

Notes

SPAD stands for Structured Prompt-Driven Engineering. It is Fernando García Varela's methodology for managing software development when it is done with the help of artificial intelligence.

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

- Sequential: work advances through phases in a fixed order. First the problem is understood, then the solution is designed, the design is reviewed, the tests are defined, the solution is built, what was built is reviewed and it is versioned.
- Blocking: no phase begins until the previous one has produced its output and someone has evaluated it. If a review says no, work goes back.
- Auditable: each phase leaves an explicit document (plan, review report, test strategy, release record) that is kept and is the input to the next one.

The illustration shows the idea with three pillars: think (the design is explicit before a single line of code is written), validate (an independent review) and execute (the AI generates code following rules, without taking design decisions). The foundations are documentation, security and traceability.

Why it matters to an executive: more and more of your company's software is written, in part, by an AI. You do not need to know how to code to require that this work has a method, that someone is accountable for it and that it leaves evidence.

SPAD is open: content under CC BY 4.0 and code under the MIT licence. There is no SPAD certification.

Question for the classroom: in your organisation, who decides today how the software that an AI generates 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

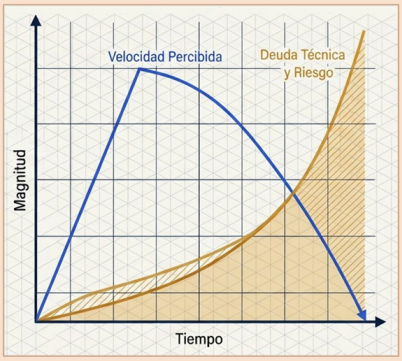
SOURCES AND LINKS

- **SPAD 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 §11 · 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 10 SLIDES

The problem: the illusion of speed

Conceptual chart, no data: without structure, perceived speed falls while technical debt and risk grow.



- **At first** AI generates plausible code very fast. The team seems to fly.
- **Later** Decisions nobody took, few tests and silent vulnerabilities appear: rework.
- **The cause** It is not the AI: it is the lack of method, which AI amplifies.
- SPAD keeps speed where it adds value and puts control where it is needed. Chart in Spanish.

Detecting an error in the plan review costs a conversation; in production, an incident.

→ The problem it solves

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 2 / 10

Notes

The chart is conceptual: it has no data. It shows two curves. Perceived speed rises very quickly at first, because AI writes plausible code in seconds. Over time it falls, while technical debt and risk grow.

What lies behind it, according to the SPAD page:

- The code works in the demo and fails in production, because nobody designed the solution: code was requested until it "worked".
- Nobody knows why the system is built this way, because the AI took the design decisions inside the code.
- Tests are scarce or only test the easy case, because they are written at the end, if time allows.
- Vulnerabilities appear that nobody reviewed, because the review was done by the same AI that wrote the code, or not done at all.
- An audit asks for evidence of the development and there is none, because the work stayed in conversations with the AI.

A simple example: a team builds an internal portal in one week with the help of an AI. Three months later, a small change breaks another part of the system and nobody knows why the portal was built that way. The week gained is lost in fixes.

Why it matters to an executive: none of these problems is caused by AI. They are problems of method that AI amplifies. With a method, quality does not depend on the skill of whoever writes the instructions or on the model of the moment, but on a process that any team can repeat.

SPAD does not promise improvement figures. At first it makes work slower, while the team learns. The hypothesis is that it reduces rework and incidents; each organisation must measure this against its own starting point.

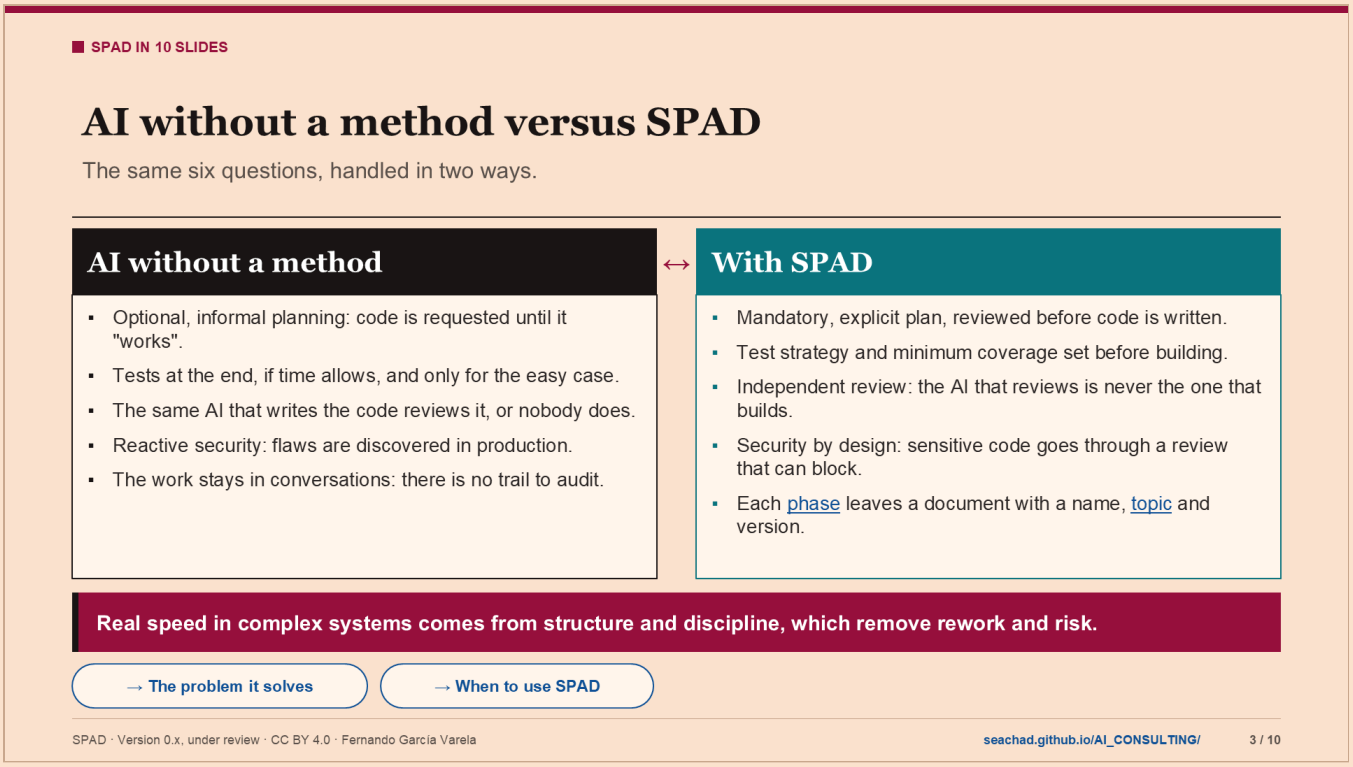
Question for the classroom: how much of your team's "speed" is real speed and how much is debt that it will pay later?

TERMS ON THIS SLIDE

- **SPAD**: SPAD (Structured Prompt-Driven Engineering) is a methodology for building software with AI help in an orderly way: sequential phases, each with its artefact, reviews with a verdict and people who validate. It is sequential, blocking and auditable. Development teams use it. The link opens its presentation document. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html

SOURCES AND LINKS

- **SPAD 00 §2 · 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 00 §10 · 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
- **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 "Traditional AI versus SPAD" illustration on the site. Each row is a question that any manager can ask their team.

- Planning. Without a method, code is requested until it seems to work. With SPAD, the plan is mandatory, explicit and reviewed by a different AI before code is written.
- Testing. Without a method, tests are written at the end. With SPAD, the test strategy and minimum coverage are set before building; later a review compares actual coverage with the required coverage.
- Review. Without a method, the AI reviews itself. With SPAD, the reviewer is never the builder and, in addition, a person reviews the code.
- Security. Without a method, flaws are seen in production. With SPAD, code that handles sensitive data, authentication or payments goes through a security review; critical and high findings block deployment.
- Traceability. Without a method, the work stays in conversations. With SPAD, each phase leaves a document stored in the folder of its topic.

When to use it: the SPAD page strongly recommends it for systems in production with customers, sensitive data, money or decisions about people, and in sectors with audit requirements (finance, insurance, health, public sector). It is useful, in a reduced version, for internal tools with business impact. It is unnecessary for throwaway prototypes or personal programs: there the cost of the process exceeds its benefit.

Why it matters to an executive: it gives you a list of simple questions to find out whether your team builds with a method or relies on luck.

Question for the classroom: choose a project you know. Which of the two columns would it be in today?

TERMS ON THIS SLIDE

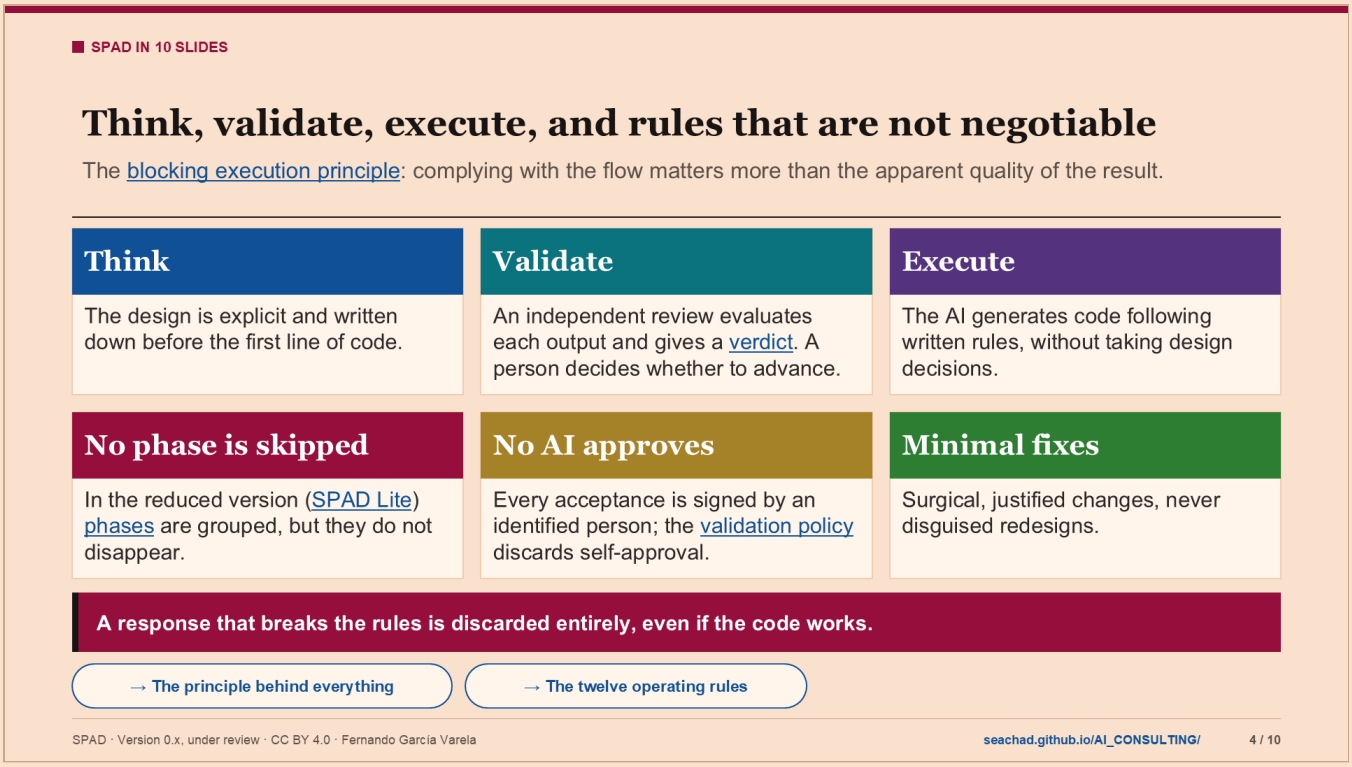
- **SPAD main cycle** (fases 0–10): SPAD's eleven phases: context, plan, plan review, code guide, test strategy, implementation, testing, test review, code review, fixes and release management. Each one leaves an artefact that

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 §2 · 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 00 §10 · 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
- **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
- **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 rests on three verbs. Think: the design is decided and written down before the code. Validate: an independent review evaluates each output and a person decides whether to advance. Execute: the AI writes the code following rules, but does not decide what the solution looks like.

The principle behind everything: in SPAD, complying with the flow is more important than the apparent quality of the result. The priority is, in this order, compliance with the process, technical correctness and efficiency.

Example: the AI is asked for a plan and delivers a good plan followed by five hundred lines of "example" code. Even if the code is good, the response is discarded entirely and the phase is repeated. Why? If it is accepted "just this once because it works", the process ceases to exist by the second time. A valid "no" from a review is worth more than an invalid "yes".

The operating guide (document 01) sets twelve non-negotiable rules. The cards show three of them. Others are:

- Every output is auditable: numbered, traceable documents.
- Every decision is explicit: there are no design decisions hidden in the code.
- Code executes, it does not decide.
- Independent reviews, also by a person.
- Mandatory tests, with minimum coverage set before building.
- Security by design.
- Each artefact states which model and which instructions produced it.

Why it matters to an executive: it is the segregation of duties of any serious process (whoever does the work does not approve it), applied to artificial intelligences as well. And it always leaves a person accountable.

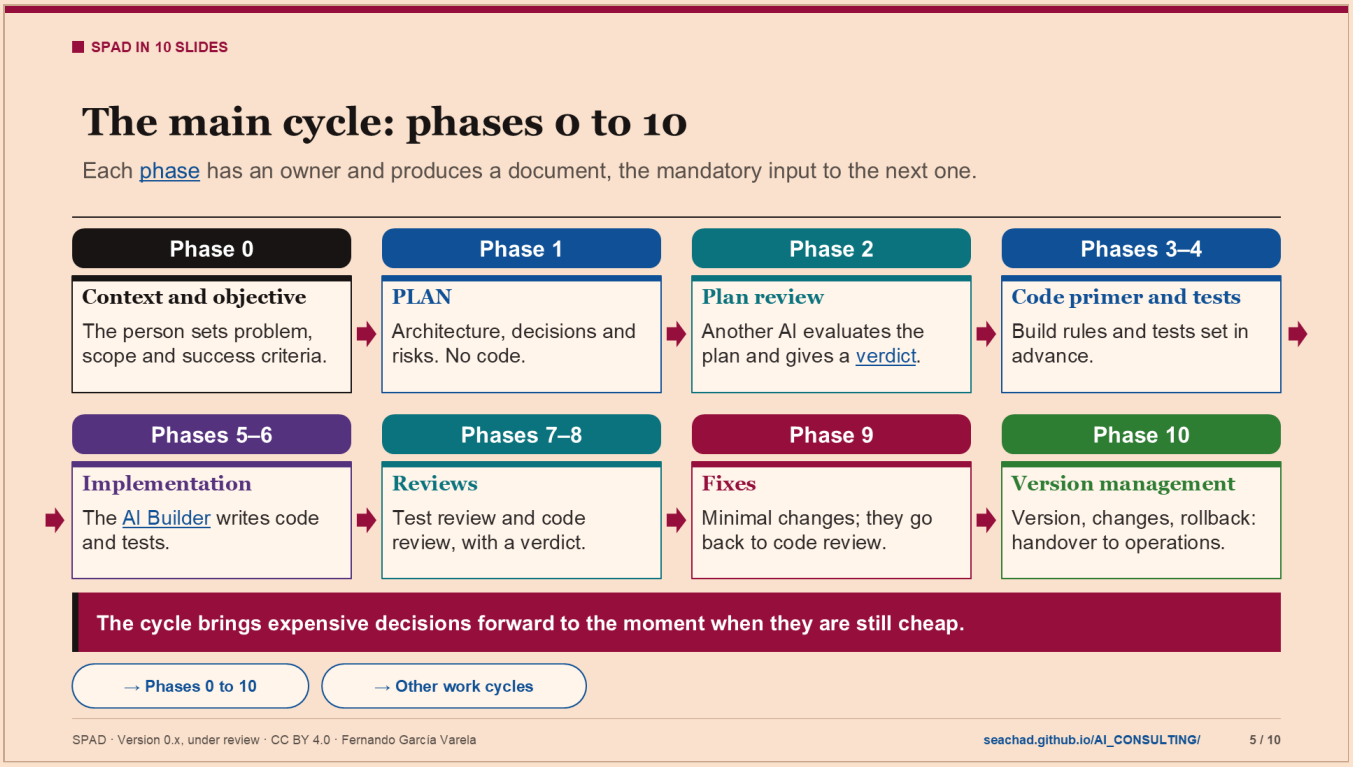
Question for the classroom: would your company accept a correct financial report that skipped the prior control? Why would software be different?

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
- **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 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
- **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
- **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 §3 · 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 01 §7 · Non-negotiable operating rules** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#non-negotiable-operating-rules
- **SPAD 03 §2 · Fundamental validation principle** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#fundamental-principle
- **The principle behind 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 operating 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, from 0 to 10. The slide groups some of them so that they fit.

- Phase 0 · Context and objective. A person (the orchestrator) writes what problem is solved, for whom, what is out of scope 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 the architecture, records each decision with its alternatives and classifies the risks. It does not write code.
- Phase 2 · Plan review. A different AI Reviewer evaluates the plan and issues a verdict. It does not fix it.
- Phases 3 and 4 · Code primer and test strategy. The decisions are translated into verifiable rules, and what will be tested and with what minimum coverage is set before building. These are the phases most often skipped and the ones that protect most value.
- Phases 5 and 6 · Implementation. The AI Builder writes the code and the tests. If the plan does not foresee something, it does not decide: it flags it and the person decides.
- Phases 7 and 8 · Test review and code review, including added dependencies and their licences.
- Phase 9 · Minimal fixes, which go back to the code review, not to rewriting everything.
- Phase 10 · Version management: version number, changes, deployment plan with tested rollback and what to monitor. It is the handover to operations.

Not all work is a new feature. SPAD has complementary cycles with the same rules: legacy (existing undocumented code), debugging (incidents), urgent fix (it speeds up the phases, it does not eliminate them) and security.

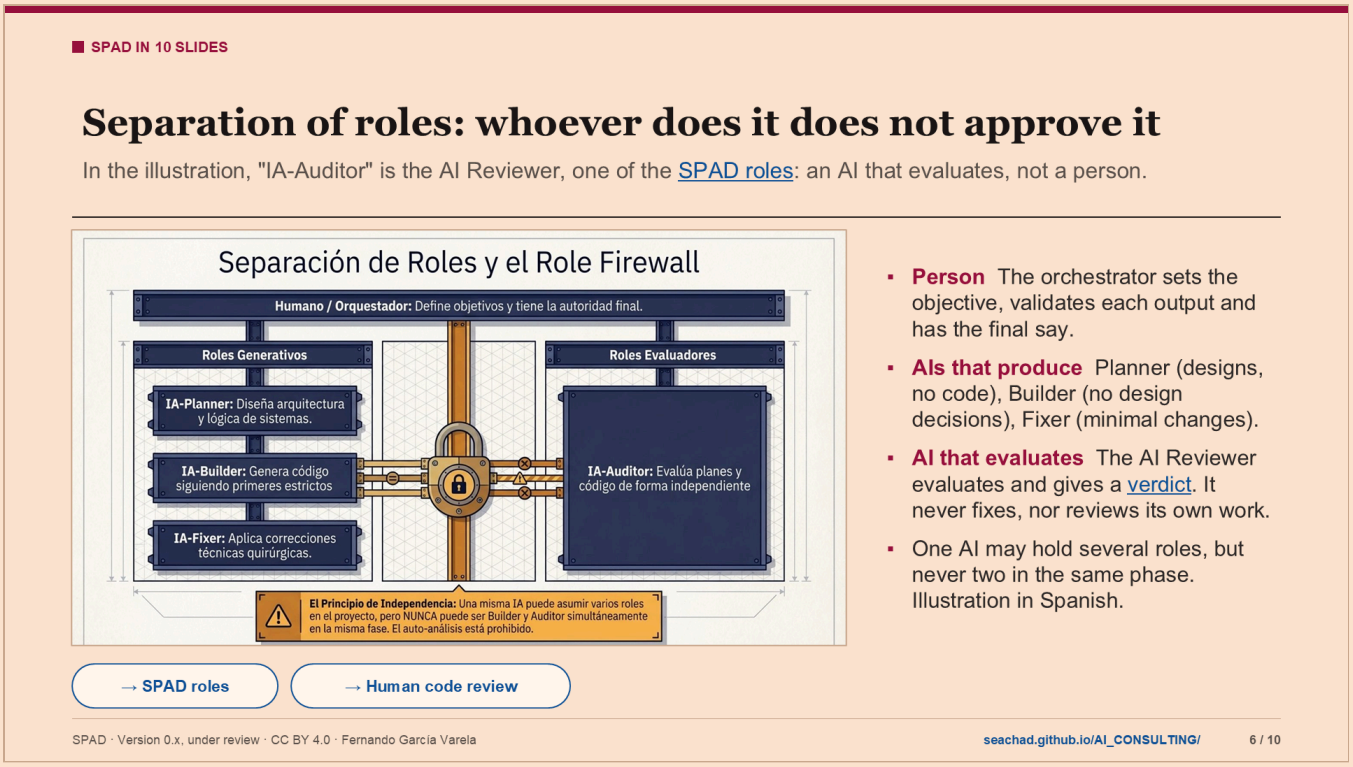
Why it matters to an executive: an architecture error detected in phase 2 costs a conversation; detected in production, it costs an incident and sometimes a customer's trust.

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 00 §5 · The main cycle** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#the-main-cycle
- **SPAD 04 · Complementary cycles** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html
- **Other work cycles** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html



Notes

SPAD separates logical roles from the people and the artificial intelligences that perform them.

- Orchestrator (person). Sets the objective and the context, validates each output, applies the validation policy and decides whether to advance. Has the final say. Cannot delegate validation to the AI that produced the output.
- AI Planner. Designs the architecture, the code primer, the test strategy and the version. Does not write code.
- AI Builder. Implements the code and the tests according to the plan. Does not take design decisions.
- AI Fixer. Applies minimal, justified fixes. Does not redesign or extend functionality.
- AI Reviewer. Reviews plans, tests and code and issues a verdict. Does not fix what it reviews or approve with known problems.
- Other roles: AI Analyst (documents legacy code), Diagnostic AI (analyses incidents) and AI Security Reviewer.

The key rule: one AI may take on several roles over a piece of work, but never two in the same phase, and the AI that reviews a document is never the one that produced it.

Human code review. Two artificial intelligences can make the same mistake. That is why, in addition, a competent person reviews the code before it is merged and leaves a record. For sensitive code (authentication, payments, personal data, agent limits) there is a second human review, by someone who did not orchestrate the generation.

A note on names: the reviewing role is called "AI Reviewer" and not "auditor", so that it is not confused with a human auditor. It is an AI: its verdict is always accepted by a person.

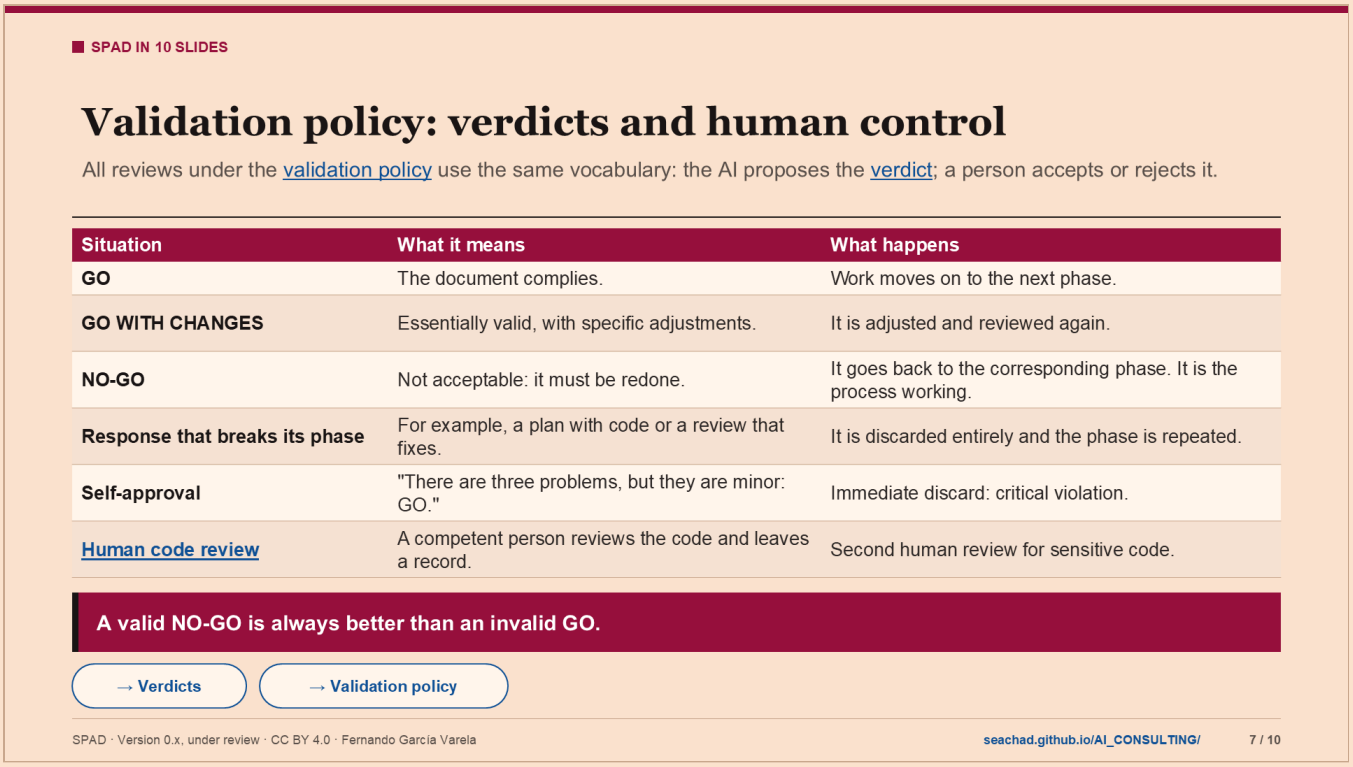
Why it matters to an executive: it is the classic control of segregation of duties. The one accountable to the organisation for the code being correct is a person, and that accountability is not delegated to a tool.

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 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 §3 · Roles** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#roles
- **SPAD 01 §3.1 · Human code review** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#3-1-human-code-review
- **SPAD 03 §3.5 · Self-approval** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#3-5-self-approval
- **SPAD roles** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#roles
- **Human code review** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#3-1-human-code-review



Notes

SPAD reviews (of the plan, the tests, the code and security) use only three verdicts.

- GO: the document complies and work advances.
- GO WITH CHANGES: it is essentially valid, but needs specific adjustments and a new review.
- NO-GO: it is not acceptable and must be redone. For example, a NO-GO in the plan review sends the work back to phase 0, because the problem is badly framed.

A single vocabulary makes it possible to compare phases, measure the process and automate checks. A verdict without written justification is incomplete and is discarded.

The validation policy is a rule for people, not a text for the AI. The orchestrator applies it, and it defines when a response is invalid even if it looks good. There are five causes:

- Phase violation: the response does something that does not belong to that phase.
- Incomplete document: for example, a review without a verdict.
- Off-plan decisions: for example, the implementation adds a cache that the plan did not foresee.
- Implementation without prior review: all the code is discarded.
- Self-approval: the AI approves itself or plays down problems. It is the most serious.

What does not invalidate: a well-founded NO-GO, a detected vulnerability or a failing test. That is the process working.

If a violation is repeated, the instruction is reinforced. If it persists after the maximum number of attempts set (starting value: three), the model is changed or a person steps in.

Why it matters to an executive: this is the point where the organisation exercises real control over what is built. A review that never says NO-GO for months is a warning sign: it has probably become a formality.

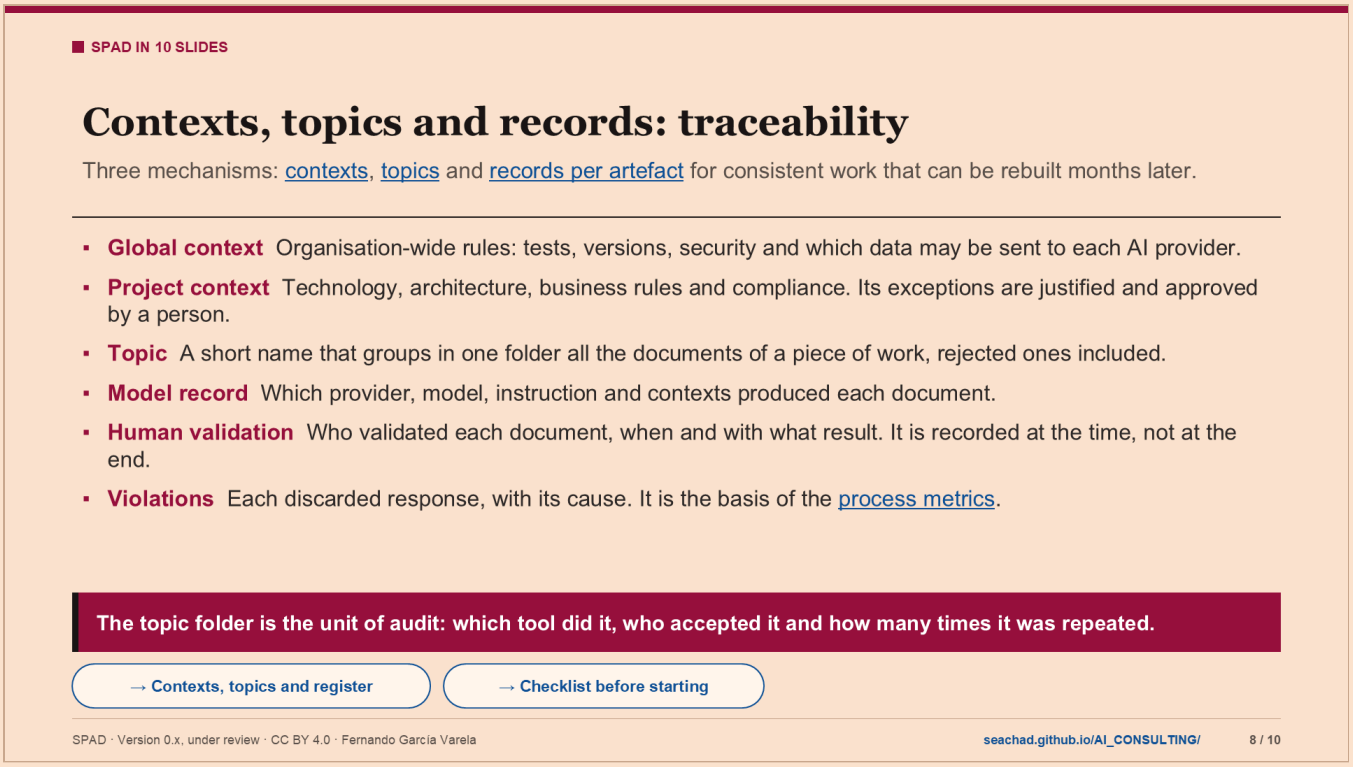
Question for the classroom: how does your organisation reward the reviewer who says "no"?

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
- **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
- **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

SOURCES AND LINKS

- **SPAD 03 §3 · Invalidation causes** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#invalidation-causes
- **SPAD 03 §4 · 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
- **SPAD 03 §5 · Procedure when a response is invalid** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#procedure-when-a-response-is-invalid



Notes

Before any phase two contexts are loaded. They are the rules the AI works with.

- Global context: rules common to all projects. Where documents are stored, minimum test coverage, versioning, baseline security, approved models per role and which classes of information may be sent to each AI provider. Illustrative example: "No real personal data in instructions to external providers; synthetic test data".
- Project context: language, architecture, business rules and compliance obligations. Illustrative example: "Customer data does not leave the European region".

If they clash, the project context prevails, but only with a written exception, with its risk, assessed by the AI Reviewer and approved by a person. No AI approves an exception.

The topic (TOPIC) is a short identifier, for example payment_gateway, that groups all the work in one folder: plan, reviews, code, tests, fixes and version. A rejected plan is not deleted: it is kept with its review and the next iteration takes the next number. This shows what was tried and why it was rejected.

Each document carries three records:

- Model record: which tool produced it. Without it, a result cannot be reproduced when the provider changes.
- Human validation record: who accepted it. Without it, traceability only proves what the AI did.
- Violation log: each discarded response, with date, phase, cause and model.

Why it matters to an executive: they answer the three questions of any development audit. In addition, the data and providers section is the one most often forgotten and the one that prevents most harm: without it, each person decides what data or secrets they paste into an external tool.

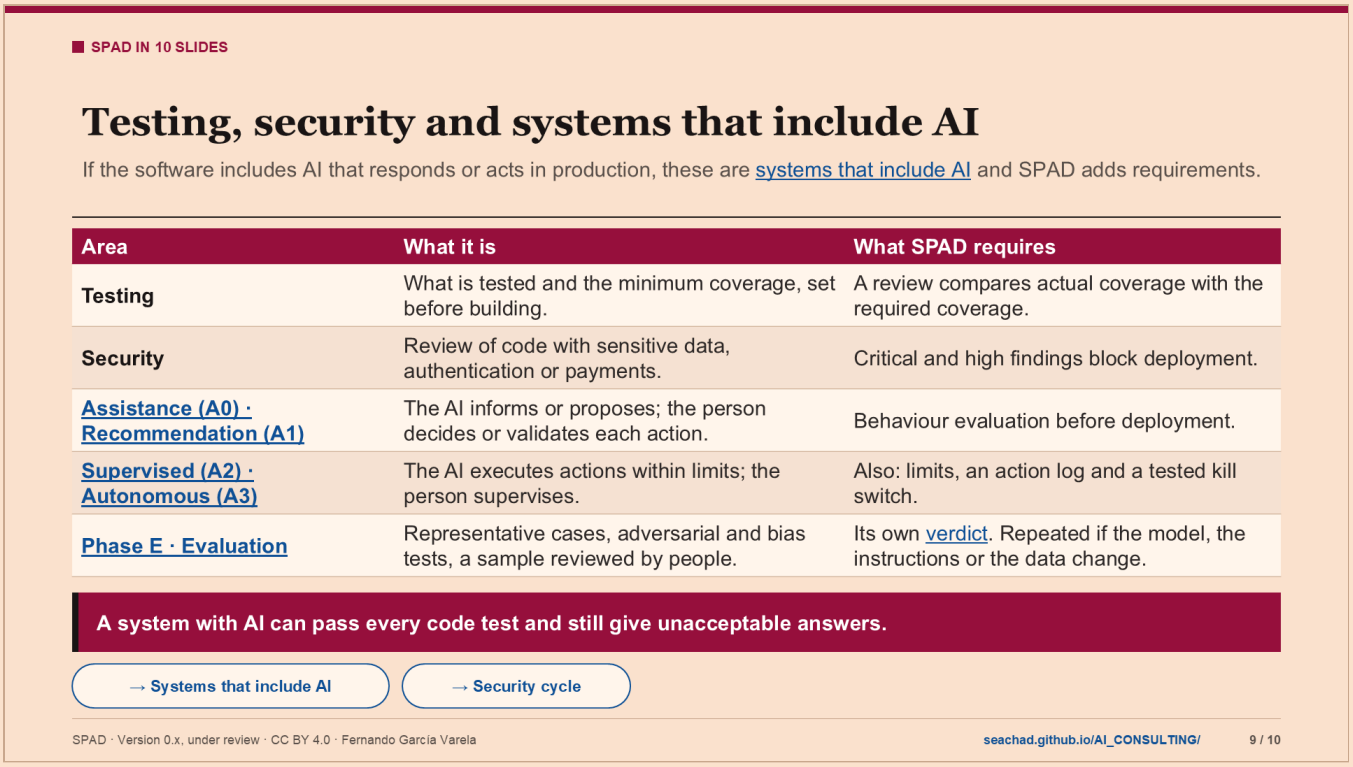
Question for the classroom: if tomorrow you were asked to explain why a critical system was built this way, where would you look?

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
- **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 02 §4 · 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
- **Checklist before starting** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#checklist-before-starting



SPAD IN 10 SLIDES

Testing, security and systems that include AI

If the software includes AI that responds or acts in production, these are systems that include AI and SPAD adds requirements.

Area	What it is	What SPAD requires
Testing	What is tested and the minimum coverage, set before building.	A review compares actual coverage with the required coverage.
Security	Review of code with sensitive data, authentication or payments.	Critical and high findings block deployment.
Assistance (A0) · Recommendation (A1)	The AI informs or proposes; the person decides or validates each action.	Behaviour evaluation before deployment.
Supervised (A2) · Autonomous (A3)	The AI executes actions within limits; the person supervises.	Also: limits, an action log and a tested kill switch.
Phase E · Evaluation	Representative cases, adversarial and bias tests, a sample reviewed by people.	Its own verdict. Repeated if the model, the instructions or the data change.

A system with AI can pass every code test and still give unacceptable answers.

→ Systems that include AI → Security cycle

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 9 / 10

Notes

Testing. In SPAD, the test strategy is written before building (phase 4): which cases, which types of test and what minimum coverage. That way tests are not written at the end to confirm what already exists. Later, an independent review (phase 7) compares actual coverage with the required coverage. Tests that merely repeat the code without verifying the expected behaviour do not count.

Security. Code that handles sensitive data, authentication, authorisation or payments goes through a security review, with recognised guides such as the OWASP Top 10 as a reference. Findings are classified by severity: critical and high ones block deployment; residual risk is always accepted by a person, never by an AI. The dependencies proposed by the AI are also reviewed (existence, origin, vulnerabilities and licence).

Systems that include AI. Using AI to write code is not the same as building a system whose behaviour depends on a model in production. In the second case there may be confidently wrong answers, prompt injection, data leakage, unauthorised actions, bias or drift. And behaviour can change without touching the code: it is enough for the provider to update the model.

That is why SPAD classifies each component with a four-level autonomy scale, from A0 (assistance) to A3 (autonomous action). Illustrative examples: A0, a summary of a case file for an analyst; A3, continuous price adjustment within an approved band. And it adds phase E, behaviour evaluation, between the test review and the code review.

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

Question for the classroom: what does your company do when the provider updates the model of an assistant that your customers already use?

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

- **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 05 · Systems that include AI** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#two-distinct-objects
- **SPAD 05 §4 · 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 04 §5 · Security cycle** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html#security-cycle
- **OWASP Top 10 for LLM Applications (2025)** https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/
- **Security cycle** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html#security-cycle



Notes

Measure. SPAD does not claim improvement figures. Each organisation measures whether the method works against its own starting point, with indicators drawn from the records: invalidation rate per phase and per model, NO-GO rate, iterations to GO, time per phase, actual versus required coverage, fixes without a finding and defects escaped to production, which is the figure that really matters. A NO-GO rate close to zero for months indicates that the review has become a formality.

Reduced version (SPAD Lite). For bounded changes, without personal data or money, with a small team. It groups the phases into three documents. It does not cut the separation between who builds and who reviews, human validation, the validation policy or the topic register.

Self-assessment. A person independent of those who built checks six conditions: decisions before code, one document per phase, independent reviews, no phase skipped, minimal fixes and no approval by an AI. It is not a certification: neither its author nor any third party issues or endorses it.

What it does for your company (the author's position, with no figures). First use: building your own software with AI with method. Second use, as a client: a company that knows how software is built with AI can require its suppliers and integrators to pass the productivity gain on to it, as lower cost or more scope for the same price. It can ask them for what SPAD already defines: the artefact of each phase, the record of the model and of human validation, the verdicts and the process metrics. It sets no percentage: it lets you negotiate with data.

A separate framework. SPAD 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.

How to get started: choose a bounded piece of work, use the reduced version and follow the checklist in document 02.

Why it matters to an executive: a board does not need to know how a plan is reviewed, but it does need to know that there is a method that does so and leaves evidence.

TERMS ON THIS SLIDE

- **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
- **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
- **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 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

SOURCES AND LINKS

- **SPAD 08 §2 · Conformity conditions** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html#conformity-conditions
- **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
- **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
- **SPAD 02 §5 · Checklist before starting** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#checklist-before-starting
- **Self-assessment and metrics** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html
- **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

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/