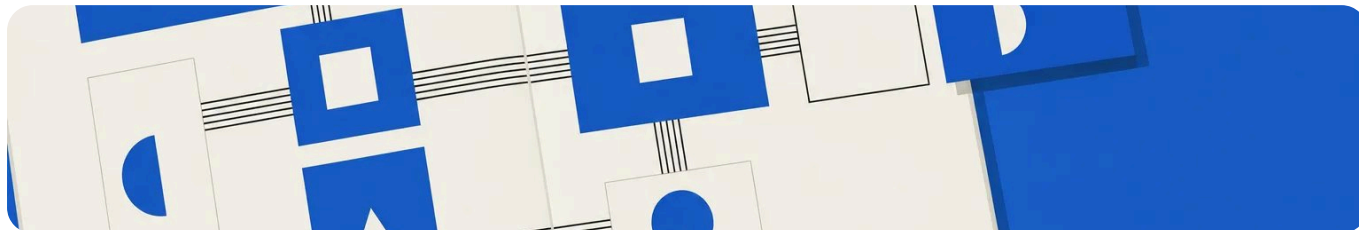
Automatic checks catch many problems, not all of them. Only rules written so a computer can read them can be checked this way, and people still make the hard calls.





What we help with.

Start with one map or one house rule, and build from there with your teams.



One map of which software does what

We list what you run, who looks after it, where things overlap and where the gaps are, and draw it as one map each team can zoom into.

A shared, current map your teams can read and keep up to date.

Experts call this: enterprise architecture, capability map, application inventory

New projects planned to fit

For a new project, we sketch how it plugs into everything around it, and review it as a conversation, not an exam.

A new system designed to fit what is already there.

Experts call this: solution architecture, design review

Decide close to the work, and write down why

Teams make the call after asking the people the decision affects, and a short note keeps what was chosen, why, and what was given up.

Decisions that don't wait for the next review meeting, with the reasons kept.

Worth knowing: Teams can make a call after asking the people the decision affects; asking is not the same as everyone agreeing. Writing down why means fewer old arguments come back, not none.

Experts call this: advice process, architecture decision records (ADRs)

A record that builds as you go

Each rule is linked to the checks that look at it, and each check leaves a dated record, kept in one place for your reviewers.

Your auditor's questions can be answered from the record rather than a last-minute dig.

Worth knowing: Checks leave a record of what was looked at. That record is a head start, not a certificate: your auditors still decide.

Experts call this: continuous compliance, compliance as code, audit evidence

For your technical team: we work with what you already use and the frameworks you follow, for example TOGAF, ArchiMate, the C4 model, Open Policy Agent, Kyverno, Azure Policy, AWS Control Tower, HashiCorp Sentinel, the AWS Well-Architected Framework and the NIST Cybersecurity Framework.

A route from today to where you want to be

We agree the destination with you and break the way there into steps that fit your business plans.

A step-by-step route, agreed with you.

Experts call this: target architecture, roadmap

A few house rules that matter

A short written list of agreed choices, such as which tools to use for common jobs and where customer details are kept, each with an owner.

A short list people can find and follow.

Worth knowing: Laws and regulators' rules are not optional; "few and clear" is about your own house rules.

Experts call this: standards, principles, reference architectures

House rules turned into automatic checks

The rules that matter most become checks that run each time someone changes the software, each set to give a hint, ask a person or stop.

Rule questions often show up while the work is still small.

Worth knowing: Automatic checks catch many problems, not all of them. Only rules written so a computer can read them can be checked this way, and people still make the hard calls.

Experts call this: policy as code, guardrails, fitness functions

Three questions for anything new, and a plan B

An agreed way to add, try and retire tools and suppliers, AI helpers included: does it fit, what does it replace, what if it breaks? Plus a list of what could go wrong with the pieces you rely on, and a plan B for the important ones.

A clear view of what comes in, why, and which pieces are risky.

Experts call this: technology governance, technology radar, technology risk, AI governance

Capabilities, from one shared map to checks that run as you go.

From seeing the whole, to agreeing the few rules that matter, to checking them while the work happens, to choosing new tools on purpose, AI helpers included.

See the whole

One map of what you run and where you are heading.

- A list of every system and who looks after it (inventory)
- A business capability map: what the business does, and which software supports each part
- A current and a target picture
- A step-by-step route between them
- A design for each new project, and a friendly review (solution design)

Agree the rules

A few house rules, with an owner and a reason.

- Agreed ways of making common choices (principles and standards)
- Ready-made patterns for common kinds of system (reference architectures)
- Short decision notes, and the habit of writing them
- A regular, optional session where teams can ask for advice

Check as you go

Rules checked while the work happens.

- House rules turned into checks that run each time software is built or a cloud setting changes (policy as code)
- Strictness set per rule: hint, ask a person or stop
- Exceptions with an owner and an end date
- A record of each check, kept as you go

Choose wisely

New tools on purpose, and a plan B for what matters.

- A list of the tools we use, try, are wary of, or are retiring (a technology radar)
- An agreed way to retire old tools
- A list of what could go wrong, and what to do about it
- A simple setup for AI tools: what they may touch, and who checks them

WHERE IT BECOMES REAL · ILLUSTRATIVE EXAMPLES

Financial services

For example, a rule like “customer records stay in the country we promised” could be checked each time the software changes, with the record kept, so a later review is a look through the folder rather than a dig through old emails.

Retail and e-commerce

For example, a business with several shop systems sees in one map which tools do the same job, and agrees which to keep before the next one arrives.

Healthcare

For example, a new system is planned to fit what is already there, and anything that touches patient records needs a named person’s yes, written down.

SaaS and product companies

For example, teams make their own design calls after asking the people affected, and a short note keeps the reasons, so the next team has them to hand.

Manufacturing and logistics

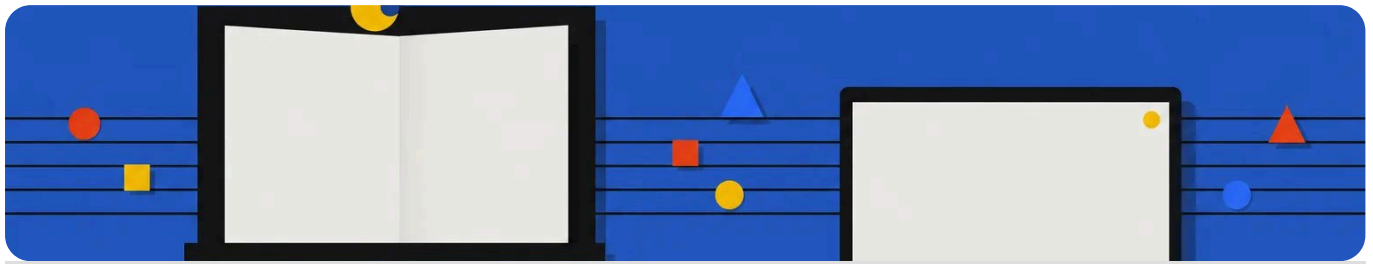
For example, older systems nobody looks after any more are listed as risks, each with a plan B, before they cause trouble.

Any team adopting AI helpers

For example, a new AI tool answers the same three questions as any new instrument: does it fit, what does it replace, what if it breaks? Plus one more: what may it touch?



Map it with you, then agree, check and coach.



- 01** **LISTEN**
Listen to how decisions are made today
We sit in on how decisions are made today, talk to the people who build and run your software, and list where things clash, overlap or wait for a review.
- 02** **MAP IT WITH YOU**
Map which software does what, with your teams
We draw one map of which software does what and how it connects, with your teams, plus the rules people already follow, written down.
- 03** **AGREE THE RULES**
Agree the few house rules that matter
We help you agree the few house rules that matter most and how strict each one is. Most start as hints. Laws and regulators' rules stay as they are.
- 04** **SET UP CHECKS**
Set up checks that run as you go
We turn the top rules into automatic checks with one pilot team (the first team to try it), and the record starts building as the work happens.
- 05** **COACH AS YOU GO**
Coach your people as they go
We coach your people as they go and hand over the map, the rules, the decision notes and the checks. We can stay on if you prefer.

Ways to start

One map. A first map of which software does what, and where the risky pieces sit.

First check. One house rule turned into an automatic check, with one team.

Unstick one review. Pick one decision that keeps waiting for a meeting; the team asks the people the decision affects in writing instead, and keeps a short note of why.

What you get

- A map of which software does what, who looks after it and how it connects
- A short list of house rules, each with an owner and a strictness level
- Decision notes for the big choices, in a format your teams keep writing
- The first automatic checks, running with a pilot team
- A record of checks your reviewers can look through
- Coaching sessions for the people who will keep it going



Questions, myths and the next step.

What is architecture and governance, in plain words?

One shared map of which software does what and how it connects, plus a few agreed house rules that are checked while the work happens, not the night before launch.

Isn't governance just more meetings and forms?

It shouldn't be. Most rules start as hints and automatic checks, teams can decide after asking the people affected, and people spend their time on the hard calls.

Do we need a big architecture framework, such as TOGAF, to start?

No. Frameworks are useful references, not a starting requirement. We start with one map and a few rules, and use frameworks as a menu.

What does "rules checked as you go" mean?

The house rules that matter most are written so a computer can check them each time someone changes the software, the same way each time, and people write and update the rules themselves. They can catch problems early, and people still make the hard calls.

Can you promise how our audit goes?

No. We can't promise an audit result. We help the rules get checked as you go and the record build along the way. Checks leave a record of what was looked at. That record is a head start, not a certificate: your auditors still decide.

Will it slow our teams down?

Most rules start as hints; only a few become a firm stop. Routine work goes ahead, and a firm stop is kept for the few rules that really matter.

Do we have to replace our tools?

Not to get started. We begin from what you have and the checks your tools already support; if the map shows two tools doing the same job, you decide which to keep.

How will we know it's working?

We agree a few plain signs with you at the start: rule questions found earlier rather than later, decisions waiting less, and the map still current. You keep track of those signs, and we make no promises about the results.

MYTHS, ANSWERED

"Governance means red tape."

Done well, it works the other way: rules turned into automatic checks answer while the work happens, so fewer things wait for a meeting. A shared beat doesn't slow the music; it lets everyone play together.

"More rules make us safer."

In our story, Sione hands out a thousand rules and the orchestra dozes off by rule forty. A few clear house rules get followed. Laws and regulators' rules are not optional; "few and clear" is about your own house rules.

"Architecture is a big diagram in a drawer."

A map only counts if teams use it. Keep it small, shared and current, and let each team look after its own part.

"Automatic checks mean no human judgement."

No. Automatic checks catch many problems, not all of them. Only rules written so a computer can read them can be checked this way, and people still make the hard calls.

Everyone in time yet?

Start with one map: email contact@algoshred.com with what your teams run and where things clash, and we'll suggest the first corner to map.

contact@algoshred.com

WATCH THE FILM

algoshred.com/architecture-governance