

Notes

SPAD (Structured Prompt-Driven Engineering) is Fernando García Varela's framework for managing software development with artificial intelligence in a controlled way. It turns AI from an informal programming assistant into a governed engineering tool: its work can be reviewed, reproduced and audited.

SPAD is not a collection of prompts for asking for code. It is a way of working with three properties:

- Sequential: work advances through phases in a fixed order.
- Blocking: no phase starts until the previous one has produced its result and a person has validated it.
- Auditable: each phase leaves an explicit artefact (plan, review report, test strategy, version record).

Why it matters: AI already writes code in many companies. The question for management is not whether it is used, but whether someone controls what it decides, what is reviewed and what evidence remains.

This presentation covers the problem, the principle, the cycle of phases, the roles, validation, the records, the complementary cycles, systems that include AI, measurement, the reduced version and what it does for a company: building with method and making demands of its suppliers.

SPAD is published under CC BY 4.0 for content and MIT for code. It is not a certification and it does not promise improvement figures.

Classroom question: in your company, who decides today how the code that an AI writes is designed?

TERMS ON THIS SLIDE

- **Blocking execution principle**: SPAD's central rule: no phase starts until the previous one has delivered its artefact and received a favourable verdict. It prevents AI from building on a plan nobody has reviewed. The link opens the principle in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#the-blocking-execution-principle

SOURCES AND LINKS

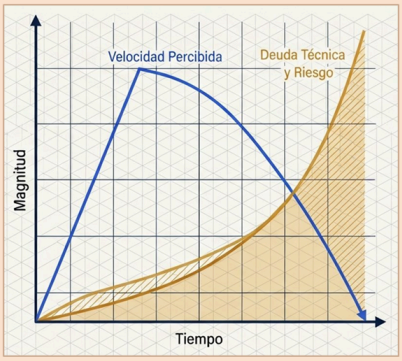
SPAD IN 20 SLIDES

- **SPAD 00 · What SPAD is and how it helps** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#what-spad-is
- **SPAD 00 · Status, licence and next steps** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#status-licence-and-next-steps
- **What SPAD is** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html
- **SPAD library** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/index.html

SPAD IN 20 SLIDES

The problem: the illusion of speed

AI generates plausible code very fast. Without a method, technical debt and risk grow without anyone seeing it.



- It works in the demo and fails in production: nobody designed the solution.
- Nobody knows why the system is built this way: the AI took the decisions inside the code.
- The audit asks for evidence and there is none: everything stayed in conversations with the AI.
- Conceptual SPAD chart, with no data. Plate in Spanish.

These are not AI problems: they are method problems that AI amplifies.

→ The problem it solves

Notes

The chart is conceptual and contains no data. It shows one idea: at first, perceived speed rises sharply because AI produces code in minutes. Over time, that speed falls, because each change collides with decisions that nobody took consciously. Meanwhile, technical debt and risk grow.

SPAD identifies six common symptoms:

- The code works in the demo and fails in production.
- Nobody knows why the system is built this way.
- Tests are scarce or only test the easy case.
- Vulnerabilities appear in code that nobody reviewed.
- A small change breaks another part of the system.
- An audit asks for evidence of the development and there is none.

Simple example (fictitious): a team builds an internal assistant in a week by asking the AI for code until it “works”. With real data it fails. Nobody knows what decisions the AI took, the tests only cover the demo case and there is no record of what was asked.

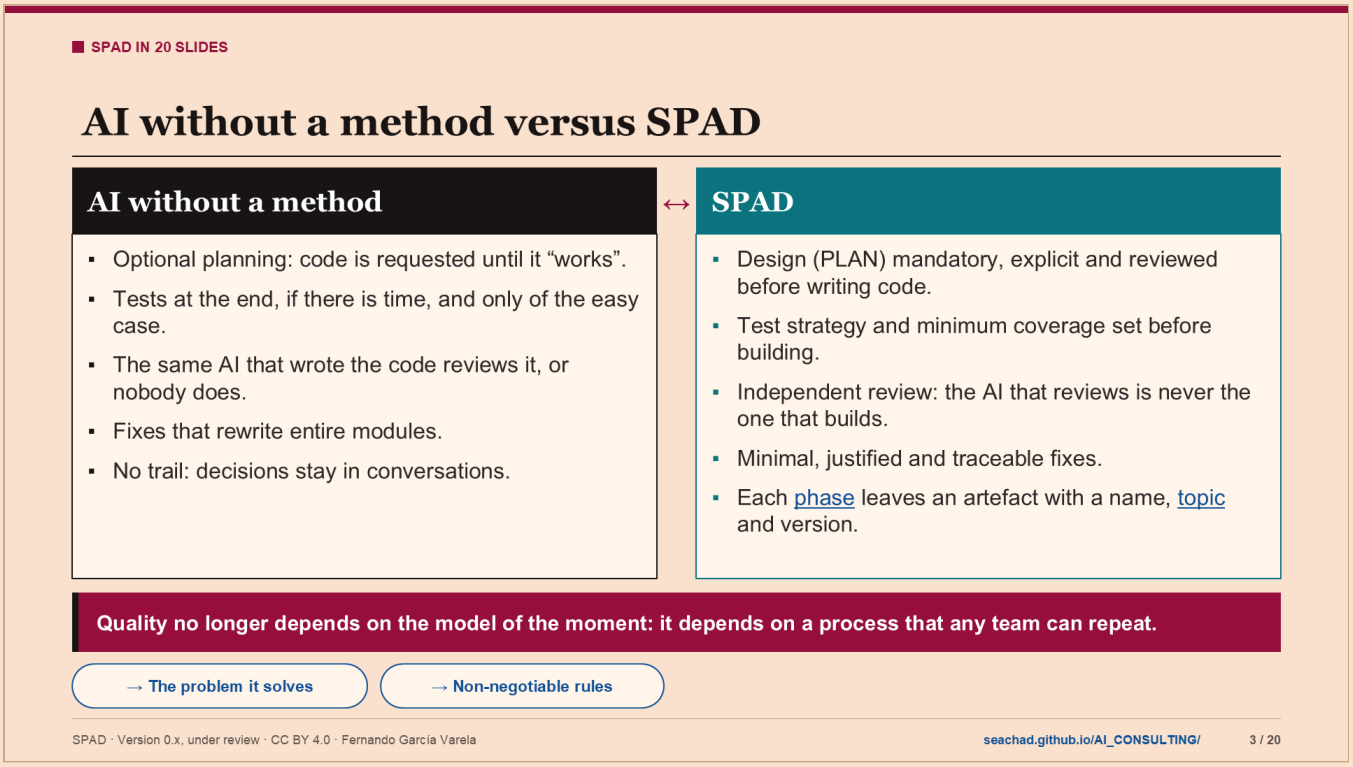
Why it matters: none of these problems belongs to the AI. They are method problems that AI amplifies. With a method, quality does not depend on the skill of whoever writes the prompts or on the model of the moment.

Common mistake: measuring progress by the amount of code generated, rather than by what is designed, tested and reviewed.

Classroom question: which symptom on the list have you seen in a real project? How much did it cost to fix it afterwards?

SOURCES AND LINKS

- **SPAD 00 · The problem it solves** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#the-problem-it-solves
- **The problem it solves** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#the-problem-it-solves



Notes

This comparison summarises the table of symptoms in SPAD document 00. Each row on the left has a specific answer on the right.

- Design: without a method, the design appears by itself, by dint of asking for code. With SPAD, the plan is mandatory, it is written without code and another AI reviews it before building. Every decision stays in the plan, not hidden in the code.
- Tests: without a method they are written at the end. With SPAD, the test strategy and its minimum coverage are set before building and a review checks that they are met.
- Review: without a method the same AI reviews, or nobody. With SPAD, the AI that reviews is never the one that builds and a person also reviews the code.
- Fixes: without a method entire modules are rewritten and other parts break. With SPAD, each fix is minimal, justified and traceable.
- Trail: without a method everything stays in conversations. With SPAD, each phase leaves an artefact stored in a common structure.

Why it matters for an executive: each row on the right is a control that can be checked. If someone states that their team works with SPAD, you can ask for the plan, its review and the record of who validated it.

Common mistake: thinking that SPAD is "writing better prompts". A good prompt helps, but what gives control is the order of the phases, the separation of roles and human validation.

Classroom question: which of the five rows do you think is broken most often in the teams you know? Why?

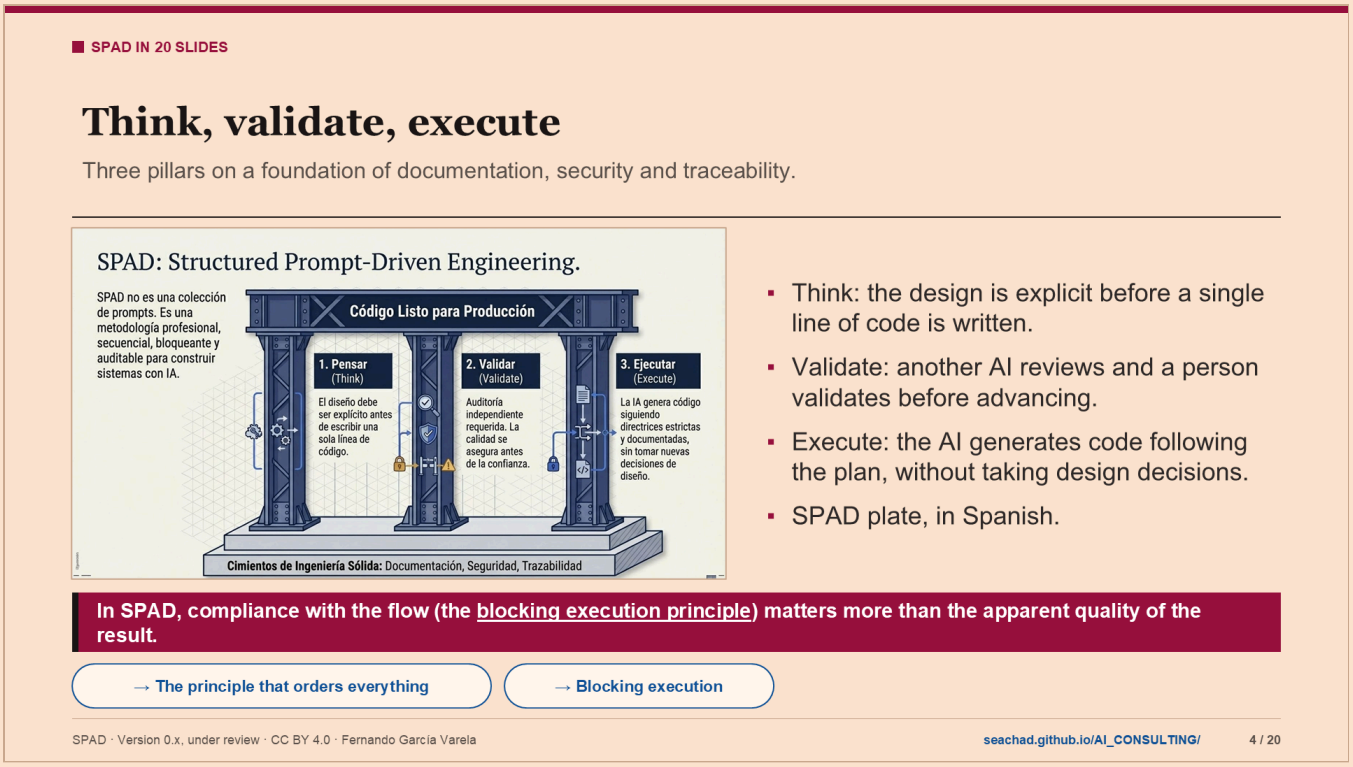
TERMS ON THIS SLIDE

- **SPAD main cycle** (fases 0–10): SPAD's eleven phases: context, plan, plan review, code guide, test strategy, implementation, testing, test review, code review, fixes and release management. Each one leaves an artefact that is reviewed. The link opens the cycle in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#main-cycle-phases-0-to-10

- **SPAD topics**: A topic is SPAD's unit of work: a feature or a change that goes through the full cycle with its own artefact folder. It lets you follow from start to finish what was asked, planned and built, and who validated it. The link opens its definition. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#topics

SOURCES AND LINKS

- **SPAD 00 · The problem it solves** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#the-problem-it-solves
- **SPAD 01 · Non-negotiable operating rules** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#non-negotiable-operating-rules
- **The problem it solves** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#the-problem-it-solves
- **Non-negotiable rules** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#non-negotiable-operating-rules



Notes

The plate summarises the logic of SPAD in three verbs.

- Think: before writing code, the problem is described and the solution is designed in a plan. The plan contains no code.
- Validate: an AI other than the one that produced the artefact reviews it and issues a verdict. A person, the one orchestrating the work, accepts or rejects that verdict. No phase starts without that validation: this is the blocking execution principle.
- Execute: the AI that builds follows the plan and the code rules. If it finds something the plan does not provide for, it does not decide: it flags it and asks for a decision.

The principle that orders everything: compliance with the flow matters more than the apparent quality of the result. A technically correct response that skips the rules of its phase is not used: it is discarded in full and the phase is repeated. The validation priority is: first, process compliance; second, technical correctness; third, efficiency.

Simple example: the AI delivers an excellent plan that includes five hundred lines of “example” code. Even if the code is good, the response is discarded, because the plan defines and does not implement.

Why it matters: if a response that breaks the rules is accepted “because it works”, the process ceases to exist the second time.

Classroom question: does it seem reasonable to discard a good result only because it did not follow the process? What is lost and what is gained?

TERMS ON THIS SLIDE

- **Blocking execution principle**: SPAD's central rule: no phase starts until the previous one has delivered its artefact and received a favourable verdict. It prevents AI from building on a plan nobody has reviewed. The link opens the principle in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#the-blocking-execution-principle

SOURCES AND LINKS

- **SPAD 00 · The principle that orders everything** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#the-principle-that-orders-everything
- **SPAD 03 · Fundamental principle** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#fundamental-principle
- **The principle that orders everything** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#the-principle-that-orders-everything

SPAD IN 20 SLIDES

Twelve rules that are not negotiable

The plate shows the first six. The operating guide adds six more.



- 7 and 8: mandatory tests, set before building; security by design in sensitive code.
- 9 to 11: contexts before each phase, one topic per job and a record per artefact of the model.
- 12: no AI approves. Every acceptance is signed by an identified person.
- SPAD plate, in Spanish: rules 1 to 6.

A valid “no” from a review is worth more than an invalid “yes”.

→ The twelve rules

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 5 / 20

Notes

The SPAD operating guide sets twelve rules. Each one protects something specific:

- 1. No phase is skipped (protects blocking). In the reduced version phases are grouped, but they do not disappear.
- 2. Every output is auditable (traceability).
- 3. Every decision is explicit: there are no design decisions hidden in the code (explainability).
- 4. Code executes, it does not decide: the AI Builder does not take design decisions (the plan).
- 5. Independent reviews: the AI Reviewer is never the builder and a person also reviews the code (quality).
- 6. Minimal fixes, never disguised refactoring (stability).
- 7. Mandatory tests, with minimum coverage set before building.
- 8. Security by design: critical and high findings block.
- 9. Global and project contexts loaded before any phase.
- 10. One topic per piece of work (the organisation's memory).
- 11. Model recorded: which model, version and instructions produced each artefact (reproducibility).
- 12. No AI approves (accountability).

Note on the plate: where it says “AI auditor” it refers to the AI Reviewer, which is an AI and not a person.

Why it matters: these rules are simple to check. An executive does not need to read code to ask whether they are met.

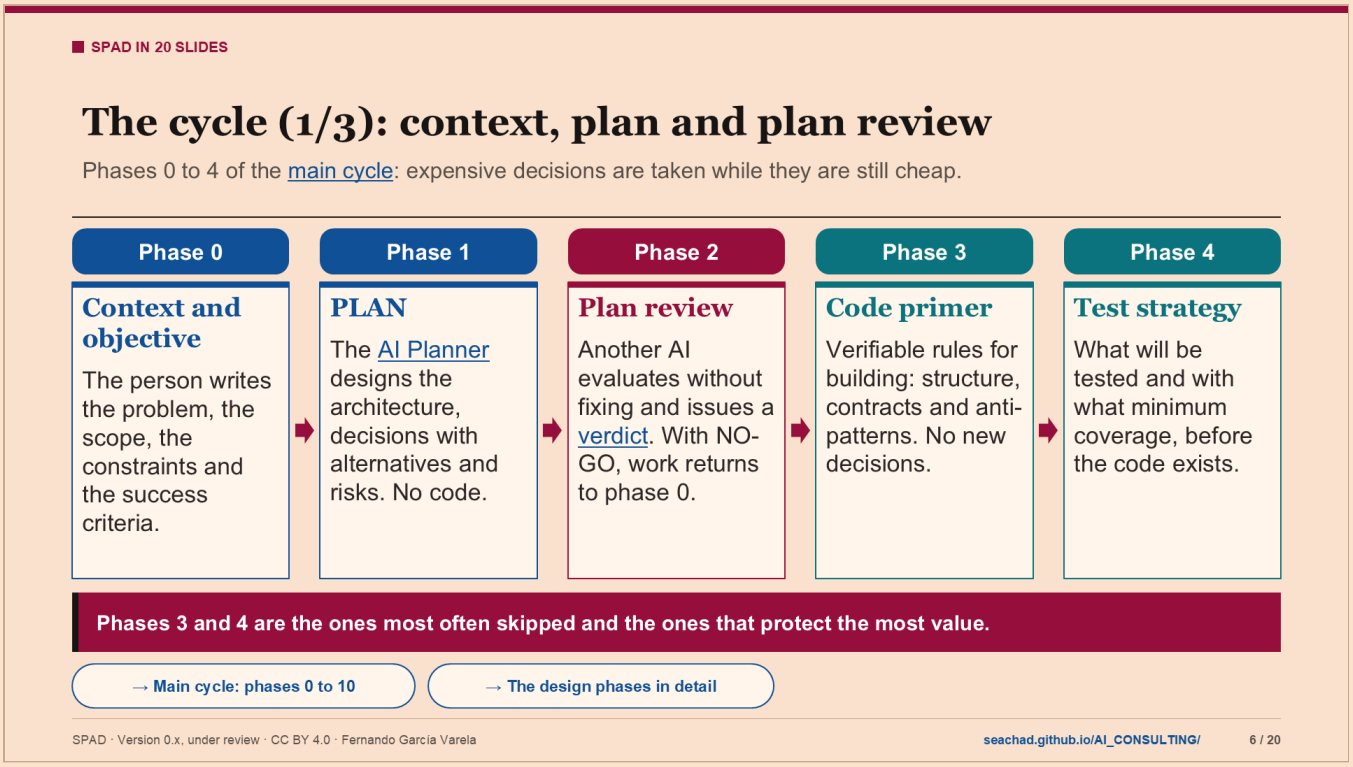
Classroom question: which of the twelve rules seems hardest to keep under deadline pressure?

TERMS ON THIS SLIDE

- **SPAD contexts**: Rule documents the AI must respect in each phase: the global context (organisation rules, data, approved models) and the project context. Exceptions are assessed by the reviewer AI and approved by a person. They ensure the AI does not work on its own assumptions. The link opens the contexts. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#contexts
- **SPAD topics**: A topic is SPAD's unit of work: a feature or a change that goes through the full cycle with its own artefact folder. It lets you follow from start to finish what was asked, planned and built, and who validated it. The link opens its definition. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#topics
- **Artefact records**: Each SPAD artefact carries three records: the model record (supplier, model, version, instruction and contexts used), the human validation record (who reviewed it and when) and the violation log (rules not complied with). They make the process auditable. The link opens the records. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#4-1-model-record

SOURCES AND LINKS

- **SPAD 01 · Non-negotiable operating rules** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#non-negotiable-operating-rules
- **SPAD 00 · The principle that orders everything** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#the-principle-that-orders-everything
- **The twelve rules** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#non-negotiable-operating-rules



Notes

The main cycle is used to build a new feature. It has eleven phases, numbered 0 to 10. This slide shows the five design phases.

- Phase 0, context and objective: the person orchestrating writes what problem is solved, for whom, what is within and outside the scope, what constraints exist (technological, legal, deadline) and how it will be known that the result is correct. Without success criteria there is no phase 1.
- Phase 1, PLAN: the AI Planner designs without writing code. Each decision is recorded with its alternatives and trade-offs, and each risk with its severity and probability.
- Phase 2, plan review: an AI Reviewer, other than the planner, evaluates coupling, cohesion, scalability, concurrency, security and observability. It does not fix. With GO WITH CHANGES, the plan goes back to phase 1; with NO-GO, work goes back to phase 0, because the problem is badly framed.
- Phase 3, code primer: translates the plan into rules that whoever builds can follow and check.
- Phase 4, test strategy: sets before building what will be tested and with what minimum coverage. Critical paths require full coverage.

Simple example: for a module that exports invoices to PDF, the plan decides where the files are generated and what happens if generation fails. The review detects that the plan does not provide for many users at once and returns the plan with changes.

Why it matters: detecting an architecture error in the plan review costs a conversation; detecting it in production costs an incident.

Classroom question: why do you think phases 3 and 4 are the ones most often skipped?

TERMS ON THIS SLIDE

- **SPAD main cycle** (fases 0–10): SPAD's eleven phases: context, plan, plan review, code guide, test strategy, implementation, testing, test review, code review, fixes and release management. Each one leaves an artefact that

is reviewed. The link opens the cycle in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#main-cycle-phases-0-to-10

- **SPAD roles**: The roles in SPAD: the orchestrator (a person who leads the cycle and validates) and several AIs with separate functions: planner, reviewer, builder, fixer, analyst and diagnostic. Whoever builds does not review what they build. The link opens the SPAD presentation document, which describes each role. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html
- **SPAD verdicts** (GO / NO-GO): Every SPAD review ends with one of three verdicts: GO (continue), GO WITH CHANGES (continue after fixing what was flagged) or NO-GO (return to the relevant phase). A single vocabulary across all reviews avoids interpretation. The link opens the verdicts in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#verdicts

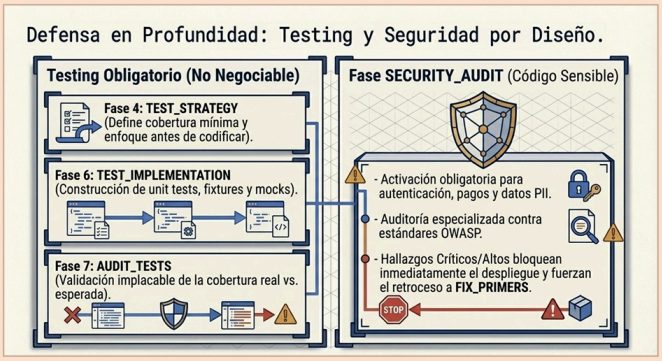
SOURCES AND LINKS

- **SPAD 01 · The design phases in detail** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#4-1-the-design-phases-in-detail
- **SPAD 00 · The main cycle** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#the-main-cycle
- **The design phases in detail** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#4-1-the-design-phases-in-detail

SPAD IN 20 SLIDES

The cycle (2/3): implementation, tests and reviews

Phases 5 to 8: what the plan says is built and another AI reviews it before continuing.



- Phase 5: the AI Builder executes the plan. If something is not provided for, it does not decide: it flags it.
- Phase 6: the tests in the strategy are implemented. Tests that only copy the code do not count.
- Phases 7 and 8: another AI reviews tests, code, dependencies and licences.
- SPAD plate, in Spanish.

Sensitive code also undergoes a security review, one of the complementary cycles: critical and high findings block deployment.

→ Build and review phases → Security cycle

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 7 / 20

Notes

With the plan approved, building starts.

- Phase 5, implementation: the AI Builder writes the code in accordance with the plan and the code primer. It also delivers the list of files created and modified, the dependencies added and a declaration of conformity with the plan. If it finds a situation that the plan does not provide for, it does not decide: it flags it as a deviation. The person orchestrating sends the work back to the plan or accepts a documented minor deviation.
- Phase 6, test implementation: the tests in the strategy are implemented, not those that happen to be convenient. A test that only reproduces the code, without verifying the expected behaviour, does not count.
- Phase 7, test review: an AI Reviewer compares actual coverage with expected coverage and reviews edge cases and error paths.
- Phase 8, code review: checks fidelity to the plan, compliance with the primer, technical risks, the debt being taken on and the dependencies: whether they exist, where they come from, whether they have known vulnerabilities and their licence.

Simple example: the AI Builder does not know how to retry a failed submission. The correct behaviour is not to invent a solution, but to write: “the plan does not say how to handle the retry; I am asking for a decision”.

Note on the plate: “audit” refers to the AI Reviewer's review and to the security review.

Why it matters: tests defined before building prevent them from being written at the end only to confirm what already exists.

Classroom question: how would you tell a test that verifies behaviour from one that only copies the code?

TERMS ON THIS SLIDE

- **SPAD roles**: The roles in SPAD: the orchestrator (a person who leads the cycle and validates) and several AIs with separate functions: planner, reviewer, builder, fixer, analyst and diagnostic. Whoever builds does not review what they build. The link opens the SPAD presentation document, which describes each role. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html
- **SPAD complementary cycles**: Variants of the cycle for special situations: legacy code (with characterisation tests), debugging, urgent fixes (with a mandatory follow-up analysis), security and supply chain. They keep control when the full cycle cannot be followed. The link opens the document. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html

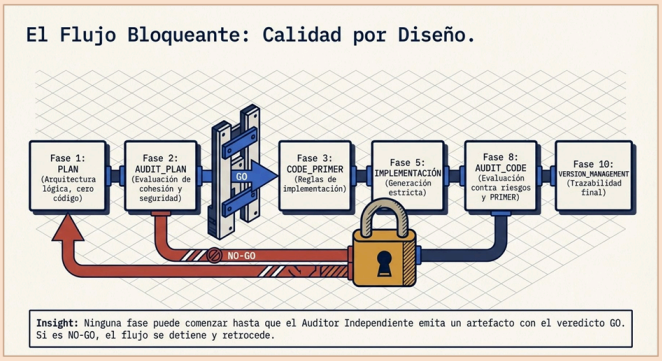
SOURCES AND LINKS

- **SPAD 01 · The build and review phases in detail** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#4-2-the-build-and-review-phases-in-detail
- **SPAD 03 · Examples of decisions** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#examples-of-decisions
- **SPAD 04 · Security cycle** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html#security-cycle
- **Build and review phases** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#4-2-the-build-and-review-phases-in-detail
- **Security cycle** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html#security-cycle

SPAD IN 20 SLIDES

The cycle (3/3): fixes and versions

Phases 9 and 10: fix only what was flagged and deliver something that can be operated and rolled back.



- Phase 9: surgical fixes. Each one goes back to the code review, not to a reimplementation.
- Phase 10: version, changes, migration, tested rollback and what to monitor. It is the handover to operations.
- A NO-GO, as a verdict of the code review, sends the work back to the PLAN (phase 1).
- SPAD plate, in Spanish: “auditor” is the AI Reviewer.

Without phase 10, the code is stable, but the system is not operated.

→ Verdicts and returns | → Handover to operations

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 8 / 20

Notes

The plate shows the blocking flow: a negative verdict stops the flow and sends it back to the corresponding phase.

- Phase 9, fixes: the AI Fixer applies minimal changes to the review findings. For each fix it writes the problem, the change, the justification, the impact and the regression risk. It cannot redesign, refactor entire modules or extend functionality. Each fix goes back to the code review, and also to the test review if the tests change.
- Phase 10, version management: version number, change log, breaking changes, migration instructions, deployment plan, rollback plan and monitoring requirements.

The returns of each verdict are fixed:

- Plan review (phase 2): GO leads to phase 3; GO WITH CHANGES, to 1; NO-GO, to 0.
- Test review (phase 7): GO leads to 8; GO WITH CHANGES, to 9; NO-GO, to 6.
- Code review (phase 8): GO leads to 10; GO WITH CHANGES, to 9; NO-GO, to 1.

Simple example: the review flags that an input is not validated. The fix adds that validation and nothing else. If it also “takes the opportunity” to reorganise the module, it is disguised refactoring and is discarded.

Why it matters: small, traceable fixes prevent a repair from breaking another part of the system. Phase 10 ensures that someone knows how to go back if something goes wrong.

Classroom question: why does a NO-GO in the code review go back to the plan and not to the implementation?

TERMS ON THIS SLIDE

- **SPAD main cycle** (fases 0–10): SPAD's eleven phases: context, plan, plan review, code guide, test strategy, implementation, testing, test review, code review, fixes and release management. Each one leaves an artefact that

is reviewed. The link opens the cycle in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#main-cycle-phases-0-to-10

- **SPAD verdicts** (GO / NO-GO): Every SPAD review ends with one of three verdicts: GO (continue), GO WITH CHANGES (continue after fixing what was flagged) or NO-GO (return to the relevant phase). A single vocabulary across all reviews avoids interpretation. The link opens the verdicts in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#verdicts
- **SPAD roles**: The roles in SPAD: the orchestrator (a person who leads the cycle and validates) and several AIs with separate functions: planner, reviewer, builder, fixer, analyst and diagnostic. Whoever builds does not review what they build. The link opens the SPAD presentation document, which describes each role. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html

SOURCES AND LINKS

- **SPAD 01 · The build and review phases in detail** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#4-2-the-build-and-review-phases-in-detail
- **SPAD 04 · Handover to operations** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html#handover-to-operations
- **Handover to operations** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html#handover-to-operations



SPAD IN 20 SLIDES

Roles: who does what, and what they cannot do

The same AI may take on several roles, but never two in the same phase. The person has the final say.

Role	What it does	What it cannot do
Orchestrator (person)	Sets the objective and contexts, validates each output and decides whether to advance.	Delegate validation to the AI that produced the output.
AI Planner	Designs the plan, the code primer, the test strategy and the version.	Write code.
AI Reviewer	Reviews plans, tests and code and issues a verdict.	Fix what it reviews or approve with known problems.
AI Builder	Implements code and tests according to the approved plan.	Take design decisions.
AI Fixer	Applies minimal, justified fixes.	Redesign or extend functionality.
AI Analyst and Diagnostic AI	Document legacy code and analyse incidents.	Propose the solution instead of the diagnosis.
AI Security Reviewer	Reviews the security of sensitive code.	Fix what it reviews.

The AI Reviewer is a tool: a person always accepts or rejects its verdict.

→ Roles in the operating guide → Self-approval: a cause of invalidity

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 9 / 20

Notes

SPAD separates logical roles from the people and the AIs that take part. There is one human role and seven AI roles.

- The orchestrator is a person. They define the objective, load the contexts, set the topic, validate each output, apply the validation policy and decide whether to advance. They have the final say in everything.
- The AIs have separate functions: one plans, another builds, another fixes and another reviews. The AI Analyst and the Diagnostic AI work in the legacy and debugging cycles. The AI Security Reviewer reviews sensitive code.

Two independence rules:

- The same AI may take on several roles over the course of a piece of work, but never two roles in the same phase.
- The AI that reviews an artefact is never the one that produced it.

Why it is called “AI Reviewer” and not “auditor”: an auditor is a person in the control function, independent of the team that builds. The AI Reviewer belongs to the function that builds and its output is only an input: it cannot approve anything.

Simple example: the same model can act as planner in the morning and as builder in the afternoon, in different phases. But it cannot review the plan it wrote itself.

Why it matters: segregation of duties is a classic control in any serious process: whoever does the work does not approve it. SPAD also applies it between artificial intelligences and always leaves a person accountable.

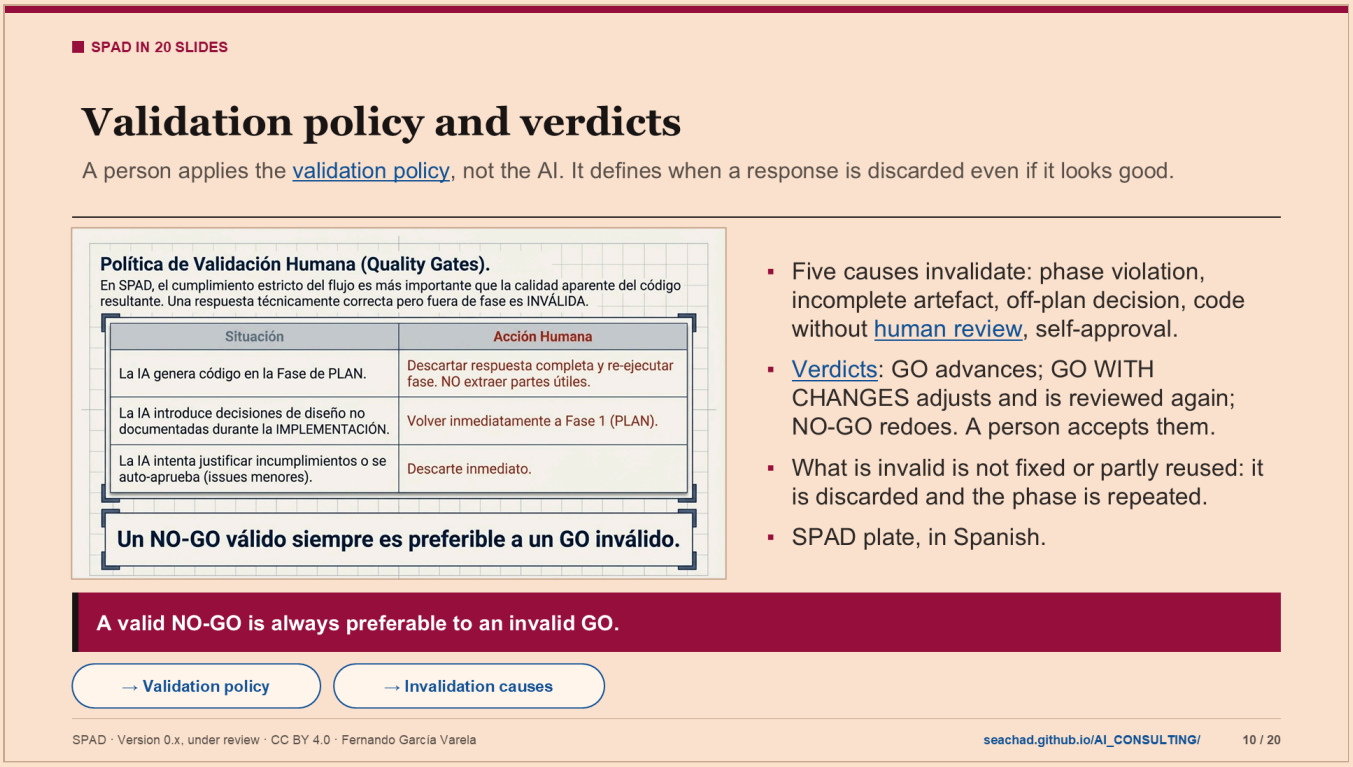
Classroom question: what risk do you see in the reviewing AI coming from the same provider as the building AI?

TERMS ON THIS SLIDE

- **SPAD roles**: The roles in SPAD: the orchestrator (a person who leads the cycle and validates) and several AIs with separate functions: planner, reviewer, builder, fixer, analyst and diagnostic. Whoever builds does not review what they build. The link opens the SPAD presentation document, which describes each role. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html
- **SPAD contexts**: Rule documents the AI must respect in each phase: the global context (organisation rules, data, approved models) and the project context. Exceptions are assessed by the reviewer AI and approved by a person. They ensure the AI does not work on its own assumptions. The link opens the contexts. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#contexts
- **SPAD verdicts** (GO / NO-GO): Every SPAD review ends with one of three verdicts: GO (continue), GO WITH CHANGES (continue after fixing what was flagged) or NO-GO (return to the relevant phase). A single vocabulary across all reviews avoids interpretation. The link opens the verdicts in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#verdicts

SOURCES AND LINKS

- **SPAD 01 · Roles** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#roles
- **SPAD 00 · Roles** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#roles
- **SPAD 03 · Self-approval** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#3-5-self-approval
- **Roles in the operating guide** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#roles
- **Self-approval: a cause of invalidity** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#3-5-self-approval



Notes

The validation policy is a human governance policy, not an instruction for the AI. The person orchestrating applies it and it is mandatory in all SPAD work, complete or reduced.

The five invalidation causes:

- Phase violation: the plan includes code or the review fixes instead of evaluating.
- Incomplete artefact: for example, a review without a verdict or a test strategy without minimum coverage.
- Off-plan decision: the implementation adds a layer that the plan did not provide for, even if it is well reasoned.
- Implementation without prior review: code without an approved plan or after a NO-GO. All that code is discarded.
- Self-approval: “there are three problems, but they are minor: GO”. It is the most serious and is treated as a critical violation.

What does not invalidate: a well-founded NO-GO, a detected vulnerability or a failing test. That is the process working. Nor does it invalidate when the AI declares a deviation and asks for a decision.

There are three verdicts, used in all reviews: GO, GO WITH CHANGES and NO-GO. A verdict without a written justification is an incomplete artefact. A GO with open critical or high findings is a self-approval. The AI Reviewer issues the verdict and the person accepts or rejects it, and records it.

If the same violation is repeated, the instruction is reinforced. After the maximum number of attempts set in the global context (starting value: three), the model is changed or a person steps in.

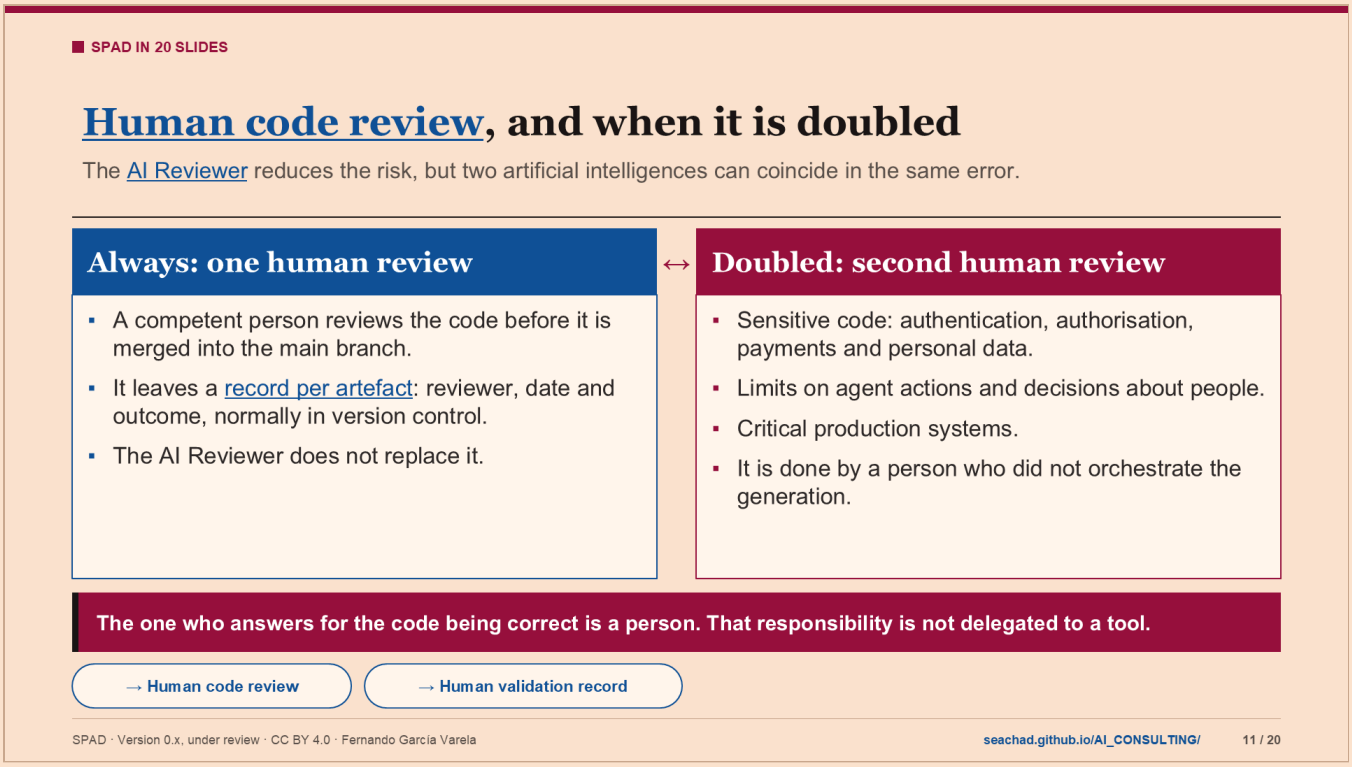
Classroom question: why is self-approval more serious than a plan that contains code?

TERMS ON THIS SLIDE

- **Validation policy**: A SPAD document that sets what a person must validate, when and how rigorously, depending on the risk of the change, and what evidence is kept of that validation. It is the guarantee that AI does not decide alone what goes into the code. The link opens the document. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html
- **Human code review**: In SPAD, code generated by AI is always reviewed by a person before it is merged, with a second review for sensitive code, autonomous agents and critical production. AI helps to build, but it does not approve itself. The link opens the rule in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#3-1-human-code-review
- **SPAD verdicts** (GO / NO-GO): Every SPAD review ends with one of three verdicts: GO (continue), GO WITH CHANGES (continue after fixing what was flagged) or NO-GO (return to the relevant phase). A single vocabulary across all reviews avoids interpretation. The link opens the verdicts in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#verdicts

SOURCES AND LINKS

- **SPAD 03 · Invalidation causes** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#invalidation-causes
- **SPAD 03 · What does not invalidate a response** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#what-does-not-invalidate-a-response
- **Invalidation causes** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#invalidation-causes



Notes

SPAD places two layers of review over the code: the AI Reviewer, which works in phases 7 and 8, and human review.

Human review is mandatory in all SPAD work. At least one competent person reviews the code before it is merged into the main branch and leaves a record of who reviewed it, when and with what outcome. This is normally kept in the version control tool.

The review is doubled when an error would be expensive: authentication, authorisation, payments or personal data code; limits on agent actions; decisions about people; and critical production systems. The second review is done by a person who did not orchestrate the generation.

Automated review never replaces human review: the AI Reviewer reduces the risk, but the accountability for merging the code lies with a person, who is noted in the human validation record.

Simple example: a change to the login screen is generated by the AI, reviewed by the AI Reviewer and reviewed by a person on the team. Because it touches authentication, a second person, who did not ask the AI for the change, reviews it as well.

Why it matters: separation between AIs is necessary but not sufficient. Final accountability to the organisation lies with a person.

Common mistake: treating code as reviewed because “another AI already reviewed it”.

Classroom question: what profile should the person doing the second review have in your organisation?

TERMS ON THIS SLIDE

- **Human code review**: In SPAD, code generated by AI is always reviewed by a person before it is merged, with a second review for sensitive code, autonomous agents and critical production. AI helps to build, but it does not approve itself. The link opens the rule in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#3-1-human-code-review

- **SPAD roles**: The roles in SPAD: the orchestrator (a person who leads the cycle and validates) and several AIs with separate functions: planner, reviewer, builder, fixer, analyst and diagnostic. Whoever builds does not review what they build. The link opens the SPAD presentation document, which describes each role. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html
- **Artefact records**: Each SPAD artefact carries three records: the model record (supplier, model, version, instruction and contexts used), the human validation record (who reviewed it and when) and the violation log (rules not complied with). They make the process auditable. The link opens the records. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#4-1-model-record

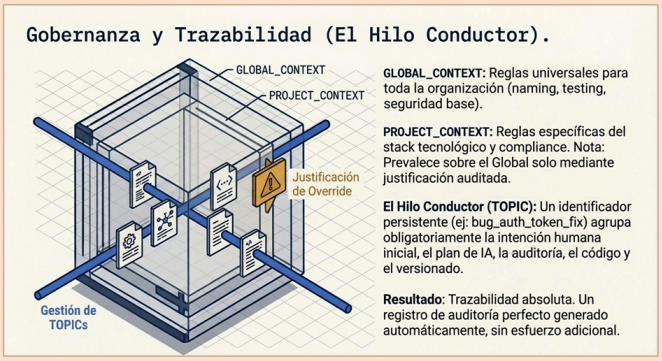
SOURCES AND LINKS

- **SPAD 02 · Human validation record** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#4-2-human-validation-record
- **Human validation record** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#4-2-human-validation-record

SPAD IN 20 SLIDES

Contexts, topics and records

What rules the AI works with, where all the artefacts are and who validated each one.



- Global and project context. The project one prevails only with a written exception approved by a person.
- Topic: a short name, such as pdf_export, that groups all the artefacts, including rejected ones.
- Three records per artefact: the model record, the human validation record and the violation log.
- SPAD plate, in Spanish.

The topic folder is the unit of audit: the intention, the plan, the reviews, the code and who validated each step.

→ Contexts, topics and register | → Records per artefact

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 12 / 20

Notes

Three mechanisms make work with SPAD consistent across projects and reconstructable afterwards.

Contexts. Two are loaded before any phase:

- Global context: rules for the whole organisation. Where artefacts are stored, minimum test coverage, versioning, baseline security, data and model providers, prohibitions and approved models.
- Project context: technology, architecture, business rules, compliance and justified exceptions.

If they clash, the project context prevails only with a written exception, including its risk. The AI Reviewer evaluates it and a person approves it. No AI approves an exception.

Topics. Each piece of work has a topic: a short name in snake_case format, up to 20 characters, such as user_auth or payment_gateway. It groups all the artefacts in one folder. A rejected plan is not deleted: it is kept with its review and the next iteration takes the next number.

Records. Each artefact carries:

- Model record: provider, model, version, date, role, instruction and version of the contexts.
- Human validation record: who validated, when, with what outcome and whether they accepted the AI's verdict.
- Violation log: each discarded response, with its cause.

Illustrative example from document 02: the global context requires 80 % unit coverage; a migration project sets 70 % for new code, with a gradual plan. The technical lead approves it and it is recorded. The figures are illustrative starting values.

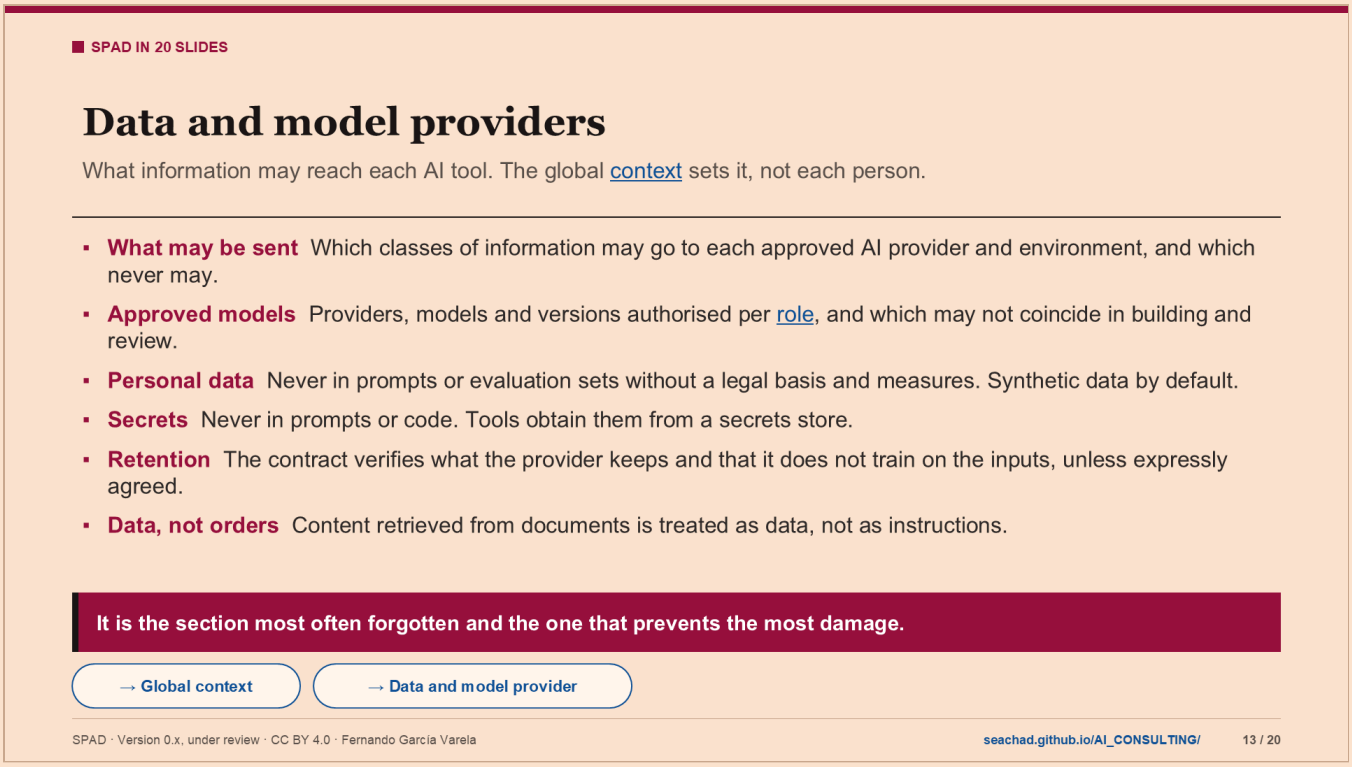
Classroom question: if an auditor asked you tomorrow why a component was built this way, where would you look for it today?

TERMS ON THIS SLIDE

- **SPAD contexts**: Rule documents the AI must respect in each phase: the global context (organisation rules, data, approved models) and the project context. Exceptions are assessed by the reviewer AI and approved by a person. They ensure the AI does not work on its own assumptions. The link opens the contexts. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#contexts
- **SPAD topics**: A topic is SPAD's unit of work: a feature or a change that goes through the full cycle with its own artefact folder. It lets you follow from start to finish what was asked, planned and built, and who validated it. The link opens its definition. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#topics
- **Artefact records**: Each SPAD artefact carries three records: the model record (supplier, model, version, instruction and contexts used), the human validation record (who reviewed it and when) and the violation log (rules not complied with). They make the process auditable. The link opens the records. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#4-1-model-record

SOURCES AND LINKS

- **SPAD 02 · Records per artefact** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#records-per-artefact
- **Contexts, topics and register** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html
- **Records per artefact** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#records-per-artefact



Notes

When a team works with AI, it sends code, documents and sometimes data to the tool. Without a common rule, each person decides alone what to paste into an external tool. SPAD solves this in the global context, with two sections: data and model providers, and approved models.

- What may be sent: the global context states which classes of information may be sent to each approved provider and environment, and which never may: for example, personal data, secrets or proprietary code, according to their classification.
- Approved models: which providers, models and versions are authorised for each role. In sensitive code, the AI Reviewer uses a model or provider different from the builder's.
- Personal data: never in prompts or evaluation sets without a documented legal basis and measures. Synthetic data by default.
- Secrets: never in prompts or code.
- Retention and use by the provider: verified contractually before the tool is used.
- Separation: system instructions and what comes from the user or from retrieved documents are kept separate. Retrieved content is treated as data, not as orders.

Illustrative example: a rule such as “no real personal data in prompts to external providers; synthetic test data”.

Why it matters: it is the link with the corporate AI use policy. And it is one of the first things a company can ask a supplier that builds with AI for it.

Classroom question: does your team know what data it may paste into an AI tool today, and who decided it?

TERMS ON THIS SLIDE

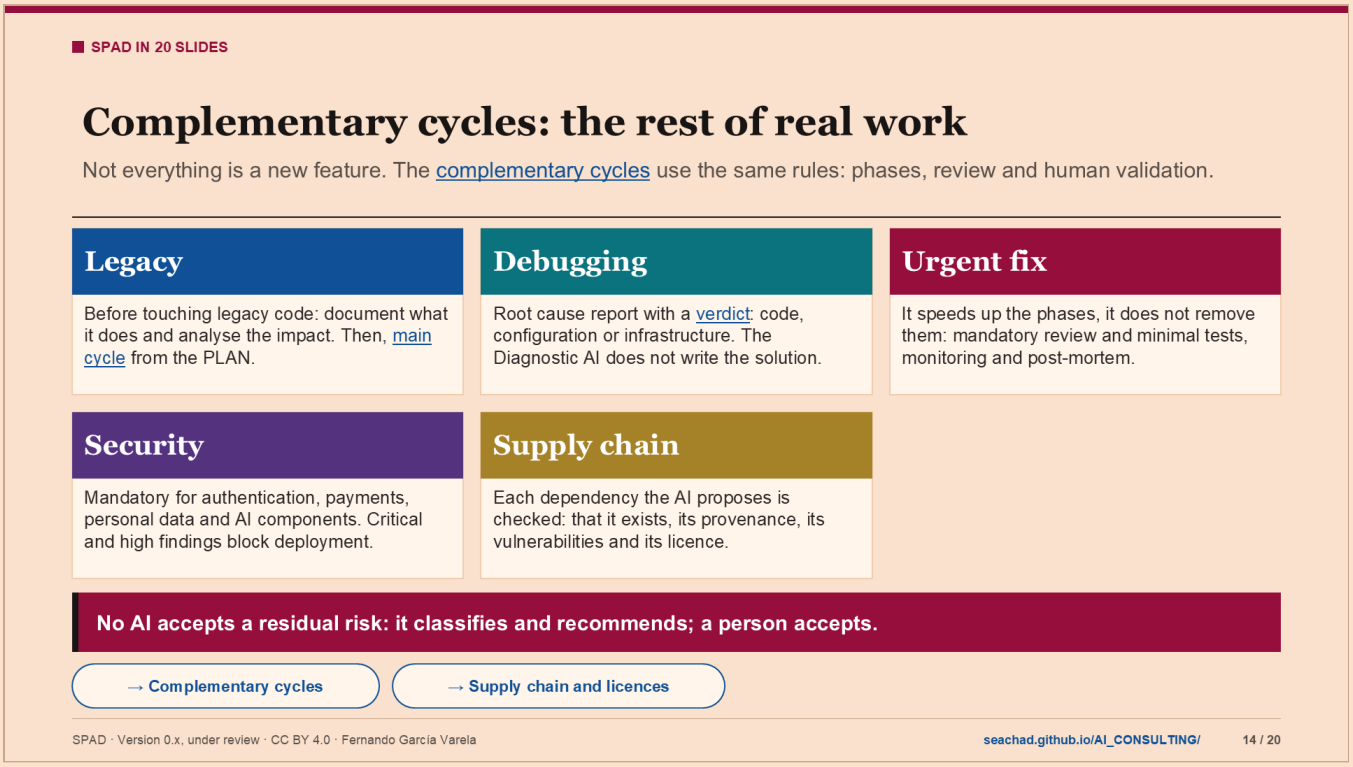
- **SPAD contexts**: Rule documents the AI must respect in each phase: the global context (organisation rules, data, approved models) and the project context. Exceptions are assessed by the reviewer AI and approved by a person.

They ensure the AI does not work on its own assumptions. The link opens the contexts. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#contexts

- **SPAD roles**: The roles in SPAD: the orchestrator (a person who leads the cycle and validates) and several AIs with separate functions: planner, reviewer, builder, fixer, analyst and diagnostic. Whoever builds does not review what they build. The link opens the SPAD presentation document, which describes each role. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html

SOURCES AND LINKS

- **SPAD 02 · Global context** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#2-1-global-context
- **SPAD 05 · Data and model provider** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#data-and-model-provider
- **Global context** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#2-1-global-context
- **Data and model provider** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#data-and-model-provider



Notes

The main cycle builds new functionality. SPAD document 04 defines four cycles for the rest of real work, plus the supply chain review.

- Legacy: existing code without enough documentation has to be changed. The AI Analyst documents what it does today and analyses the impact of the change. It describes, it does not propose solutions. If there are no tests of the current behaviour, the test strategy includes characterisation tests before modifying it.
- Debugging: a problem appears. The Diagnostic AI produces a root cause report with a verdict. If the cause is the code, the main cycle is entered; if it is configuration or infrastructure, a recorded change with rollback is made.
- Urgent fix: for critical incidents. Rapid diagnosis, simplified plan, accelerated but mandatory review, implementation with minimal tests, deployment with emergency approval by a person, monitoring and post-mortem. A provisional fix always opens a topic for the definitive plan.
- Security: mandatory for sensitive code. Critical findings block and nobody can accept them as residual risk. High findings block unless the risk authority gives an exceptional written acceptance.
- Supply chain: the AI may suggest non-existent or malicious packages. Each dependency and its licence are checked.

Simple example: a production failure caused by a configuration file is fixed as a configuration change. The root cause report is kept anyway: it is organisational learning.

Classroom question: in an emergency, which control would your team tend to skip first?

TERMS ON THIS SLIDE

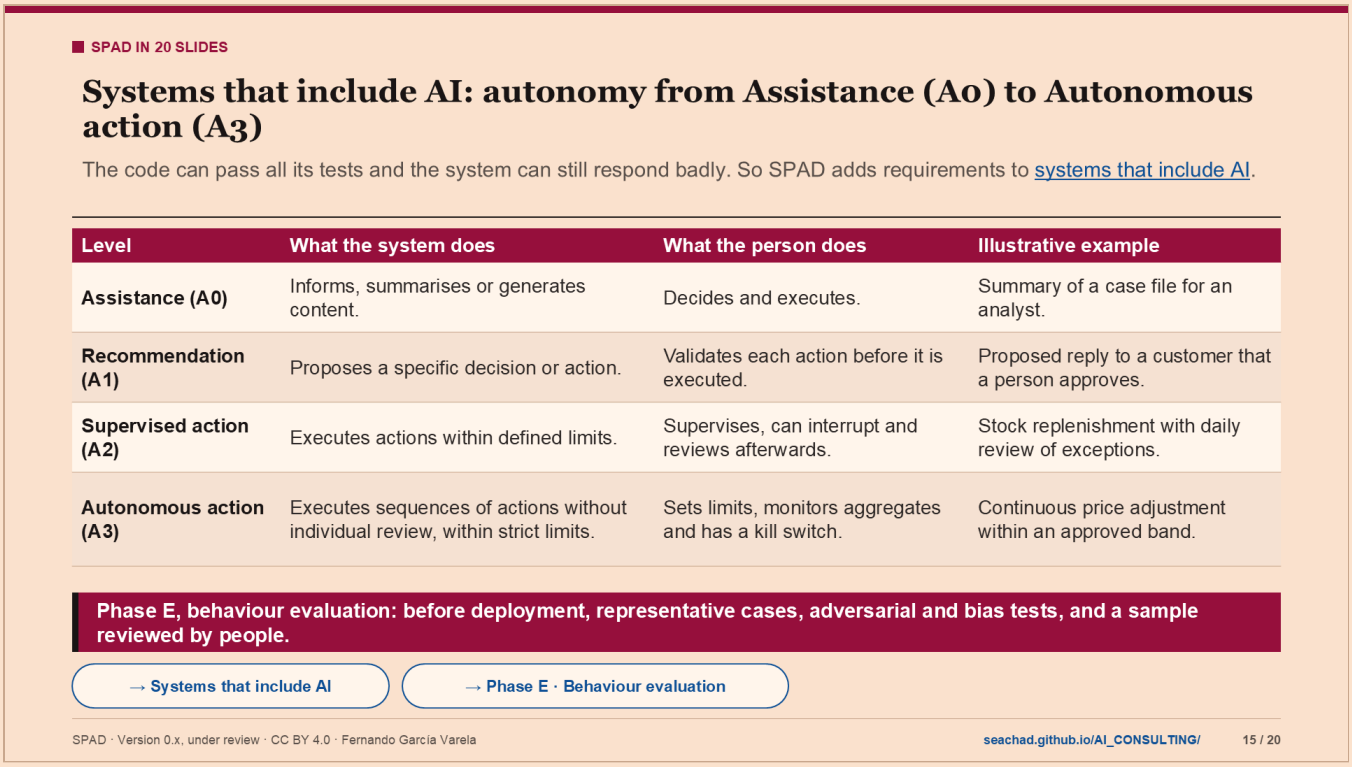
- **SPAD complementary cycles**: Variants of the cycle for special situations: legacy code (with characterisation tests), debugging, urgent fixes (with a mandatory follow-up analysis), security and supply chain. They keep control

when the full cycle cannot be followed. The link opens the document. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html

- **SPAD main cycle** (fases 0–10): SPAD's eleven phases: context, plan, plan review, code guide, test strategy, implementation, testing, test review, code review, fixes and release management. Each one leaves an artefact that is reviewed. The link opens the cycle in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#main-cycle-phases-0-to-10
- **SPAD verdicts** (GO / NO-GO): Every SPAD review ends with one of three verdicts: GO (continue), GO WITH CHANGES (continue after fixing what was flagged) or NO-GO (return to the relevant phase). A single vocabulary across all reviews avoids interpretation. The link opens the verdicts in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#verdicts

SOURCES AND LINKS

- **SPAD 04 · Legacy cycle** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html#legacy-cycle
- **SPAD 04 · Debugging cycle** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html#debugging-cycle
- **SPAD 04 · Urgent fix cycle** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html#urgent-fix-cycle
- **SPAD 04 · Severities, blocking and risk acceptance** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html#5-3-severities-blocking-and-risk-acceptance
- **OWASP Top 10 for Large Language Model Applications (2025 edition)** https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/
- **Supply chain and licences** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html#5-4-supply-chain-and-licences



SPAD IN 20 SLIDES

Systems that include AI: autonomy from Assistance (A0) to Autonomous action (A3)

The code can pass all its tests and the system can still respond badly. So SPAD adds requirements to systems that include AI.

Level	What the system does	What the person does	Illustrative example
Assistance (A0)	Informs, summarises or generates content.	Decides and executes.	Summary of a case file for an analyst.
Recommendation (A1)	Proposes a specific decision or action.	Validates each action before it is executed.	Proposed reply to a customer that a person approves.
Supervised action (A2)	Executes actions within defined limits.	Supervises, can interrupt and reviews afterwards.	Stock replenishment with daily review of exceptions.
Autonomous action (A3)	Executes sequences of actions without individual review, within strict limits.	Sets limits, monitors aggregates and has a kill switch.	Continuous price adjustment within an approved band.

Phase E, behaviour evaluation: before deployment, representative cases, adversarial and bias tests, and a sample reviewed by people.

→ Systems that include AI → Phase E · Behaviour evaluation

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 15 / 20

Notes

SPAD was created to govern code written with the help of AI. More and more, what is built is a system that includes AI: an application whose behaviour depends, in production, on a model, on instructions, on a knowledge base or on an agent that executes actions. That system can change without the code changing: when the provider updates the model, or when the instructions or the data change.

Every AI component that decides or acts is classified from the plan onwards with a four-level autonomy scale (A0 to A3). The level sets the requirements: A0 and A1 require behaviour evaluation; A2 and A3 also require action limits, an action log and a tested kill switch.

Phase E, behaviour evaluation, sits between the test review and the code review. The system is run on a versioned set of representative cases, including those it must refuse, and compared with thresholds set in advance. It includes adversarial tests (for example, attempts to inject instructions through a document) and bias tests with counterfactual pairs when the system decides about people or communicates with them. People review a sample. It ends with a verdict.

Simple example: a customer service assistant passes all the code tests, but the evaluation shows that it answers questions outside its scope with confidence. It is an instructions NO-GO: it goes back to implementation.

Why it matters: without evaluation, the only proof that an assistant answers well is a demo with the questions that occurred to someone.

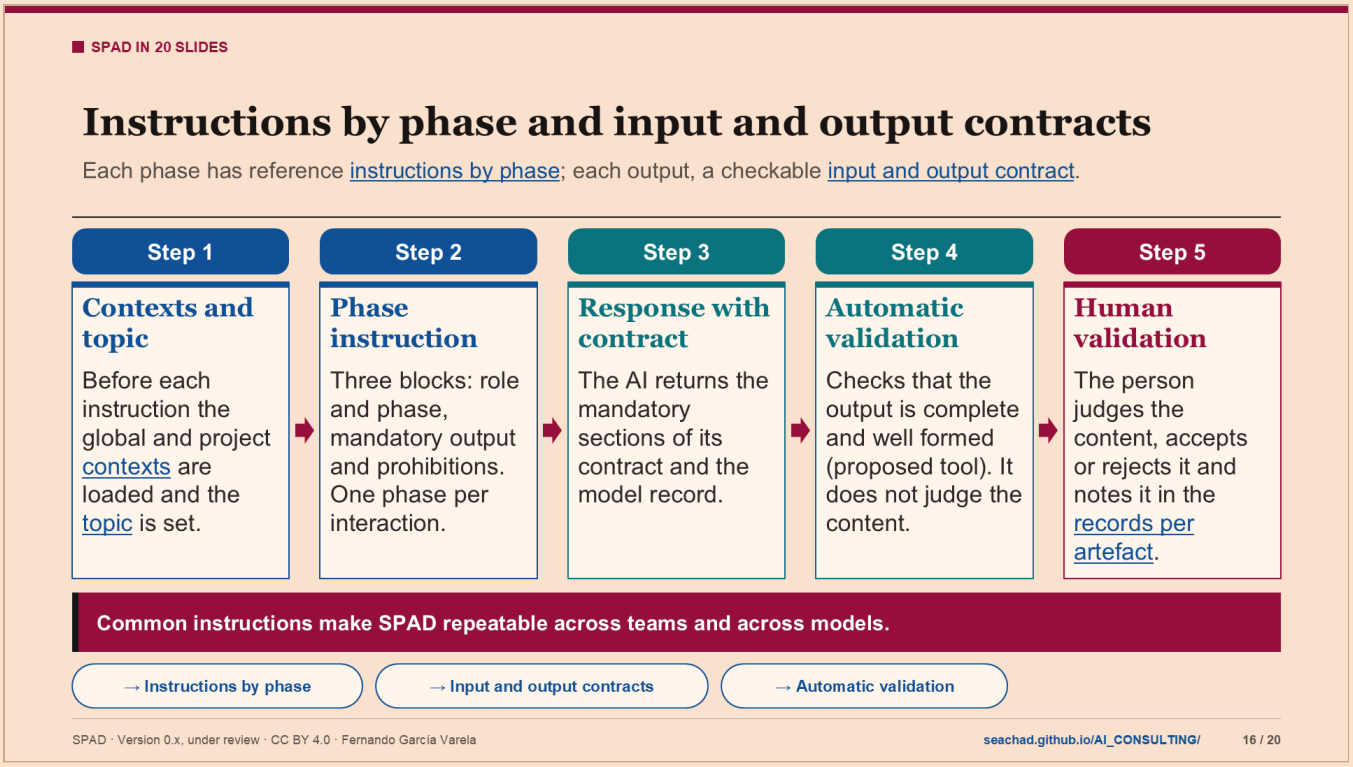
Classroom question: what should happen when the provider updates the model your system uses?

TERMS ON THIS SLIDE

- **Systems that include AI**: What SPAD adds when what is being built is a system that uses AI: autonomy levels, a behaviour evaluation phase with test sets, adversarial and bias tests, an agent plan and control of changes that do not touch code (such as changing the model). The link opens the document. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html

SOURCES AND LINKS

- **SPAD 05 · Two distinct objects** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#two-distinct-objects
- **SPAD 05 · Autonomy level** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#autonomy-level
- **SPAD 05 · Phase E · Behaviour evaluation** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#phase-e-behaviour-evaluation
- **SPAD 05 · Agents** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#agents
- **Phase E · Behaviour evaluation** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#phase-e-behaviour-evaluation



Notes

This slide gives the general idea of two technical SPAD documents: 06, instructions by phase, and 07, input and output contracts.

Instructions by phase. Each phase opens with a reference instruction (prompt) with three blocks:

- Role and phase: which role the AI takes on and which phase it is in.
- Mandatory output: the sections the response must contain, in order.
- Prohibitions: what the phase does not allow, for example, writing code in the plan.

All of them start with a common preamble that recalls the phase, the role, the contexts and the topic, and asks the AI not to decide when something is not covered. Each phase runs in its own interaction: phases are not merged. Organisations adapt the text, but keep the three blocks.

Contracts. Each artefact has a structure with mandatory sections and fields. The canonical form is JSON, although it can be presented in Markdown. All share a common block: topic, phase, iteration, date and model record, and use the same verdict and severity scales.

Automatic validation: a validator checks that nothing is missing and detects, for example, a GO with open critical problems. It does not judge the content: that belongs to the AI Reviewer and to the person. The tool is proposed as a next step of SPAD; in the meantime, the check can be done by hand.

Why it matters: “incomplete artefact” is the invalidation cause that is easiest to automate. The contract turns it into a mechanical step and leaves to the person the judgement that really requires a person.

Classroom question: what are the advantages of two teams using the same instruction for the same phase?

TERMS ON THIS SLIDE

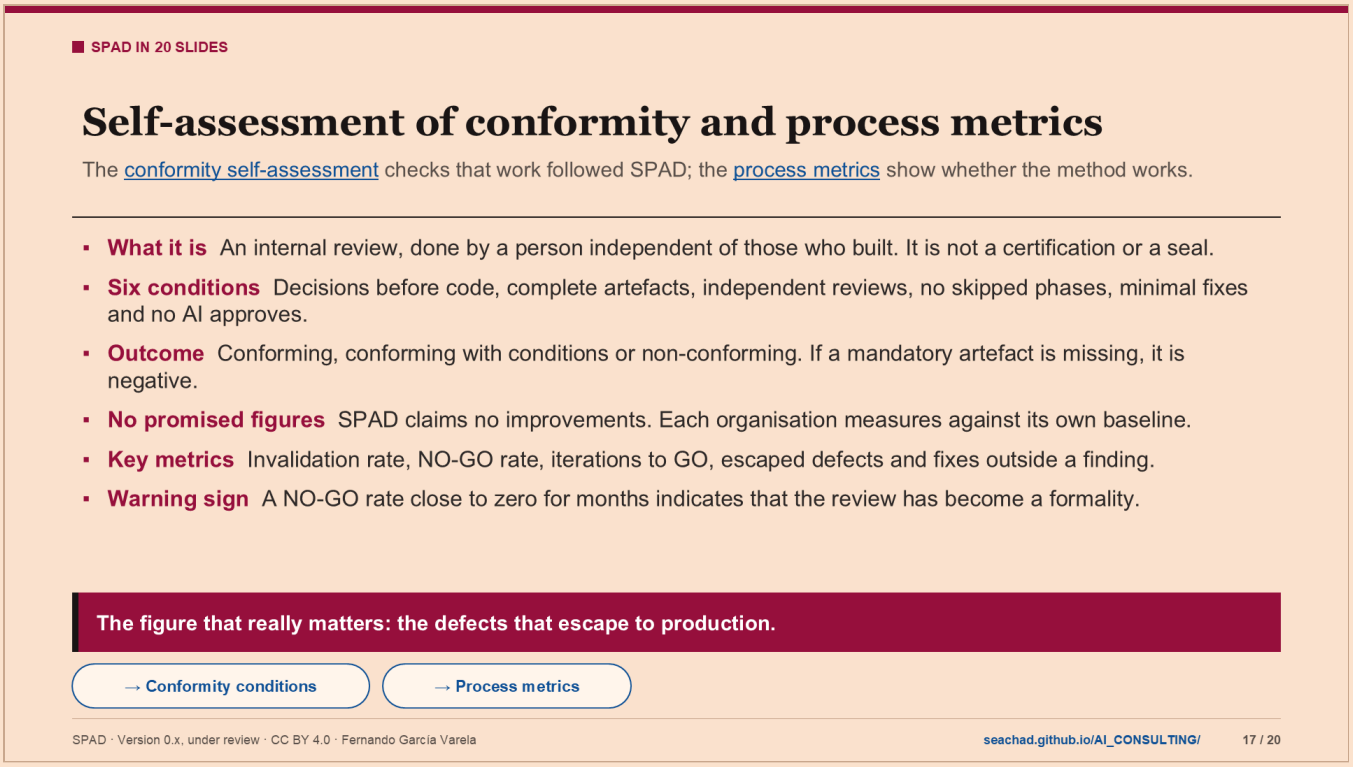
- **Instructions by phase**: The instructions given to the AI in each SPAD phase, with a common preamble and what is specific to each phase: what it must read, what it must produce and what it must not do. They give repeatable

results that can be compared across teams. The link opens the document. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/06_SPAD_Instrucciones_por_fase.html

- **Input and output contracts**: The structure that what goes into and comes out of each SPAD phase must have, with a common block, so that a tool can validate it automatically. That way an incomplete artefact does not move to the next phase. The link opens the document. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/07_SPAD_Contratos_de_entrada_y_salida.html
- **SPAD contexts**: Rule documents the AI must respect in each phase: the global context (organisation rules, data, approved models) and the project context. Exceptions are assessed by the reviewer AI and approved by a person. They ensure the AI does not work on its own assumptions. The link opens the contexts. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#contexts
- **SPAD topics**: A topic is SPAD's unit of work: a feature or a change that goes through the full cycle with its own artefact folder. It lets you follow from start to finish what was asked, planned and built, and who validated it. The link opens its definition. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#topics
- **Artefact records**: Each SPAD artefact carries three records: the model record (supplier, model, version, instruction and contexts used), the human validation record (who reviewed it and when) and the violation log (rules not complied with). They make the process auditable. The link opens the records. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#4-1-model-record

SOURCES AND LINKS

- **SPAD 06 · How to use this document** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/06_SPAD_Instrucciones_por_fase.html#how-to-use-this-document
- **SPAD 06 · Common preamble** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/06_SPAD_Instrucciones_por_fase.html#common-preamble
- **SPAD 07 · Common conventions** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/07_SPAD_Contratos_de_entrada_y_salida.html#common-conventions
- **SPAD 07 · Automatic validation** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/07_SPAD_Contratos_de_entrada_y_salida.html#automatic-validation
- **Automatic validation** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/07_SPAD_Contratos_de_entrada_y_salida.html#automatic-validation



Notes

The self-assessment of conformity is done by the organisation itself. A designated person, independent of those who built and orchestrated, checks six conditions against the evidence of the topic:

- Decisions were taken before code was written.
- Each phase produced its complete artefact.
- The reviews were independent.
- No phase was skipped or merged.
- Fixes were minimal, justified and traceable.
- No AI approved anything.

If the system includes AI, there is a seventh: the behaviour evaluation was done before deployment and is repeated with each change of model, instructions or data.

It is not a certification. Neither its author nor any third party issues or endorses it, and it does not prove compliance with any standard. It is always recorded as “conforming with SPAD (self-assessment)”. Its validity is set by the global context (starting value: twelve months).

Process metrics. SPAD claims no improvement figures. Each organisation measures against its own baseline using the records: invalidation rate per phase and per model, NO-GO rate, iterations to GO, time per phase, defects escaped to production, actual versus required coverage and fixes without an associated finding. As outcomes, rework, incidents, audit preparation time and time to production are compared before and after.

Common mistake: celebrating a zero NO-GO rate. Sustained for months, it indicates a nominal review.

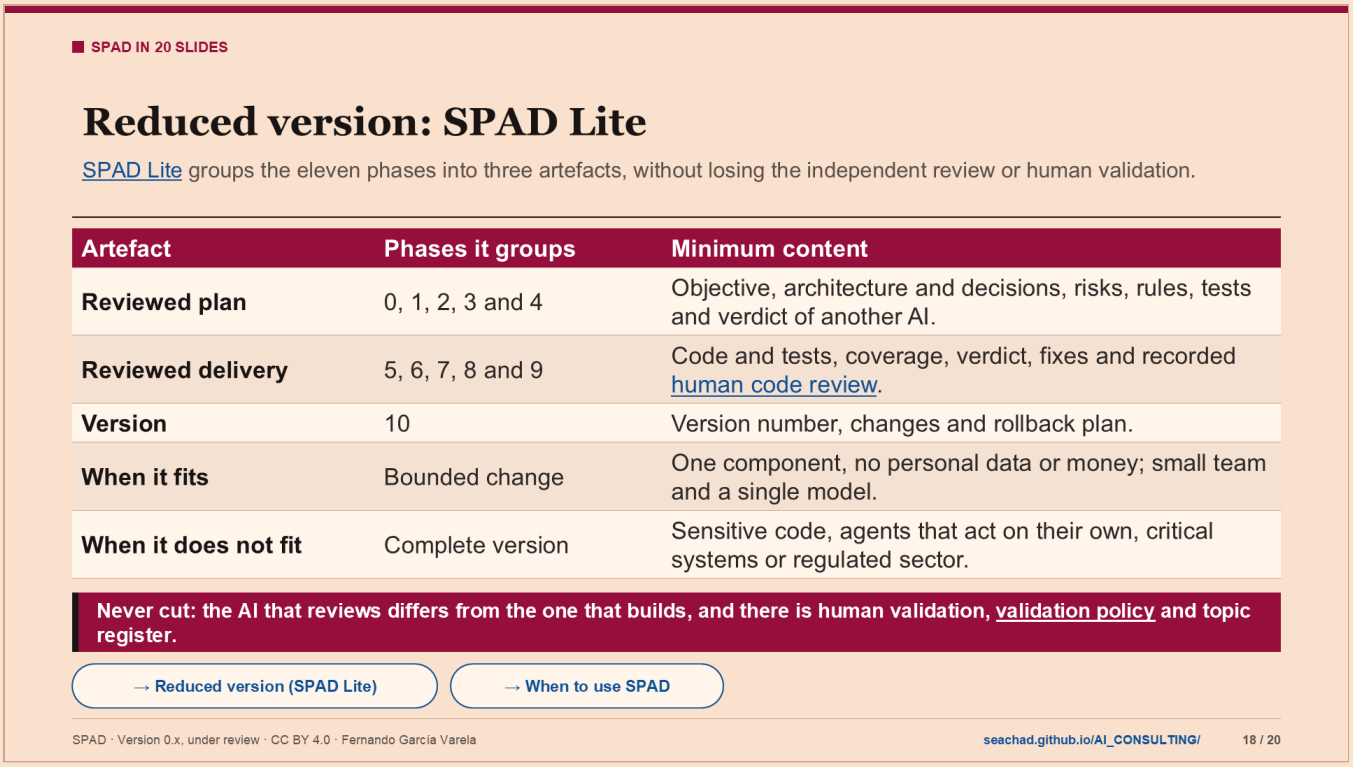
Classroom question: which metric would you present to a management committee to decide whether to keep using SPAD?

TERMS ON THIS SLIDE

- **Conformity self-assessment**: A review with which a team checks whether it applies SPAD as it claims: which phases it follows, which records it keeps and what fails. It is valid for twelve months and nobody outside grants it. It helps improve the process with data rather than impressions. The link opens the self-assessment. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html#what-the-self-assessment-is
- **Process metrics**: Indicators measured by a team using SPAD, such as how many reviews end in NO-GO, how many defects reach production or how long a topic takes. SPAD promises no figures: it proposes measuring them and setting your own targets. The link opens the metrics. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html#process-metrics

SOURCES AND LINKS

- **SPAD 08 · Conformity conditions** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html#conformity-conditions
- **SPAD 08 · When SPAD is worth using, according to the metrics** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html#when-spad-is-worth-using-according-to-the-metrics
- **Conformity conditions** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html#conformity-conditions



Notes

Not all work needs eleven separate artefacts. The reduced version groups the phases into three artefacts: reviewed plan, reviewed delivery and version.

Criteria for using it: a bounded change in one component, with no personal data or money and no effect on decisions about people; a team of one or two people with a single model; and no external audit obligations on the development.

Criteria that require the complete version: sensitive code, agents with Supervised action (A2) or Autonomous action (A3) autonomy, critical production systems, several teams or model providers, or a regulated sector with development traceability requirements.

What is not cut in any version: the separation between the AI that builds and the one that reviews, the human validation of each artefact, the validation policy and the topic register.

The grouping only holds if the minimum content of each artefact is kept and the human review is recorded.

Simple example: adding a date filter to an internal report, with no personal data. A single plan reviewed by another AI, a reviewed delivery with its human review and a version with its rollback plan are enough.

Why it matters: without a reduced version, teams apply SPAD to large projects and avoid it in small ones, which are the majority and where silent debt accumulates.

Classroom question: which pieces of work in your organisation would fit the reduced version?

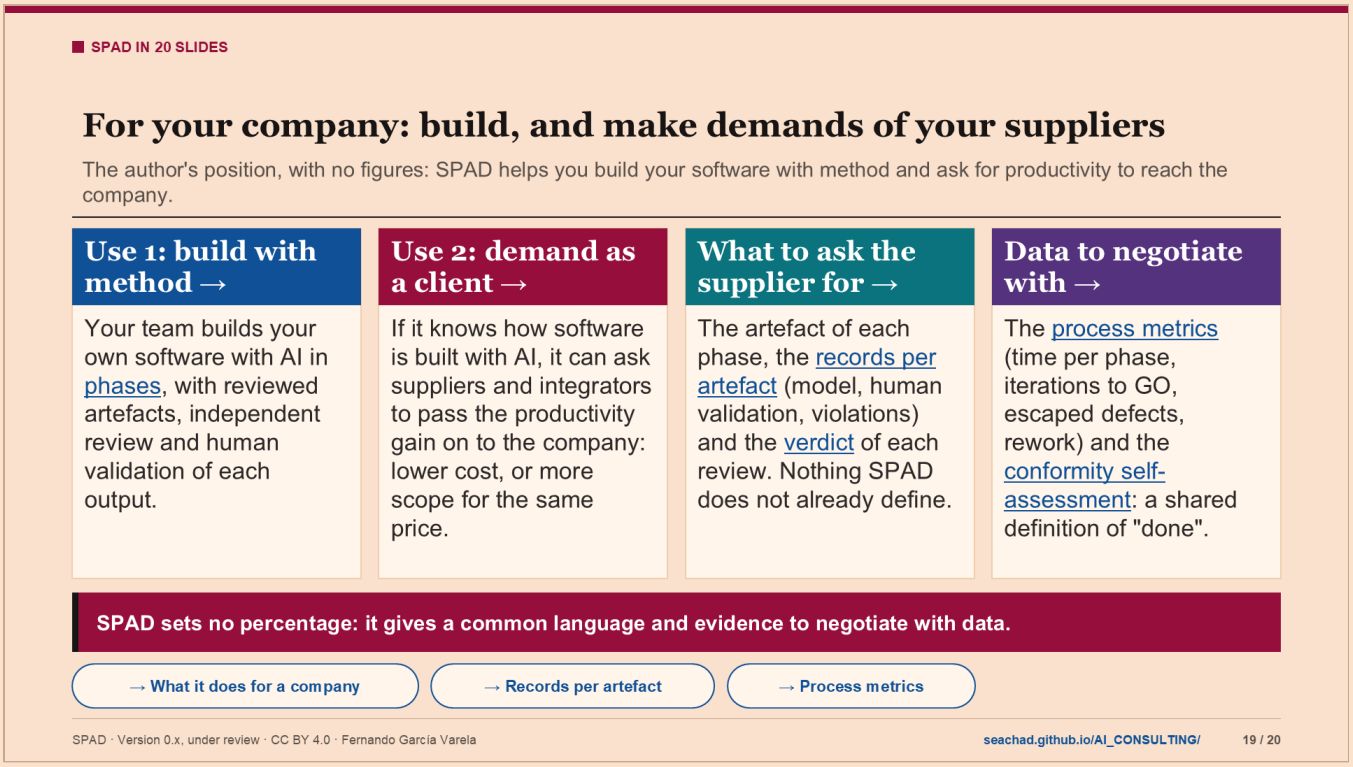
TERMS ON THIS SLIDE

- **SPAD Lite**: A reduced version of SPAD for small, low-risk changes, with only three artefacts. It keeps review and human validation, but with fewer steps. It avoids loading a minor change with the whole cycle. The link opens the reduced version in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#reduced-version-spad-lite

- **Human code review**: In SPAD, code generated by AI is always reviewed by a person before it is merged, with a second review for sensitive code, autonomous agents and critical production. AI helps to build, but it does not approve itself. The link opens the rule in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#3-1-human-code-review
- **Validation policy**: A SPAD document that sets what a person must validate, when and how rigorously, depending on the risk of the change, and what evidence is kept of that validation. It is the guarantee that AI does not decide alone what goes into the code. The link opens the document. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html

SOURCES AND LINKS

- **SPAD 00 · When to use SPAD** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#when-to-use-spad
- **When to use SPAD** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#when-to-use-spad



Notes

SPAD solves a specific problem: how software is built with AI. In the author's position, it serves a company in two ways. SPAD promises no savings or productivity figures: each organisation measures its own.

- Use 1, building your own software with AI. Your team works in phases, with a reviewed artefact in each one, an AI Reviewer different from the builder and human validation of each output.
- Use 2, demanding as a client. AI changes what it costs to build software. A company that knows how software is built with AI can ask its suppliers and integrators to pass that productivity gain on to it: lower cost, or more scope for the same price.

What it can ask for, without inventing anything SPAD does not already define:

- The artefact of each phase: plan, plan review, test strategy, code review and version record. This shows that the design was decided before the code.
- The model record of each artefact: which part the AI did and with which tool.
- The human validation record and the violation log: that a person approved what is delivered and which responses were discarded.
- The verdict of each review: where work went back and why.
- The process metrics (time per phase, iterations to GO, escaped defects, rework) and the self-assessment of conformity: a shared definition of "done".

Why it matters: if the company does not know how software is built with AI, it cannot know whether the change in cost reaches its invoice or stays with the supplier. SPAD sets no percentage: it lets you negotiate with the supplier's own data.

Illustrative example: faced with a development offer, the company asks the integrator for the time per phase and the model record of a previous piece of work, and with that data discusses price and scope.

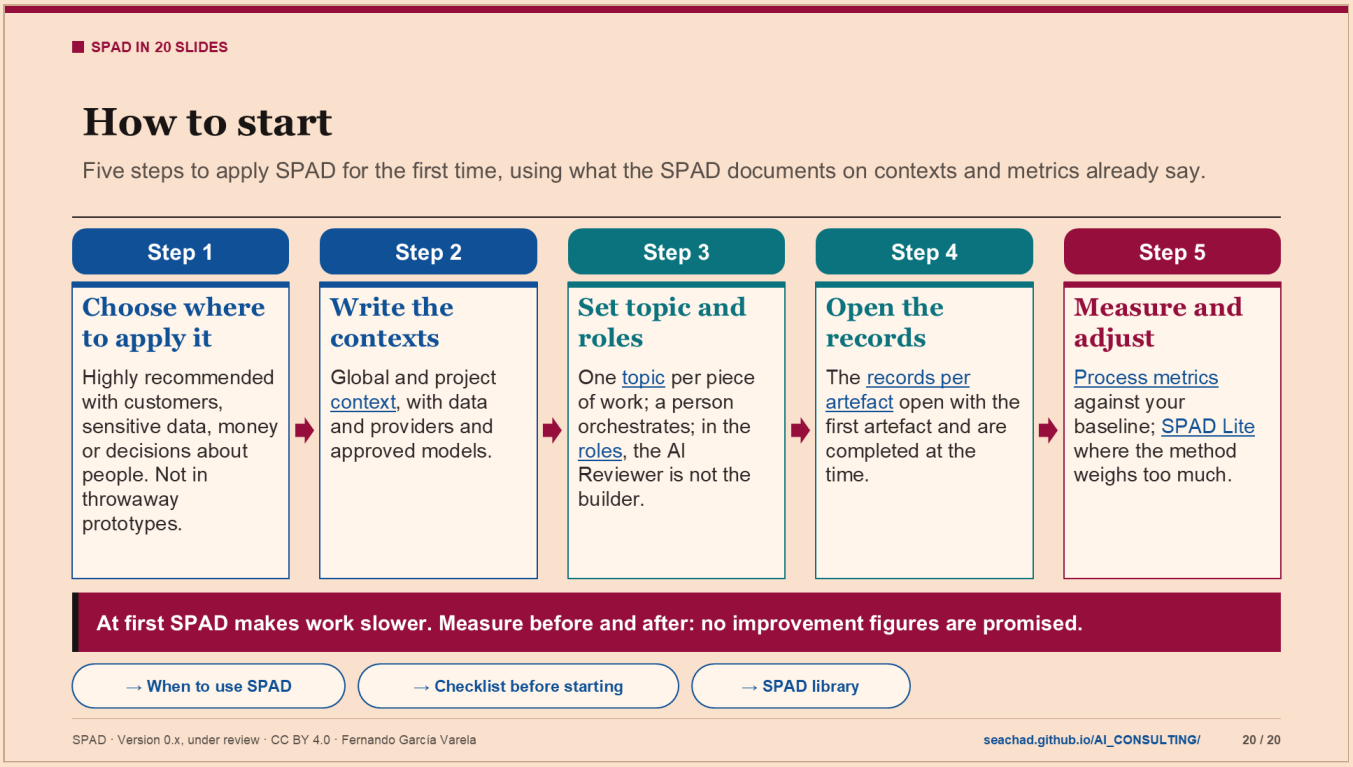
Classroom question: what would you ask your main software supplier today to find out whether it uses AI and who keeps the gain?

TERMS ON THIS SLIDE

- **SPAD main cycle** (fases 0–10): SPAD's eleven phases: context, plan, plan review, code guide, test strategy, implementation, testing, test review, code review, fixes and release management. Each one leaves an artefact that is reviewed. The link opens the cycle in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#main-cycle-phases-0-to-10
- **Artefact records**: Each SPAD artefact carries three records: the model record (supplier, model, version, instruction and contexts used), the human validation record (who reviewed it and when) and the violation log (rules not complied with). They make the process auditable. The link opens the records. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#4-1-model-record
- **SPAD verdicts** (GO / NO-GO): Every SPAD review ends with one of three verdicts: GO (continue), GO WITH CHANGES (continue after fixing what was flagged) or NO-GO (return to the relevant phase). A single vocabulary across all reviews avoids interpretation. The link opens the verdicts in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#verdicts
- **Process metrics**: Indicators measured by a team using SPAD, such as how many reviews end in NO-GO, how many defects reach production or how long a topic takes. SPAD promises no figures: it proposes measuring them and setting your own targets. The link opens the metrics. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html#process-metrics
- **Conformity self-assessment**: A review with which a team checks whether it applies SPAD as it claims: which phases it follows, which records it keeps and what fails. It is valid for twelve months and nobody outside grants it. It helps improve the process with data rather than impressions. The link opens the self-assessment. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html#what-the-self-assessment-is

SOURCES AND LINKS

- **SPAD 00 · What it does for a company** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#what-it-does-for-a-company
- **SPAD 02 · Records per artefact** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#records-per-artefact
- **What it does for a company** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#what-it-does-for-a-company
- **Records per artefact** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#records-per-artefact



Notes

To start, you do not need to ask for permission: SPAD is published under CC BY 4.0 (content) and MIT (code), with the only condition of citing the authorship: SPAD · Fernando García Varela.

- Step 1, choose where to apply it. Document 00 says so: highly recommended in production systems with customers, sensitive data, money or decisions about people, and in sectors with audit requirements; recommended with several developers or a lot of technical debt; useful, in the reduced version, in internal tools; unnecessary in throwaway prototypes, personal programs or pure research.
- Step 2, write the global and project contexts. The data and providers section and the approved models section must not be missing.
- Step 3, set the topic in phase 0 and assign the roles. The AI Reviewer is not the builder and a person orchestrates.
- Step 4, open the validation record and complete it at the time, not at the end.
- Step 5, measure. The process metrics are calculated from the records and compared with the baseline.

The checklist in document 02 summarises the minimum before starting: contexts with a version, exceptions approved by a person, topic and folder, models assigned per role, validation record open and class of information authorised for each provider.

A separate framework. SPAD deals only with how software is built; it is not the governance of the company's AI. Whoever wants to govern what is built, with what risk and how its value is measured has a separate framework, SEVEN-G. The only bridge is its document 53, and SPAD verdicts do not replace its decision gates.

Common mistake: starting with the largest and most critical project. At first the method makes work slower while the team learns.

Classroom question: which piece of work in your organisation would you choose to apply SPAD for the first time, and why?

TERMS ON THIS SLIDE

- **SPAD contexts**: Rule documents the AI must respect in each phase: the global context (organisation rules, data, approved models) and the project context. Exceptions are assessed by the reviewer AI and approved by a person. They ensure the AI does not work on its own assumptions. The link opens the contexts. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#contexts
- **SPAD topics**: A topic is SPAD's unit of work: a feature or a change that goes through the full cycle with its own artefact folder. It lets you follow from start to finish what was asked, planned and built, and who validated it. The link opens its definition. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#topics
- **SPAD roles**: The roles in SPAD: the orchestrator (a person who leads the cycle and validates) and several AIs with separate functions: planner, reviewer, builder, fixer, analyst and diagnostic. Whoever builds does not review what they build. The link opens the SPAD presentation document, which describes each role. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html
- **Artefact records**: Each SPAD artefact carries three records: the model record (supplier, model, version, instruction and contexts used), the human validation record (who reviewed it and when) and the violation log (rules not complied with). They make the process auditable. The link opens the records. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#4-1-model-record
- **Process metrics**: Indicators measured by a team using SPAD, such as how many reviews end in NO-GO, how many defects reach production or how long a topic takes. SPAD promises no figures: it proposes measuring them and setting your own targets. The link opens the metrics. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html#process-metrics
- **SPAD Lite**: A reduced version of SPAD for small, low-risk changes, with only three artefacts. It keeps review and human validation, but with fewer steps. It avoids loading a minor change with the whole cycle. The link opens the reduced version in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#reduced-version-spad-lite

SOURCES AND LINKS

- **SPAD 00 · When to use SPAD** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#when-to-use-spad
- **SPAD 02 · Checklist before starting** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#checklist-before-starting
- **SEVEN-G 53 · Building solutions with AI (the only bridge)** https://seachad.github.io/AI_CONSULTING/SEVEN-G/html/en/53_SEVEN-G_Construccion_de_soluciones_con_IA.html
- **When to use SPAD** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#when-to-use-spad
- **Checklist before starting** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#checklist-before-starting
- **SPAD library** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/index.html

LEGAL NOTICE

Legal notice: a generic, free methodological aid provided "as is" for information purposes; it is not legal advice and has not been legally reviewed for any case. Each organisation adapts it and is responsible for its legal review and its regulatory compliance; it may not reflect recent regulatory changes. To the fullest extent permitted by law, the author accepts no liability whatsoever for the effects of its application. It does not grant certification of any kind. Details in document 93.

CC BY 4.0 (content) · MIT (code) · Fernando García Varela · seachad.github.io/AI_CONSULTING/